Zero-Knowledge Arguments for Lattice-Based PRFs and Applications to E-Cash

Benoît Libert, San Ling, Khoa Nguyen, Huaxiong Wang

Zero-Knowledge Arguments for Lattice-Based PRFs and Applications to E-Cash

Benoît Libert[1,2], San Ling[3], Khoa Nguyen[3], and Huaxiong Wang[3]

[1] CNRS, Laboratoire LIP, France
[2] ENS de Lyon, Laboratoire LIP (U. Lyon, CNRS, ENSL, INRIA, UCBL), France
[3] School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore

Abstract. Beyond their security guarantees under well-studied assumptions, algebraic pseudo-random functions are motivated by their compatibility with efficient zero-knowledge proof systems, which is useful in a number of privacy applications like digital cash. We consider the problem of proving the correct evaluation of lattice-based PRFs based on the Learning-With-Rounding (LWR) problem introduced by Banerjee *et al.* (Eurocrypt'12). Namely, we are interested zero-knowledge arguments of knowledge of triples (y, k, x) such that $y = F_k(x)$ is the correct evaluation of a PRF for a secret input x and a committed key k. While analogous statements admit efficient zero-knowledge protocols in the discrete logarithm setting, they have never been addressed in lattices so far. We provide such arguments for the key homomorphic PRF of Boneh *et al.* (Crypto'13) and the generic PRF implied by the LWR-based pseudo-random generator. As an application of our ZK arguments, we design the first compact e-cash system based on lattice assumptions. By "compact", we mean that the complexity is at most logarithmic in the value of withdrawn wallets. Our system can be seen as a lattice-based analogue of the first compact e-cash construction due to Camenisch, Hohenberger and Lysyanskaya (Eurocrypt'05).

Keywords. Lattices, pseudo-random functions, zero-knowledge arguments, e-cash systems, anonymity.

1 Introduction

Since the seminal results of Ajtai [2] and Regev [88], lattice-based cryptography has been a very active area which undergone quite rapid development, notably with the advent of lattice trapdoors [54,79] and homomorphic encryption [53]. Not only does it enable powerful functionalities, it also offers many advantages over conventional number-theoretic techniques, like simpler arithmetic operations, its conjectured resistance to quantum attacks or a better asymptotic efficiency.

The design of numerous cryptographic protocols appeals to zero-knowledge proofs [57] to prove properties about encrypted or committed values so as to enforce honest behavior on behalf of participants or protect the privacy of users. In the lattice settings, efficient zero-knowledge proofs are non-trivial to construct. While natural solutions exist for proving knowledge of secret keys [80,76,67,73],

they are only known to work for very specific languages. When it comes to proving general circuit satisfiability, the best known methods rely on the ring variants [92,14] of the Learning-With-Errors (LWE) and Short Integer Solution (SIS) problems and are not known to readily carry over to standard lattices. In the standard model, the problem is even trickier as we do not have a lattice-based counterpart of Groth-Sahai proofs [60] and efficient non-interactive proof systems are only available for specific problems [87].

In this paper, we consider the natural problem of proving the correct evaluation of lattice-based pseudo-random functions (PRFs) w.r.t. committed keys and inputs. This problem arises in numerous protocols where a user has to deterministically generate a random-looking value without betraying his identity.

We provide zero-knowledge arguments of correct evaluation for the LWE-based PRF of Boneh, Lewi, Montgomery and Raghunathan (BLMR) [17] as well as the construction generically obtained from pseudo-random generators via the Goldreich-Goldwasser-Micali (GGM) methodology [56]. As an application of our arguments, we provide the first lattice-based realization of the compact e-cash primitive of Camenisch, Hohenberger and Lysyanskaya [23].

Introduced by Chaum [34,35], electronic cash is the digital counterpart of regular money. As envisioned in [34], digital cash involve a bank and several users and merchants. It allows users to withdraw digital coins from the bank in such a way that e-coins can later be spent at merchants. In the on-line setting [34,36,37], merchants contact the bank before accepting any payment so that the bank is involved in all transactions to prevent double-spendings. In the (usually preferred) off-line model [38], the merchant accepts payments without any interaction with the bank: the deposit phase is postponed to a later moment where the merchant can return many coins at once. In all cases, when a merchant returns coins back to the bank, the latter should infer no information as to when and by whom the returned coins were withdrawn. Should the bank collude with the merchant, it remains unable to link a received coin to a particular execution of the withdrawal protocol. Of course, dishonest users should not be able to spend more coins than they withdrew without being identified. While fair e-cash systems [91] resort to an off-line trusted authority to call out cheaters, classical e-cash [38] allows identifying double-spenders without any TTP. In 2005, Camenisch, Hohenberger and Lysyanskaya [23] advocated e-cash solutions with *compactness* property: namely, a compact e-cash scheme allows a user to withdraw a wallet of 2^L coins in such a way that the complexity of spending and withdrawal protocols does not exceed $\mathcal{O}(L + \lambda)$, where λ is the security parameter. The constructions of [23] elegantly combine signature schemes with efficient protocols [26,27], number theoretic pseudo-random functions [46] and zero-knowledge proofs, making it possible to store a wallet using only $\mathcal{O}(L + \lambda)$ bits.

1.1 Our Contributions

Our Results. We describe the first compact e-cash system [23] based on lattice assumptions. Here, consistently with the literature on e-cash, "compactness" refers to schemes where the withdrawal, spending and deposit phases have at most

logarithmic complexities in the maximal value of withdrawn wallets (analogously to the solutions of [23] where the term "compact" was introduced). The security of our scheme is proved in the random oracle model [11] under the Short Integer Solution (SIS) and LWE assumptions.

As a crucial ingredient of our solution, we provide zero-knowledge arguments vouching for the correct evaluation of lattice-based pseudo-random functions. More precisely, we construct arguments of knowledge of a committed seed $\mathbf{k}$, a secret input J and an output $\mathbf{y}$ satisfying $\mathbf{y} = F_{\mathbf{k}}(J)$. We describe such arguments for the key-homomorphic PRF of Boneh *et al.* [17] and the PRF obtained by applying the Goldreich-Goldwasser-Micali (GGM) [56] paradigm. As a building block, we provide zero-knowledge arguments for statements related to the Learning-With-Rounding (LWR) problem of Banerjee, Peikert and Rosen [8]. Given a public random matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, it requires to tell apart vectors $\lfloor \mathbf{A}^T \cdot \mathbf{s} \rceil_p = \lfloor (p/q) \cdot \mathbf{A}^T \cdot \mathbf{s} \rceil \in \mathbb{Z}_p^m$ from the uniform distribution $U(\mathbb{Z}_p^m)$ over $\mathbb{Z}_p^m$, where $q > p \geq 2$. A crucial step of our argument system consists in demonstrating the correct computation of the rounding step: i.e., proving that $\mathbf{y} = \lfloor \mathbf{x} \rceil_p$, for $\mathbf{x} \in \mathbb{Z}_q^m$ satisfying some additional context-dependent constraints.

We believe that our zero-knowledge arguments can find use cases in many other applications involving PRFs, and where zero-knowledge proofs constrain participants not to deviate from the protocol. Examples include privacy-preserving de-centralized e-cash systems [12,59,40], stateful anonymous credentials [41], n-times periodic anonymous authentication [22], traceable ring signatures [52], anonymous survey systems [61], password-protected secret sharing [64] or unlinkable pseudonyms for privacy-preserving distributed databases [25]. We also think of distributed PRFs [78,83], where servers holding a polynomial share $\mathbf{k}_i$ of the seed $\mathbf{k}$ can prove the correctness of their contribution w.r.t. to their committed share $\mathbf{k}_i$. Our argument for the key homomorphic PRF of [17] allows proving the correctness of partial evaluations at the expense of a small amount of interaction.[4] Our arguments may also prove useful in the context of oblivious PRF evaluations [51,65], where one party holds a PRF key $\mathbf{k}$ and must convince the other party that $\mathbf{k}$ was correctly used in oblivious computations.

Our Techniques. In order to convince a verifier of the correct evaluation of LWR-based PRFs, the first step is to provide evidence that the underlying rounding operation is properly carried out. For dimension $m > 1$ and moduli $q > p \geq 2$, identify $\mathbb{Z}_q$, $\mathbb{Z}_p$ as the set $[0, q-1]$ and $[0, p-1]$, respectively, and consider the function $\lfloor \cdot \rceil_p : \mathbb{Z}_q^m \to \mathbb{Z}_p^m : \mathbf{x} \mapsto \mathbf{y} = \lfloor (p/q) \cdot \mathbf{x} \rceil \bmod p$. We observe that, one knows secret vector $\mathbf{x} \in [0, q-1]^m$ such that $\lfloor \mathbf{x} \rceil_p = \mathbf{y}$ for a given $\mathbf{y} \in [0, p-1]^m$, if and only if one knows $\mathbf{x}, \mathbf{z} \in [0, q-1]^m$ such that

$$p \cdot \mathbf{x} = q \cdot \mathbf{y} + \mathbf{z} \mod pq. \tag{1}$$

[4] The key homomorphic PRFs of Boneh *et al.* [17] were motivated by the design of non-interactive distributed PRFs in the standard model. By combining our interactive arguments and their construction, we can still achieve distributed PRFs in the standard model with less interaction than previous standard model realizations.

This crucial observation gives us a modular equation where the secret vectors $\mathbf{x}, \mathbf{z}$ are "small" relatively to the modulus pq. To prove that we know such secret vectors (where $\mathbf{x}$ may satisfy additional statements, e.g., it is committed, or certified, or is the output of other algorithms), we exploit Ling et al.'s decomposition-extension framework [73], which interacts well with Stern's permuting technique [89]. Specifically, we employ a matrix $\mathbf{H}_{m,q-1} \in \mathbb{Z}_q^{m \times \bar{m}}$, where $\bar{m} = m\lceil \log q \rceil$, that allows to compute $\tilde{\mathbf{x}}, \tilde{\mathbf{z}} \in \{0,1\}^{\bar{m}}$ such that $\mathbf{H}_{m,q-1} \cdot \tilde{\mathbf{x}} = \mathbf{x}$ and $\mathbf{H}_{m,q-1} \cdot \tilde{\mathbf{z}} = \mathbf{z}$. Then, we let $\mathsf{B}^2_{\bar{m}}$ be the set of all vectors in $\{0,1\}^{2\bar{m}}$ that have fixed Hamming weight $\bar{m}$, and append $\bar{m}$ suitable entries to $\tilde{\mathbf{x}}, \tilde{\mathbf{z}}$ to obtain $\widehat{\mathbf{x}}, \widehat{\mathbf{z}} \in \mathsf{B}^2_{\bar{m}}$. Now, equation (1) is rewritten as:

$$\left(p \cdot [\mathbf{H}_{m,q-1} \mid \mathbf{0}^{m \times \bar{m}}]\right) \cdot \widehat{\mathbf{x}} - [\mathbf{H}_{m,q-1} \mid \mathbf{0}^{m \times \bar{m}}] \cdot \widehat{\mathbf{z}} = q \cdot \mathbf{y} \bmod pq. \tag{2}$$

Note that, one knows $\mathbf{x}, \mathbf{z} \in [0, q-1]^m$ satisfying (1) if and only if one can compute $\widehat{\mathbf{x}}, \widehat{\mathbf{z}} \in \mathsf{B}^2_{\bar{m}}$ satisfying (2). Moreover, as the constraint of $\widehat{\mathbf{x}}, \widehat{\mathbf{z}}$ is invariant under permutation (namely, $\widehat{\mathbf{x}}, \widehat{\mathbf{z}} \in \mathsf{B}^2_{\bar{m}}$ if and only if $\pi_x(\widehat{\mathbf{x}}), \pi_z(\widehat{\mathbf{z}}) \in \mathsf{B}^2_{\bar{m}}$, where π_x, π_z are permutations of $2\bar{m}$ elements), the latter statement can be handled via Stern's technique. Our method is readily extended to prove that the underlying vector $\mathbf{x}$ satisfies additional statements.

Let us now consider the problem of proving a correct evaluation of the Boneh *et al.* PRF [17]. The function uses public binary matrices $\mathbf{P}_0, \mathbf{P}_1 \in \{0,1\}^{m \times m}$ and a secret seed $\mathbf{k} \in \mathbb{Z}_q^m$ which allows mapping an input $J \in \{0,1\}^L$ to

$$F_{\mathbf{k}}(J) = \left\lfloor \mathbf{P}_{J[L]} \cdot \mathbf{P}_{J[L-1]} \cdots \mathbf{P}_{J[1]} \cdot \mathbf{k} \right\rfloor_p.$$

We consider the evaluation process iteratively and transform intermediate witnesses using the decomposition-extension framework [73], so that they nicely interact with Stern's permuting technique [89]. Namely, we define a sequence $\{\mathbf{x}_i\}_{i=0}^L$ which is initialized with $\mathbf{x}_0 = \mathbf{k} \in \mathbb{Z}_q^m$, iteratively computed as $\mathbf{x}_i = \mathbf{P}_{J[i]} \cdot \mathbf{x}_{i-1} \in \mathbb{Z}_q^m$, for each $i \in [1, L]$, and eventually yields the output $\mathbf{y} = \lfloor \mathbf{x}_L \rfloor_p$. For each $i \in [1, L]$, we translate the equation $\mathbf{x}_i = \mathbf{P}_{J[i]} \cdot \mathbf{x}_{i-1} \bmod q$ into

$$\mathbf{x}_i = \left[\mathbf{P}_0 \mid \mathbf{P}_1\right] \cdot \mathbf{t}_{i-1} \bmod q, \qquad \text{with} \qquad \mathbf{t}_{i-1} = \begin{pmatrix} \overline{J[i]} \cdot \mathbf{x}_{i-1} \\ J[i] \cdot \mathbf{x}_{i-1} \end{pmatrix}$$

and where $J[i]$ and $\overline{J[i]} = 1 - J[i]$ are part of the witnesses. Using suitable decomposition-extension techniques [73,72] on vectors $\{\mathbf{x}_i\}_{i=0}^L, \{\mathbf{t}_i\}_{i=1}^L$, we manage to express all the L iterative equations by just one equation of the form $\mathbf{M}_1 \cdot \mathbf{w}_1 = \mathbf{u}_1 \bmod q$, for some public matrix $\mathbf{M}_1$ and vector $\mathbf{u}_1$ over $\mathbb{Z}_q$, while $\mathbf{w}_1$ is a binary vector containing secret bits of all the witnesses and fitting a certain pattern. Meanwhile, the rounding step $\mathbf{y} = \lfloor \mathbf{x}_L \rfloor_p$, as discussed above, would yield an equation of the form $\mathbf{M}_2 \cdot \mathbf{w}_2 = \mathbf{u}_2 \bmod pq$, where $\mathbf{w}_2$ is correlated to $\mathbf{w}_1$. Furthermore, our applications require to additionally prove that a binary representation of the seed $\mathbf{x}_0 = \mathbf{k}$ is properly committed or certified, while the commitment or signature scheme may use a different modulus. Thus, we eventually have to handle relations of the form $\mathbf{M}_i \cdot \mathbf{w}_i = \mathbf{u}_i \bmod q_i$ for several moduli $q_1, \ldots, q_N$ when, for distinct $i, j \in [N]$, witnesses $\mathbf{w}_i, \mathbf{w}_j$ may have entries

in common. An abstraction of Stern's protocol was recently suggested by Libert *et al.* [71] to address a similar setting when one has to prove a number of linear relations. Unfortunately, their framework, which deals with a unique modulus, does not directly cover our setting. To overcome this problem, we thus put forward a generalization of Libert *et al.*'s framework, so as to handle correlated witnesses across relations modulo distinct integers.

The above techniques thus smoothly interact with the pseudo-random functions of Boneh *et al.* [17] and the PRG of [8]. Unfortunately, we did not manage to extend them to other existing PRFs [8,7,47] based on the hardness of LWR. In the synthesizer-based construction of Banerjee *et al.* [8], the difficulty is the lack of public matrices which would help us reduce the statement to an assertion of the form $\mathbf{M} \cdot \mathbf{w} = \mathbf{u}$, for some witness $\mathbf{w} \in \mathbb{Z}^m$ and public $\mathbf{M} \in \mathbb{Z}_q^{n \times m}$, $\mathbf{u} \in \mathbb{Z}_q^n$. Our zero-knowledge arguments do not appear to carry over to the key homomorphic functions of Banerjee and Peikert [7] either as they rely on a more complex tree-like structure. The fact that our techniques do *not* apply to all known lattice-based PRFs emphasizes that they are far more innovative than just an application of generic zero-knowledge paradigms, which would resort to a circuit decomposition of the evaluation algorithm and proceed in a gate-by-gate manner. Indeed, we process all statements without decomposing the arithmetic operations into a circuit.

Our compact e-cash construction builds on the design principle of Camenisch *et al.* [23] which combines signatures with efficient protocols [26,27], algebraic pseudo-random functions [46] and zero-knowledge proofs. In the lattice setting, we take the same approach by combining (a variant of) the signature scheme with efficient protocols of [71] and the PRF of [17]. While the GGM-based PRF of [8] would allow a more efficient choice of parameters, we chose to instantiate our system with the realization of Boneh *et al.* [17] since it simplifies the description and the security proof (specifically, we do not have to rely on the pseudo-randomness of the function in one of the security properties). However, our scheme can be modified to rely on the PRF built on the LWR-based PRG.

As in [23], the withdrawal phase allows the user to obtain a wallet of value $2^L - 1$ which consists of two PRF seeds, a counter and a signature generated by the bank on committed values. In the withdrawal protocol, the PRF seeds are obliviously signed (and bound to the user's secret key) by the bank using a signature scheme with efficient protocols [26,27]. The first seed $\mathbf{k}$ is used to derive each coin's serial number $\mathbf{y}_S = F_{\mathbf{k}}(J) \in \mathbb{Z}_p^m$ as a pseudo-random function of an L-bit counter $J \in \{0,1\}^L$ which denotes the number of previously spent coins. By spending the same coin twice, the user is thus forced to use the same serial number in two distinct transactions, making the cheating attempt detectable.

The second PRF seed $\mathbf{t}$ is used to compute a security tag $\mathbf{y}_T$ that allows identifying double-spenders. This tag is a vector $\mathbf{y}_T = PK_{\mathcal{U}} + H(\mathtt{info}) \cdot F_{\mathbf{t}}(J) \in \mathbb{Z}_p^m$, where $PK_{\mathcal{U}}$ is the user's public key and $H(\mathtt{info}) \in \mathbb{Z}_p^{m \times m}$ is a matrix generated by hashing some transaction-specific information supplied by the merchant. From two coins that share the same serial number $\mathbf{y}_S$ and distinct security tags $\mathbf{y}_{T,1} = PK_{\mathcal{U}} + H(\mathtt{info}_1) \cdot F_{\mathbf{t}}(J)$ and $\mathbf{y}_{T,2} = PK_{\mathcal{U}} + H(\mathtt{info}_2) \cdot F_{\mathbf{t}}(J)$, the differ-

ence $\mathbf{y}_{T,1} - \mathbf{y}_{T,2} = (H(\mathtt{info}_1) - H(\mathtt{info}_2)) \cdot F_{\mathbf{t}}(J)$ allows computing the PRF value $F_{\mathbf{t}}(J) = (H(\mathtt{info}_1) - H(\mathtt{info}_2))^{-1} \cdot (\mathbf{y}_{T,1} - \mathbf{y}_{T,2}) \in \mathbb{Z}_p^m$ (and then $PK_{\mathcal{U}}$) whenever $H(\mathtt{info}_1) - H(\mathtt{info}_2)$ is invertible over $\mathbb{Z}_p$. This property is precisely ensured by the Full-Rank Difference function of Agrawal *et al.* [1], which comes in handy to instantiate $H : \{0,1\}^* \to \mathbb{Z}_p^{m \times m}$. In contrast with [1], the Full-Rank Difference function is utilized in the scheme while [1] uses it in security proofs.

1.2 Related Work

E-Cash. Chaum's pioneering work [34,35] inspired a large body of research towards efficient e-cash systems [38,85,39,49,86,90] with better properties during two decades. The first compact realization was given by Camenisch *et al.* [23] whose techniques served as a blueprint for many subsequent e-cash systems with additional features such as refined accountability-anonymity tradeoffs [24], coin endorsement [28], or security proofs in the standard model [10]. The authors of [23] extended their schemes with a coin tracing mechanism whereby all the coins of a double-spender can be traced once this user has been identified.

Divisible e-cash [85] allow users to withdraw a wallet of value 2^L in such a way that each spending may involve transactions of variable amounts. While the early constructions [85,86] only provided weaker anonymity properties, Canard and Gouget gave truly anonymous realizations [29] using tree-based techniques which were subsequently made scalable [30,31]. The recent adoption of de-centralized payment systems [82] has triggered a new line of research towards strengthening the privacy of Bitcoin (see [81,12,59] and references therein).

To our knowledge, all truly private compact e-cash systems rely on discrete-logarithm-based techniques, either because of the underlying pseudo-random function [23,10] or via accumulators [5] (or both). In the lattice setting, we are not aware of any compact e-cash realization and neither do we know of any proofs of correct PRF evaluation with or without random oracles. In particular, it remains an open problem to build verifiable random functions [77] from lattices.

Lattices and Zero-knowledge Proofs. Existing methods of proving relations appearing in lattice-based cryptosystems belong to two main families. The first family, introduced by Lyubashevsky [76], uses "rejection sampling" techniques, and recently lead to relatively efficient proofs of knowledge of small secret vectors [13,14,9,43,45]. However, due to the nature of "rejection sampling" mechanisms, even the honest prover may fail to convince the verifier with a tiny probability: i.e., protocols in this family do not have perfect completeness. Furthermore, when proving knowledge of vectors having norm bound β, the knowledge extractor of these protocols is only guaranteed to produce witnesses of norm bound $g \cdot \beta$, for some factor $g > 1$. This factor, called the "soundness slack" in [9,43], may have an undesirable consequence: if an extracted witness has to be used in the security proof to solve a challenge SIS instance, we have to rely on the $\mathsf{SIS}_{g \cdot \beta}$ assumption, which is stronger than the SIS_β assumption required by the protocol itself. Moreover, in some advanced protocols such as those considered in this work, the coordinates of extracted vectors are expected to be in $\{0,1\}$ and/or satisfy a specific pattern. Such issues seem hard to tackle

using this family of protocols.

The second family, initiated by Ling *et al.* [73], rely on "decomposition-extension" techniques in lattice-based analogues [67] of Stern's protocol [89]. Stern-like systems are less efficient than those of the first family because each protocol execution admits a constant soundness error, requiring the protocols to be repeated $\omega(\log \lambda)$ times in order to achieve a negligible soundness error. On the upside, Stern-like protocols do have perfect completeness and are capable of handling a wide range of lattice-based relations [69,74,72,71,70], especially when the witnesses are not only required to be small or binary, but should also have prescribed arrangements of coordinates. Moreover, unlike protocols of the first family, the extractor of Stern-like protocols are able to output witness vectors having exactly the same properties as those expected from valid witnesses. This feature is often crucial in the design of advanced cryptographic constructions involving zero-knowledge proofs. Additionally, the "soundness slack" issue is completely avoided, so that the hardness assumptions are kept "in place".

When it comes to proving the correct evaluation of AES-like secret key primitives, several works [66,50,33] built zero-knowledge proofs upon garbled circuits or multi-party computation [63,55], which may lead to truly practical proofs [55] even for non-algebraic statements. However, the garbled circuit paradigm [66] inherently requires interactive proofs (and cannot be made non-interactive via Fiat-Shamir [48]), making it unsuitable to our applications where coins must carry a non-interactive proof. While Giacomelli *et al.* [55] successfully designed efficient non-interactive proofs for SHA-256 evaluations, these remain of linear length in the circuit size and efficiently combining them with proofs of algebraic statements is non-trivial here: in the e-cash setting, our goal is to prove the correct evaluation of LWE-based symmetric primitives for committed inputs and keys. To our knowledge, known results on the smooth integration of algebraic and non-algebraic statements [33] are obtained by tweaking the approach of Jawurek *et al.* [66], which requires interaction.

Despite the scarcity of truly efficient zero-knowledge proofs in the lattice-setting, a recent body of work successfully designed proof systems in privacy-preserving protocols [67,58,68,13,84,74]. These results, however, only considered ring signatures [19,67], group signatures [58,68,69,13,84,74], group encryption [70] or building blocks [71] for anonymous credentials [36]. As of the time of writing, lattice-based realizations of anonymous e-cash still remain lacking.

2 Background and Definitions

Vectors are denoted in bold lower-case letters and bold upper-case letters will denote matrices. The Euclidean and infinity norm of any vector $\mathbf{b} \in \mathbb{R}^n$ will be denoted by $\|\mathbf{b}\|$ and $\|\mathbf{b}\|_\infty$, respectively. The Euclidean norm of matrix $\mathbf{B} \in \mathbb{R}^{m \times n}$ with columns $(\mathbf{b}_i)_{i \leq n}$ is $\|\mathbf{B}\| = \max_{i \leq n} \|\mathbf{b}_i\|$. When $\mathbf{B}$ has full column-rank, we let $\widetilde{\mathbf{B}}$ denote its Gram-Schmidt orthogonalization.

When S is a finite set, we denote by $U(S)$ the uniform distribution over S and by $x \hookleftarrow U(S)$ the action of sampling x according to this distribution.

For any $\mathbf{x} \in \mathbb{Z}_q^m$, the notation $\lfloor \mathbf{x} \rceil_p$ stands for the result of the rounding operation $\lfloor \mathbf{x} \rceil_p = \lfloor (p/q) \cdot \mathbf{x} \rfloor \bmod p$. Intuitively, the mapping $\lfloor \cdot \rceil_p : \mathbb{Z}_q^m \to \mathbb{Z}_p^m$ can be seen as dividing $\mathbb{Z}_q$ into p intervals of size (q/p) and sending each coordinate of $\mathbf{x} \in \mathbb{Z}_q^m$ to the interval it belongs to.

The column concatenation of matrices $\mathbf{A} \in \mathbb{R}^{n \times k}$ and $\mathbf{B} \in \mathbb{R}^{n \times m}$ is denoted by $[\mathbf{A} \,|\, \mathbf{B}] \in \mathbb{R}^{n \times (k+m)}$. When concatenating column vectors $\mathbf{x} \in \mathbb{R}^k$ and $\mathbf{y} \in \mathbb{R}^m$, for simplicity, we often use the notation $(\mathbf{x} \| \mathbf{y}) \in \mathbb{R}^{k+m}$ (instead of $(\mathbf{x}^\top \| \mathbf{y}^\top)^\top$).

2.1 Lattices

A lattice L is the set of integer linear combinations of linearly independent basis vectors $(\mathbf{b}_i)_{i \leq n}$ living in $\mathbb{R}^m$. We work with q-ary lattices, for some prime q.

Definition 1. *Let $m \geq n \geq 1$, a prime $q \geq 2$ and $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and $\mathbf{u} \in \mathbb{Z}_q^n$, define $\Lambda_q(\mathbf{A}) := \{\mathbf{e} \in \mathbb{Z}^m \mid \exists \mathbf{s} \in \mathbb{Z}_q^n \;\; s.t. \;\; \mathbf{A}^T \cdot \mathbf{s} = \mathbf{e} \bmod q\}$ as well as*

$$\Lambda_q^\perp(\mathbf{A}) := \{\mathbf{e} \in \mathbb{Z}^m \mid \mathbf{A} \cdot \mathbf{e} = \mathbf{0}^n \bmod q\}, \quad \Lambda_q^{\mathbf{u}}(\mathbf{A}) := \{\mathbf{e} \in \mathbb{Z}^m \mid \mathbf{A} \cdot \mathbf{e} = \mathbf{u} \bmod q\}$$

For any arbitrary $\mathbf{t} \in \Lambda_q^{\mathbf{u}}(\mathbf{A})$, we also define the shifted lattice $\Lambda_q^{\mathbf{u}}(\mathbf{A}) = \Lambda_q^\perp(\mathbf{A}) + \mathbf{t}$.

For a lattice L, a vector $\mathbf{c} \in \mathbb{R}^m$ and a real number $\sigma > 0$, define the function $\rho_{\sigma,\mathbf{c}}(\mathbf{x}) = \exp(-\pi \|\mathbf{x} - \mathbf{c}\|^2 / \sigma^2)$. The discrete Gaussian distribution of support L, center $\mathbf{c}$ and parameter σ is defined as $D_{L,\sigma,\mathbf{c}}(\mathbf{y}) = \rho_{\sigma,\mathbf{c}}(\mathbf{y}) / \rho_{\sigma,\mathbf{c}}(L)$ for any $\mathbf{y} \in L$, where $\rho_{\sigma,\mathbf{c}}(L) = \sum_{\mathbf{x} \in L} \rho_{\sigma,\mathbf{c}}(\mathbf{x})$. We denote by $D_{L,\sigma}(\mathbf{y})$ the distribution centered in $\mathbf{c} = \mathbf{0}^m$ and exploit the fact that samples from $D_{L,\sigma}$ are short w.h.p.

Lemma 1 ([6, Le. 1.5]). *For any lattice $L \subseteq \mathbb{R}^m$ and positive real number $\sigma > 0$, we have $\Pr_{\mathbf{b} \hookleftarrow D_{L,\sigma}}[\|\mathbf{b}\| \leq \sqrt{m}\sigma] \geq 1 - 2^{-\Omega(m)}$.*

It is well-known that Gaussian distributions with lattice support can be efficiently sampled from a sufficiently short basis of the lattice.

Lemma 2 ([20, Le. 2.3]). *There exists a PPT algorithm* GPVSample *that takes as inputs a basis $\mathbf{B}$ of a lattice $L \subseteq \mathbb{Z}^n$ and a rational $\sigma \geq \|\widetilde{\mathbf{B}}\| \cdot \Omega(\sqrt{\log n})$, and outputs vectors $\mathbf{b} \in L$ with distribution $D_{L,\sigma}$.*

We rely on the trapdoor generation algorithm of Alwen and Peikert [4].

Lemma 3 ([4, Th. 3.2]). *There exists a PPT algorithm* TrapGen *that takes as inputs 1^n, 1^m and an integer $q \geq 2$ with $m \geq \Omega(n \log q)$, and outputs a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and a basis $\mathbf{T_A}$ of $\Lambda_q^\perp(\mathbf{A})$ such that $\mathbf{A}$ is within statistical distance $2^{-\Omega(n)}$ to $U(\mathbb{Z}_q^{n \times m})$, and $\|\widetilde{\mathbf{T_A}}\| \leq \mathcal{O}(\sqrt{n \log q})$.*

We utilize the basis delegation algorithm [32] that inputs a trapdoor for $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and produces a trapdoor for any $\mathbf{B} \in \mathbb{Z}_q^{n \times m'}$ containing $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ as a submatrix.

Lemma 4 ([32, Le. 3.2]). *There exists a PPT algorithm that inputs a matrix $\mathbf{B} \in \mathbb{Z}_q^{n \times m'}$ whose first m columns span $\mathbb{Z}_q^n$, and a basis $\mathbf{T_A}$ of $\Lambda_q^\perp(\mathbf{A})$ where $\mathbf{A}$ is an $n \times m$ submatrix of $\mathbf{B}$, and outputs a basis $\mathbf{T_B}$ of $\Lambda_q^\perp(\mathbf{B})$ with $\|\widetilde{\mathbf{T_B}}\| \leq \|\widetilde{\mathbf{T_A}}\|$.*

Our security proofs use a technique introduced by Agrawal *et al.* [1].

Lemma 5 ([1, Th. 19]). *There exists a PPT algorithm that inputs matrices* $\mathbf{A}, \mathbf{C} \in \mathbb{Z}_q^{n \times m}$*, a small-norm matrix* $\mathbf{R} \in \mathbb{Z}^{m \times m}$*, a short basis* $\mathbf{T_C} \in \mathbb{Z}^{m \times m}$ *of* $\Lambda_q^{\perp}(\mathbf{C})$*, a vector* $\mathbf{u} \in \mathbb{Z}_q^n$ *and a rational* σ *such that* $\sigma \geq \|\widetilde{\mathbf{T_C}}\| \cdot \Omega(\sqrt{\log n})$*, and outputs vectors* $\mathbf{b} \in \mathbb{Z}^{2m}$ *such that* $\left[\mathbf{A} \;\middle|\; \mathbf{A} \cdot \mathbf{R} + \mathbf{C}\right] \cdot \mathbf{b} = \mathbf{u} \bmod q$ *and with distribution statistically close to* $D_{L,\sigma}$ *where* L *denotes the shifted lattice* $\{\mathbf{x} \in \mathbb{Z}^{2m} : \left[\mathbf{A} \;\middle|\; \mathbf{A} \cdot \mathbf{R} + \mathbf{C}\right] \cdot \mathbf{x} = \mathbf{u} \bmod q\}$.

2.2 Hardness Assumptions

Definition 2. *Let* $m, n, q \in \mathbb{N}$ *with* $m > n$ *and* $\beta > 0$*. The Short Integer Solution problem* $\mathsf{SIS}_{m,q,\beta}$ *is, given* $\mathbf{A} \hookleftarrow U(\mathbb{Z}_q^{n \times m})$*, find* $\mathbf{x} \in \Lambda^{\perp}(\mathbf{A})$ *with* $0 < \|\mathbf{x}\| \leq \beta$*.*

Definition 3. *Let* q, α *be functions of a parameter* n*. For a secret* $\mathbf{s} \in \mathbb{Z}_q^n$*, the distribution* $A_{q,\alpha,\mathbf{s}}$ *over* $\mathbb{Z}_q^n \times \mathbb{Z}_q$ *is obtained by sampling* $\mathbf{a} \hookleftarrow U(\mathbb{Z}_q^n)$ *and a noise* $e \hookleftarrow D_{\mathbb{Z},\alpha q}$*, and returning* $(\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e)$*. The Learning-With-Errors problem* $\mathsf{LWE}_{q,\alpha}$ *is, for* $\mathbf{s} \hookleftarrow U(\mathbb{Z}_q^n)$*, to distinguish between arbitrarily many independent samples from* $U(\mathbb{Z}_q^n \times \mathbb{Z}_q)$ *and the same number of samples from* $A_{q,\alpha,\mathbf{s}}$*.*

If $q \geq \sqrt{n}\beta$ and $m, \beta \leq \mathsf{poly}(n)$, then standard worst-case lattice problems with approximation factors $\gamma = \widetilde{\mathcal{O}}(\beta\sqrt{n})$ reduce to $\mathsf{SIS}_{m,q,\beta}$ (see, e.g., [54, Se. 9]). Similarly, if $\alpha q = \Omega(\sqrt{n})$, standard worst-case lattice problems with approximation factors $\gamma = \mathcal{O}(\alpha/n)$ reduce [88,20] to $\mathsf{LWE}_{q,\alpha}$. In the design of deterministic primitives like PRFs, the following variant of LWE comes in handy.

Definition 4 ([8]). *Let* q, p, m *be functions of a security parameter* n *such that* $q > p \geq 2$ *and* $m > n$*. The Learning-With-Rounding (*LWR*) problem is to distinguish the distribution* $\{(\mathbf{A},\ \lfloor \mathbf{A}^T \cdot \mathbf{s} \rfloor_p) \mid \mathbf{A} \hookleftarrow U(\mathbb{Z}_q^{n \times m}),\ \mathbf{s} \hookleftarrow U(\mathbb{Z}_q^n)\}$ *from the distribution* $\{(\mathbf{A},\ \mathbf{y}) \mid \mathbf{A} \hookleftarrow U(\mathbb{Z}_q^{n \times m}),\ \mathbf{y} \hookleftarrow U(\mathbb{Z}_p^m)\}$*.*

Banerjee *et al.* [8] proved that LWR is as hard as LWE when q and the modulus-to-error ratio are super-polynomial. Alwen *et al.* [3] showed that, when the number m of samples is fixed in advance, LWR retains its hardness for polynomial moduli. Bogdanov *et al.* [15] generalized the result of [3] to get rid of restrictions on the modulus q. For such parameters, their result implies the security of the LWR-based PRG, which stretches the seed $\mathbf{s} \in \mathbb{Z}_q^n$ into $\lfloor \mathbf{A}^T \cdot \mathbf{s} \rfloor_p \in \mathbb{Z}_p^m$.

2.3 Syntactic Definitions for Off-line Compact E-Cash

An off-line e-cash system involves a bank $\mathcal{B}$, many users $\mathcal{U}$ and merchants $\mathcal{M}$. In the syntax defined by Camenisch, Hohenberger and Lysyanskaya [23], all these parties interact together via the following algorithms and protocols:

$\mathsf{ParGen}(1^\lambda)$: inputs a security parameter 1^λ and outputs public parameters par.

In the following, we assume that par are available to all parties although we sometimes omit them from the inputs of certain algorithms.

BKeygen(par)**:** generates a bank's key pair $(SK_{\mathcal{B}}, PK_{\mathcal{B}})$ which allows $\mathcal{B}$ to issue wallets of value $2^L \in \mathsf{poly}(\lambda)$ (we assume that L is part of par).

UKeygen(par)**:** generates a user key pair $(SK_{\mathcal{U}}, PK_{\mathcal{U}})$.

Withdraw$\big(\mathcal{U}(PK_{\mathcal{B}}, SK_{\mathcal{U}}), \mathcal{B}(PK_{\mathcal{U}}, SK_{\mathcal{B}})\big)$**:** is an interactive protocol between a user $\mathcal{U}$ and the bank $\mathcal{B}$. The user $\mathcal{U}$ obtains either a wallet $\mathcal{W}$ of 2^L coins or an error message $\perp$. The bank outputs some state information $\mathsf{T}_{\mathcal{W}}$ which allows identifying $\mathcal{U}$, should he overspend.

Spend$\big(\mathcal{U}(\mathcal{W}, PK_{\mathcal{B}}, PK_{\mathcal{M}}, \mathtt{info}), \mathcal{M}(SK_{\mathcal{M}}, PK_{\mathcal{B}}, 2^L)\big)$**:** is a protocol whereby the user $\mathcal{U}$, on input of public keys $PK_{\mathcal{M}}, PK_{\mathcal{B}}$ and some transaction-specific meta data $\mathtt{info}$, spends a coin from his wallet $\mathcal{W}$ to merchant $\mathcal{M}$. The merchant obtains a coin *coin* comprised of a serial number and a proof of validity. $\mathcal{U}$'s output is an updated wallet $\mathcal{W}'$.

VerifyCoin$(\mathsf{par}, PK_{\mathcal{M}}, PK_{\mathcal{B}}, coin)$**:** is a non-interactive coin verification algorithm. On input of a purported coin and the public keys $PK_{\mathcal{M}}$, $PK_{\mathcal{B}}$ of the bank and the merchant, it outputs 0 or 1.

Deposit$\big(\mathcal{M}(SK_{\mathcal{M}}, coin, PK_{\mathcal{B}}), \mathcal{B}(PK_{\mathcal{M}}, SK_{\mathcal{B}}, \mathsf{state}_{\mathcal{B}})\big)$**:** is a protocol allowing the merchant $\mathcal{M}$ to deposit a received coin *coin* into its account at the bank $\mathcal{B}$. $\mathcal{M}$ outputs $\perp$ if the protocol fails and nothing if it succeeds. The bank $\mathcal{B}$ outputs "accept" and updates its state $\mathsf{state}_{\mathcal{B}}$ by adding an entry $(PK_{\mathcal{M}}, coin)$ if $\mathsf{VerifyCoin}(\mathsf{par}, PK_{\mathcal{M}}, PK_{\mathcal{B}}, coin) = 1$ and no double-spending is detected. Otherwise, if $\mathsf{VerifyCoin}(\mathsf{par}, PK_{\mathcal{M}}, PK_{\mathcal{B}}, coin) = 1$ and $\mathsf{state}_{\mathcal{B}}$ already contains a coin with the same serial number, it outputs "user". If $\mathsf{VerifyCoin}(\mathsf{par}, PK_{\mathcal{M}}, PK_{\mathcal{B}}, coin) = 0$ or $\mathsf{state}_{\mathcal{B}}$ already contains an entry $(PK_{\mathcal{M}}, coin)$, it outputs "$\mathsf{merchant}$".

Identify$\big(\mathsf{par}, PK_{\mathcal{B}}, coin_1, coin_2\big)$**:** is an algorithm that allows the bank $\mathcal{B}$ to identify a double-spender on input of two coins $coin_1$, $coin_2$ with identical serial numbers. The bank outputs the double-spender's public key $PK_{\mathcal{U}}$ and a proof Π_G that $\mathcal{U}$ indeed overspent.

Like [23], we assume that wallets $\mathcal{W}$ contain a counter J, initialized to 0, which indicates the number of previously spent coins. We also assume that each coin contains a serial number S, a proof of validity π as well as some information on the merchant's public key $PK_{\mathcal{M}}$ and some meta-data $\mathtt{info}$.

Following [23], we say that an off-line e-cash system is *compact* if the bitlength of the wallet $\mathcal{W}$ and the communication/computational complexities of all protocols is at most logarithmic in the value of the wallet (i.e., linear in L).

We use the security definitions of [23], which formalize security requirements called *anonymity*, *balance*, *double-spender identification* and *exculpability*.

Informally, the *balance* property considers colluding users interacting with a honest bank and attempting to spend more coins than they withdraw. This property is broken if the adversary manages to spend a coin of which the serial number does not match the serial number of any legally withdrawn coin. *Double-spender identification* complements the balance property by requiring that a malicious user be unable to output two coins with the same serial number without being caught by the $\mathsf{Identify}$ algorithm. *Anonymity* mandates that, when a merchant returns a received coin to the bank, even if they collude, they cannot

infer anything as to when and by whom the coin was withdrawn. The *exculpability* property captures that honest users cannot be falsely accused of being double-spenders: the adversary controls the bank and wins if it outputs two coins with the same serial number and such that $\mathsf{Identify}$ points to a well-behaved user. The formal definitions of these properties are recalled in Appendix A.

3 Warm-up: Permutations, Decompositions, Extensions

This section presents various notations and techniques that appeared (in slightly different forms) in earlier works on Stern-like protocols [73,69,74,72,71], and that will be used extensively throughout this work.

PERMUTATIONS. For any positive integer $\mathfrak{m}$, we define the following sets.

- $\mathcal{S}_{\mathfrak{m}}$: the set of all permutations of $\mathfrak{m}$ elements.
- $\mathsf{B}^2_{\mathfrak{m}}$: the set of binary vectors in $\{0,1\}^{2\mathfrak{m}}$ with Hamming weight $\mathfrak{m}$. Note that for any $\mathbf{v} \in \mathbb{Z}^{2\mathfrak{m}}$ and $\pi \in \mathcal{S}_{2m}$, we have:

$$\mathbf{v} \in \mathsf{B}^2_{\mathfrak{m}} \iff \pi(\mathbf{v}) \in \mathsf{B}^2_{\mathfrak{m}}. \tag{3}$$

- $\mathsf{B}^3_{\mathfrak{m}}$: the set of vectors in $\{-1,0,1\}^{3\mathfrak{m}}$ that have exactly $\mathfrak{m}$ coordinates equal to j, for every $j \in \{-1,0,1\}$. Note that for any $\mathbf{w} \in \mathbb{Z}^{3\mathfrak{m}}$ and $\phi \in \mathcal{S}_{3m}$:

$$\mathbf{w} \in \mathsf{B}^3_{\mathfrak{m}} \iff \phi(\mathbf{w}) \in \mathsf{B}^3_{\mathfrak{m}}. \tag{4}$$

For bit $c \in \{0,1\}$ and integer vector $\mathbf{v}$ of any dimension $\mathfrak{m}$, we denote by $\mathsf{Expand}(c,\mathbf{v})$ the vector $\begin{pmatrix} \overline{c}\cdot\mathbf{v} \\ c\cdot\mathbf{v} \end{pmatrix} \in \mathbb{Z}^{2\mathfrak{m}}$, where $\overline{c}$ denotes the bit $1-c$.

For any positive integer $\mathfrak{m}$, bit $b \in \{0,1\}$, and permutation $\pi \in \mathcal{S}_{\mathfrak{m}}$, we denote by $T_{b,\pi}$ the permutation that transforms the vector $\mathbf{v} = \begin{pmatrix} \mathbf{v}_0 \\ \mathbf{v}_1 \end{pmatrix} \in \mathbb{Z}^{2\mathfrak{m}}$, where $\mathbf{v}_0, \mathbf{v}_1 \in \mathbb{Z}^{\mathfrak{m}}$, into the vector $T_{b,\pi}(\mathbf{v}) = \begin{pmatrix} \pi(\mathbf{v}_b) \\ \pi(\mathbf{v}_{\overline{b}}) \end{pmatrix}$. Namely, $T_{b,\pi}$ first rearranges the 2 blocks of $\mathbf{v}$ according to b (it keeps the arrangement of blocks if $b=0$ and swaps them if $b=1$), then it permutes each block according to π.

Observe that the following equivalence holds for all $\mathfrak{m} \in \mathbb{Z}_+$, $b,c \in \{0,1\}$, $\pi \in \mathcal{S}_{\mathfrak{m}}$, $\mathbf{v} \in \mathbb{Z}^{\mathfrak{m}}$:

$$\mathbf{z} = \mathsf{expand}(c,\mathbf{v}) \iff T_{b,\pi}(\mathbf{z}) = \mathsf{expand}(c \oplus b, \pi(\mathbf{v})), \tag{5}$$

where $\oplus$ denotes the addition operation modulo 2.

DECOMPOSITIONS. For any $B \in \mathbb{Z}_+$, define $\delta_B := \lfloor \log_2 B \rfloor + 1 = \lceil \log_2(B+1) \rceil$ and the sequence $B_1, \ldots, B_{\delta_B}$, where $B_j = \lfloor \frac{B+2^{j-1}}{2^j} \rfloor$, for each $j \in [1, \delta_B]$. As observed in [75,73], it satisfies $\sum_{j=1}^{\delta_B} B_j = B$ and any integer $v \in [0,B]$ can be decomposed to $\mathsf{idec}_B(v) = (v^{(1)}, \ldots, v^{(\delta_B)})^\top \in \{0,1\}^{\delta_B}$ such that $\sum_{j=1}^{\delta_B} B_j \cdot v^{(j)} = v$. We describe this decomposition procedure in a deterministic manner as follows:

1. $v' := v$
2. For $j = 1$ to δ_B do:
   (i) If $v' \geq B_j$ then $v^{(j)} := 1$, else $v^{(j)} := 0$;
   (ii) $v' := v' - B_j \cdot v^{(j)}$.
3. Output $\mathsf{idec}_B(v) = (v^{(1)}, \ldots, v^{(\delta_B)})^\top$.

Next, for any positive integers $\mathfrak{m}, B$, we define the matrix:

$$\mathbf{H}_{\mathfrak{m},B} := \begin{bmatrix} B_1 \ldots B_{\delta_B} & & & \\ & B_1 \ldots B_{\delta_B} & & \\ & & \ddots & \\ & & & B_1 \ldots B_{\delta_B} \end{bmatrix} \in \mathbb{Z}^{\mathfrak{m} \times \mathfrak{m}\delta_B}, \tag{6}$$

and the following injective functions:

(i) $\mathsf{vdec}_{\mathfrak{m},B} : [0,B]^{\mathfrak{m}} \to \{0,1\}^{\mathfrak{m}\delta_B}$ that maps the vector $\mathbf{v} = (v_1, \ldots, v_{\mathfrak{m}})$ to $\big(\mathsf{idec}_B(v_1) \| \ldots \| \mathsf{idec}_B(v_{\mathfrak{m}})\big)$. Note that $\mathbf{H}_{\mathfrak{m},B} \cdot \mathsf{vdec}_{\mathfrak{m},B}(\mathbf{v}) = \mathbf{v}$.

(ii) $\mathsf{vdec}'_{\mathfrak{m},B} : [-B,B]^{\mathfrak{m}} \to \{-1,0,1\}^{\mathfrak{m}\delta_B}$ that decomposes $\mathbf{w} = (w_1, \ldots, w_{\mathfrak{m}})$ into the vector $\big(\sigma(w_1) \cdot \mathsf{idec}_B(|w_1|) \| \ldots \| \sigma(w_{\mathfrak{m}}) \cdot \mathsf{idec}_B(|w_{\mathfrak{m}}|)\big)$ such that, for each $i \in [\mathfrak{m}]$, we have: $\sigma(w_i) = 0$ if $w_i = 0$; $\sigma(w_i) = -1$ if $w_i < 0$; $\sigma(w_i) = 1$ if $w_i > 0$. Note that $\mathbf{H}_{\mathfrak{m},B} \cdot \mathsf{vdec}'_{\mathfrak{m},B}(\mathbf{w}) = \mathbf{w}$.

Extensions. We define following extensions of matrices and vectors.

– For any $\mathfrak{m}, B \in \mathbb{Z}_+$, define $\widehat{\mathbf{H}}_{\mathfrak{m},B} \in \mathbb{Z}^{\mathfrak{m} \times 2\mathfrak{m}\delta_B}$, $\breve{\mathbf{H}}_{\mathfrak{m},B} \in \mathbb{Z}^{\mathfrak{m} \times 3\mathfrak{m}\delta_B}$ as follows:

$$\widehat{\mathbf{H}}_{\mathfrak{m},B} := \big[\, \mathbf{H}_{\mathfrak{m},B} \,\big|\, \mathbf{0}^{\mathfrak{m} \times \mathfrak{m}\delta_B} \,\big]; \qquad \breve{\mathbf{H}}_{\mathfrak{m},B} := \big[\, \mathbf{H}_{\mathfrak{m},B} \,\big|\, \mathbf{0}^{\mathfrak{m} \times 2\mathfrak{m}\delta_B} \,\big].$$

– Given $\mathbf{v} \in \{0,1\}^{\mathfrak{m}}$, define $\mathsf{TwoExt}(\mathbf{v}) := (\mathbf{v} \| \mathbf{0}^{\mathfrak{m}-n_0} \| \mathbf{1}^{\mathfrak{m}-n_1}) \in \mathsf{B}^2_{\mathfrak{m}}$, where n_0, n_1 are the number of coordinates in $\mathbf{v}$ equal to 0 and 1, respectively.

– Given $\mathbf{v} \in [-1,0,1]^{\mathfrak{m}}$, define

$$\mathsf{ThreeExt}(\mathbf{v}) := (\mathbf{v} \| \mathbf{0}^{\mathfrak{m}-n_0} \| \mathbf{1}^{\mathfrak{m}-n_1} \| -\mathbf{1}^{\mathfrak{m}-n_{-1}}) \in \mathsf{B}^3_{\mathfrak{m}},$$

where n_0, n_1, n_{-1} are the number of coordinates in $\mathbf{v}$ equal to 0, 1, and -1, respectively.

Note that, if $\mathbf{x} \in [0,B]^m$ and $\mathbf{y} \in [-B,B]^m$, then we have:

$$\mathsf{TwoExt}\big(\mathsf{vdec}_{m,B}(\mathbf{x})\big) \in \mathsf{B}^2_{m\delta_B} \text{ and } \widehat{\mathbf{H}}_{m,B} \cdot \mathsf{TwoExt}\big(\mathsf{vdec}_{m,B}(\mathbf{x})\big) = \mathbf{x}, \tag{7}$$

$$\mathsf{ThreeExt}\big(\mathsf{vdec}'_{m,B}(\mathbf{y})\big) \in \mathsf{B}^3_{m\delta_B} \text{ and } \breve{\mathbf{H}}_{m,B} \cdot \mathsf{ThreeExt}\big(\mathsf{vdec}'_{m,B}(\mathbf{y})\big) = \mathbf{y}. \tag{8}$$

In the framework of Stern-like protocols [89,67,73,74,71], the above techniques are useful when it comes proving in zero-knowledge the possession of integer vectors satisfying several different constraints:

Case 1: $\mathbf{x} \in [0,B]^m$. We equivalently prove $\hat{\mathbf{x}} = \mathsf{TwoExt}\big(\mathsf{vdec}_{m,B}(\mathbf{x})\big) \in \mathsf{B}^2_{m\delta_B}$. To do this, pick $\pi \hookleftarrow U(\mathcal{S}_{2m\delta_B})$, and convince the verifier that $\pi(\hat{\mathbf{x}}) \in \mathsf{B}^2_{m\delta_B}$.

Case 2: $\mathbf{x} \in [-B, B]^m$. We equivalently prove $\breve{\mathbf{x}} = \mathsf{ThreeExt}\big(\mathsf{vdec}'_{m,B}(\mathbf{y})\big) \in \mathsf{B}^3_{m\delta_B}$. To do this, pick $\pi \hookleftarrow U(\mathcal{S}_{3m\delta_B})$, and convince the verifier that $\pi(\breve{\mathbf{x}}) \in \mathsf{B}^3_{m\delta_B}$.

Case 3: $\mathbf{x} = \mathsf{expand}(c, \mathbf{v})$, where $\mathbf{v}$ satisfies one of the above two constraints. To hide $\mathbf{v}$, we use the respective decomposition-extension-permutation technique. To hide the bit c, we pick a "one-time pad" $b \hookleftarrow U(\{0,1\})$ and exploit the equivalence observed in (5). Looking ahead, this technique will be used in Section 4.3 to hide the bits of the PRF input J and those of a signature component $\tau \in \{0,1\}^\ell$ in Section 5.

4 Zero-Knowledge Arguments for Lattice-Based PRFs

Here, we first give an abstraction of Stern's protocol [89]. With this abstraction in mind, we then present our techniques for achieving zero-knowledge arguments for the BLMR PRF [17].

In Appendix C.3, we adapt these techniques to the PRF generically implied by the GGM [56] paradigm. While slightly more complex to describe, the GGM-based construction allows for a better choice of parameters since, owing to the result of Bogdanov *et al.* [15], it allows instantiating the LWR-based PRG with polynomial-size moduli.

4.1 An Abstraction of Stern's Protocol

In [89], Stern proposed a zero-knowledge protocol for the Syndrome Decoding problem, in which the main idea is to use a random permutation over coordinates of a secret vector to prove that the latter satisfies a given constraint (e.g., having fixed Hamming weight). Later on, Stern's protocol was adapted to the lattice setting by Kawachi *et al.* [67] and refined by Ling *et al.* [73] to handle statements related to the SIS and LWE problems. Subsequently, the protocol was further developed to design several lattice-based systems [69,74,72]. Recently, Libert *et al.* [71] suggested an abstraction that addresses the setting where one has to prove knowledge of small secret vectors satisfying a number of modular linear equations with respect to one modulus. While their generalization subsumes many relations that naturally appear in privacy-preserving protocols involving lattices, it is not immediately applicable to the statements considered in this paper since we have to work with more than one modulus.

We thus put forward a new abstraction of Stern's protocol [89] that handles modular equations with respect to $N \geq 1$ moduli $q_1, \ldots, q_N$, where secret witnesses may simultaneously appear across multiple equations.

Let n_i and $d_i \geq n_i$ be positive integers, and let $d = d_1 + \cdots + d_N$. Suppose that VALID is a subset of $\{-1, 0, 1\}^d$ and $\mathcal{S}$ is a finite set such that every $\phi \in \mathcal{S}$ can be associated with a permutation Γ_ϕ of d elements satisfying the conditions

$$\begin{cases} \mathbf{w} \in \mathsf{VALID} \iff \Gamma_\phi(\mathbf{w}) \in \mathsf{VALID}; \\ \text{If } \mathbf{w} \in \mathsf{VALID} \text{ and } \phi \text{ is uniform in } \mathcal{S}, \text{ then } \Gamma_\phi(\mathbf{w}) \text{ is uniform in } \mathsf{VALID}. \end{cases} \tag{9}$$

In our abstract protocol, for public matrices $\{\mathbf{M}_i \in \mathbb{Z}_{q_i}^{n_i \times d_i}\}_{i\in[N]}$ and vectors $\mathbf{u}_i \in \mathbb{Z}_{q_i}^{n_i}$, the prover argues in zero-knowledge the possession of integer vectors $\{\mathbf{w}_i \in \{-1,0,1\}^{d_i}\}_{i\in[N]}$ such that:

$$\mathbf{w} = \big(\mathbf{w}_1 \| \ldots \| \mathbf{w}_N\big) \in \mathsf{VALID}, \tag{10}$$

$$\forall i \in [N] : \mathbf{M}_i \cdot \mathbf{w}_i = \mathbf{u}_i \bmod q_i. \tag{11}$$

Looking ahead, all the statements considered in Sections 4.3, 5 (as well as those in Appendix C.3) will be reduced to the above setting, wherein secret vectors $\mathbf{w}_1, \ldots, \mathbf{w}_N$ are mutually related, e.g., some entries of $\mathbf{w}_i$ also appear in $\mathbf{w}_j$.

1. **Commitment:** $\mathcal{P}$ samples $\phi \leftarrow U(\mathcal{S})$, $\mathbf{r}_1 \leftarrow U(\mathbb{Z}_{q_1}^{d_1}), \ldots, \mathbf{r}_N \leftarrow U(\mathbb{Z}_{q_N}^{d_N})$, and computes $\mathbf{r} = (\mathbf{r}_1 \| \ldots \| \mathbf{r}_N)$, $\mathbf{z} = \mathbf{w} \boxplus \mathbf{r}$.
   Then $\mathcal{P}$ samples randomness ρ_1, ρ_2, ρ_3 for COM, and sends $\mathrm{CMT} = \big(C_1, C_2, C_3\big)$ to $\mathcal{V}$, where $C_1 = \mathsf{COM}(\phi, \{\mathbf{M}_i \cdot \mathbf{r}_i \bmod q_i\}_{i\in[N]}; \rho_1)$, and
   $$C_2 = \mathsf{COM}(\Gamma_\phi(\mathbf{r}); \rho_2), \quad C_3 = \mathsf{COM}(\Gamma_\phi(\mathbf{z}); \rho_3).$$
2. **Challenge:** $\mathcal{V}$ sends a challenge $Ch \leftarrow U(\{1,2,3\})$ to $\mathcal{P}$.
3. **Response:** $\mathcal{P}$ sends RSP computed according to Ch, as follows:
   - $Ch = 1$: $\mathrm{RSP} = (\mathbf{t}, \mathbf{s}, \rho_2, \rho_3)$, where $\mathbf{t} = \Gamma_\phi(\mathbf{w})$ and $\mathbf{s} = \Gamma_\phi(\mathbf{r})$.
   - $Ch = 2$: $\mathrm{RSP} = (\pi, \mathbf{x}, \rho_1, \rho_3)$, where $\pi = \phi$ and $\mathbf{x} = \mathbf{z}$.
   - $Ch = 3$: $\mathrm{RSP} = (\psi, \mathbf{y}, \rho_1, \rho_2)$, where $\psi = \phi$ and $\mathbf{y} = \mathbf{r}$.

Verification: Receiving RSP, $\mathcal{V}$ proceeds as follows:

- $Ch = 1$: Check that $\mathbf{t} \in \mathsf{VALID}$, and $C_2 = \mathsf{COM}(\mathbf{s}; \rho_2)$, $C_3 = \mathsf{COM}(\mathbf{t} \boxplus \mathbf{s}; \rho_3)$.
- $Ch = 2$: Parse $\mathbf{x} = (\mathbf{x}_1 \| \ldots \| \mathbf{x}_N)$, where $\mathbf{x}_i \in \mathbb{Z}_{q_i}^{d_i}$ for all $i \in [N]$, and check that
  $$C_1 = \mathsf{COM}(\pi, \{\mathbf{M}_i \cdot \mathbf{x}_i - \mathbf{u}_i \bmod q_i\}_{i\in[N]}; \rho_1), \quad C_3 = \mathsf{COM}(\Gamma_\pi(\mathbf{x}); \rho_3).$$
- $Ch = 3$: Parse $\mathbf{y} = (\mathbf{y}_1 \| \ldots \| \mathbf{y}_N)$, where $\mathbf{y}_i \in \mathbb{Z}_{q_i}^{d_i}$ for all $i \in [N]$, and check that
  $$C_1 = \mathsf{COM}(\psi, \{\mathbf{M}_i \cdot \mathbf{y}_i \bmod q_i\}_{i\in[N]}; \rho_1), \quad C_2 = \mathsf{COM}(\Gamma_\psi(\mathbf{y}); \rho_2).$$

In each case, $\mathcal{V}$ outputs 1 if and only if all the conditions hold.

Fig. 1: Our abstract protocol.

The main ideas driving our protocol are as follows. To prove (10), the prover samples $\phi \leftarrow U(\mathcal{S})$ and provides evidence that $\Gamma_\phi(\mathbf{w}) \in \mathsf{VALID}$. The verifier should be convinced while learning nothing else, owing to the aforementioned properties of the sets VALID and $\mathcal{S}$. Meanwhile, to prove that equations (11) hold, the prover uses masking vectors $\{\mathbf{r}_i \leftarrow U(\mathbb{Z}_{q_i}^{d_i})\}_{i\in[N]}$ and demonstrates instead that $\mathbf{M}_i \cdot (\mathbf{w}_i + \mathbf{r}_i) = \mathbf{u}_i + \mathbf{M}_i \cdot \mathbf{r}_i \bmod q_i$.

The interaction between prover $\mathcal{P}$ and verifier $\mathcal{V}$ is described in Figure 1. The common input consists of $\{\mathbf{M}_i \in \mathbb{Z}_{q_i}^{n_i \times d_i}\}_{i\in[N]}$ and $\mathbf{u}_i \in \mathbb{Z}_{q_i}^{n_i}$, while $\mathcal{P}$'s secret input is $\mathbf{w} = \big(\mathbf{w}_1 \| \ldots \| \mathbf{w}_N\big)$. The protocol makes use of a statistically

hiding and computationally binding string commitment scheme COM such as the SIS-based commitment of [67]. For simplicity of presentation, for vectors $\mathbf{w} = \big(\mathbf{w}_1 \| \dots \| \mathbf{w}_N\big) \in \mathbb{Z}^d$ and $\mathbf{r} = \big(\mathbf{r}_1 \| \dots \| \mathbf{r}_N\big) \in \mathbb{Z}^d$, we denote by $\mathbf{w} \boxplus \mathbf{r}$ the operation that computes $\mathbf{z}_i = \mathbf{w}_i + \mathbf{r}_i \bmod q_i$ for all $i \in [N]$, and outputs d-dimensional integer vector $\mathbf{z} = \big(\mathbf{z}_1 \| \dots \| \mathbf{z}_N\big)$. We note that, for all $\phi \in \mathcal{S}$, if $\mathbf{t} = \Gamma_\phi(\mathbf{w})$ and $\mathbf{s} = \Gamma_\phi(\mathbf{r})$, then we have $\Gamma_\phi(\mathbf{w} \boxplus \mathbf{r}) = \mathbf{t} \boxplus \mathbf{s}$.

The properties of our protocol are summarized in the following theorem.

Theorem 1. *Suppose that* COM *is a statistically hiding and computationally binding string commitment. Then, the protocol of Figure 1 is a zero-knowledge argument of knowledge for the given statement, with perfect completeness, soundness error* $2/3$*, and communication cost* $\mathcal{O}\big(\sum_{i=1}^{N} d_i \cdot \log q_i\big)$*. In particular:*

- *There exists an efficient simulator that, on input* $\{\mathbf{M}_i, \mathbf{u}_i\}_{i \in [N]}$*, outputs an accepted transcript statistically close to that produced by the real prover.*
- *There exists an efficient knowledge extractor that, on input a commitment* CMT *as well as valid responses* $(\mathrm{RSP}_1, \mathrm{RSP}_2, \mathrm{RSP}_3)$ *to all three possible values of the challenge* Ch*, outputs a witness* $\mathbf{w}' = (\mathbf{w}'_1 \| \dots \| \mathbf{w}'_N) \in \mathsf{VALID}$ *such that* $\mathbf{M}_i \cdot \mathbf{w}'_i = \mathbf{u}_i \bmod q_i$*, for all* $i \in [N]$*.*

The proof of Theorem 1 employs standard simulation and extraction techniques of Stern-like protocols [67,73,72,71], and is deferred to Appendix C.1.

4.2 Transforming the LWR Relation

Let $q \geq p \geq 2$, $m \geq 1$, and let $\mathbb{Z}_q = [0, q-1]$ and $\mathbb{Z}_p = [0, p-1]$. Consider the LWR rounding function: $\lfloor \cdot \rfloor_p : \mathbb{Z}_q^m \to \mathbb{Z}_p^m \ : \ \mathbf{x} \mapsto \mathbf{y} = \lfloor (p/q) \cdot \mathbf{x} \rfloor \bmod p$.

On the road towards zero-knowledge arguments for LWR-based PRFs, we have to build a sub-protocol that allows proving knowledge of a secret vector $\mathbf{x} \in \mathbb{Z}_q^m$ satisfying, among other statements, the property of rounding to a given $\mathbf{y} \in \mathbb{Z}_p^m$: i.e., $\lfloor \mathbf{x} \rfloor_p = \mathbf{y}$. To our knowledge, such a sub-protocol is not available in the literature for the time being and must be designed from scratch.

Our crucial observation is that one knows $\mathbf{x} \in [0, q-1]^m$ such that $\lfloor \mathbf{x} \rfloor_p = \mathbf{y}$, if and only if one can compute $\mathbf{x}, \mathbf{z} \in [0, q-1]^m$ such that:

$$p \cdot \mathbf{x} = q \cdot \mathbf{y} + \mathbf{z} \bmod pq. \tag{12}$$

This observation allows us to transform the LWR relation into an equivalent form that can be handled using the Stern-like techniques provided in Section 3. Let $\widehat{\mathbf{x}} = \mathsf{TwoExt}\big(\mathsf{vdec}_{m,q-1}(\mathbf{x})\big)$ and $\widehat{\mathbf{z}} = \mathsf{TwoExt}\big(\mathsf{vdec}_{m,q-1}(\mathbf{z})\big)$. Then we have $\mathbf{x} = \widehat{\mathbf{H}}_{m,q-1} \cdot \widehat{\mathbf{x}}$ and $\mathbf{z} = \widehat{\mathbf{H}}_{m,q-1} \cdot \widehat{\mathbf{z}}$, so that equation (12) can be written as:

$$(p \cdot \widehat{\mathbf{H}}_{m,q-1}) \cdot \widehat{\mathbf{x}} - \widehat{\mathbf{H}}_{m,q-1} \cdot \widehat{\mathbf{z}} = q \cdot \mathbf{y} \bmod pq. \tag{13}$$

Note that one knows $\mathbf{x}, \mathbf{z} \in [0, q-1]^m$ satisfying (12) if and only if one can compute $\widehat{\mathbf{x}}, \widehat{\mathbf{z}} \in \mathsf{B}^2_{m\delta_{q-1}}$ satisfying (13). Furthermore, Stern's framework allows proving the latter in zero-knowledge using random permutations.

4.3 Argument of Correct Evaluation for the BLMR PRF

We now consider the problem of proving the correct evaluation of the BLMR pseudo-random function from [17]. Namely, we would like to prove that a given $\mathbf{y} = \left\lfloor \prod_{i=1}^{L} \mathbf{P}_{J[L+1-i]} \cdot \mathbf{k} \right\rfloor_p \in \mathbb{Z}_p^m$ is the correct evaluation for a committed seed $\mathbf{k} \in \mathbb{Z}_q^m$ and a secret input $J[1] \dots J[L] \in \{0,1\}^L$, where $\mathbf{P}_0, \mathbf{P}_1 \in \{0,1\}^{m \times m}$ are public binary matrices, while revealing neither $\mathbf{k}$ nor $J[1] \dots J[L]$. We assume public matrices $\mathbf{D}_0 \in \mathbb{Z}_{q_s}^{n \times m_0}$, $\mathbf{D}_1 \in \mathbb{Z}_{q_s}^{n \times \bar{m}}$, for some modulus q_s and integers m_0 and $\bar{m} = m\delta_{q-1}$, which are used to compute a KTX commitment [67] $\mathbf{c} = \mathbf{D}_0 \cdot \mathbf{r} + \mathbf{D}_1 \cdot \tilde{\mathbf{k}} \in \mathbb{Z}_{q_s}^n$ to the decomposition $\tilde{\mathbf{k}} = \mathsf{vdec}_{m,q-1}(\mathbf{k}) \in \{0,1\}^{\bar{m}}$ of the seed $\mathbf{k}$, where $\mathbf{r} \in [-\beta, \beta]^{m_0}$ is a discrete Gaussian vector (for some small integer β), and $\tilde{\mathbf{k}}$ satisfies $\mathbf{H}_{m,q-1} \cdot \tilde{\mathbf{k}} = \mathbf{k}$.

We first note that, in the evaluation process of $\mathbf{y} = \left\lfloor \prod_{i=1}^{L} \mathbf{P}_{J[L+1-i]} \cdot \mathbf{k} \right\rfloor_p$, one works with vectors $\{\mathbf{x}_i \in \mathbb{Z}_q^m\}_{i=0}^{L}$ such that $\mathbf{x}_0 = \mathbf{k}$, $\mathbf{x}_i = \mathbf{P}_{J[i]} \cdot \mathbf{x}_{i-1} \bmod q$ for each $i \in \{1, \dots, L\}$, and $\mathbf{y} = \lfloor \mathbf{x}_L \rfloor_p$. We further observe that the iterative equation $\mathbf{x}_i = \mathbf{P}_{J[i]} \cdot \mathbf{x}_{i-1} \bmod q$ is equivalent to:

$$\mathbf{x}_i = \mathbf{P}_0 \cdot (\overline{J[i]} \cdot \mathbf{x}_{i-1}) + \mathbf{P}_1 \cdot (J[i] \cdot \mathbf{x}_{i-1}) = \left[\mathbf{P}_0 \,|\, \mathbf{P}_1 \right] \cdot \begin{pmatrix} \overline{J[i]} \cdot \mathbf{x}_{i-1} \\ J[i] \cdot \mathbf{x}_{i-1} \end{pmatrix} \bmod q. \quad (14)$$

Intuitively, this observation allows us to move the secret bit $J[i]$ from the "matrix side" to the "vector side" in order to make the equation compatible with Stern-like protocols. Next, for each $i \in \{0, \dots, L\}$, we form the vector $\widehat{\mathbf{x}}_i = \mathsf{TwoExt}\big(\mathsf{vdec}_{m,q-1}(\mathbf{x}_i)\big) \in \mathsf{B}_{\bar{m}}^2$. Equation (14) can then be written as:

$$\widehat{\mathbf{H}}_{m,q-1} \cdot \widehat{\mathbf{x}}_i = \left[\mathbf{P}_0 \cdot \widehat{\mathbf{H}}_{m,q-1} \,|\, \mathbf{P}_1 \cdot \widehat{\mathbf{H}}_{m,q-1} \right] \cdot \mathsf{expand}(J[i], \widehat{\mathbf{x}}_{i-1}) \bmod q.$$

Let $\mathbf{P} = \left[\mathbf{P}_0 \cdot \widehat{\mathbf{H}}_{m,q-1} \,|\, \mathbf{P}_1 \cdot \widehat{\mathbf{H}}_{m,q-1} \right] \in \mathbb{Z}_q^{m \times 4\bar{m}}$, and $\{\mathbf{s}_{i-1} = \mathsf{expand}(J[i], \widehat{\mathbf{x}}_{i-1})\}_{i=1}^{L}$, we have the L equations:

$$\begin{cases} \mathbf{P} \cdot \mathbf{s}_0 - \widehat{\mathbf{H}}_{m,q-1} \cdot \widehat{\mathbf{x}}_1 = \mathbf{0} \bmod q, \\ \qquad\qquad \vdots \\ \mathbf{P} \cdot \mathbf{s}_{L-1} - \widehat{\mathbf{H}}_{m,q-1} \cdot \widehat{\mathbf{x}}_L = \mathbf{0} \bmod q, \end{cases} \quad (15)$$

Regarding the rounding step $\lfloor \mathbf{x}_L \rfloor_p = \mathbf{y} \in \mathbb{Z}_p^m$, using the transformations of Section 4.2, we obtain the following equation for $\widehat{\mathbf{z}} \in \mathsf{B}_{\bar{m}}^2$:

$$(p \cdot \widehat{\mathbf{H}}_{m,q-1}) \cdot \widehat{\mathbf{x}}_L - \widehat{\mathbf{H}}_{m,q-1} \cdot \widehat{\mathbf{z}} = q \cdot \mathbf{y} \bmod pq, \quad (16)$$

As for the commitment relation, we have the equation $\mathbf{c} = \mathbf{D}_0 \cdot \mathbf{r} + \mathbf{D}_1 \cdot \tilde{\mathbf{k}} \bmod q_s$, where $\mathbf{H}_{m,q-1} \cdot \tilde{\mathbf{k}} = \mathbf{x}_0$. We let $\breve{\mathbf{r}} = \mathsf{ThreeExt}\big(\mathsf{vdec}'_{m_0,\beta}(\mathbf{r})\big) \in \mathsf{B}_{m_0\delta_\beta}^3$ and remark that $\mathsf{TwoExt}\big(\tilde{\mathbf{k}}\big) = \widehat{\mathbf{x}}_0$. Then, we have:

$$[\mathbf{D}_0 \cdot \breve{\mathbf{H}}_{m_0,\beta}] \cdot \breve{\mathbf{r}} + [\mathbf{D}_1 \mid \mathbf{0}^{n \times \bar{m}}] \cdot \widehat{\mathbf{x}}_0 = \mathbf{c} \bmod q_s. \quad (17)$$

Our goal is now reduced to proving the possession of $J[1]\dots J[L] \in \{0,1\}^L$, $\widehat{\mathbf{x}}_0, \dots, \widehat{\mathbf{x}}_L, \widehat{\mathbf{z}} \in \mathsf{B}^2_{\bar{m}}$ and $\check{\mathbf{r}} \in \mathsf{B}^3_{m_0\delta_\beta}$, satisfying equations (17), (15) and (16). Next, we let $q_1 = q_s$, $q_2 = q$, $q_3 = pq$, and proceed as follows.

Regarding equation (17), letting $\mathbf{M}_1 = \left[\mathbf{D}_0 \cdot \check{\mathbf{H}}_{m_0,\beta} \,\middle|\, \mathbf{D}_1 \mid \mathbf{0}^{n\times\bar{m}}\right]$, $\mathbf{u}_1 = \mathbf{c}$ and $\mathbf{w}_1 = \left(\check{\mathbf{r}} \| \widehat{\mathbf{x}}_0\right)$, the equation becomes:

$$\mathbf{M}_1 \cdot \mathbf{w}_1 = \mathbf{u}_1 \bmod q_1.$$

Next, we unify the L equations in (15). To this end, we define

$$\mathbf{M}_2 = \begin{bmatrix} \mathbf{P} & -\widehat{\mathbf{H}}_{m,q-1} & & \\ & \ddots & \ddots & \\ & & \mathbf{P} & -\widehat{\mathbf{H}}_{m,q-1} \end{bmatrix}, \quad \mathbf{u}_2 = \mathbf{0},$$

and $\mathbf{w}_2 = \left(\mathbf{s}_0 \| \widehat{\mathbf{x}}_1 \| \cdots \| \mathbf{s}_{L-1} \| \widehat{\mathbf{x}}_L\right)$. Then, (15) can be equivalently written as:

$$\mathbf{M}_2 \cdot \mathbf{w}_2 = \mathbf{u}_2 \bmod q_2.$$

As for equation (16), let $\mathbf{M}_3 = \left[(p \cdot \widehat{\mathbf{H}}_{m,q-1}) | -\widehat{\mathbf{H}}_{m,q-1}\right]$, $\mathbf{u}_3 = q \cdot \mathbf{y}$ and $\mathbf{w}_3 = \left(\widehat{\mathbf{x}}_L \| \widehat{\mathbf{z}}\right)$. Then, we obtain:

$$\mathbf{M}_3 \cdot \mathbf{w}_3 = \mathbf{u}_3 \bmod q_3.$$

Now, we let $d_1 = 3m_0\delta_\beta + 2\bar{m}$, $d_2 = 6L\bar{m}$ and $d_3 = 4\bar{m}$ be the dimensions of $\mathbf{w}_1, \mathbf{w}_2$ and $\mathbf{w}_3$, respectively, and $d = d_1 + d_2 + d_3$. We form the vector $\mathbf{w} = (\mathbf{w}_1 \| \mathbf{w}_2 \| \mathbf{w}_3) \in \{-1,0,1\}^d$, which has the form:

$$\mathbf{w} = \left(\check{\mathbf{r}} \,\|\, \widehat{\mathbf{x}}_0 \,\|\, \mathbf{s}_0 \,\|\, \widehat{\mathbf{x}}_1 \,\|\, \cdots \,\|\, \mathbf{s}_{L-1} \,\|\, \widehat{\mathbf{x}}_L \,\|\, \widehat{\mathbf{x}}_L \,\|\, \widehat{\mathbf{z}}\right). \tag{18}$$

At this point, we have come close to reducing our statement to an instance of the one considered in Section 4.1. Next, let us specify the set VALID containing $\mathbf{w}$, the set $\mathcal{S}$ and the associated permutation Γ_ϕ satisfying conditions in 9.

Let VALID be the set of all vectors in $\{-1,0,1\}^d$ having the form (18), where

- $\check{\mathbf{r}} \in \mathsf{B}^3_{m_0\delta_\beta}$, and $\widehat{\mathbf{x}}_0, \dots, \widehat{\mathbf{x}}_L, \widehat{\mathbf{z}} \in \mathsf{B}^2_{\bar{m}}$.
- $\{\mathbf{s}_{i-1} = \mathsf{expand}(J[i], \widehat{\mathbf{x}}_{i-1})\}_{i=1}^L$, for some $J[1]\dots J[L] \in \{0,1\}^L$.

It can be seen that our vector $\mathbf{w}$ belongs to this tailored set VALID.

Now, we define $\mathcal{S} := \mathcal{S}_{3m_0\delta_\beta} \times (\mathcal{S}_{2\bar{m}})^{L+2} \times \{0,1\}^L$. Then, for any set element $\phi = (\phi_r, \phi_0, \phi_1, \dots, \phi_L, \phi_z, b_1 \dots b_L) \in \mathcal{S}$, let Γ_ϕ be the permutation that transforms vector $\mathbf{w} \in \mathbb{Z}^d$ of the form (18) to vector $\Gamma_\phi(\mathbf{w})$ of the form:

$$\Gamma_\phi(\mathbf{w}) = \big(\phi_r(\check{\mathbf{r}}) \,\|\, \phi_0(\widehat{\mathbf{x}}_0) \,\|\, T_{b_1,\phi_0}(\mathbf{s}_0) \,\|\, \phi_1(\widehat{\mathbf{x}}_1) \,\|\, \cdots \,\|\, T_{b_L,\phi_{L-1}}(\mathbf{s}_{L-1}) \,\|\, \phi_L(\widehat{\mathbf{x}}_L) \\ \|\, \phi_L(\widehat{\mathbf{x}}_L) \,\|\, \phi_z(\widehat{\mathbf{z}})\big).$$

Thanks to the equivalences (3), (4), (5) from Section 3, we have $\mathbf{w} \in \mathsf{VALID}$ if and only if $\Gamma_\phi(\mathbf{w}) \in \mathsf{VALID}$. Furthermore, if $\phi \leftarrow U(\mathcal{S})$, then $\Gamma_\phi(\mathbf{w})$ is uniform in VALID. Said otherwise, the conditions in (9) are satisfied.

Given the above transformations and specifications, we can now run the abstract protocol of Figure 1 to prove knowledge of $\mathbf{w} = (\mathbf{w}_1 \| \mathbf{w}_2 \| \mathbf{w}_3) \in \mathsf{VALID}$ satisfying $\{\mathbf{M}_i \cdot \mathbf{w}_i = \mathbf{u}_i \bmod q_i\}_{i=1,2,3}$, where public matrices/vectors $\{\mathbf{M}_i, \mathbf{u}_i\}_{i=1,2,3}$ are as constructed above. As a result, we obtain a statistical zero-knowledge argument of knowledge for the statement described at the beginning of this section. For simulation, we run the simulator of Theorem 1 with public input $\{\mathbf{M}_i, \mathbf{u}_i\}_{i=1,2,3}$. For extraction (see also Appendix C.2), we first run the knowledge extractor of Theorem 1, to obtain $\mathbf{w}' = (\mathbf{w}'_1 \| \mathbf{w}'_2 \| \mathbf{w}'_3) \in \mathsf{VALID}$ such that $\{\mathbf{M}_i \cdot \mathbf{w}'_i = \mathbf{u}_i \bmod q_i\}_{i=1,2,3}$ and then reverse the witness transformations to get $\mathbf{k}' \in \mathbb{Z}_q^m$, $J'[1] \ldots J'[L] \in \{0,1\}^L$ and $\mathbf{r}' \in [-\beta, \beta]^{m_0}$, $\tilde{\mathbf{k}}' \in \{0,1\}^{\bar{m}}$ satisfying:

$$\mathbf{y} = \Big\lfloor \prod_{i=1}^{L} \mathbf{P}_{J'[L+1-i]} \cdot \mathbf{k}' \Big\rceil_p, \quad \mathbf{c} = \mathbf{D}_0 \cdot \mathbf{r}' + \mathbf{D}_1 \cdot \mathbf{k}' \bmod q_s, \quad \mathbf{H}_{m,q-1} \cdot \tilde{\mathbf{k}}' = \mathbf{k}'.$$

The protocol has communication cost $\mathcal{O}(d_1 \cdot \log q_1 + d_2 \cdot \log q_2 + d_3 \cdot \log q_3)$. For a typical setting of parameters (as in Section 5), this cost is of order $\widetilde{\mathcal{O}}(\lambda \cdot L)$, where λ is the security parameter (and L is the input length of the PRF).

5 Description of Our Compact E-cash System

This section describes our e-cash system. We do not present a general construction from lower level primitives because such a construction is already implicit in the work of Camenisch *et al.* [23] of which we follow the blueprint. To avoid repeating it, we directly show how to apply the same design principle in lattices using carefully chosen primitives that interact with our zero-knowledge proofs.

Like [23], our scheme combines signatures with efficient protocols and pseudo-random functions which support proofs of correct evaluation. Our e-cash system builds on the signature scheme with efficient protocols of Libert *et al.* [71]. The latter is a variant of the SIS-based signatures described by Boyen [18] and Böhl *et al.* [16]. We actually use a simplified version of their scheme which is recalled in Appendix B and dispenses with the need to encode messages in a special way.

As in [23], our withdrawal protocol involves a step where the bank and the user jointly compute a seed $\mathbf{k} = \mathbf{k}_0 + \mathbf{k}_1 \in \mathbb{Z}_q^m$, which will be uniform over $\mathbb{Z}_q^m$ as long as one of the two parties is honest. The reason is that the identification of double-spenders can only be guaranteed if two distinct small-domain PRFs with independent random keys never collide, except with negligible probability. To jointly generate the PRF seed $\mathbf{k}$, the protocol of [23] relies on the homomorphic property of the commitment scheme used in their oblivious signing protocol. In our setting, one difficulty is that the underlying KTX commitment [67] has message space $\{0,1\}^{m\lceil \log q \rceil}$ and is not homomorphic over $\mathbb{Z}_q^m$. To solve this problem, our withdrawal protocol lets the user obtain the bank's signature on a message containing the binary decompositions of $\mathbf{k}_0$ and $\mathbf{k}_1$, so that the sum $\mathbf{k} = \mathbf{k}_0 + \mathbf{k}_1$ is only reconstructed during the spending phase.

At the withdrawal step, the user also chooses a second PRF seed $\mathbf{t} \in \mathbb{Z}_q^m$ of its own. The withdrawal protocol ends with the user obtaining a signature on the

committed messages $(\mathbf{e}_u, \tilde{\mathbf{k}}_0, \tilde{\mathbf{k}}_1, \tilde{\mathbf{t}})$, where $(\tilde{\mathbf{k}}_0, \tilde{\mathbf{k}}_1, \tilde{\mathbf{t}})$ are bitwise decompositions of PRF seeds and $\mathbf{e}_u$ is the user's private key for which the corresponding public key is a GPV syndrome $PK_{\mathcal{U}} = \mathbf{F} \cdot \mathbf{e}_u \in \mathbb{Z}_p^m$, for a random matrix $\mathbf{F} \in \mathbb{Z}_p^{m \times m\lceil \log q \rceil}$.

In each spent coin, the user computes a serial number $\mathbf{y}_S = F_{\mathbf{k}}(J) \in \mathbb{Z}_p^m$ consisting of a PRF evaluation under $\mathbf{k} \in \mathbb{Z}_p^m$ and generates a NIZK argument that $\mathbf{y}_S$ is the correct evaluation for the secret index J and the key $\mathbf{k} = \mathbf{k}_0 + \mathbf{k}_1$ contained in the certified wallet. Note that the argument does not require a separate commitment to $\mathbf{k}$ since the bank's signature $sig_{\mathcal{B}}$ on the message $(\mathbf{e}_u, \tilde{\mathbf{k}}_0, \tilde{\mathbf{k}}_1, \tilde{\mathbf{t}})$ already contains a commitment to the bits of $(\mathbf{k}_0, \mathbf{k}_1)$. Since $sig_{\mathcal{B}}$ and $(\mathbf{e}_u, \tilde{\mathbf{k}}_0, \tilde{\mathbf{k}}_1, \tilde{\mathbf{t}})$ are part of the witnesses that the user argues knowledge of, it is eventually the bank's public key that commits the user to the seed $\mathbf{k}$.

In each coin, the identification of double-spenders is enabled by a security tag $\mathbf{y}_T = PK_{\mathcal{U}} + H_{\mathrm{FRD}}(R) \cdot F_{\mathbf{t}}(J) \in \mathbb{Z}_p^m$, where $H_{\mathrm{FRD}}(R)$ is a Full-Rank Difference function [1,42] of some transaction-specific information. If two coins share the same serial number $\mathbf{y}_S$, the soundness of the argument system implies that the two security tags $\mathbf{y}_{T,1}, \mathbf{y}_{T,2}$ hide the same $PK_{\mathcal{U}}$. By the Full Rank Difference property, subtracting $\mathbf{y}_{T,1} - \mathbf{y}_{T,2}$ exposes $F_{\mathbf{t}}(J) \in \mathbb{Z}_p^m$ and, in turn, $PK_{\mathcal{U}} \in \mathbb{Z}_p^m$.

The details of the underlying argument system are given in Section 5.2, where we show that the considered statement reduces to an instance of the abstraction given in Section 4.1. On the way, we use a combination our transformation techniques for the BLMR PRF from Section 4.3 and the Stern-like techniques for the signature signature scheme of [71].

5.1 Description

In the description below, we use the injective function $\mathsf{vdec}_{\mathtt{n},\mathtt{q}-1}(\cdot)$ defined in Section 3, which maps a vector $\mathbf{v} \in \mathbb{Z}_{\mathtt{q}}^{\mathtt{n}}$ to the vector $\mathsf{vdec}_{\mathtt{n},\mathtt{q}-1}(\mathbf{v}) \in \{0,1\}^{\mathtt{n}\lceil \log_2 \mathtt{q} \rceil}$.

ParGen$(1^\lambda, 1^L)$**:** Given a security parameter $\lambda > 0$ and an integer $L > 0$ such that 2^L is the desired value of wallets, public parameters are chosen as follows.

1. Choose a lattice parameter $n = \mathcal{O}(\lambda)$. Choose parameters that will be used by the BLMR pseudo-random function [17]: an LWE parameter $\alpha = 2^{-\omega(\log^{1+c}(n))}$ for some constant $c > 0$; moduli $p = 2^{\log^{1+c}(n)}$ and $q = \mathcal{O}(\sqrt{n}/\alpha)$ such that p divides q; and dimension $m = \lceil n \log q \rceil$. Pick another prime modulus $q_s = \widetilde{\mathcal{O}}(n^4)$ to be used by the signature scheme. Pick an integer $\ell = \Theta(\lambda)$, a Gaussian parameter $\sigma = \Omega(\sqrt{n \log q_s} \log n)$, and an infinity norm bound $\beta = \sigma \cdot \omega(\log n)$. Let $\delta_{q_s-1} = \lceil \log_2(q_s) \rceil$, $\delta_{q-1} = \lceil \log_2(q) \rceil$, $\delta_{p-1} = \lceil \log_2(p) \rceil$.

   We will use an instance of the signature scheme with efficient protocols from [71], where matrices $(\mathbf{A}, \{\mathbf{A}_j\}_{j=0}^{\ell}, \mathbf{D})$, $\{\mathbf{D}_k\}_{k=0}^{4}$ do not all have the same number of columns. Specifically, let $m_s = m_0 = 2n\delta_{q_s-1}$ and define the length of message blocks to be $m_1 = m_2 = m_3 = m_4 = \bar{m} = m\delta_{q-1}$. We also use an additional matrix $\mathbf{F} \in \mathbb{Z}_p^{m \times m_f}$, where $m_f = \bar{m} = m\delta_{q-1}$.

2. Choose a commitment key CK for a statistically hiding commitment where the message space is $\{0,1\}^{m_1} \times \{0,1\}^{m_2} \times \{0,1\}^{m_3}$. This commitment key $CK = \big([\mathbf{D}'_0 \mid \mathbf{D}''_0],\ \mathbf{D}_1,\ \mathbf{D}_2,\ \mathbf{D}_3,\ \mathbf{D}_4\big)$ consists of random matrices $\mathbf{D}'_0, \mathbf{D}''_0 \hookleftarrow U(\mathbb{Z}_{q_s}^{n\times m_0})$, $\mathbf{D}_1 \hookleftarrow U(\mathbb{Z}_{q_s}^{n\times m_1})$, $\mathbf{D}_2 \hookleftarrow U(\mathbb{Z}_{q_s}^{n\times m_2})$, $\mathbf{D}_3 \hookleftarrow U(\mathbb{Z}_{q_s}^{n\times m_3})$, $\mathbf{D}_4 \hookleftarrow U(\mathbb{Z}_{q_s}^{n\times m_4})$.
3. Select two binary matrices $\mathbf{P}_0, \mathbf{P}_1 \in \{0,1\}^{m\times m}$ uniformly among $\mathbb{Z}_q$-invertible matrices.
4. Finally, choose a full-rank difference function $H_{\mathrm{FRD}} : \mathbb{Z}_p^m \to \mathbb{Z}_p^{m\times m}$ such as [1], a collision-resistant hash function $H_0 : \{0,1\}^* \to \mathbb{Z}_p^m$ and another hash function $H : \{0,1\}^* \to \{1,2,3\}^\kappa$, for some $\kappa = \omega(\log \lambda)$, which will be modeled as a random oracle in the security analysis.

We define

$$\mathsf{par} := \Big(\mathbf{F},\ \{\mathbf{P}_0,\ \mathbf{P}_1\},\ H_{\mathrm{FRD}},\ H_0,\ H,\ CK\Big).$$

where $CK = (\mathbf{D}_0 = [\mathbf{D}'_0 \mid \mathbf{D}''_0],\ \mathbf{D}_1, \mathbf{D}_2,\ \mathbf{D}_3,\ \mathbf{D}_4)$.

BKeygen$(1^\lambda, \mathsf{par})$**:** The bank $\mathcal{B}$ generates a key pair for the signature scheme with efficient protocols. This is done as follows.

1. Run $\mathsf{TrapGen}(1^n, 1^{m_s}, q_s)$ to get $\mathbf{A} \in \mathbb{Z}_{q_s}^{n\times m_s}$ and a short basis $\mathbf{T}_\mathbf{A}$ of $\Lambda_{q_s}^\perp(\mathbf{A})$. This basis allows computing short vectors in $\Lambda_{q_s}^\perp(\mathbf{A})$ with a Gaussian parameter σ. Next, choose matrices $\mathbf{A}_0, \mathbf{A}_1, \ldots, \mathbf{A}_\ell \hookleftarrow U(\mathbb{Z}_{q_s}^{n\times m_s})$.
2. Choose $\mathbf{D} \hookleftarrow U\big(\mathbb{Z}_{q_s}^{n\times (m_s/2)}\big)$ and a random vector $\mathbf{u} \hookleftarrow U(\mathbb{Z}_{q_s}^n)$.

The private key consists of $SK_\mathcal{B} := \mathbf{T}_\mathbf{A}$ while the public key is

$$PK_\mathcal{B} := \big(\mathbf{A},\ \{\mathbf{A}_j\}_{j=0}^\ell,\ \mathbf{D},\ \mathbf{u}\big).$$

UKeygen$(1^\lambda, \mathsf{par})$**:** As a secret key, the user picks $SK_\mathcal{U} := \mathbf{e}_u \hookleftarrow U(\{0,1\}^{m_f})$ at random and computes his public key $PK_\mathcal{U}$ as a syndrome $PK_\mathcal{U} = \mathbf{F}\cdot\mathbf{e}_u \in \mathbb{Z}_p^m$.

Withdraw$\big(\mathcal{U}(PK_\mathcal{B}, SK_\mathcal{U}, 2^L), \mathcal{B}(PK_\mathcal{U}, SK_\mathcal{B}, 2^L)\big)$**:** The bank $\mathcal{B}$, which has a key pair $(SK_\mathcal{B}, PK_\mathcal{B})$, interacts with $\mathcal{U}$, who has $SK_\mathcal{U} = \mathbf{e}_u$, as follows.

1. $\mathcal{U}$ picks $\mathbf{t}, \mathbf{k}_0 \hookleftarrow U(\mathbb{Z}_q^m)$ and computes $\tilde{\mathbf{t}} = \mathsf{vdec}_{m,q-1}(\mathbf{t}) \in \{0,1\}^{\bar{m}}$, $\tilde{\mathbf{k}}_0 = \mathsf{vdec}_{m,q-1}(\mathbf{k}_0) \in \{0,1\}^{\bar{m}}$. Then, he generates a commitment to the 3-block message $(\mathbf{e}_u, \tilde{\mathbf{t}}, \tilde{\mathbf{k}}_0) \in \{0,1\}^{m_f} \times \{0,1\}^{\bar{m}} \times \{0,1\}^{\bar{m}}$. To this end, $\mathcal{U}$ samples $\mathbf{r}_0 \hookleftarrow D_{\mathbb{Z}^{m_s},\sigma}$ and computes

$$\mathbf{c}_\mathcal{U} = \mathbf{D}'_0 \cdot \mathbf{r}_0 + \mathbf{D}_1 \cdot \mathbf{e}_u + \mathbf{D}_2 \cdot \tilde{\mathbf{t}} + \mathbf{D}_3 \cdot \tilde{\mathbf{k}}_0 \in \mathbb{Z}_{q_s}^n, \tag{19}$$

which is sent to $\mathcal{B}$. In addition, $\mathcal{U}$ generates an interactive zero-knowledge argument of knowledge of an opening

$$(\mathbf{r}_0, \mathbf{e}_u, \tilde{\mathbf{t}}, \tilde{\mathbf{k}}_0) \in D_{\mathbb{Z}^{m_s},\sigma} \times \{0,1\}^{m_f} \times \{0,1\}^{\bar{m}} \times \{0,1\}^{\bar{m}}$$

of $\mathbf{c}_\mathcal{U} \in \mathbb{Z}_{q_s}^n$ satisfying (19) and such that $PK_\mathcal{U} = \mathbf{F}\cdot\mathbf{e}_u \in \mathbb{Z}_p^m$. We note that this argument system is obtained via a straightforward adaptation of the Stern-like protocol from [71].

2. If the argument of step 1 verifies, $\mathcal{B}$ samples $\mathbf{r}_1 \hookleftarrow D_{\mathbb{Z}^{m_s},\sigma}$, $\mathbf{k}_1 \hookleftarrow U(\mathbb{Z}_q^m)$ and computes $\tilde{\mathbf{k}}_1 = \mathsf{vdec}_{m,q-1}(\mathbf{k}_1) \in \{0,1\}^{\bar{m}}$ and a re-randomized version of $\mathbf{c}_{\mathcal{U}}$ which is obtained as $\mathbf{c}'_{\mathcal{U}} = \mathbf{c}_{\mathcal{U}} + \mathbf{D}''_0 \cdot \mathbf{r}_1 + \mathbf{D}_4 \cdot \tilde{\mathbf{k}}_1 \in \mathbb{Z}_{q_s}^n$. It defines $\mathbf{u}_{\mathcal{U}} = \mathbf{u} + \mathbf{D} \cdot \mathsf{vdec}_{n,q_s-1}(\mathbf{c}'_{\mathcal{U}}) \in \mathbb{Z}_{q_s}^n$. Next, $\mathcal{B}$ randomly picks $\tau \hookleftarrow \{0,1\}^\ell$ and uses $\mathbf{T}_{\mathbf{A}}$ to compute a delegated basis $\mathbf{T}_\tau \in \mathbb{Z}^{2m_s \times 2m_s}$ for the matrix $\mathbf{A}_\tau \in \mathbb{Z}_{q_s}^{n \times 2m_s}$ defined as

$$\mathbf{A}_\tau = [\mathbf{A} \mid \mathbf{A}_0 + \sum_{j=1}^{\ell} \tau[j] \cdot \mathbf{A}_j] \in \mathbb{Z}_{q_s}^{n \times 2m_s}. \tag{20}$$

Using $\mathbf{T}_\tau \in \mathbb{Z}^{2m_s \times 2m_s}$, $\mathcal{B}$ samples a short vector $\mathbf{v} \in \mathbb{Z}^{2m_s}$ in $D_{\Lambda_{q_s}^{\mathbf{u}_{\mathcal{U}}}(\mathbf{A}_\tau),\sigma}$. It returns $\mathbf{k}_1 \in \mathbb{Z}_q^m$ and the vector $(\tau, \mathbf{v}, \mathbf{r}_1) \in \{0,1\}^\ell \times \mathbb{Z}^{2m_s} \times \mathbb{Z}^{m_s}$ to $\mathcal{U}$.

3. $\mathcal{U}$ computes $\mathbf{r} = \left[\begin{array}{c}\mathbf{r}_0\\ \hline \mathbf{r}_1\end{array}\right] \in \mathbb{Z}^{2m_s}$ and verifies that

$$\begin{aligned}\mathbf{A}_\tau \cdot \mathbf{v} = \mathbf{u} + \mathbf{D} \cdot \mathsf{vdec}_{n,q_s-1}\Big(&\mathbf{D}_0 \cdot \mathbf{r} + \mathbf{D}_1 \cdot \mathbf{e}_u + \mathbf{D}_2 \cdot \mathsf{vdec}_{m,q-1}(\mathbf{t})\\ &+ \mathbf{D}_3 \cdot \mathsf{vdec}_{m,q-1}(\mathbf{k}_0) + \mathbf{D}_4 \cdot \mathsf{vdec}_{m,q-1}(\mathbf{k}_1)\Big) \bmod q_s\end{aligned}$$

and $\|\mathbf{v}\| \leq \sigma\sqrt{2m_s}$, $\|\mathbf{r}_1\| \leq \sigma\sqrt{m_s}$. If so, $\mathcal{U}$ saves the wallet

$$\mathcal{W} := \Big(\mathbf{e}_u, \mathbf{t}, \mathbf{k}_0, \mathbf{k}_1, sig_{\mathcal{B}} = (\tau, \mathbf{v}, \mathbf{r}), J = 0\Big),$$

where $J \in \{0, \ldots, 2^L - 1\}$ is a counter initialized to 0 (otherwise, it outputs $\perp$). The bank $\mathcal{B}$ records a debit of 2^L for the account $PK_{\mathcal{U}}$.

Spend$\big(\mathcal{U}(\mathcal{W}, PK_{\mathcal{B}}, PK_{\mathcal{M}}, \mathtt{info}), \mathcal{M}(SK_{\mathcal{M}}, PK_{\mathcal{B}}, 2^L)\big)$**:** The user $\mathcal{U}$, on input of a wallet $\mathcal{W} = \big(\mathbf{e}_u, \mathbf{t}, \mathbf{k}_0, \mathbf{k}_1, sig_{\mathcal{B}} = (\tau, \mathbf{v}, \mathbf{r}), J\big)$, outputs $\perp$ if $J > 2^L - 1$. Otherwise, it runs the following protocol with $\mathcal{M}$.

1. Hash $\mathsf{info} \in \{0,1\}^*$ and $PK_{\mathcal{M}}$ to obtain $R = H_0(PK_{\mathcal{M}}, \mathsf{info}) \in \mathbb{Z}_p^m$.
2. Compute $\mathbf{k} = \mathbf{k}_0 + \mathbf{k}_1 \bmod q$, which will serve a PRF seed $\mathbf{k} \in \mathbb{Z}_q^m$. Using $\mathbf{k}$, compute the serial number

$$\mathbf{y}_S = \lfloor \prod_{i=1}^{L} \mathbf{P}_{J[L+1-i]} \cdot \mathbf{k} \rfloor_p \ \in \mathbb{Z}_p^m, \tag{21}$$

where $J[1] \ldots J[L] \in \{0,1\}^L$ is the representation of $J \in \{0, \ldots, 2^L - 1\}$.
3. Using the PRF seed $\mathbf{t} \in \mathbb{Z}_q^m$, compute the security tag

$$\mathbf{y}_T = PK_{\mathcal{U}} + H_{\mathrm{FRD}}(R) \cdot \lfloor \prod_{i=1}^{L} \mathbf{P}_{J[L+1-i]} \cdot \mathbf{t} \rfloor_p \ \in \mathbb{Z}_p^m. \tag{22}$$

4. Generate a non-interactive argument of knowledge π_K to prove that:

(i) The given serial number $\mathbf{y}_S$ is the correct output of the PRF with key $\mathbf{k} = \mathbf{k}_0 + \mathbf{k}_1 \bmod q$ and input $J[1] \dots J[L]$ (Equation (21));
(ii) The same input $J[1] \dots J[L]$ and another key $\mathbf{t}$ involve in the generation of the security tag $\mathbf{y}_T$ of the form (22);
(iii) The PRF keys $\mathbf{k}_0, \mathbf{k}_1, \mathbf{t}$ and the secret key $\mathbf{e}_u$ that corresponds to $PK_{\mathcal{U}}$ in (22) were certified by the bank.

This is done by running the interactive zero-knowledge argument presented in Section 5.2, which can be seen as a combination of 2 instances of the protocol for the PRF layer from Section 4.3 and one instance of the protocol for the signature layer from [71]. The argument is repeated $\kappa = \omega(\log \lambda)$ times to achieve negligible soundness error, and then made non-interactive using the Fiat-Shamir heuristic [48] as a triple $\pi_K = (\{\mathsf{Comm}_{K,j}\}_{j=1}^{\kappa}, \mathsf{Chall}_K, \{\mathsf{Resp}_{K,j}\}_{j=1}^{\kappa})$ where

$$\mathsf{Chall}_K = H(R, \mathbf{y}_S, \mathbf{y}_T, \{\mathsf{Comm}_{K,j}\}_{j=1}^{t}) \in \{1,2,3\}^{\kappa}.$$

$\mathcal{U}$ sends $coin = \big(R, \mathbf{y}_S, \mathbf{y}_T, \pi_K\big)$ to $\mathcal{M}$ who outputs $coin$ if $\mathsf{VerifyCoin}$ accepts it and $\perp$ otherwise. $\mathcal{U}$ outputs an updated wallet $\mathcal{W}'$, where J is incremented. We note that $coin$ has bit-size $\widetilde{\mathcal{O}}(L \cdot \lambda + \lambda^2)$, which is inherited from that of the underlying zero-knowledge argument system of Section 5.2.

$\mathsf{VerifyCoin}(\mathsf{par}, PK_{\mathcal{M}}, PK_{\mathcal{B}}, coin)$: Parse the coin as $coin = \big(R, \mathbf{y}_S, \mathbf{y}_T, \pi_K\big)$ and output 1 if and only if π_K properly verifies.

$\mathsf{Deposit}\big(\mathcal{M}(SK_{\mathcal{M}}, coin, PK_{\mathcal{B}})), \mathcal{B}(PK_{\mathcal{M}}, SK_{\mathcal{B}}, \mathsf{state}_{\mathcal{B}})\big)$: $coin = \big(R, \mathbf{y}_S, \mathbf{y}_T, \pi_K\big)$ is sent by $\mathcal{M}$ to the bank $\mathcal{B}$. If $\mathsf{VerifyCoin}(\mathsf{par}, PK_{\mathcal{M}}, PK_{\mathcal{B}}, coin) = 1$ and if serial number $\mathbf{y}_S$ does not already appear in any coin of the list $\mathsf{state}_{\mathcal{B}}$, $\mathcal{B}$ accepts $coin$, adds $(R, \mathbf{y}_S)$ in $\mathsf{state}_{\mathcal{B}}$ and credits $PK_{\mathcal{M}}$'s account. Otherwise, $\mathcal{B}$ returns "user" or "$\mathsf{merchant}$" depending on which party is declared faulty.

$\mathsf{Identify}\big(\mathsf{par}, PK_{\mathcal{B}}, coin_1, coin_2)\big)$: Given two coins $coin_1 = \big(R_1, \mathbf{y}_S, \mathbf{y}_{T,1}, \pi_{K,1}\big)$, $coin_2 = \big(R_2, \mathbf{y}_S, \mathbf{y}_{T,2}, \pi_{K,2}\big)$ with verifying proofs $\pi_{K,1}, \pi_{K,2}$ and the same serial number $\mathbf{y}_S \in \mathbb{Z}_p^m$ in distinct transactions $R_1 \neq R_2$, output $\perp$ if $\mathbf{y}_{T,1} = \mathbf{y}_{T,2}$. Otherwise, compute

$$\mathbf{y}'_T = \big(H_{\mathrm{FRD}}(R_1) - H_{\mathrm{FRD}}(R_2)\big)^{-1} \cdot (\mathbf{y}_{T,1} - \mathbf{y}_{T,2}) \in \mathbb{Z}_p^m$$

and then $PK_{\mathcal{U}} = \mathbf{y}_{T,1} - H_{\mathrm{FRD}}(R_1) \cdot \mathbf{y}'_T \in \mathbb{Z}_p^m$. The proof Π_G that $\mathcal{U}$ is guilty simply consists of the two coins $coin_1$, $coin_2$ and the public key $PK_{\mathcal{U}}$.

In Appendix E, we show how to extend the scheme with a mechanism allowing to trace all the coins of an identified double-spender. Like Camenisch *et al.* [23], we can add this feature via a verifiable encryption step during the withdrawal phase. For this purpose, however, [23] crucially relies on properties of groups with a bilinear map that are not available here. To overcome this difficulty, we slightly depart from the design principle of [23] in that we rather use a secret-key verifiable encryption based on the hardness of LWE.

5.2 The Underlying Argument System of Our E-Cash Scheme

We now present the argument system employed by the Spend algorithm of the e-cash scheme in Section 5.1. This protocol is summarized as follows.

Let parameters λ, n, p, q, q_s, m, β, L, ℓ, $\bar{m} = m\delta_{q-1}$, $m_s = 2n\delta_{q_s-1}$ be as specified in Section 5.1. The public input consists of:

$$\begin{cases} \mathbf{D} \in \mathbb{Z}_{q_s}^{n\times(m_s/2)};\ \mathbf{D}_0 \in \mathbb{Z}_{q_s}^{n\times 2m_s};\ \{\mathbf{D}_k \in \mathbb{Z}_{q_s}^{n\times\bar{m}}\}_{k=1}^{4};\ \mathbf{A}, \{\mathbf{A}_j\}_{j=0}^{\ell} \in \mathbb{Z}_{q_s}^{n\times m_s}; \\ \mathbf{F} \in \mathbb{Z}_p^{m\times\bar{m}};\ \mathbf{u} \in \mathbb{Z}_{q_s}^{n};\ \mathbf{P}_0, \mathbf{P}_1 \in \{0,1\}^{m\times m};\ H_{\mathrm{FRD}}(R) \in \mathbb{Z}_p^{m\times m};\ \mathbf{y}_S, \mathbf{y}_T \in \mathbb{Z}_p^{m}. \end{cases}$$

The prover's goal is to prove in zero-knowledge the possession of

$$\begin{cases} \mathbf{v}_1, \mathbf{v}_2 \in [-\beta,\beta]^{m_s};\ \ \mathbf{r} \in [-\beta,\beta]^{2m_s};\ \ \tilde{\mathbf{w}} \in \{0,1\}^{m_s/2}; \\ \mathbf{e}_u,\ \tilde{\mathbf{k}}_0,\ \tilde{\mathbf{k}}_1,\ \tilde{\mathbf{t}} \in \{0,1\}^{\bar{m}};\ \ \mathbf{y}'_T \in \mathbb{Z}_p^m;\ \ \mathbf{k} \in \mathbb{Z}_q^m; \\ \mathbf{k}_0 = \mathbf{H}_{m,q-1}\cdot\tilde{\mathbf{k}}_0 \in \mathbb{Z}_q^m;\ \ \mathbf{k}_1 = \mathbf{H}_{m,q-1}\cdot\tilde{\mathbf{k}}_1 \in \mathbb{Z}_q^m;\ \ \mathbf{t} = \mathbf{H}_{m,q-1}\cdot\tilde{\mathbf{t}} \in \mathbb{Z}_q^m; \\ \tau[1]\ldots\tau[\ell] \in \{0,1\}^{\ell};\ \ J[1]\ldots J[L] \in \{0,1\}^{L}, \end{cases}$$

such that the following equations hold:

$$\mathbf{A}\cdot\mathbf{v}_1 + \mathbf{A}_0\cdot\mathbf{v}_2 + \sum_{j=1}^{\ell}\mathbf{A}_j\cdot(\tau[j]\cdot\mathbf{v}_2) - \mathbf{D}\cdot\tilde{\mathbf{w}} = \mathbf{u} \in \mathbb{Z}_{q_s}^n, \tag{23}$$

$$\mathbf{D}_0\cdot\mathbf{r} + \mathbf{D}_1\cdot\mathbf{e}_u + \mathbf{D}_2\cdot\tilde{\mathbf{t}} + \mathbf{D}_3\cdot\tilde{\mathbf{k}}_0 + \mathbf{D}_4\cdot\tilde{\mathbf{k}}_1 - \mathbf{H}_{n,q_s-1}\cdot\tilde{\mathbf{w}} = \mathbf{0} \in \mathbb{Z}_{q_s}^n, \tag{24}$$

$$\mathbf{k} = \mathbf{k}_0 + \mathbf{k}_1 \in \mathbb{Z}_q^m; \quad \mathbf{y}_S = \Big\lfloor \prod_{i=1}^{L}\mathbf{P}_{J[L+1-i]}\cdot\mathbf{k} \Big\rfloor_p \in \mathbb{Z}_p^m, \tag{25}$$

$$\mathbf{y}'_T = \Big\lfloor \prod_{i=1}^{L}\mathbf{P}_{J[L+1-i]}\cdot\mathbf{t} \Big\rfloor_p \in \mathbb{Z}_p^m, \quad \mathbf{y}_T = \mathbf{F}\cdot\mathbf{e}_u + H_{\mathrm{FRD}}(R)\cdot\mathbf{y}'_T \ \in \mathbb{Z}_p^m. \tag{26}$$

Our strategy is to reduce the above statement to an instance of the abstraction in Section 4.1. To this end, we will combine the zero-knowledge proofs of signatures from the Stern-like techniques of [71] and our techniques for the PRF layer from Section 4.3. Specifically, we let $q_1 = q_s$, $q_2 = q$, $q_3 = pq$, $q_4 = p$, and perform the following transformations.

Regarding the two equations of the signature relation in (23)-(24), we apply the following decompositions and/or extensions to the underlying secret vectors:

$$\begin{cases} \{\breve{\mathbf{v}}_i = \mathsf{ThreeExt}\big(\mathsf{vdec}'_{m_s,\beta}(\mathbf{v}_i)\big) \in \mathsf{B}^3_{m_s\delta_\beta}\}_{i=1}^{2};\quad \{\mathbf{c}_j = \mathsf{expand}(\tau[j], \breve{\mathbf{v}}_2)\}_{j=1}^{\ell}; \\ \breve{\mathbf{r}} = \mathsf{ThreeExt}\big(\mathsf{vdec}'_{2m_s,\beta}(\mathbf{r})\big) \in \mathsf{B}^3_{2m_s\delta_\beta};\quad \widehat{\mathbf{w}} = \mathsf{TwoExt}\big(\tilde{\mathbf{w}}\big) \in \mathsf{B}^2_{m_s/2}; \\ \widehat{\mathbf{e}} = \mathsf{TwoExt}(\mathbf{e}_u) \in \mathsf{B}^2_{\bar{m}};\quad \forall\alpha \in \{\mathbf{t}, \mathbf{k}_0, \mathbf{k}_1\} : \widehat{\alpha} = \mathsf{TwoExt}\big(\tilde{\alpha}\big) \in \mathsf{B}^2_{\bar{m}}. \end{cases} \tag{27}$$

At the same time, we also transform the associated public matrices $\mathbf{A}$, $\{\mathbf{A}_j\}_{j=0}^{\ell}$, $\mathbf{D}$, $\{\mathbf{D}_j\}_{j=0}^{4}$, $\mathbf{H}_{n,q_s-1}$ accordingly, so that the equations are preserved. Next, we combine the vectors obtained in (27) into:

$$\mathbf{w}_1 = \big(\breve{\mathbf{v}}_1 \,\|\, \breve{\mathbf{v}}_2 \,\|\, \mathbf{c}_1 \,\|\, \ldots \,\|\, \mathbf{c}_\ell \,\|\, \breve{\mathbf{r}} \,\|\, \widehat{\mathbf{w}} \,\|\, \widehat{\mathbf{e}} \,\|\, \widehat{\mathbf{t}} \,\|\, \widehat{\mathbf{k}}_0 \,\|\, \widehat{\mathbf{k}}_1\big) \in \{-1,0,1\}^{d_1}, \tag{28}$$

where $d_1 = 6(\ell+2)m_s\delta_\beta + m_s + 8\bar{m}$. We observe that the two equations can be unified into just one equation of the form $\mathbf{M}_1 \cdot \mathbf{w}_1 = \mathbf{u}_1 \bmod q_1$, where $\mathbf{M}_1 \in \mathbb{Z}_{q_1}^{2n \times d_1}$ is built from public matrices, and $\mathbf{u}_1 = (\mathbf{u} \,\|\, \mathbf{0}) \in \mathbb{Z}_{q_1}^{2n}$.

We now consider equations in (25) and (26), which involve PRF evaluations. We note that, for all $\alpha \in \{\mathbf{t}, \mathbf{k}_0, \mathbf{k}_1\}$ appearing in this layer, we have the connection

$$\alpha = \mathbf{H}_{m,q-1} \cdot \tilde{\alpha} = \widehat{\mathbf{H}}_{m,q-1} \cdot \widehat{\alpha},$$

where $\widehat{\alpha}$ is constructed in 27. To transform the equation $\mathbf{k} = \mathbf{k}_0 + \mathbf{k}_1 \in \mathbb{Z}_q^m$ in (25), we let $\widehat{\mathbf{k}} = \mathsf{TwoExt}\big(\mathsf{vdec}_{m,q-1}(\mathbf{k})\big) \in \mathsf{B}_{\bar{m}}^2$, and rewrite the equation as

$$\widehat{\mathbf{H}}_{m,q-1} \cdot \widehat{\mathbf{k}}_0 + \widehat{\mathbf{H}}_{m,q-1} \cdot \widehat{\mathbf{k}}_1 - \widehat{\mathbf{H}}_{m,q-1} \cdot \widehat{\mathbf{k}} = \mathbf{0} \bmod q.$$

Letting $\mathbf{M}_{k,2} = [\widehat{\mathbf{H}}_{m,q-1} \mid \widehat{\mathbf{H}}_{m,q-1} \mid -\widehat{\mathbf{H}}_{m,q-1}]$ and $\mathbf{u}_{k,2} = \mathbf{0}$, we have the equation $\mathbf{M}_{k,2} \cdot \mathbf{w}_{k,2} = \mathbf{u}_{k,2} \bmod q_2$, where:

$$\mathbf{w}_{k,2} = (\widehat{\mathbf{k}}_0 \,\|\, \widehat{\mathbf{k}}_1 \,\|\, \widehat{\mathbf{k}}). \tag{29}$$

The evaluation process of $\mathbf{y}_S$ in (25) is handled as in Section 4.3, resulting in equations $\mathbf{M}_{S,2} \cdot \mathbf{w}_{S,2} = \mathbf{u}_{S,2} \bmod q_2$, and $\mathbf{M}_{S,3} \cdot \mathbf{w}_{S,3} = \mathbf{u}_{S,3} \bmod q_3$, where

$$\mathbf{w}_{S,2} = \big(\mathbf{s}_{S,0} \,\|\, \widehat{\mathbf{x}}_{S,1} \,\|\, \cdots \,\|\, \mathbf{s}_{S,L-1} \,\|\, \widehat{\mathbf{x}}_{S,L}\big); \quad \mathbf{w}_{S,3} = \big(\widehat{\mathbf{x}}_{S,L} \,\|\, \widehat{\mathbf{z}}_S\big), \tag{30}$$

satisfy $\{\widehat{\mathbf{x}}_{S,i}\}_{i=1}^{L}$, $\widehat{\mathbf{z}}_S \in \mathsf{B}_{\bar{m}}^2$ and

$$\mathbf{s}_{S,0} = \mathsf{expand}(J[1], \widehat{\mathbf{k}}); \qquad \{\mathbf{s}_{S,i-1} = \mathsf{expand}(J[i], \widehat{\mathbf{x}}_{S,i-1})\}_{i=2}^{L}.$$

Regarding the evaluation of $\mathbf{y}'_T$ in (26), equations appearing in the iteration step can also be unified into one of the form $\mathbf{M}_{T,2} \cdot \mathbf{w}_{T,2} = \mathbf{u}_{T,2} \bmod q_2$, where

$$\mathbf{w}_{T,2} = \big(\mathbf{s}_{T,0} \| \widehat{\mathbf{x}}_{T,1} \| \cdots \| \mathbf{s}_{T,L-1} \| \widehat{\mathbf{x}}_{T,L}\big), \tag{31}$$

satisfy

$$\{\widehat{\mathbf{x}}_{T,i} \in \mathsf{B}_{\bar{m}}^2\}_{i=1}^{L}; \quad \mathbf{s}_{T,0} = \mathsf{expand}(J[1], \widehat{\mathbf{t}}); \quad \{\mathbf{s}_{T,i-1} = \mathsf{expand}(J[i], \widehat{\mathbf{x}}_{T,i-1})\}_{i=2}^{L}.$$

Meanwhile, the rounding step is handled in a slightly different manner as the output $\mathbf{y}'_T \in \mathbb{Z}_p^m$ is hidden. Letting $\widehat{\mathbf{y}}'_T = \mathsf{TwoExt}\big(\mathsf{vdec}_{m,p-1}(\mathbf{y}'_T)\big) \in \mathsf{B}_{m\delta_{p-1}}^2$, we are presented with the equation

$$(p \cdot \widehat{\mathbf{H}}_{m,q-1}) \cdot \widehat{\mathbf{x}}_{T,L} - \widehat{\mathbf{H}}_{m,q-1} \cdot \widehat{\mathbf{z}}_T - (q \cdot \widehat{\mathbf{H}}_{m,p-1}) \cdot \widehat{\mathbf{y}}'_T = \mathbf{0} \bmod pq,$$

where $\widehat{\mathbf{z}}_T \in \mathsf{B}^2_{\bar{m}}$. This equation can be written as $\mathbf{M}_{T,3} \cdot \mathbf{w}_{T,3} = \mathbf{u}_{T,3} \bmod q_3$, where $\mathbf{M}_{T,3} = \big[p \cdot \widehat{\mathbf{H}}_{m,q-1} \mid -\widehat{\mathbf{H}}_{m,q-1} \mid -q \cdot \widehat{\mathbf{H}}_{m,p-1}\big]$, $\mathbf{u}_{T,3} = \mathbf{0}$, and

$$\mathbf{w}_{T,3} = (\widehat{\mathbf{x}}_{T,L} \parallel \widehat{\mathbf{z}}_T \parallel \widehat{\mathbf{y}}'_T). \tag{32}$$

Furthermore, we observe that, the three equations modulo q_2, as well as the two equations modulo q_3 we have obtained above can be unified as follows. Let

$$\mathbf{M}_2 = \begin{bmatrix} \mathbf{M}_{k,2} & & \\ & \mathbf{M}_{S,2} & \\ & & \mathbf{M}_{T,2} \end{bmatrix};\ \mathbf{u}_2 = \begin{pmatrix} \mathbf{u}_{k,2} \\ \mathbf{u}_{S,2} \\ \mathbf{u}_{T,2} \end{pmatrix};\ \mathbf{M}_3 = \begin{bmatrix} \mathbf{M}_{S,3} & \\ & \mathbf{M}_{T,3} \end{bmatrix};\ \mathbf{u}_3 = \begin{pmatrix} \mathbf{u}_{S,3} \\ \mathbf{u}_{T,3} \end{pmatrix},$$

then we have $\mathbf{M}_2 \cdot \mathbf{w}_2 = \mathbf{u}_2 \bmod q_2$ and $\mathbf{M}_3 \cdot \mathbf{w}_3 = \mathbf{u}_3 \bmod q_3$, where

$$\mathbf{w}_2 = (\mathbf{w}_{k,2} \parallel \mathbf{w}_{S,2} \parallel \mathbf{w}_{T,2}) \in \{-1,0,1\}^{d_2}; \tag{33}$$
$$\mathbf{w}_3 = (\mathbf{w}_{S,3} \parallel \mathbf{w}_{T,3}) \in \{-1,0,1\}^{d_3}, \tag{34}$$

for $\mathbf{w}_{k,2}, \mathbf{w}_{S,2}, \mathbf{w}_{T,2}, \mathbf{w}_{S,3}, \mathbf{w}_{T,3}$ defined by (29)-(32), and for $d_2 = 6\bar{m}(2L+1)$, $d_3 = 8\bar{m} + 2m\delta_{p-1}$.

Now, the remaining equation in (26) can be written as:

$$\big[\mathbf{F} \mid \mathbf{0}^{m \times \bar{m}}\big] \cdot \widehat{\mathbf{e}} + \big(H_{\mathrm{FRD}}(R) \cdot \widehat{\mathbf{H}}_{m,p-1}\big) \cdot \widehat{\mathbf{y}}'_T = \mathbf{y}_T \bmod p,$$

where $\widehat{\mathbf{e}}$ and $\widehat{\mathbf{y}}'_T$ are as constructed earlier. We therefore obtain the equation $\mathbf{M}_4 \cdot \mathbf{w}_4 = \mathbf{u}_4 \bmod q_4$, where $\mathbf{M}_4 = \big[H_{\mathrm{FRD}}(R) \cdot \widehat{\mathbf{H}}_{m,p-1} \mid \mathbf{F} \mid \mathbf{0}^{m \times \bar{m}}\big]$, $\mathbf{u}_4 = \mathbf{y}_T$ and, for $d_4 = 2\bar{m} + 2m\delta_{p-1}$,

$$\mathbf{w}_4 = (\widehat{\mathbf{y}}'_T \parallel \widehat{\mathbf{e}}) \in \{-1,0,1\}^{d_4}. \tag{35}$$

At this point, we have transformed all the considered equations into four equations $\{\mathbf{M}_i \cdot \mathbf{w}_i = \mathbf{u}_i \bmod q_i\}_{i=1}^4$. We then let $d = \sum_{i=1}^4 d_i$ and, for $\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3, \mathbf{w}_4$ defined by (28), (33), (34), (35), respectively, let

$$\mathbf{w} = (\mathbf{w}_1 \| \mathbf{w}_2 \| \mathbf{w}_3 \| \mathbf{w}_4) \in \{-1,0,1\}^d. \tag{36}$$

Let us now specify the set VALID containing $\mathbf{w}$, the set $\mathcal{S}$ and the associated permutation Γ_ϕ, satisfying conditions in 9.

Let VALID be the set of all vectors in $\{-1,0,1\}^d$ having the form (36) (which follows from (28)-(35)), whose block-vectors satisfy the following conditions:

$$\begin{cases} \check{\mathbf{v}}_1, \check{\mathbf{v}}_2 \in \mathsf{B}^3_{m_s\delta_\beta};\ \ \check{\mathbf{r}} \in \mathsf{B}^3_{2m_s\delta_\beta};\ \ \widehat{\mathbf{w}} \in \mathsf{B}^2_{m_s/2};\ \ \mathbf{y}'_T \in \mathsf{B}^2_{m\delta_{p-1}}; \\ \widehat{\mathbf{e}},\ \widehat{\mathbf{t}},\ \widehat{\mathbf{k}}_0,\ \widehat{\mathbf{k}}_1,\ \widehat{\mathbf{k}},\ \widehat{\mathbf{x}}_{S,1}, \ldots, \widehat{\mathbf{x}}_{S,L},\ \widehat{\mathbf{x}}_{T,1}, \ldots, \widehat{\mathbf{x}}_{T,L},\ \widehat{\mathbf{z}}_S,\ \widehat{\mathbf{z}}_T \in \mathsf{B}^2_{\bar{m}}; \\ \{\mathbf{c}_j = \mathsf{expand}(\tau[j], \check{\mathbf{v}}_2)\}_{j=1}^{\ell};\ \ \mathbf{s}_{S,0} = \mathsf{expand}(J[1], \widehat{\mathbf{k}});\ \ \mathbf{s}_{T,0} = \mathsf{expand}(J[1], \widehat{\mathbf{t}}); \\ \{\mathbf{s}_{S,i-1} = \mathsf{expand}(J[i], \widehat{\mathbf{x}}_{S,i-1}),\ \ \mathbf{s}_{T,i-1} = \mathsf{expand}(J[i], \widehat{\mathbf{x}}_{T,i-1})\}_{i=2}^{L}, \end{cases}$$

for some $\tau[1] \ldots \tau[\ell] \in \{0,1\}^\ell$ and some $J[1] \ldots J[L] \in \{0,1\}^L$. By construction, our vector $\mathbf{w}$ belongs to this tailored set VALID.

Now, we define

$$\mathcal{S} := (\mathcal{S}_{3m_s\delta_\beta})^2 \times \mathcal{S}_{6m_s\delta_\beta} \times \mathcal{S}_{m_s} \times \mathcal{S}_{2m\delta_{p-1}} \times (\mathcal{S}_{2\bar{m}})^{2L+7} \times \{0,1\}^\ell \times \{0,1\}^L.$$

Then, for any element $\phi \in \mathcal{S}$ of the form

$$\begin{aligned}\phi = \big(&\phi_{\check{\mathbf{v}}_1},\ \phi_{\check{\mathbf{v}}_2},\ \phi_{\check{\mathbf{r}}},\ \phi_{\widehat{\mathbf{w}}},\ \phi_{\mathbf{y}'_T},\ \phi_{\widehat{\mathbf{e}}},\ \phi_{\widehat{\mathbf{t}}},\ \phi_{\widehat{\mathbf{k}}_0},\ \phi_{\widehat{\mathbf{k}}_1},\ \phi_{\widehat{\mathbf{k}}},\ \phi_{\widehat{\mathbf{x}}_{S,1}},\ \ldots,\ \phi_{\widehat{\mathbf{x}}_{S,L}},\\ &\phi_{\widehat{\mathbf{x}}_{T,1}},\ \ldots,\ \phi_{\widehat{\mathbf{x}}_{T,L}},\ \phi_{\widehat{\mathbf{z}}_S},\ \phi_{\widehat{\mathbf{z}}_T},\ a[1]\ldots a[\ell],\ b[1]\ldots b[L]\big),\end{aligned}$$

let Γ_ϕ be the permutation that, on input vector $\mathbf{w} \in \mathbb{Z}^d$ of the form (18) (which is implied by (28)-(35)), it transforms the block-vectors of $\mathbf{w}$ as follows:

- Apply permutation ϕ_α to block α, for all

$$\alpha \in \big\{\check{\mathbf{v}}_1,\ \check{\mathbf{v}}_2,\ \check{\mathbf{r}},\ \widehat{\mathbf{w}},\ \mathbf{y}'_T,\ \widehat{\mathbf{e}},\ \widehat{\mathbf{t}},\ \widehat{\mathbf{k}}_0,\ \widehat{\mathbf{k}}_1,\ \widehat{\mathbf{k}},\ \widehat{\mathbf{x}}_{S,1},\ldots,\widehat{\mathbf{x}}_{S,L},\ \widehat{\mathbf{x}}_{T,1},\ldots,\widehat{\mathbf{x}}_{T,L},\ \widehat{\mathbf{z}}_S,\ \widehat{\mathbf{z}}_T\big\}.$$

- For $j \in [\ell]$, apply permutation $T_{a[j],\phi_{\check{\mathbf{v}}_2}}$ to block $\mathbf{c}_j$.
- Apply permutation $T_{b[1],\phi_{\widehat{\mathbf{k}}}}$ to block $\mathbf{s}_{S,0}$, and $T_{b[1],\phi_{\widehat{\mathbf{t}}}}$ to block $\mathbf{s}_{T,0}$.
- For $i \in [2,L]$, apply permutation $T_{b[i],\phi_{\widehat{\mathbf{x}}_{S,i-1}}}$ to block $\mathbf{s}_{S,i-1}$, and permutation $T_{b[i],\phi_{\widehat{\mathbf{x}}_{T,i-1}}}$ to block $\mathbf{s}_{T,i-1}$.

It can be checked that, we have $\mathbf{w} \in \mathsf{VALID}$ if and only if $\Gamma_\phi(\mathbf{w}) \in \mathsf{VALID}$, thanks to the equivalences (3), (4), (5) from Section 3. Furthermore, if $\phi \leftarrow U(\mathcal{S})$, then $\Gamma_\phi(\mathbf{w})$ is uniform in VALID. In other words, the conditions in 9 are satisfied.

Given the above transformations and specifications, we can run the abstract protocol of Figure 1 to prove knowledge of $\mathbf{w} = (\mathbf{w}_1 \| \mathbf{w}_2 \| \mathbf{w}_3 \mathbf{w}_4) \in \mathsf{VALID}$ satisfying $\{\mathbf{M}_i \cdot \mathbf{w}_i = \mathbf{u}_i \bmod q_i\}_{i\in[4]}$, where public matrices/vectors $\{\mathbf{M}_i, \mathbf{u}_i\}_{i\in[4]}$ are as constructed above. As a result, we obtain a statistical zero-knowledge argument of knowledge for the considered statement.

Each round of the protocol has communication cost $\mathcal{O}(\sum_{i=1}^4 d_i \cdot \log q_i)$. For the parameters in Section 5.1, this cost is of order $\widetilde{\mathcal{O}}(L \cdot \lambda + \lambda^2)$. In the Spend algorithm, the protocol is repeated $\kappa = \omega(\log \lambda)$ to achieve negligible soundness error. The global communication cost is $\widetilde{\mathcal{O}}(L \cdot \lambda + \lambda^2) \cdot \omega(\log \lambda) = \widetilde{\mathcal{O}}(L \cdot \lambda + \lambda^2)$.

6 Security

Theorem 2. *The scheme guarantees balance under the* SIS *assumption in the random oracle model.* (The proof is in Appendix D.1).

Theorem 3 shows that, under the SIS assumption and assuming the collision-resistance of H_0, double-spenders can always be identified by the bank. Analogously to the security proof of Camenisch *et al.* [23] which relies on a similar feature of the Dodis-Yampolskiy PRF [46], the proof uses some range-disjointness property of the underlying small-domain PRF: namely, two functions keyed by independent keys should have disjoint ranges with high probability. In Appendix D.2[Lemma 6], we prove this property unconditionally for the BLMR PRF [17].

Theorem 3. *If H_0 is a collision-resistant hash function and H is modeled as a random oracle, the scheme guarantees the identification of double spenders under the* SIS *assumption.* (The proof is in Appendix D.2.)

Theorem 4. *The scheme provides strong exculpability under the* SIS *assumption in the random oracle model.* (The proof is given in Appendix D.3.)

Theorem 5. *The scheme provides anonymity under the* LWE *assumption in the random oracle model.* (The proof is in Appendix D.4.)

Our scheme can be modified so as to use the more efficient LWR-based PRF based on the GGM technique. This allows significantly improving the choice of parameters at the expense of a longer description and a more complex proof for the identification property. The reason is that, in the GGM-based PRF, the range disjointness property (for small domains) does not appear to be provable in the statistical sense. This can be addressed by relying on the pseudo-randomness of the function, as in the security proof of Belenkiy *et al.* [10]. Relying on the pseudo-randomness is perhaps counter-intuitive since the adversary knows the PRF seed in the proof of the identification property. Nevertheless, the reduction still works as in [10, Appendix F] when the domain has polynomial size.

Acknowledgements

Part of this research was funded by Singapore Ministry of Education under Research Grant MOE2016-T2-2-014(S), by the French ANR ALAMBIC project (ANR-16-CE39-0006) and by BPI-France in the context of the national project RISQ (P141580).

References

1. S. Agrawal, D. Boneh, and X. Boyen. Efficient lattice (H)IBE in the standard model. In *Eurocrypt*, 2010.
2. M. Ajtai. Generating hard instances of the short basis problem. In *ICALP*, volume 1644, pages 1–9, 1999.
3. J. Alwen, S. Krenn, K. Pietrzak, and D. Wichs. Learning with rounding revisited – new reduction, properties and applications. In *Crypto 2013*.
4. J. Alwen and C. Peikert. Generating shorter bases for hard random lattices. In *STACS 2009*.
5. M. H. Au, Q. Wu, W. Susilo, and Y. Mu. Compact e-cash from bounded accumulator. In *CT-RSA*, 2007.
6. W. Banaszczyk. New bounds in some transference theorems in the geometry of number. 296, 1993.
7. A. Banerjee and C. Peikert. New and improved key-homomorphic pseudo-random functions. In *Crypto*, 2014.
8. A. Banerjee, C. Peikert, and A. Rosen. Pseudo-random functions and lattices. In *Eurocrypt*, 2012.

9. C. Baum, I. Damgård, K.-G. Larsen, and M. Nielsen. How to prove knowledge of small secrets. In *Crypto 2016*.
10. M. Belenkiy, M. Chase, M. Kohlweiss, and A. Lysyanskaya. Compact e-cash and simulatable vrfs revisited. In *Pairing*, 2009.
11. M. Bellare and P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *ACM-CCS'93*, 1993.
12. E. Ben-Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *IEEE S&P*, 2014.
13. F. Benhamouda, J. Camenisch, S. Krenn, V. Lyubashevsky, and G. Neven. Better zero-knowledge proofs for lattice encryption and their application to group signatures. In *Asiacrypt*, 2014.
14. F. Benhamouda, S. Krenn, V. Lyubashevsky, and K. Pietrzak. Efficient Zero-Knowledge Proofs for Commitments from Learning With Errors over Rings. In *ESORICS*, 2015.
15. A. Bogdanov, S. Guo, D. Masny, S. Richelson, and A. Rosen. On the hardness of leanring with rounding over small modulus. In *TCC 2016*.
16. F. Böhl, D. Hofheinz, T. Jager, J. Koch, and C. Striecks. Confined guessing: New signatures from standard assumptions. *Journal of Cryptology*, 28(1), 2015.
17. D. Boneh, K. Lewi, H. Montgomery, and A. Raghunathan. Key-homomorphic prfs and their applications. In *Crypto*, 2013.
18. X. Boyen. Lattice mixing and vanishing trapdoors: A framework for fully secure short signatures and more. In *PKC 2010*, 2010.
19. Z. Brakerski and Y. T. Kalai. A Framework for Efficient Signatures, Ring Signatures and Identity Based Encryption in the Standard Model. *IACR Cryptology ePrint Archive*, 2010:86, 2010.
20. Z. Brakerski, A. Langlois, C. Peikert, O. Regev, and D. Stehlé. On the classical hardness of learning with errors. In *STOC*, 2013.
21. E. Brickell, D. Pointcheval, S. Vaudenay, and M. Yung. Design validations for discrete logarithm based signature schemes. In *PKC*, 2000.
22. J. Camenisch, S. Hohenberger, M. Kohlweiss, A. Lysyanskaya, and M. Meyerovich. How to win the clone wars: Efficient periodic n-times anonymous authentication. In *ACM-CCS*, 2006.
23. J. Camenisch, S. Hohenberger, and A. Lysyanskaya. Compact e-cash. In *Eurocrypt 2005*.
24. J. Camenisch, S. Hohenberger, and A. Lysyanskaya. Balancing accountability and privacy using e-cash. In *SCN*, 2005.
25. J. Camenisch and A. Lehmann. (un)linkable pseudonyms for governmental databases. In *ACM-CCS*, 2015.
26. J. Camenisch and A. Lysyanskaya. A signature scheme with efficient protocols. In *SCN*, 2002.
27. J. Camenisch and A. Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *Crypto*, 2004.
28. J. Camenisch, A. Lysyanskaya, and M. Meyerovich. Endorsed e-cash. In *IEEE S&P*, 2007.
29. S. Canard and A. Gouget. Divisible e-cash systems can be truly anonymous. In *Eurocrypt*, 2007.
30. S. Canard, D. Pointcheval, O. Sanders, and J. Traoré. Divisible e-cash made practical. In *PKC*, 2015.
31. S. Canard, D. Pointcheval, O. Sanders, and J. Traoré. Scalable divisible e-cash. In *ACNS*, 2015.

32. D. Cash, D. Hofheinz, E. Kiltz, and C. Peikert. Bonsai trees, or how to delegate a lattice basis. In *Eurocrypt*, 2010.
33. M. Chase, C. Ganesh, and P. Mohassel. Efficient zero-knowledge proof of algebraic and non-algebraic statements with applications to privacy preserving credentials. In *Crypto*, 2016.
34. D. Chaum. Blind signatures for untraceable payments. In *Crypto*, 1982.
35. D. Chaum. Blind signature system. In *Crypto*, 1983.
36. D. Chaum. Security without identification: Transactions ssystem to make big brother obsolete. *Comm. of the ACM*, 28(10), 1985.
37. D. Chaum. On-line cash checks. In *Eurocrypt*, 1989.
38. D. Chaum, A. Fiat, and M. Naor. Untraceable electronic cash. In *Crypto*, 1988.
39. D. Chaum and T. Pedersen. Transferred cash grows in size. In *Eurocrypt*, 1992.
40. A. Chiesa, M. Green, J. Liu, P. Miao, I. Miers, and P. Mishra. Decentralized anonymous micropayments. In *Eurocrypt*, 2017.
41. S. Coull, M. Green, and S. Hohenberger. Controlling access to an oblivious database using stateful anonymous credentials. In *PKC*, 2009.
42. R. Cramer and I. Damgård. On the amortized complexity of zero-knowledge protocols. In *Crypto*, 2009.
43. R. Cramer, I. Damgård, C. Xing, and C. Yuan. Amortized complexity of zero-knowledge proofs revisited: Achieving linear soundness slack. In *Eurocrypt 2017*.
44. I. Damgård. Efficient concurrent zero-knowledge in the auxiliary string model. In *Eurocrypt*, 2000.
45. R. del Pino and V. Lyubashevsky. Amortization with fewer equations for proving knowledge of small secrets. Cryptology ePrint Archive: Report 2017/280, 2017.
46. Y. Dodis and A. Yampolskiy. A verifiable random function with short proofs and keys. In *PKC*, 2005.
47. N. Döttling and D. Schröder. Efficient pseudorandom functions via on-the-fly adaptation. In *Crypto*, 2015.
48. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Crypto*, 1987.
49. M. Franklin and M. Yung. Secure and efficient off-line digital money. In *ICALP*, 1993.
50. T. Frederiksen, J.-B. Nielsen, and C. Orlandi. Privacy-free garbled circuits with applications to efficient zero-knowledge. In *Eurocrypt*, 2015.
51. M. Freedman, Y. Ishai, B. Pinkas, and O. Reingold. Keyword search and oblivious pseudorandom functions. In *TCC*, 2005.
52. E. Fujisaki and K. Suzuki. Traceable ring signature. In *PKC*, 2007.
53. C. Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, 2009.
54. C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, 2008.
55. I. Giacomelli, J. Madsen, and C. Orlandi. Zkboo: Faster zero-knowledge for boolean circuits. In *USENIX Security Symposium*, 2016.
56. O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. In *J. of ACM*, volume 33, 1986.
57. S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof-systems. In *STOC*, 1985.
58. S. D. Gordon, J. Katz, and V. Vaikuntanathan. A group signature scheme from lattice assumptions. In *Asiacrypt*, 2010.
59. M. Green and I. Miers. Bolt: Anonymous payment channels for decentralized currencies. Cryptology ePrint Archive: Report 2016/701, 2016.

60. J. Groth and A. Sahai. Efficient non-interactive proof systems for bilinear groups. In *Eurocrypt 2008*.
61. S. Hohenberger, S. Myers, R. Pass, and a. shelat. ANONIZE: A large-scale anonymous survey system. In *IEEE S&P 2014*, 2014.
62. S. Hohenberger and B. Waters. Short and stateless signatures from the RSA assumption. In *Crypto*, 2009.
63. Y. Ishai, E. Kushilevitz, R. Ostrovksy, and A. Sahai. Zero-knowledge from secure multiparty computation. In *STOC*, 2007.
64. S. Jarecki, A. Kiayias, and H. Krawczyk. Round-optimal password-protected secret sharing and T-PAKE in the password-only model. In *Asiacrypt*, 2014.
65. S. Jarecki and X. Liu. Efficient oblivious pseudorandom function with applications to adaptive OT and secure computation of set intersection. In *TCC*, 2009.
66. M. Jawurek, F. Kerschbaum, and C. Orlandi. Zero-knowledge using garbled circuits: how to prove non-algebraic statements efficiently. In *ACM-CCS*, 2013.
67. A. Kawachi, K. Tanaka, and K. Xagawa. Concurrently secure identification schemes based on the worst-case hardness of lattice problems. In *Asiacrypt*, 2008.
68. F. Laguillaumie, A. Langlois, B. Libert, and D. Stehlé. Lattice-based group signatures with logarithmic signature size. In *Asiacrypt*, 2013.
69. A. Langlois, S. Ling, K. Nguyen, and H. Wang. Lattice-based group signature scheme with verifier-local revocation. In *PKC 2014*.
70. B. Libert, S. Ling, F. Mouhartem, K. Nguyen, and H. Wang. Zero-knowledge arguments for matrix-vector relations and lattice-based group encryption. In *Asiacrypt 2016*.
71. B. Libert, S. Ling, F. Mouhartem, K. Nguyen, and H. Wang. Signature schemes with efficient protocols and dynamic group signatures from lattice assumptions. In *Asiacrypt*, 2016.
72. B. Libert, S. Ling, K. Nguyen, and H. Wang. Zero-knowledge arguments for lattice-based accumulators: Logarithmic-size ring signatures and group signatures without trapdoors. In *Eurocrypt*, 2016.
73. S. Ling, K. Nguyen, D. Stehlé, and H. Wang. Improved zero-knowledge proofs of knowledge for the ISIS problem, and applications. In *PKC 2013*, 2013.
74. S. Ling, K. Nguyen, and H. Wang. Group signatures from lattices: Simpler, tighter, shorter, ring-based. In *PKC*. 2015.
75. H. Lipmaa, N. Asokan, and V. Niemi. Secure vickrey auctions without threshold trust. In *Financial Cryptography*, 2002.
76. V. Lyubashevsky. Lattice-Based Identification Schemes Secure Under Active Attacks. In *PKC*, 2008.
77. S. Micali, M. Rabin, and S. Vadhan. Verifiable random functions. In *FOCS*, 1999.
78. S. Micali and R. Sidney. A simple method for generating and sharing pseudo-random functions. In *Crypto*, 1995.
79. D. Micciancio and C. Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *Eurocrypt*, 2012.
80. D. Micciancio and S. Vadhan. Statistical zero-knowledge proofs with efficient provers: Lattice problems and more. In *Crypto*, 2003.
81. I. Miers, C. Garman, M. Green, and A. Rubin. Zerocoin: Anonymous distributed e-cash from bitcoin. In *IEEE S&P*, 2013.
82. S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. `www.bitcoin.org`.
83. M. Naor, B. Pinkas, and O. Reingold. Distributed pseudo-random functions and KDCs. In *Eurocrypt*, 1999.
84. P. Nguyen, J. Zhang, and Z. Zhang. Simpler efficient group signatures from lattices. In *PKC*, 2015.

85. K. Ohta and T. Okamoto. Universal electronic cash. In *Crypto*, 1991.
86. T. Okamoto. An efficient divisible electronic cash scheme. In *Crypto*, 1995.
87. C. Peikert and V. Vaikuntanathan. Non-interactive statistical zero-knowledge proofs for lattice problems. In *Crypto*, 2008.
88. O. Regev. On lattices, learning with errors, random linear codes, and cryptography. In *STOC*, 2005.
89. J. Stern. A new paradigm for public key identification. *Information Theory, IEEE Transactions on*, 42(6), 1996.
90. Y. Tsiounis. Efficient electronic cash: New notions and techniques. PhD thesis, Northeastern University, 1997.
91. S. von Solms and D. Naccache. On blind signatures and perfect crimes. *Computers & Security*, 11, 1992.
92. X. Xie, R. Xue, and M. Wang. Zero knowledge proofs from ring-LWE. In *CANS*, 2013.

A Security Definitions for Off-line E-Cash

The standard correctness condition mandates that honest users interacting with a honest bank in the withdrawal protocol always obtain the requested wallets. Moreover, honest merchants always accept coins from honest users and a honest bank always accepts faithfully generated coins.

Balance. This property ensures that no coalition of users can spend more coins than they withdrew. The adversary $\mathcal{A}$ is thus granted access to a withdrawal oracle which acts on behalf of the bank. To formalize the balance property, Camenisch *et al.* [23] require that part of the withdrawal protocol is a proof of knowledge of the serial numbers of the coins withdrawn by the user (or, at least, some information that determines them). In Definition 5, the $\mathcal{Q}_{\mathsf{withdraw}}$ oracle thus appeals to the knowledge extractor $\mathcal{E}^{\mathcal{A}}_{\mathsf{Withdraw}}$ of this proof of knowledge.

Definition 5. *An e-cash system provides the* **balance** *property if, for any PPT adversary $\mathcal{A}$ and any $L \in \mathbb{N}$ such that $2^L \in \mathsf{poly}(\lambda)$, there exists a knowledge extractor $\mathcal{E}^{\mathcal{A}}_{\mathsf{Withdraw}}$ for which $\mathcal{A}$'s winning probability in the game below is negligible:*

1. *The challenger generates public parameters $\mathsf{par} \leftarrow \mathsf{ParGen}(1^\lambda, 1^L)$ and a public key $(PK_{\mathcal{B}}, SK_{\mathcal{B}}) \leftarrow \mathsf{BKeyen}(\mathsf{par})$. It gives par and $PK_{\mathcal{B}}$ to $\mathcal{A}$ and initializes an empty set $\mathrm{DB}_{\mathcal{B}} \leftarrow \emptyset$.*
2. *The adversary $\mathcal{A}$ is interacts with the following oracles:*
   - *$\mathcal{Q}_{withdraw}\big(PK_{\mathcal{U}}, SK_{\mathcal{B}}\big)$: the oracle plays the role of the bank in an execution of the Withdraw protocol, on input $\big(\mathcal{A}(\mathtt{states}), \mathcal{B}(\mathsf{par}, PK_{\mathcal{U}}, SK_{\mathcal{B}})\big)$, in interaction with the adversary which acts as a cheating user and maintains a state $\mathtt{states}$. The oracle is allowed to invoked the knowledge extractor $\mathcal{E}^{\mathcal{A}}_{\mathsf{Withdraw}}$ so as to extract values by rewinding $\mathcal{A}$ to an earlier state. After each query, $\mathcal{Q}_{withdraw}$ stores the tracing information $\mathsf{T}_w = PK_{\mathcal{U}}$ (or $\mathsf{T}_w = \perp$ if the protocol fails) in a database T where it also stores the output of $\mathcal{E}^{\mathcal{A}}_{\mathsf{Withdraw}}$, which consists of a list of serial numbers $w = (S_1, \ldots, S_{2^L})$.*

- $\mathcal{Q}_{\mathsf{deposit}}\big(PK_{\mathcal{B}}, PK_{\mathcal{M}}, coin, \mathrm{DB}_{\mathcal{B}}\big)$*: the oracle plays the role of the bank and* $\mathcal{A}$ *plays the role of the merchant. On behalf of* $\mathcal{B}$*, the oracle thus outputs either "*accept*", "*user*" or "*$\mathsf{merchant}$*" and, in the first case, stores* $(PK_{\mathcal{M}}, coin)$*, which includes a serial number* S*, in the database* $\mathrm{DB}_{\mathcal{B}}$*.*

3. *After polynomially many queries,* $\mathcal{A}$ *halts and is declared successful if* $\mathrm{DB}_{\mathcal{B}}$ *contains a serial number that does not appear anywhere in the database* T*.*

Anonymity. The model of [23] adopts a simulation-based formulation of anonymity. No coalition of banks and merchants should distinguish a real execution of the Spend protocol from a simulated one, where the simulator is withheld access to users' wallets and secret keys. In the experiment, the adversary is allowed to introduce honest users in the system via a $\mathcal{Q}_{\mathsf{GetKey}}$ oracle and withdraw and spend coins via oracles $\mathcal{Q}_{\mathsf{withdraw}}, \mathcal{Q}_{\mathsf{Spend}}$, respectively.

Definition 6. *An e-cash system provides* **anonymity** *if there exists an efficient simulator* $\mathcal{S} = (\mathsf{SimParGen}, \mathsf{SimSpend})$ *such that no PPT adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *has non-negligible advantage in the following game.*

1. *The challenger flips a fair coin* $d \hookleftarrow U(\{0,1\})$*. If* $d = 1$*, it runs* $\mathsf{par} \leftarrow \mathsf{ParGen}(1^\lambda, 1^L)$ *whereas, if* $d = 0$*, it runs* $(\mathsf{par}, \tau_{sim}) \leftarrow \mathsf{SimParGen}(1^\lambda, 1^L)$*. In either case, it gives* par *to* $\mathcal{A}$ *and, if* $d = 0$*, it keeps* τ_{sim} *to itself.*
2. $\mathcal{A}$ *outputs a public key* $PK_{\mathcal{B}}$ *of its own and starts adaptively invoking the following oracles:*
   - $\mathcal{Q}_{\mathsf{GetKey}}(i)$*: If no public key has been created for user* $\mathcal{U}_i$*, the oracle generates* $(SK_{\mathcal{U}_i}, PK_{\mathcal{U}_i}) \leftarrow \mathsf{UKeygen}(\mathsf{par})$ *and returns* $PK_{\mathcal{U}_i}$*.*
   - $\mathcal{Q}_{\mathsf{withdraw}}(PK_{\mathcal{B}}, i)$*: given a bank's public key* $PK_{\mathcal{B}}$ *and a user index* i*, this oracle plays the role of user* $\mathcal{U}_j$ *– and creates the key pair* $(SK_{\mathcal{U}_i}, PK_{\mathcal{U}_i})$ *if it does not exist yet – in an execution of the withdrawal protocol* $\mathsf{Withdraw}\big(\mathcal{U}(\mathsf{par}, PK_{\mathcal{B}}, SK_{\mathcal{U}_i}), \mathcal{A}(\mathtt{states})\big)$*, while the adversary* $\mathcal{A}$ *assumes the role of the bank. At the* j*-th such query, we denote by* $\mathcal{W}_j$ *the user's output which may be a valid wallet or an error message if* $\mathcal{A}$ *did not correctly run the protocol on behalf of* $\mathcal{B}$*.*
   - $\mathcal{Q}_{\mathsf{Spend}}\big(PK_{\mathcal{B}}, i, j, PK_{\mathcal{M}}, \mathtt{info}\big)$*: the oracle first checks if the wallet* $\mathcal{W}_j$ *has been issued to* $\mathcal{U}_i$ *by the bank* $\mathcal{B}$ *via an invocation of* $\mathcal{Q}_{\mathsf{withdraw}}(PK_{\mathcal{B}}, i)$*. If not, the oracle outputs* $\perp$*. Otherwise,* $\mathcal{Q}_{\mathsf{Spend}}$ *checks if the internal counter* J *of* $\mathcal{W}_j$ *satisfies* $J < 2^\ell - 1$*. If* $J = 2^\ell - 1$*, it outputs* $\perp$*. Otherwise,* $\mathcal{Q}_{\mathsf{Spend}}$ *responds as follows:*
     - *If* $d = 1$*, it runs* $\mathsf{Spend}\big(\mathcal{U}_i(\mathcal{W}_j, PK_{\mathcal{B}}, PK_{\mathcal{M}}, \mathtt{info}), \mathcal{A}(\mathtt{states})\big)$ *in interaction with the merchant-executing* $\mathcal{A}$ *in order to spend a coin from* $\mathcal{W}_j$*. If the spending protocol fails, the oracle outputs* $\perp$*.*
     - *If* $d = 0$*,* $\mathcal{Q}_{\mathsf{Spend}}$ *runs* $\mathsf{SimSpend}\big(\mathsf{par}, \tau_{sim}, PK_{\mathcal{B}}, j, PK_{\mathcal{M}}, \mathtt{info}\big)$ *without using the user's wallet* $\mathcal{W}_j$ *or his public key.*
3. *When* $\mathcal{A}$ *halts, it outputs a bit* $d' \in \{0,1\}$ *and wins if* $d' = d$*. The adversary's advantage is the distance* $\mathbf{Adv}^{\mathrm{anon}}_{\mathcal{A}}(\lambda) := |\Pr[d' = d] - 1/2|$*, where the probability is taken over all coin tosses.*

IDENTIFICATION. This property requires that, given two fraudulent valid coins, the bank be able identify the double-spender. This property is defined via an experiment where the adversary $\mathcal{A}$ is the double-spender and has access to a $\mathcal{Q}_{\mathsf{withdraw}}$ oracle. Its goal is to deposit a coin twice while escaping identification. For simplicity, Definition 7 assumes that each coin explicitly contains the merchant's public key $PK_{\mathcal{M}}$ and the transaction meta data `info`.

Definition 7. *An e-cash system ensures the* **identification** *of double-spenders if, for any PPT adversary $\mathcal{A}$, this experiment outputs 0 with negligible probability:*

1. *The challenger generates public parameters* $\mathsf{par} \leftarrow \mathsf{ParGen}(1^\lambda, 1^L)$ *and a public key* $(PK_{\mathcal{B}}, SK_{\mathcal{B}}) \leftarrow \mathsf{BKeyen}(\mathsf{par})$. *It gives* par *and* $PK_{\mathcal{B}}$ *to* $\mathcal{A}$.
2. $\mathcal{A}$ *is granted access to the same* $\mathcal{Q}_{\mathsf{withdraw}}$ *oracle as in Definition 5. At each invocation, the oracle stores the tracing information* $\mathsf{T}_{\mathcal{W}} = PK_{\mathcal{U}}$ *(or* $\mathsf{T}_{\mathcal{W}} = \perp$ *if the interaction fails) about the issued wallet in a database* T.
3. *After polynomially many queries,* $\mathcal{A}$ *outputs two coins* $coin_1$, $coin_2$, *which are parsed as* $(S, \pi_1, PK_{\mathcal{M}_1}, \mathtt{info}_1)$, $(S, \pi_2, PK_{\mathcal{M}_2}, \mathtt{info}_2)$, *respectively. If* $(PK_{\mathcal{M}_1}, \mathtt{info}_1) = (PK_{\mathcal{M}_2}, \mathtt{info}_2)$, $\mathsf{VerifyCoin}(\mathsf{par}, PK_{\mathcal{M}}, PK_{\mathcal{B}}, coin_1) = 0$ *or* $\mathsf{VerifyCoin}(\mathsf{par}, PK_{\mathcal{M}}, PK_{\mathcal{B}}, coin_2) = 0$, *the experiment outputs* 0. *Otherwise, the experiment outputs* 1 *if and only if* $\mathsf{Identify}\big(\mathsf{par}, PK_{\mathcal{B}}, coin_1, coin_2\big)$ *does not output a public key* $PK_{\mathcal{U}}$ *that belongs to the database* T.

EXCULPABILITY. This property captures that no coalition of bank and merchants should be able to frame an honest user $\mathcal{U}$ by producing two coins $(coin_1, coin_2)$ such that $\mathsf{Identify}(\mathsf{par}, PK_{\mathcal{B}}, coin_1, coin_2) = PK_{\mathcal{U}}$ although $\mathcal{U}$ did not overspent. We define a game where the challenger plays the user and the adversary embodies the bank and the merchants. The adversary $\mathcal{A}$ has oracles $\mathcal{Q}_{\mathsf{GetKey}}, \mathcal{Q}_{\mathsf{withdraw}}, \mathcal{Q}_{\mathsf{Spend}}$ allowing to obtain users' keys, create wallets and spend coins.

Definition 8. *an e-cash system provides* **weak exculpability** *if no PPT adversary $\mathcal{A}$ has noticeable advantage in the following game.*

1. *The challenger runs* $\mathsf{par} \leftarrow \mathsf{ParGen}(1^\lambda, 1^L)$, *gives* par *to* $\mathcal{A}$ *and initializes a set* $\mathcal{HU}$, *which is initially empty.*
2. $\mathcal{A}$ *generates* $PK_{\mathcal{B}}$ *on behalf of the bank and interacts with these oracles:*
   - $\mathcal{Q}_{\mathsf{GetKey}}(i)$: *If no key pair has been created for* $\mathcal{U}_i$, $(SK_{\mathcal{U}_i}, PK_{\mathcal{U}_i}) \leftarrow \mathsf{UKeygen}(\mathsf{par})$ *is generated and the oracle returns* $PK_{\mathcal{U}_i}$, *which is added in* $\mathcal{HU}$ *of honest users.*
   - $\mathcal{Q}_{\mathsf{withdraw}}(PK_{\mathcal{B}}, i)$: *given* $PK_{\mathcal{B}}$ *and a user index* i, *this oracle plays the role of* $\mathcal{U}_j$ *– and creates* $(SK_{\mathcal{U}_i}, PK_{\mathcal{U}_i})$ *if it does not exist yet – in an execution of* $\mathsf{Withdraw}\big(\mathcal{U}(\mathsf{par}, PK_{\mathcal{B}}, SK_{\mathcal{U}_i}), \mathcal{A}(\mathtt{states})\big)$ *where* $\mathcal{A}$ *plays the role of the bank. At the* j*-th such query, we denote by* $\mathcal{W}_j$ *the user's output (which is kept secret from* $\mathcal{A}$*) that may be* $\perp$ *if* $\mathcal{A}$ *did not correctly run the protocol.*
   - $\mathcal{Q}_{\mathsf{Spend}}\big(PK_{\mathcal{B}}, i, j, PK_{\mathcal{M}}, \mathtt{info}\big)$: *the oracle first checks if the wallet* $\mathcal{W}_j$ *has been provided to* $\mathcal{U}_i$ *via an invocation of* $\mathcal{Q}_{\mathsf{withdraw}}(\mathsf{par}, PK_{\mathcal{B}}, i)$. *If not, the oracle outputs* $\perp$. *Otherwise,* $\mathcal{Q}_{\mathsf{Spend}}$ *checks if the internal counter* J *of* $\mathcal{W}_j$

satisfies $J < 2^\ell - 1$. If $J = 2^\ell - 1$, it outputs $\perp$. Otherwise, $\mathcal{Q}_{\mathsf{Spend}}$ spends a coin from $\mathcal{W}_j$ by running $\mathsf{Spend}\big(\mathcal{U}_i(\mathcal{W}_j, PK_{\mathcal{B}}, PK_{\mathcal{M}}, \mathtt{info}), \mathcal{A}(\mathtt{states})\big)$ in interaction with $\mathcal{A}$ which plays the merchant's side. If the protocol fails, $\mathcal{Q}_{\mathsf{Spend}}$ outputs $\perp$.

3. *When $\mathcal{A}$ halts, it outputs two coins $coin_1, coin_2$. The adversary wins if $\mathsf{Identify}(\mathsf{par}, PK_{\mathcal{B}}, coin_1, coin_2) \in \mathcal{HU}$ and its advantage is its probability of success taken over all random choices.*

Definition 8 only protects well-behaved users (who never double-spend) from being falsely accused. Camenisch *et al.* [23] defined a more stringent notion of **strong exculpability** where the $\mathcal{Q}_{\mathsf{Spend}}$ oracle may overspent for specific wallets. In this case, the adversary wins if it can produce $coin_1, coin_2$ that share the same serial number S, but S was not used more than once by $\mathcal{Q}_{\mathsf{Spend}}$. More precisely, let $\mathcal{W}_j$ be a wallet that reached the maximal state $J = 2^L - 1$ after $2^L - 1$ queries to the $\mathcal{Q}_{\mathsf{Spend}}$-oracle. If this wallet $\mathcal{W}_j$ is queried one more time (and thus overspends upon adversarial request), its state J is reset to $J = 0$. However, the serial number S of the illegally spent coin is added to a list of double-spendings $\mathsf{T}_{ds} := \mathsf{T}_{ds} \cup \{(i, j, S)\}$. The adversary wins if it can produce $coin_1, coin_2$, which share a common serial number S, such that $\mathsf{Identify}$ points to $PK_{\mathcal{U}_{i^\star}}$, but no entry $(i^\star, ., S)$ exists in T_{ds}.

B Signatures with Efficient Protocols from SIS

In [71], Libert, Ling, Mouhartem, Nguyen and Wang recently described a lattice-based signature scheme with efficient protocols.[5] Our e-cash system builds on a simplified variant of their scheme. The main difference is that, in our variant, we use a modified statistically hiding commitment where: (i) The matrix $\mathbf{D}_0$, which carries the randomness of the commitment, does not have the same number of columns as message-carrying matrices $\{\mathbf{D}_k\}_{k=1}^N$; (ii) The discrete Gaussian vector $\mathbf{r} \in \mathbb{Z}^{2ms}$ that serves as the randomness of the commitment $\mathbf{c}_{\mathfrak{m}} = \mathbf{D}_0 \cdot \mathbf{r} + \sum_{k=1}^N \mathbf{D}_k \cdot \mathfrak{m}_k \in \mathbb{Z}_{q_s}^n$ is split into two parts $\mathbf{r}_0, \mathbf{r}_1 \in \mathbb{Z}^{m_s}$.

Our scheme departs from [71] in the protocol for signing committed messages. As in many signatures with efficient protocols [26,27], the signer must re-randomize the user's commitment before generating a signature on it. While [71] lets the signer additively re-randomize the commitment $\mathbf{c}_{\mathfrak{m}}$ into $\mathbf{c}_{\mathfrak{m}} + \mathbf{D}_0 \cdot \mathbf{r}'$ mod q_s using a discrete Gaussian vector $\mathbf{r}' \in \mathbb{Z}^{m_s}$ of larger standard deviation than $\mathbf{r}$, we rather split the matrix $\mathbf{D}_0 \in \mathbb{Z}_{q_s}^{n \times 2m_s}$ into two parts $\mathbf{D}_0', \mathbf{D}_0'' \in \mathbb{Z}_{q_s}^{n \times m_s}$. In the oblivious signing protocol, when the signer receives the user's commitment $\mathbf{c}_{\mathfrak{m}} = \mathbf{D}_0' \cdot \mathbf{r}_0 + \sum_{k=1}^N \mathbf{D}_k \cdot \mathfrak{m}_k \in \mathbb{Z}_{q_s}^n$, he chooses a discrete Gaussian vector $\mathbf{r}_1 \hookleftarrow \mathcal{D}_{\mathbb{Z}^{m_s}, \sigma}$ of his own and computes $\mathbf{c}_{\mathfrak{m}} + \mathbf{D}_0'' \cdot \mathbf{r}_1$ mod q_s, so that the final signature contains the concatenation $\mathbf{r} = [\mathbf{r}_0^T \mid \mathbf{r}_1^T]^T$ instead of the sum $\mathbf{r}_0 + \mathbf{r}_1$.

[5] Note that the latter primitive differs from blind signatures in that it does not seek to provide *blindness*: i.e., the signer can possibly link an observed plain signature to the specific execution of the protocol where it was issued. For this reason, signatures with efficient protocols are always used in combination with zero-knowledge proofs.

This modification allows simplifying the security proof as we do not need to choose vectors $\mathbf{r}_0, \mathbf{r}_1 \in \mathbb{Z}^{m_s}$ of distinct standard deviations and neither do we need to encode messages in a special way. While [71] assumes that each message block $\mathfrak{m} \in \{0,1\}^N$ encodes a message of length $N/2$ where each bit is coded as $\bar{b}||b \in \{0,1\}^2$, our modification eliminates the need for such an encoding. The reason is that our security proof does not "program" the signer's contribution $\mathbf{r}_1 \in \mathbb{Z}^{m_s}$ to $\mathbf{r} \in \mathbb{Z}^{2m_s}$ in the same way.

Another difference w.r.t. [71] is that we choose the matrices $\{\mathbf{D}_k\}_{k=1}^N$ of smaller dimensions since we do not need to encode the message blocks with a two-fold expansion. We finally use the vector decomposition function $\mathsf{vdec}_{n,q_s-1} : \mathbb{Z}_{q_s}^n \to \{0,1\}^{n\lceil \log q_s \rceil}$. instead of the standard binary decomposition.

Our modified scheme goes as follows.

ParGen$(1^\lambda, 1^N)$**:** Given a security parameter $\lambda > 0$ and the number of blocks $N = \mathsf{poly}(\lambda)$, choose the following parameters: $n = \mathcal{O}(\lambda)$; a prime modulus $q_s = \widetilde{\mathcal{O}}(N \cdot n^4)$; dimension $m_s = 2n\lceil \log q_s \rceil$; an integer $\ell = \Theta(\lambda)$; and Gaussian parameters $\sigma = \Omega(\sqrt{n \log q} \log n)$. Define the message space as $(\{0,1\}^{\bar{m}})^N$, for some $\bar{m} \in \mathsf{poly}(n)$. Then, pick uniformly random matrices $\mathbf{D}_0 = [\mathbf{D}_0' | \mathbf{D}_0''] \hookleftarrow U(\mathbb{Z}_{q_s}^{n \times 2m_s})$ and $\mathbf{D}_1, \ldots, \mathbf{D}_N \hookleftarrow U(\mathbb{Z}_{q_s}^{n \times \bar{m}})$, which form a commitment key $CK := \{\mathbf{D}_k\}_{k=0}^N$. Output

$$\mathsf{par} := \Big(n, m_s, \ell, N, q_s, CK\Big).$$

Keygen$(1^\lambda, \mathsf{par})$**:** To generate a key pair (SK, PK) conduct the following steps.

1. Run $\mathsf{TrapGen}(1^n, 1^{m_s}, q_s)$ to get $\mathbf{A} \in \mathbb{Z}_{q_s}^{n \times m_s}$ and a short basis $\mathbf{T}_\mathbf{A}$ of $\Lambda_{q_s}^\perp(\mathbf{A})$. This basis makes it possible to sample short vectors in $\Lambda_{q_s}^\perp(\mathbf{A})$ with a Gaussian parameter σ.
2. Choose random $\mathbf{A}_0, \mathbf{A}_1, \ldots, \mathbf{A}_\ell \hookleftarrow U(\mathbb{Z}_{q_s}^{n \times m_s})$ and $\mathbf{D} \hookleftarrow U(\mathbb{Z}_{q_s}^{n \times (m_s/2)})$. Also, choose a random vector $\mathbf{u} \hookleftarrow U(\mathbb{Z}_{q_s}^n)$.

The key pair is comprised of $SK := \mathbf{T}_\mathbf{A}$ and

$$PK := \big(\mathbf{A},\ \{\mathbf{A}_j\}_{j=0}^\ell,\ \mathbf{D},\ \mathbf{u}\big).$$

Sign$\big(SK, \mathsf{Msg}\big)$**:** To sign an N-block message $\mathsf{Msg} = (\mathfrak{m}_1, \ldots, \mathfrak{m}_N) \in (\{0,1\}^{\bar{m}})^N$,

1. Pick a random string $\tau \hookleftarrow U(\{0,1\}^\ell)$. Using $SK := \mathbf{T}_\mathbf{A}$, compute a short delegated basis $\mathbf{T}_\tau \in \mathbb{Z}^{2m_s \times 2m_s}$ for the matrix

$$\mathbf{A}_\tau = [\mathbf{A} \mid \mathbf{A}_0 + \sum_{j=1}^{\ell} \tau[j] \cdot \mathbf{A}_j] \in \mathbb{Z}_{q_s}^{n \times 2m_s}. \tag{37}$$

2. Sample a discrete Gaussian vector $\mathbf{r} = [\mathbf{r}_0^T \mid \mathbf{r}_1^T]^T \hookleftarrow D_{\mathbb{Z}^{2m_s},\sigma}$. Compute $\mathbf{c}_M \in \mathbb{Z}_{q_s}^n$ as a chameleon hash of $(\mathfrak{m}_1, \ldots, \mathfrak{m}_N)$. Namely, compute

$$\begin{aligned}\mathbf{c}_M &= \mathbf{D}_0 \cdot \mathbf{r} + \sum_{k=1}^{N} \mathbf{D}_k \cdot \mathfrak{m}_k \mod q_s \\ &= \mathbf{D}'_0 \cdot \mathbf{r}_0 + \mathbf{D}''_0 \cdot \mathbf{r}_1 + \sum_{k=1}^{N} \mathbf{D}_k \cdot \mathfrak{m}_k \in \mathbb{Z}_{q_s}^n,\end{aligned}$$

which allows defining $\mathbf{u}_M = \mathbf{u} + \mathbf{D} \cdot \mathsf{vdec}(\mathbf{c}_M) \in \mathbb{Z}_{q_s}^n$. Then, using the delegated basis $\mathbf{T}_\tau \in \mathbb{Z}^{2m_s \times 2m_s}$, sample $\mathbf{v} = \begin{bmatrix}\mathbf{v}_1 \\ \mathbf{v}_2\end{bmatrix} \in \mathbb{Z}^{2m_s}$ in $D_{\Lambda_{q_s}^{\mathbf{u}_M}(\mathbf{A}_\tau),\sigma}$.

Output the signature $sig = (\tau, \mathbf{v}, \mathbf{r}) \in \{0,1\}^\ell \times \mathbb{Z}^{2m_s} \times \mathbb{Z}^{2m_s}$.

Verify$\big(PK, \mathsf{Msg}, sig\big)$**:** Given PK, a message $\mathsf{Msg} = (\mathfrak{m}_1, \ldots, \mathfrak{m}_N) \in (\{0,1\}^{\bar{m}})^N$ and a purported signature $sig = (\tau, \mathbf{v}, \mathbf{r}) \in \{0,1\}^\ell \times \mathbb{Z}^{2m_s} \times \mathbb{Z}^{2m_s}$, return 1 if

$$\mathbf{A}_\tau \cdot \mathbf{v} = \mathbf{u} + \mathbf{D} \cdot \mathsf{vdec}_{n,q_s-1}\big(\mathbf{D}_0 \cdot \mathbf{r} + \sum_{k=1}^{N} \mathbf{D}_k \cdot \mathfrak{m}_k\big) \bmod q_s.$$

and $\|\mathbf{v}\| < \sigma\sqrt{2m_s}$, $\|\mathbf{r}\| < \sigma\sqrt{2m_s}$.

The basic signature scheme is extended with the following protocol which allows a user to obtain a signature on a committed message.

Protocol: The signer S interacts with the user $\mathcal{U}$ in the following interactive protocol which allows $\mathcal{U}$ to obtain a signature on the message $(\mathfrak{m}_1, \ldots, \mathfrak{m}_N)$.

1. $\mathcal{U}$ samples $\mathbf{r}_0 \hookleftarrow D_{\mathbb{Z}^{m_s},\sigma}$ and computes $\mathbf{c}_\mathfrak{m} = \mathbf{D}'_0 \cdot \mathbf{r}_0 + \sum_{k=1}^{N} \mathbf{D}_k \cdot \mathfrak{m}_k \in \mathbb{Z}_{q_s}^n$. Then, $\mathcal{U}$ sends $\mathbf{c}_\mathfrak{m}$ to S and generates an interactive zero-knowledge argument of knowledge of $(\mathfrak{m}_1, \ldots, \mathfrak{m}_N, \mathbf{r}_0) \in (\{0,1\}^{\bar{m}})^N \times \mathbb{Z}^{m_s}$ which open the statistically hiding commitment $\mathbf{c}_\mathfrak{m}$.
2. If the argument of step 1 verifies, S samples $\mathbf{r}_1 \hookleftarrow D_{\mathbb{Z}^{m_s},\sigma}$ and computes a vector $\mathbf{u}_\mathfrak{m} = \mathbf{u} + \mathbf{D} \cdot \mathsf{vdec}_{n,q_s-1}\big(\mathbf{c}_\mathfrak{m} + \mathbf{D}''_0 \cdot \mathbf{r}_1\big) \in \mathbb{Z}_{q_s}^n$. Then, S picks $\tau \hookleftarrow \{0,1\}^\ell$ and uses $\mathbf{T}_\mathbf{A}$ to compute a delegated basis $\mathbf{T}_\tau \in \mathbb{Z}^{2m_s \times 2m_s}$ for the matrix $\mathbf{A}_\tau \in \mathbb{Z}_{q_s}^{n \times 2m_s}$ of (37). Using $\mathbf{T}_\tau \in \mathbb{Z}^{2m_s \times 2m_s}$, S samples a short vector $\mathbf{v} \in \mathbb{Z}^{2m}$ in $D^{\mathbf{u}_M}_{\Lambda_{q_s}^\perp(\mathbf{A}_\tau),\sigma}$. It returns $(\tau, \mathbf{v}, \mathbf{r}_1) \in \{0,1\}^\ell \times \mathbb{Z}^{2m_s} \times \mathbb{Z}^{m_s}$ to $\mathcal{U}$.
3. $\mathcal{U}$ defines $\mathbf{r} = \begin{bmatrix}\mathbf{r}_0 \\ \mathbf{r}_1\end{bmatrix} \in \mathbb{Z}^{2m_s}$ and verifies that

$$\mathbf{A}_\tau \cdot \mathbf{v} = \mathbf{u} + \mathbf{D} \cdot \mathsf{vdec}_{n,q_s-1}\big(\mathbf{D}_0 \cdot \mathbf{r} + \sum_{k=1}^{N} \mathbf{D}_k \cdot \mathfrak{m}_k\big) \bmod q_s.$$

In this case, it outputs $(\tau, \mathbf{v}, \mathbf{r}) \in \{0,1\}^\ell \times \mathbb{Z}^{2m_s} \times \mathbb{Z}^{2m_s}$. Otherwise, it outputs $\perp$.

In order to prove possession of a message-signature pair, the user can use the same technique as [71], which extend [73,74].

Our security proof builds on the same strategy as [71]. At each oblivious signing query, the reduction can use the knowledge extractor of the proof system to extract the opening $(\mathfrak{m}_1, \ldots, \mathfrak{m}_N, \mathbf{r}_0)$ of $\mathbf{c}_{\mathfrak{m}}$ at step 1 of the signing protocol. When the adversary proves knowledge of a fake message-signature pair $\big((\mathfrak{m}_1^\star, \ldots, \mathfrak{m}_N^\star), (\tau^\star, \mathbf{v}^\star, \mathbf{r}^\star)\big)$, the reduction appeals to the corresponding knowledge extractor so as to obtain the underlying witnesses. At this point, three cases can be distinguished.

• If $\tau^\star$ was not used in any signing query, the reduction can solve SIS in the same way as in the security proof of Boyen's signature [18].

• If $\tau^\star$ was used in some signing query, the reduction can predict with probability $1/Q$ the index $i^\dagger \in [1, Q]$ of the query for which the adversary recycles $\tau^\star = \tau^{(i^\dagger)}$. At this point, if the reduction expects that

$$\mathbf{D}_0 \cdot \mathbf{r}^\star + \sum_{k=1}^{N} \mathbf{D}_k \cdot \mathfrak{m}_k^\star \neq \mathbf{D}_0 \cdot \mathbf{r}^{(i^\dagger)} + \sum_{k=1}^{N} \mathbf{D}_k \cdot \mathfrak{m}_k^{(i^\dagger)} \bmod q_s, \tag{38}$$

it can solve SIS as in the proof of Böhl *et al.* [16]. It sets up $\{\mathbf{A}, \{\mathbf{A}_j\}_{j=0}^{\ell}\}$ so that the matrix $\mathbf{A}_{\tau^{(i^\dagger)}}$ of (37) does not depend on any GPV trapdoor at the $i^\dagger$-th signing query. It also runs the the trapdoor generation algorithm to obtain a statistically uniform $\mathbf{D}_0'' \in \mathbb{Z}_{q_s}^{n \times m_s}$ and a short basis $\mathbf{T}_{\mathbf{D}_0''}$ of $\Lambda_{q_s}^{\perp}(\mathbf{D}_0'')$. To generate PK, it also samples $\mathbf{v} \hookleftarrow D_{\mathbb{Z}^{2m_s}, \sigma}$, $\mathbf{c}_M \hookleftarrow U(\mathbb{Z}_{q_s}^n)$ and "programs" $\mathbf{u} = \mathbf{A}_{\tau^{(i^\dagger)}} \cdot \mathbf{v} - \mathbf{D} \cdot \mathsf{vdec}_{n, q_s - 1}(\mathbf{c}_M) \in \mathbb{Z}_{q_s}^n$ in the setup phase. When the $i^\dagger$-th query occurs, the reduction receives $\mathbf{c}_{\mathfrak{m}}^{(i^\dagger)}$ and uses $\mathbf{T}_{\mathbf{D}_0''}$ to compute a short $\mathbf{r}_1^{(i^\dagger)} \in \mathbb{Z}^{m_s}$ such that

$$\mathbf{c}_M = \mathbf{c}_{\mathfrak{m}}^{(i^\dagger)} + \mathbf{D}_0'' \cdot \mathbf{r}_1^{(i^\dagger)} \in \mathbb{Z}_{q_s}^n,$$

which yields a valid response $(\tau^{(i^\dagger)}, \mathbf{v}, \mathbf{r}_1^{(i^\dagger)})$ at the $i^\dagger$-th query since

$$\mathbf{A}_{\tau^{(i^\dagger)}} \cdot \mathbf{v} = \mathbf{u} + \mathsf{vdec}_{n, q_s - 1}\big(\mathbf{c}_{\mathfrak{m}}^{(i^\dagger)} + \mathbf{D}_0'' \cdot \mathbf{r}_1^{(i^\dagger)}\big).$$

If the adversary eventually produces a forgery that recycles $\tau^{(i^\dagger)}$ and for which the inequality (38) holds, it can solve SIS as in [16], by using the vector decomposition of $\mathbf{D}_0 \cdot \mathbf{r}^\star + \sum_{k=1}^{N} \mathbf{D}_k \cdot \mathfrak{m}_k^\star$ as the forgery message of [16].

• If $\tau^\star$ was used in some signing query and the reduction expects to have the collision

$$\mathbf{D}_0 \cdot \mathbf{r}^\star + \sum_{k=1}^{N} \mathbf{D}_k \cdot \mathfrak{m}_k^\star = \mathbf{D}_0 \cdot \mathbf{r}^{(i^\dagger)} + \sum_{k=1}^{N} \mathbf{D}_k \cdot \mathfrak{m}_k^{(i^\dagger)} \bmod q_s,$$

it can break SIS by breaking the binding property in the statistically hiding commitment of Kawachi *et al.* [67].

Our security proof thus differs from [71] at the $i^\dagger$-th query when the reduction expects situation 2. Instead of additively "programming" the integer vector $\mathbf{r}_1^{(i\dagger)} \in \mathbb{Z}^{m_s}$ to make the sum $\mathbf{r}_0^{(i^\dagger)} + \mathbf{r}_1^{(i^\dagger)}$ hit a predetermined value, we use the trapdoor $\mathbf{T}_{\mathbf{D}_0''}$ to find a valid $\mathbf{r}_1^{(i\dagger)}$.

The detailed security proof is incorporated into the proof of the balance property of our e-cash system.

C Deferred Details of Our Zero-Knowledge Protocols

C.1 Analysis of the Abstract Protocol

We prove Theorem 1 using standard simulation and extraction techniques for Stern-like protocols [67,73,72,71].

Proof. It can be checked that the protocol has perfect completeness: If an honest prover follows the protocol, then he always gets accepted by the verifier. It is also easy to see that the communication cost is bounded by $\mathcal{O}\big(\sum_{i=1}^{N} d_i \cdot \log q_i\big)$. We now prove that the protocol is a statistical zero-knowledge argument of knowledge.

Zero-Knowledge Property. We will construct a PPT simulator SIM interacting with a (possibly dishonest) verifier $\widehat{\mathcal{V}}$, such that, given the public input $\{\mathbf{M}_i \in \mathbb{Z}_{q_i}^{n_i \times d_i}\}_{i\in[N]}$ and $\mathbf{u}_i \in \mathbb{Z}_{q_i}^{n_i}$, it outputs with probability negligibly close to 2/3 a simulated transcript that is statistically close to the one produced by the honest prover in the real interaction.

The simulator first chooses a random $\overline{Ch} \in \{1,2,3\}$ as a prediction of the challenge value that $\widehat{\mathcal{V}}$ will *not* choose.

Case $\overline{Ch} = 1$: For every $i \in [N]$, the simulator uses basic linear algebra over $\mathbb{Z}_{q_i}$ to compute vector $\mathbf{w}_i' \in \mathbb{Z}_{q_i}^{d_i}$ such that $\mathbf{M}_i \cdot \mathbf{w}_i' = \mathbf{u}_i \bmod q_i$. Let $\mathbf{w}' = (\mathbf{w}_1' \| \ldots \| \mathbf{w}_N')$.

Next, it samples $\phi \leftarrow U(\mathcal{S})$, $\mathbf{r}_1 \leftarrow U(\mathbb{Z}_{q_1}^{d_1}), \ldots, \mathbf{r}_N \leftarrow U(\mathbb{Z}_{q_N}^{d_N})$, and computes $\mathbf{r} = (\mathbf{r}_1 \| \ldots \| \mathbf{r}_N)$, $\mathbf{z}' = \mathbf{w}' \boxplus \mathbf{r}$. Then, it samples randomness ρ_1, ρ_2, ρ_3 for COM and sends the commitment $\mathrm{CMT} = \big(C_1', C_2', C_3'\big)$ to $\widehat{\mathcal{V}}$, where

$$C_1' = \mathsf{COM}(\phi, \{\mathbf{M}_i \cdot \mathbf{r}_i \bmod q_i\}_{i\in[N]};\, \rho_1),$$

$$C_2' = \mathsf{COM}(\Gamma_\phi(\mathbf{r});\, \rho_2), \quad C_3' = \mathsf{COM}(\Gamma_\phi(\mathbf{z}');\, \rho_3).$$

Receiving a challenge Ch from $\widehat{\mathcal{V}}$, the simulator responds as follows:

- If $Ch = 1$: Output $\perp$ and abort.
- If $Ch = 2$: Send $\mathrm{RSP} = \big(\phi, \mathbf{z}', \rho_1, \rho_3\big)$.
- If $Ch = 3$: Send $\mathrm{RSP} = \big(\phi, \mathbf{r}, \rho_1, \rho_2\big)$.

Case $\overline{Ch} = 2$: SIM samples $\mathbf{w}' \leftarrow U(\mathsf{VALID})$, together with $\phi \leftarrow U(\mathcal{S})$, and $\mathbf{r}_1 \leftarrow U(\mathbb{Z}_{q_1}^{d_1}), \ldots, \mathbf{r}_N \leftarrow U(\mathbb{Z}_{q_N}^{d_N})$. It computes $\mathbf{r} = (\mathbf{r}_1 \| \ldots \| \mathbf{r}_N)$, $\mathbf{z}' = \mathbf{w}' \boxplus \mathbf{r}$. Then, it samples randomness ρ_1, ρ_2, ρ_3 for COM and sends the commitment $\mathrm{CMT} = \big(C_1', C_2', C_3'\big)$ to $\widehat{\mathcal{V}}$, where

$$C_1' = \mathsf{COM}(\phi, \{\mathbf{M}_i \cdot \mathbf{r}_i \bmod q_i\}_{i \in [N]}; \rho_1),$$

$$C_2' = \mathsf{COM}(\Gamma_\phi(\mathbf{r}); \rho_2), \quad C_3' = \mathsf{COM}(\Gamma_\phi(\mathbf{z}'); \rho_3).$$

Receiving a challenge Ch from $\widehat{\mathcal{V}}$, the simulator responds as follows:

- If $Ch = 1$: Send $\mathrm{RSP} = \big(\Gamma_\phi(\mathbf{w}'), \Gamma_\phi(\mathbf{r}), \rho_2, \rho_3\big)$.
- If $Ch = 2$: Output $\perp$ and abort.
- If $Ch = 3$: Send $\mathrm{RSP} = \big(\phi, \mathbf{r}, \rho_1, \rho_2\big)$.

Case $\overline{Ch} = 3$: SIM samples $\mathbf{w}' \leftarrow U(\mathsf{VALID})$ and parse it as $\mathbf{w}' = (\mathbf{w}_1' \| \ldots \| \mathbf{w}_N')$, where $\mathbf{w}_i' \in \mathbb{Z}^{d_i}$ for all $i \in [N]$.

Next, it picks $\phi \leftarrow U(\mathcal{S})$, $\mathbf{r}_1 \leftarrow U(\mathbb{Z}_{q_1}^{d_1}), \ldots, \mathbf{r}_N \leftarrow U(\mathbb{Z}_{q_N}^{d_N})$, and computes $\mathbf{r} = (\mathbf{r}_1 \| \ldots \| \mathbf{r}_N)$, $\mathbf{z}' = \mathbf{w}' \boxplus \mathbf{r}$. Then, it samples randomness ρ_1, ρ_2, ρ_3 for COM and sends the commitment $\mathrm{CMT} = \big(C_1', C_2', C_3'\big)$ to $\widehat{\mathcal{V}}$, where

$$C_2' = \mathsf{COM}(\Gamma_\phi(\mathbf{r}); \rho_2), \quad C_3' = \mathsf{COM}(\Gamma_\phi(\mathbf{z}'); \rho_3),$$

as in the previous two cases, while

$$C_1' = \mathsf{COM}(\phi, \{\mathbf{M}_i \cdot (\mathbf{w}_i' + \mathbf{r}_i) - \mathbf{u}_i \bmod q_i\}_{i \in [N]}; \rho_1).$$

Receiving a challenge Ch from $\widehat{\mathcal{V}}$, it responds as follows:

- If $Ch = 1$: Send RSP computed as in the case $(\overline{Ch} = 2, Ch = 1)$.
- If $Ch = 2$: Send RSP computed as in the case $(\overline{Ch} = 1, Ch = 2)$.
- If $Ch = 3$: Output $\perp$ and abort.

We observe that, in every case we have considered above, since COM is statistically hiding, the distribution of the commitment CMT and the distribution of the challenge Ch from $\widehat{\mathcal{V}}$ are statistically close to those in the real interaction. Hence, the probability that the simulator outputs $\perp$ is negligibly close to $1/3$. Moreover, one can check that whenever the simulator does not halt, it will provide an accepted transcript, the distribution of which is statistically close to that of the prover in the real interaction. In other words, we have constructed a simulator that can successfully impersonate the honest prover with probability negligibly close to $2/3$.

Argument of Knowledge. Suppose that

$$\mathrm{RSP}_1 = (\mathbf{t}, \mathbf{s}, \rho_2^{(1)}, \rho_3^{(1)}), \quad \mathrm{RSP}_2 = (\pi, \mathbf{x}, \rho_1^{(2)}, \rho_3^{(2)}), \quad \mathrm{RSP}_3 = (\psi, \mathbf{y}, \rho_1^{(3)}, \rho_2^{(3)})$$

are 3 valid responses to the same commitment $\mathrm{CMT} = (C_1, C_2, C_3)$, with respect to all 3 possible values of the challenge. Let us parse $\mathbf{x}$ and $\mathbf{y}$ as $\mathbf{x} = (\mathbf{x}_1 \| \ldots \| \mathbf{x}_N)$, $\mathbf{y} = (\mathbf{y}_1 \| \ldots \| \mathbf{y}_N)$, where $\mathbf{x}_i, \mathbf{y}_i \in \mathbb{Z}_{q_i}^{d_i}$ for all $i \in [N]$.

The validity of the given responses implies that:

$$\begin{cases} \mathbf{t} \in \mathsf{VALID}; \\ C_1 = \mathsf{COM}(\pi, \{\mathbf{M}_i \cdot \mathbf{x}_i - \mathbf{u}_i \bmod q_i\}_{i\in[N]}; \rho_1^{(2)}); \\ C_1 = \mathsf{COM}(\psi, \{\mathbf{M}_i \cdot \mathbf{y}_i \bmod q_i\}_{i\in[N]}; \rho_1^{(3)}); \\ C_2 = \mathsf{COM}(\mathbf{s}; \rho_2^{(1)}) = \mathsf{COM}(\Gamma_\psi(\mathbf{y}); \rho_2^3); \\ C_3 = \mathsf{COM}(\mathbf{t} \boxplus \mathbf{s}; \rho_3^{(1)}) = \mathsf{COM}(\Gamma_\pi(\mathbf{x}); \rho_3^{(2)}). \end{cases}$$

Since COM is computationally binding, we can deduce that:

$$\mathbf{t} \in \mathsf{VALID}; \quad \pi = \psi; \quad \mathbf{s} = \Gamma_\psi(\mathbf{y}); \quad \mathbf{t} \boxplus \mathbf{s} = \Gamma_\pi(\mathbf{x});$$
$$\forall i \in [N] : \mathbf{M}_i \cdot \mathbf{x}_i - \mathbf{u}_i = \mathbf{M}_i \cdot \mathbf{y}_i \bmod q_i.$$

Let $\mathbf{w}' = [\Gamma_\psi]^{-1}(\mathbf{t})$. Since $\mathbf{t} \in \mathsf{VALID}$, we have $\mathbf{w}' \in \mathsf{VALID}$. Furthermore, note that $\Gamma_\psi(\mathbf{w}') \boxplus \Gamma_\psi(\mathbf{y}) = \Gamma_\psi(\mathbf{x})$, which implies that $\mathbf{w}' \boxplus \mathbf{y} = \mathbf{x}$.

Now, parse $\mathbf{w}'$ as $\mathbf{w}' = (\mathbf{w}'_1 \| \dots \| \mathbf{w}'_N)$, where $\mathbf{w}'_i \in \mathbb{Z}^{d_i}$, for all $i \in [N]$. Then, for all $i \in [N]$, we have $\mathbf{w}'_i + \mathbf{y}_i = \mathbf{x}_i \bmod q$, and that

$$\mathbf{M}_i \cdot \mathbf{w}'_i = \mathbf{M}_i \cdot \mathbf{x}_i - \mathbf{M}_i \cdot \mathbf{y}_i = \mathbf{u}_i \bmod q_i.$$

As a result, we have $\mathbf{w}' \in \mathsf{VALID}$ and $\mathbf{M}_i \cdot \mathbf{w}'_i = \mathbf{u}_i$, for all $i \in [N]$. This concludes the proof. □

C.2 Knowledge Extraction of the Protocol for the BLMR PRF

We provide more details on the knowledge extractor of the zero-knowledge argument of Section 4.3. It first runs the knowledge extractor of the abstract protocol to obtain a witness $\mathbf{w}' = (\mathbf{w}'_1 \| \mathbf{w}'_2 \| \mathbf{w}'_3) \in \mathsf{VALID}$ satisfying

$$\{\mathbf{M}_i \cdot \mathbf{w}'_i = \mathbf{u}_i \bmod q_i\}_{i=1,2,3}.$$

Since $\mathbf{w}' \in \mathsf{VALID}$, vectors $\mathbf{w}'_1, \mathbf{w}'_2, \mathbf{w}'_3$ can be parsed as

$$\mathbf{w}'_1 = \left(\breve{\mathbf{r}}' \| \widehat{\mathbf{x}}'_0\right); \quad \mathbf{w}'_2 = \left(\mathbf{s}'_0 \| \widehat{\mathbf{x}}'_1 \| \cdots \| \mathbf{s}'_{L-1} \| \widehat{\mathbf{x}}'_L\right); \quad \mathbf{w}'_3 = \left(\widehat{\mathbf{x}}'_L \| \widehat{\mathbf{z}}'\right),$$

where

- $\breve{\mathbf{r}}' \in \mathsf{B}^3_{m_0\delta_\beta}$, and $\widehat{\mathbf{x}}'_0, \dots, \widehat{\mathbf{x}}'_L, \widehat{\mathbf{z}}' \in \mathsf{B}^2_{\bar{m}}$.
- $\{\mathbf{s}'_{i-1} = \mathsf{expand}(J'[i], \widehat{\mathbf{x}}'_{i-1})\}_{i=1}^L$, for $J'[1] \dots J'[L] \in \{0,1\}^L$.

Next, the knowledge extractor backtracks the transformations conducted in Section 4.3 as follows.

Equation $\mathbf{M}_3 \cdot \mathbf{w}'_3 = \mathbf{u}_3 \bmod q_3$ is equivalent to

$$(p \cdot \widehat{\mathbf{H}}_{m,q-1}) \cdot \widehat{\mathbf{x}}'_L - \widehat{\mathbf{H}}_{m,q-1} \cdot \widehat{\mathbf{z}}' = q \cdot \mathbf{y} \bmod pq.$$

Letting $\mathbf{x}'_L = \widehat{\mathbf{H}}_{m,q-1} \cdot \widehat{\mathbf{x}}'_L$ and $\mathbf{z}' = \widehat{\mathbf{H}}_{m,q-1} \cdot \widehat{\mathbf{z}}'$, we have $\mathbf{x}'_L, \mathbf{z}' \in \mathbb{Z}_q^m$ and $p \cdot \mathbf{x}' = q \cdot \mathbf{y} + \mathbf{z} \bmod pq$. This implies $\lfloor \mathbf{x}'_L \rfloor_p = \mathbf{y} \in \mathbb{Z}_p^m$.

Equation $\mathbf{M}_2 \cdot \mathbf{w}'_2 = \mathbf{u}_2 \bmod q_2$ is equivalent to

$$\begin{cases} \mathbf{P} \cdot \mathbf{s}'_0 - \widehat{\mathbf{H}}_{m,q-1} \cdot \widehat{\mathbf{x}}'_1 = \mathbf{0} \bmod q, \\ \qquad\qquad \vdots \\ \mathbf{P} \cdot \mathbf{s}'_{L-1} - \widehat{\mathbf{H}}_{m,q-1} \cdot \widehat{\mathbf{x}}'_L = \mathbf{0} \bmod q, \end{cases}$$

which can be written as

$$\widehat{\mathbf{H}}_{m,q-1} \cdot \widehat{\mathbf{x}}'_i = \left[\mathbf{P}_0 \cdot \widehat{\mathbf{H}}_{m,q-1} \,|\, \mathbf{P}_1 \cdot \widehat{\mathbf{H}}_{m,q-1} \right] \cdot \mathsf{expand}(J'[i], \widehat{\mathbf{x}}'_{i-1}) \bmod q \qquad \forall i \in [L].$$

For each $i \in \{0, \ldots, L\}$, let $\mathbf{x}'_i = \widehat{\mathbf{H}}_{m,q-1} \cdot \widehat{\mathbf{x}}'_i \in \mathbb{Z}_q^m$. Then, for all $i \in \{1, \ldots, L\}$, we have:

$$\mathbf{x}'_i = \mathbf{P}_0 \cdot (\overline{J'[i]} \cdot \mathbf{x}'_{i-1}) + \mathbf{P}_1 \cdot (J'[i] \cdot \mathbf{x}'_{i-1}) = \left[\mathbf{P}_0 \,|\, \mathbf{P}_1 \right] \cdot \begin{pmatrix} \overline{J'[i]} \cdot \mathbf{x}'_{i-1} \\ J'[i] \cdot \mathbf{x}'_{i-1} \end{pmatrix} \bmod q,$$

which means that $\mathbf{x}'_i = \mathbf{P}_{J'[i]} \cdot \mathbf{x}'_{i-1} \bmod q$.

At this point, the knowledge extractor sets $\mathbf{k}' = \mathbf{x}'_0$. Then $\mathbf{k}'$ and $J'[1] \ldots J'[L]$ satisfy

$$\mathbf{y} = \Big\lfloor \prod_{i=1}^{L} \mathbf{P}_{J'[L+1-i]} \cdot \mathbf{k}' \Big\rfloor_p.$$

The equation $\mathbf{M}_1 \cdot \mathbf{w}'_1 = \mathbf{u}_1 \bmod q_1$ is equivalent to:

$$[\mathbf{D}_0 \cdot \breve{\mathbf{H}}_{m_0,\beta}] \cdot \breve{\mathbf{r}}' + [\mathbf{D}_1 \mid \mathbf{0}^{n \times \bar{m}}] \cdot \widehat{\mathbf{x}}'_0 = \mathbf{c} \bmod q_s.$$

Let $\mathbf{r}' = \breve{\mathbf{H}}_{m_0,\beta} \cdot \breve{\mathbf{r}} \in [-\beta, \beta]^{m_0}$, and drop the last $\bar{m}$ coordinates from $\widehat{\mathbf{x}}'_0$ to obtain vector $\tilde{\mathbf{k}}' \in \{0,1\}^{\bar{m}}$. Note that we have $\mathbf{H}_{m,q-1} \cdot \tilde{\mathbf{k}}' = \widehat{\mathbf{H}}_{m,q-1} \cdot \widehat{\mathbf{x}}'_0 = \mathbf{k}'$, and:

$$\mathbf{c} = \mathbf{D}_0 \cdot \mathbf{r}' + \mathbf{D}_1 \cdot \mathbf{k}' \bmod q_s.$$

Finally, the knowledge extractor outputs $\mathbf{k}' \in \mathbb{Z}_q^m$, $J'[1] \ldots J'[L] \in \{0,1\}^L$ and $\mathbf{r}' \in [-\beta, \beta]^{m_0}$, $\tilde{\mathbf{k}}' \in \{0,1\}^{\bar{m}}$, satisfying the considered statement.

C.3 Proving the Correct Evaluation of GGM-Based PRF

We now consider the problem of proving the correct evaluation of an LWR-based PRF based on the GGM construction [56]. Recall that the latter uses a length-doubling pseudo-random generator $G : \{0,1\}^n \to \{0,1\}^{2n}$ to build a PRF that, for any seed $s \in \{0,1\}^n$ and input $J = J[1] \ldots J[L] \in \{0,1\}^L$, outputs

$$\mathrm{GGM}_k(J) = G_{J[L]}\big(G_{J[L-1]}\big(\ldots G_{J[2]}(G_{J[1]}(s))\ldots\big)\big)$$

where, for each $b \in \{0,1\}$, $G_b(s) \in \{0,1\}^n$ is defined to be the n-bit string such that $G(s) = G_0(s)||G_1(s)$.

For a matrix $\mathbf{A} \hookleftarrow U(\mathbb{Z}_q^{n\times m})$, Banerjee *et al.* [8] define the PRG

$$G_{\mathbf{A}} : \mathbb{Z}_q^n \to \mathbb{Z}_p^m, \quad G_{\mathbf{A}}(\mathbf{s}) = \lfloor \mathbf{A}^\top \cdot \mathbf{s} \rceil_p \ \in \mathbb{Z}_p^m,$$

which is a length-doubling PRG if we choose $q = p^2$ and $m = 4n$.

Next, in order to iteratively apply the PRG, we need to re-encode the output $\mathbf{y} \in \mathbb{Z}_p^m = \mathbb{Z}_p^{4n}$ of the PRG as a pair of elements of $\mathbb{Z}_q^n = \mathbb{Z}_{p^2}^n$. To this end, we define the dimension-modulus switching matrix $\mathbf{M}_{p,n} = \mathbf{I}_n \otimes \begin{bmatrix} 1 & p \end{bmatrix} \in \mathbb{Z}^{n\times 2n}$, which allows re-encoding any $\bar{\mathbf{y}} \in \mathbb{Z}_p^{2n}$ into $\mathbb{Z}_q^n = \mathbb{Z}_{p^2}^n$ by computing $\mathbf{M}_{p,n} \cdot \bar{\mathbf{y}} \bmod q$.

Now, to evaluate the function $\mathrm{GGM}_{\mathbf{A},\mathbf{s}}(J)$ for the input $J[1]\dots J[L] \in \{0,1\}^L$ and the seed $\mathbf{s} \in \mathbb{Z}_q^n$, we can set $\mathbf{s}_0 = \mathbf{s}$ and conduct the following steps:

1. Compute $\mathbf{y}_0 = G_{\mathbf{A}}(\mathbf{s}_0) \in \mathbb{Z}_p^m$ and let $\mathbf{y}_0 = \begin{pmatrix} \mathbf{y}_0^{(0)} \\ \mathbf{y}_0^{(1)} \end{pmatrix}$, where $\mathbf{y}_0^{(0)}, \mathbf{y}_0^{(1)} \in \mathbb{Z}_p^{2n}$.
2. For $i = 1$ to L, recursively compute the following:

$$\mathbf{s}_i = \mathbf{M}_{p,n} \cdot \mathbf{y}_{i-1}^{(J[i])} \in \mathbb{Z}_q^n; \qquad \mathbf{y}_i = G_{\mathbf{A}}(\mathbf{s}_i) \in \mathbb{Z}_p^m.$$

3. Output $\mathbf{y} = \mathbf{y}_L \in \mathbb{Z}_p^m$.

Our goal is to prove the correctness of the output $\mathbf{y} = \mathrm{GGM}_{\mathbf{A},\mathbf{s}}(J)$ for a committed seed $\mathbf{s}$. We will use the strategy explained in Section 4.3, where the main technical difficulty is to transform the equations and secret vectors underlying the evaluation process in such a way that we can prove the whole statement by running the abstract protocol of Section 4.1. As the commitment relation can be smoothly integrated into the picture, we focus on the PRF evaluation layer.

In the initial step of the evaluation, we are presented with secret vectors $\mathbf{s}_0 \in \mathbb{Z}_q^n$ and $\mathbf{y}_0 = \lfloor \mathbf{A}^\top \cdot \mathbf{s}_0 \rceil_p$. Let $\mathbf{x}_0 = \mathbf{A}^\top \cdot \mathbf{s}_0 \in \mathbb{Z}_q^m$, then we have $\mathbf{y}_0 = \lfloor \mathbf{x}_0 \rceil_p$, which yields the equation $p \cdot \mathbf{x}_0 - q \cdot \mathbf{y}_0 - \mathbf{z}_0 \bmod pq$, for $\mathbf{z}_0 \in \mathbb{Z}_q^m$ - following our observation in Section 4.2.

The L iterative steps are much more involved, since for each $i \in [L]$, we additionally have to deal with the equation $\mathbf{s}_i = \mathbf{M}_{p,n} \cdot \mathbf{y}_{i-1}^{(J[i])} \bmod q$, which determines which half of vector $\mathbf{y}_{i-1}$ goes to the next step based on the secret bit $J[i]$. To this end, we transform the equation into

$$\begin{aligned}
\mathbf{s}_i = \mathbf{M}_{p,n} \cdot \mathbf{y}_{i-1}^{(J[i])} &= \mathbf{M}_{p,n} \cdot \big(\overline{J[i]} \cdot \mathbf{y}_{i-1}^{(0)} + J[i] \cdot \mathbf{y}_{i-1}^{(1)}\big) \\
&= \big[\mathbf{M}_{p,n} \mid \mathbf{0}^{n\times 2n}\big] \cdot \begin{pmatrix} \overline{J[i]} \cdot \mathbf{y}_{i-1}^{(0)} \\ J[i] \cdot \mathbf{y}_{i-1}^{(0)} \end{pmatrix} + \big[\mathbf{0}^{n\times 2n} \mid \mathbf{M}_{p,n}\big] \cdot \begin{pmatrix} \overline{J[i]} \cdot \mathbf{y}_{i-1}^{(1)} \\ J[i] \cdot \mathbf{y}_{i-1}^{(1)} \end{pmatrix} \\
&= \big[\mathbf{M}_{p,n} \mid \mathbf{0}^{n\times 2n}\big] \cdot \mathsf{expand}\big(J[i], \mathbf{y}_{i-1}^{(0)}\big) + \big[\mathbf{0}^{n\times 2n} \mid \mathbf{M}_{p,n}\big] \cdot \mathsf{expand}\big(J[i], \mathbf{y}_{i-1}^{(1)}\big) \\
&= \big[\mathbf{M}_{p,n} \mid \mathbf{0}^{n\times 2n}\big] \cdot \mathbf{t}_i^{(0)} + \big[\mathbf{0}^{n\times 2n} \mid \mathbf{M}_{p,n}\big] \cdot \mathbf{t}_i^{(1)}
\end{aligned}$$

where the constraints of the secret vectors $\mathbf{t}_i^{(0)} = \mathsf{expand}\big(J[i], \mathbf{y}_{i-1}^{(0)}\big) \in \mathbb{Z}_p^{4n}$ and $\mathbf{t}_i^{(1)} = \mathsf{expand}\big(J[i], \mathbf{y}_{i-1}^{(1)}\big) \in \mathbb{Z}_p^{4n}$ are compatible with Stern-like techniques.

It follows from the above discussion that the evaluation process can be captured by $\mathcal{O}(L)$ equations modulo q and pq, with respect to secret vectors $\{\mathbf{s}_i\}_{i=0}^{L}$, $\{\mathbf{x}_i\}_{i=0}^{L}$, $\{\mathbf{z}_i\}_{i=0}^{L}$, $\mathbf{y}_0 = (\mathbf{y}_0^{(0)} \| \mathbf{y}_0^{(1)}), \ldots, \mathbf{y}_{L-1} = (\mathbf{y}_{L-1}^{(0)} \| \mathbf{y}_{L-1}^{(1)}) \in \mathbb{Z}_p^{4n}$; and $\mathbf{t}_1^{(0)}, \mathbf{t}_1^{(1)}, \ldots, \mathbf{t}_L^{(0)}, \mathbf{t}_L^{(1)} \in \mathbb{Z}_p^{4n}$ of the form described above.

Now, as in Section 4.3, we can unify all the considered equations into only two equations modulo q and pq. If we commit to the seed $\mathbf{s}_0$ via the same mechanism as in Section 4.3, we obtain another equation modulo q_s. Then, we can use the decomposition-extension techniques of Section 3 to transform the underlying witnesses so that the obtained vectors have their coordinates in $\{-1, 0, 1\}$ and satisfy constraints that are invariant under permutation. Next, we concatenate all the resulting vectors into one vector $\mathbf{w}$ and specify suitable sets VALID and $\mathcal{S}$ for which the conditions in (9) are satisfied. Finally, we run the abstract protocol of Section 4.1 to prove our statement.

For recommended parameter settings of the PRG layer [8,3,15] and the commitment layer [67,71], where $n = \mathcal{O}(\lambda)$, $q, q_s \in \mathsf{poly}(\lambda)$, the protocol has communication cost $\widetilde{\mathcal{O}}(L \cdot \lambda)$.

D Deferred Proofs for the Compact E-Cash System

D.1 Proof of Theorem 2

Proof. Let $\mathcal{A}$ be an adversary that executes Q_w withdrawal protocols in interaction with a honest bank and wins the game of Definition 5. We show that there exists an extractor $\mathcal{E}^{\mathcal{A}}_{\mathsf{Withdraw}}$ for which $\mathcal{A}$ implies an algorithm $\mathcal{B}^{\mathsf{SIS}}$ that solves an instance $(\bar{\mathbf{A}}, \beta) \in \mathbb{Z}_{q_s}^{n \times m_s} \times \mathbb{R}$ of the $\mathsf{SIS}^{\infty}_{n,m_s,q_s,\beta}$ problem.

We define $\mathcal{E}^{\mathcal{A}}_{\mathsf{Withdraw}}$ to be the knowledge extractor of the argument of knowledge generated by the prover at step 1 of the withdrawal protocol (i.e., the argument of knowledge of $(\mathbf{e}_u, \tilde{\mathbf{t}}, \tilde{\mathbf{k}}_0, \tilde{\mathbf{k}}_1) \in \{0,1\}^{m_f} \times (\{0,1\}^{\bar{m}})^3$ forming an opening of $\mathbf{c}_{\mathcal{U}}$ such that $PK_{\mathcal{U}} = \mathbf{F} \cdot \mathbf{e}_u$). It is easy to see that, if $\mathcal{E}^{\mathcal{A}}_{\mathsf{Withdraw}}$ successfully extracts the witnesses $(\mathbf{e}_u, \tilde{\mathbf{t}}, \tilde{\mathbf{k}}_0, \tilde{\mathbf{k}}_1)$ by rewinding $\mathcal{A}$ at step 1, it can reconstruct the seed $\mathbf{k} = \mathbf{k}_0 + \mathbf{k}_1 \bmod q$, where $\mathbf{k}_0 = \mathbf{H}_{m,q-1} \cdot \tilde{\mathbf{k}}_0 \bmod q$ and $\mathbf{k}_1 = \mathbf{H}_{m,q-1} \cdot \tilde{\mathbf{k}}_1 \bmod q$. In turn, $\mathbf{k} \in \mathbb{Z}_q^m$ determines all the serial number $\mathbf{y}_S \in \mathbb{Z}_p^m$ that can be generated using the wallet. Note that, when $\mathcal{A}$ terminates, by hypothesis, $\mathrm{DB}_{\mathcal{B}}$ must contain a coin $(R^\star, \mathbf{y}_S^\star, \mathbf{y}_T^\star, \pi_K^\star)$ such that the serial number $\mathbf{y}_S^\star$ never appears in the database T of serial numbers generated by the knowledge extractor $\mathcal{E}^{\mathcal{A}}_{\mathsf{Withdraw}}$ when answering $\mathcal{Q}_{\mathsf{withdraw}}$-queries.

We also call $\mathcal{E}^{\mathcal{A}}_{\mathsf{Spend}}$ the knowledge extractor of the argument system used in the Spend protocol, which $\mathcal{B}^{\mathsf{SIS}}$ will invoke when $\mathcal{A}$ halts. Since $\mathcal{E}^{\mathcal{A}}_{\mathsf{Spend}}$ is applied to non-interactive Fiat-Shamir-like proofs, it proceeds by programming the random oracle H in such a way that the responses in repeated executions fork with respect to those of the initial execution.

Unless the computational soundness of the argument system can be broken (recall that the soundness of these arguments relies on the binding property of the underlying commitment, which relies itself on the $\mathsf{SIS}^{\infty}_{n,m_s,q_s,\beta}$ assumption), $\mathcal{B}^{\mathsf{SIS}}$ can expect that the knowledge extractors $\mathcal{E}^{\mathcal{A}}_{\mathsf{Withdraw}}$ and $\mathcal{E}^{\mathcal{A}}_{\mathsf{Spend}}$ will manage to

extract witnesses whenever they are called. If $\mathcal{E}^{\mathcal{A}}_{\mathsf{Spend}}$ indeed succeeds, it extracts $\tau^\star \in \{0,1\}^\ell$, $\mathbf{e}_u^\star \in \{0,1\}^{m_f}$, $\tilde{\mathbf{w}}_s{}^\star \in \{0,1\}^{m_s/2}$, $\tilde{\mathbf{t}}^\star, \tilde{\mathbf{k}_0}^\star, \tilde{\mathbf{k}_1}^\star \in \{0,1\}^{\bar{m}}$, $\mathbf{v}^\star \in \mathbb{Z}^{2m_s}$ and $\mathbf{r}^\star \in \mathbb{Z}^{2m_s}$ satisfying

$$\begin{aligned} \mathbf{A}_{\tau^\star} \cdot \mathbf{v}^\star &= \mathbf{u} + \mathbf{D} \cdot \tilde{\mathbf{w}}_s{}^\star \bmod q_s \\ \mathbf{H}_{n,q_s-1} \cdot \tilde{\mathbf{w}}_s{}^\star &= \mathbf{D}_0 \cdot \mathbf{r}^\star + \mathbf{D}_1 \cdot \mathbf{e}_u^\star + \mathbf{D}_2 \cdot \tilde{\mathbf{t}}^\star \\ &\quad + \mathbf{D}_3 \cdot \tilde{\mathbf{k}_0}^\star + \mathbf{D}_4 \cdot \tilde{\mathbf{k}_1}^\star \bmod q_s. \end{aligned} \tag{39}$$

Note that, if $(\mathbf{e}_u^{(i)}, \tilde{\mathbf{t}}^{(i)}, \tilde{\mathbf{k}_0}^{(i)}, \tilde{\mathbf{k}_1}^{(i)})$ are the vectors that are certified in the i-th wallet returned by $\mathcal{Q}_{\mathsf{withdraw}}$ oracle, for each $i \in [1, Q_w]$, it must be the case that

$$\mathbf{k}^\star = \mathbf{H}_{m,q-1} \cdot (\tilde{\mathbf{k}_0}^\star + \tilde{\mathbf{k}_1}^\star) \bmod q \neq \mathbf{H}_{m,q-1} \cdot \left(\tilde{\mathbf{k}_0}^{(i)} + \tilde{\mathbf{k}_1}^{(i)}\right) \bmod q = \mathbf{k}^{(i)}$$

(and *a fortiori* $\left(\tilde{\mathbf{k}_0}^\star, \tilde{\mathbf{k}_1}^\star\right) \neq \left(\tilde{\mathbf{k}_0}^{(i)}, \tilde{\mathbf{k}_1}^{(i)}\right)$) since, otherwise, the serial number $\mathbf{y}_S^\star$ would appear in the database T.

Before starting its interaction with $\mathcal{A}$, $\mathcal{B}^{\mathsf{SIS}}$ randomly chooses a coin $\vartheta \hookleftarrow U(\{0,1,2\})$ and an index $i^\star \hookleftarrow U([1, Q_w])$ in an attempt to foresee the strategy behind $\mathcal{A}$'s attack. If $\vartheta = 0$, $\mathcal{B}^{\mathsf{SIS}}$ expects that $\mathcal{E}^{\mathcal{A}}_{\mathsf{Spend}}$ will output a fresh identifier $\tau^\star \in \{0,1\}^\ell$ that was never used by the $\mathcal{Q}_{\mathsf{withdraw}}$ oracle. If $\vartheta = 1$, $\mathcal{B}^{\mathsf{SIS}}$ bets that:

- $\tau^\star$ will be recycled from a wallet generated by the $\mathcal{Q}_{\mathsf{withdraw}}$ oracle at the $i^\star$-th $\mathcal{Q}_{\mathsf{withdraw}}$-query (i.e., $\tau^\star = \tau^{(i^\star)}$).
- If $\mathbf{e}_u^{(i^\star)}$, $\tilde{\mathbf{t}}^{(i^\star)}$, $\tilde{\mathbf{k}_0}^{(i^\star)}$, $\tilde{\mathbf{k}_1}^{(i^\star)}$, $\mathbf{v}^{(i^\star)}$, $\mathbf{r}^{(i^\star)}$ are the vectors involved in the $i^\star$-th query (note that $\mathcal{B}^{\mathsf{SIS}}$ can reconstruct these values via $\mathcal{E}^{\mathcal{A}}_{\mathsf{Withdraw}}$), we have

$$\begin{aligned} \mathbf{D}_0 \cdot \mathbf{r}^\star + \mathbf{D}_1 \cdot \mathbf{e}_u^\star + \mathbf{D}_2 \cdot \tilde{\mathbf{t}}^\star + \mathbf{D}_3 \cdot \tilde{\mathbf{k}_0}^\star + \mathbf{D}_4 \cdot \tilde{\mathbf{k}_1}^\star \\ \neq \mathbf{D}_0 \cdot \mathbf{r}^{(i^\star)} + \mathbf{D}_1 \cdot \mathbf{e}_u^{(i^\star)} + \mathbf{D}_2 \cdot \tilde{\mathbf{t}}^{(i^\star)} \\ + \mathbf{D}_3 \cdot \tilde{\mathbf{k}_0}^{(i^\star)} + \mathbf{D}_4 \cdot \tilde{\mathbf{k}_1}^{(i^\star)} \end{aligned} \tag{40}$$

If $\vartheta = 2$, $\mathcal{B}^{\mathsf{SIS}}$ rather foresees the opposite case of (40): rewinding $\mathcal{A}$ allows $\mathcal{E}^{\mathcal{A}}_{\mathsf{Spend}}$ to reveal $\tau^\star = \tau^{(i^\star)}$ but

$$\begin{aligned} \mathbf{D}_0 \cdot \mathbf{r}^\star + \mathbf{D}_1 \cdot \mathbf{e}_u^\star + \mathbf{D}_2 \cdot \tilde{\mathbf{t}}^\star + \mathbf{D}_3 \cdot \tilde{\mathbf{k}_0}^\star + \mathbf{D}_4 \cdot \tilde{\mathbf{k}_1}^\star \\ = \mathbf{D}_0 \cdot \mathbf{r}^{(i^\star)} + \mathbf{D}_1 \cdot \mathbf{e}_u^{(i^\star)} + \mathbf{D}_2 \cdot \tilde{\mathbf{t}}^{(i^\star)} \\ + \mathbf{D}_3 \cdot \tilde{\mathbf{k}_0}^{(i^\star)} + \mathbf{D}_4 \cdot \tilde{\mathbf{k}_1}^{(i^\star)} \end{aligned} \tag{41}$$

Before starting its interaction with $\mathcal{A}$, $\mathcal{B}^{\mathsf{SIS}}$ computes re-randomized versions $\mathbf{A}_{\mathsf{withdraw}}$, $\mathbf{A}_{\mathsf{spend}}$ of the matrix $\bar{\mathbf{A}}$ (by multiplying them on the right by discrete Gaussian matrices with the appropriate number of columns), which will be used to instantiate the statistically hiding commitment of the argument system in the Withdraw and Spend protocols. By doing so, if one of the knowledge extractors $\mathcal{E}^{\mathcal{A}}_{\mathsf{Withdraw}}$ and $\mathcal{E}^{\mathcal{A}}_{\mathsf{Spend}}$ fails to extract witnesses by rewinding $\mathcal{A}$, $\mathcal{B}^{\mathsf{SIS}}$ immediately obtains a SIS solution.

Initialization. Depending on the value of $\vartheta \in \{0,1,2\}$, $\mathcal{B}^{\mathsf{SIS}}$ sets up par and $PK_{\mathcal{B}}$ using three different strategies.

• If $\vartheta = 0$, the public parameters par are honestly generated by faithfully running ParGen. To generate the bank's public key $PK_{\mathcal{B}}$, $\mathcal{B}^{\mathsf{SIS}}$ first chooses the ℓ-bit tags $\tau^{(1)}, \ldots, \tau^{(Q_w)} \hookleftarrow \{0,1\}^{\ell}$ that will be used by the $\mathcal{Q}_{\mathsf{withdraw}}$ oracle. As in the prefix-guessing technique of [62], it guesses the shortest prefix such that the string $\tau^{\star}$ produced by $\mathcal{E}^{\mathcal{A}}_{\mathsf{Withdraw}}$ differs from all prefixes of $\tau^{(1)}, \ldots, \tau^{(Q_w)}$. To this end, $\mathcal{B}$ picks $i^{\dagger} \hookleftarrow [1, Q_w]$ and $t^{\dagger} \hookleftarrow [1, \ell]$ in such a way that, with probability $1/(Q_w \cdot \ell)$, the string $\tau^{\star}[1] \ldots \tau^{\star}[t^{\dagger}-1] \in \{0,1\}^{t^{\dagger}-1}$ is the longest common prefix between $\tau^{\star}$ and any of the $\{\tau^{(i)}\}_{i=1}^{Q_w}$. Said otherwise, $t^{\dagger} \in [1, \ell]$ is the smallest integer such that the $t^{\dagger}$-bit string $\tau^{\star} = \tau^{\star}[1] \ldots \tau^{\star}[t^{\dagger}]$ differs from the length-$t^{\dagger}$ prefixes of all $\{\tau^{(i)}\}_{i=1}^{Q_w}$ with probability $1/(Q_w \cdot \ell)$.

Then, $\mathcal{B}$ runs $\mathsf{TrapGen}(1^n, 1^{m_s}, q_s)$ to obtain $\mathbf{C} \in \mathbb{Z}_{q_s}^{n \times m_s}$ and a basis $\mathbf{T_C}$ of $\Lambda_{q_s}^{\perp}(\mathbf{C})$ such that $\|\widetilde{\mathbf{T_C}}\| \leq \mathcal{O}(\sqrt{n \log q_s})$. Then, it samples $\ell + 1$ matrices $\mathbf{Q}_0, \ldots, \mathbf{Q}_{\ell} \in \mathbb{Z}^{m_s \times m_s}$, where the columns of each matrix $\mathbf{Q}_i$ are sampled independently from $D_{\mathbb{Z}^{m_s}, \sigma}$. These are used to define the matrices $\{\mathbf{A}_j\}_{j=0}^{\ell}$ as

$$\begin{cases} \mathbf{A}_0 = \bar{\mathbf{A}} \cdot \mathbf{Q}_0 + (\sum_{j=1}^{t^{\dagger}} \tau^{\star}[j]) \cdot \mathbf{C} & \\ \mathbf{A}_j = \bar{\mathbf{A}} \cdot \mathbf{Q}_j + (-1)^{\tau^{\star}[j]} \cdot \mathbf{C}, & \text{for } j \in [j, t^{\dagger}] \\ \mathbf{A}_j = \bar{\mathbf{A}} \cdot \mathbf{Q}_j, & \text{for } j \in [t^{\dagger}+1, \ell] \end{cases}$$

It also sets $\mathbf{A} = \bar{\mathbf{A}}$. Note that

$$\begin{aligned} \mathbf{A}_{\tau^{(i)}} &= \Big[\bar{\mathbf{A}} \Big| \mathbf{A}_0 + \textstyle\sum_{j=1}^{\ell} \tau^{(i)}[j] \mathbf{A}_j \Big] \\ &= \Big[\bar{\mathbf{A}} \;\Big|\; \bar{\mathbf{A}} \cdot (\mathbf{Q}_0 + \textstyle\sum_{j=1}^{\ell} \tau^{(i)}[j] \mathbf{Q}_j) + (\sum_{j=1}^{t^{\dagger}} \tau^{\star}[j] + (-1)^{\tau^{\star}[j]} \tau^{(i)}[j]) \cdot \mathbf{C} \Big] \\ &= \Big[\bar{\mathbf{A}} \;\Big|\; \bar{\mathbf{A}} \cdot (\mathbf{Q}_0 + \textstyle\sum_{j=1}^{\ell} \tau^{(i)}[j] \mathbf{Q}_j) + h_{\tau^{(i)}} \cdot \mathbf{C} \Big] \end{aligned}$$

where $h_{\tau^{(i)}} \in [1, t^{\dagger}] \subset [1, \ell]$ is the Hamming distance between $\tau^{(i)}_{|t^{\dagger}}$ and $\tau^{\star}_{|t^{\dagger}}$. With probability $1/(Q_w \cdot \ell)$ and since $q_s > \ell$, we have $h_{\tau^{(i)}} \neq 0 \bmod q_s$ whenever $\tau^{(i)}_{|t^{\dagger}} \neq \tau^{\star}_{|t^{\dagger}}$.

Next, $\mathcal{B}$ samples a short matrix $\mathbf{R} \in \mathbb{Z}^{m_s \times (m_s/2)}$ from $D_{\mathbb{Z}^{m_s}, \sigma} \times \cdots \times D_{\mathbb{Z}^{m_s}, \sigma}$ and computes $\mathbf{D} = \bar{\mathbf{A}} \cdot \mathbf{R} \in \mathbb{Z}_{q_s}^{n \times (m_s/2)}$. Finally, $\mathcal{B}$ samples $\mathbf{e}_u \in D_{\mathbb{Z}^{m_s}, \sigma}$ and computes the vector $\mathbf{u} \in \mathbb{Z}_q^n$ as $\mathbf{u} = \bar{\mathbf{A}} \cdot \mathbf{e}_u \in \mathbb{Z}_{q_s}^n$.

• If $\vartheta = 1$, the reduction $\mathcal{B}^{\mathsf{SIS}}$ similarly defines $\mathbf{D} = \bar{\mathbf{A}} \cdot \mathbf{R} \in \mathbb{Z}_{q_s}^{n \times (m_s/2)}$, using a small-norm matrix $\mathbf{R} \in \mathbb{Z}^{m_s \times (m_s/2)}$ sampled from $D_{\mathbb{Z}^{m_s}, \sigma} \times \cdots \times D_{\mathbb{Z}^{m_s}, \sigma}$. It chooses random $\tau^{(1)}, \ldots, \tau^{(Q_w)} \hookleftarrow U(\{0,1\}^{\ell})$ and aborts in the event that they are not all distinct (which occurs with negligible probability $Q_w^2/2^{\ell}$). It also picks a random index $i^{\star} \hookleftarrow [1, Q_w]$ as a guess that $\mathcal{E}^{\mathcal{A}}_{\mathsf{Withdraw}}$ will extract an ℓ-bit string $\tau^{\star}$ that coincides with the tag $\tau^{(i^{\star})} \in \{0,1\}^{\ell}$ of the $i^{\star}$-th $\mathcal{Q}_{\mathsf{withdraw}}$-query. To generate par, $\mathcal{B}^{\mathsf{SIS}}$ chooses $CK = \big(\mathbf{D}_0 = [\mathbf{D}'_0 \mid \mathbf{D}''_0], \mathbf{D}_1, \mathbf{D}_3, \mathbf{D}_4\big)$ by picking $\mathbf{D}'_0 \hookleftarrow \mathbb{Z}_{q_s}^{n \times m_s}$, $\mathbf{D}_1, \mathbf{D}_2, \mathbf{D}_3, \mathbf{D}_4 \hookleftarrow U(\mathbb{Z}_{q_s}^{n \times \bar{m}})$ uniformly and running the

trapdoor generation algorithm $(\mathbf{D}_0'', \mathbf{T}_{\mathbf{D}_0''}) \leftarrow \mathsf{TrapGen}(1^n, 1^{m_s}, q_s)$ to generate a statistically uniform $\mathbf{D}_0'' \in \mathbb{Z}_{q_s}^{n \times m_s}$ together with a short basis $\mathbf{T}_{\mathbf{D}_0''} \in \mathbb{Z}^{m_s \times m_s}$ of $\Lambda_{q_s}^{\perp}(\mathbf{D}_0'')$. Other components of par are chosen as in the actual ParGen algorithm. To generate the bank's public key $PK_{\mathcal{B}}$, $\mathcal{B}$ picks $h_0, h_1, \ldots, h_\ell \in \mathbb{Z}_{q_s}$ at random subject to the constraints

$$\begin{aligned} h_0 + \sum_{j=1}^{\ell} \tau^{(i^\star)}[j] \cdot h_j &= 0 \bmod q_s \\ h_0 + \sum_{j=1}^{\ell} \tau^{(i)}[j] \cdot h_j &\neq 0 \bmod q_s \qquad\qquad i \in [1, Q_w] \setminus \{i^\star\} \end{aligned}$$

Next, $\mathcal{B}^{\mathsf{SIS}}$ runs $(\mathbf{C}, \mathbf{T}_{\mathbf{C}}) \leftarrow \mathsf{TrapGen}(1^n, 1^{m_s}, q_s)$ in order to obtain a statistically random matrix $\mathbf{C} \in \mathbb{Z}_{q_s}^{n \times m_s}$ with a trapdoor $\mathbf{T}_{\mathbf{C}}$ consisting of a short basis of $\Lambda_{q_s}^{\perp}(\mathbf{C})$. Then, $\mathcal{B}^{\mathsf{SIS}}$ defines $\mathbf{A} = \bar{\mathbf{A}}$ and generates "re-randomizations" of $\bar{\mathbf{A}}$ by sampling short matrices $\mathbf{S}_0, \mathbf{S}_1, \ldots, \mathbf{S}_\ell \hookleftarrow \mathbb{Z}^{m_s \times m_s}$ from the distribution $D_{\mathbb{Z}^{m_s},\sigma} \times \ldots \times D_{\mathbb{Z}^{m_s},\sigma}$: from $\bar{\mathbf{A}} \in \mathbb{Z}_{q_s}^{n \times m_s}$, $\mathcal{B}$ defines

$$\begin{aligned} \mathbf{A}_0 &= \bar{\mathbf{A}} \cdot \mathbf{S}_0 + h_0 \cdot \mathbf{C} \\ \mathbf{A}_j &= \bar{\mathbf{A}} \cdot \mathbf{S}_j + h_j \cdot \mathbf{C} \qquad\qquad \forall j \in \{1, \ldots, \ell\} \end{aligned} \tag{42}$$

In addition, $\mathcal{B}$ picks a random vector $\mathbf{c}'_{\mathcal{U}} \hookleftarrow U(\mathbb{Z}_{q_s}^n)$. It samples short vectors $\mathbf{v}_1, \mathbf{v}_2 \hookleftarrow D_{\mathbb{Z}^m,\sigma}$ and computes $\mathbf{u} \in \mathbb{Z}_{q_s}^n$ as

$$\mathbf{u} = \mathbf{A}_{\tau^{(i^\star)}} \cdot \begin{bmatrix} \mathbf{v}_1 \\ \hline \mathbf{v}_2 \end{bmatrix} - \mathbf{D} \cdot \mathsf{vdec}_{n, q_s - 1}(\mathbf{c}'_{\mathcal{U}}) \bmod q_s, \tag{43}$$

where

$$\begin{aligned} \mathbf{A}_{\tau^{(i^\star)}} &= \left[\mathbf{A} \;\middle|\; \mathbf{A}_0 + \textstyle\sum_{j=1}^{\ell} \tau^{(i^\star)}[j] \cdot \mathbf{A}_j \right] \\ &= \left[\bar{\mathbf{A}} \;\middle|\; \bar{\mathbf{A}} \cdot (\mathbf{S}_0 + \textstyle\sum_{j=1}^{\ell} \tau^{(i^\star)}[j] \cdot \mathbf{S}_j) \right]. \end{aligned} \tag{44}$$

• If $\vartheta = 2$, $\mathcal{B}^{\mathsf{SIS}}$ defines $\mathbf{D}'_0 = \bar{\mathbf{A}} \in \mathbb{Z}_{q_s}^{n \times m_s}$. It also chooses small-norm matrices $\mathbf{Q}_0'' \in \mathbb{Z}^{m_s \times m_s}$, $\mathbf{Q}_1, \mathbf{Q}_2, \mathbf{Q}_3, \mathbf{Q}_4 \in \mathbb{Z}^{m_s \times \bar{m}}$ by sampling their columns from $D_{\mathbb{Z}^{m_s},\sigma}$. These are used to define

$$\begin{aligned} \mathbf{D}_0'' &= \mathbf{D}'_0 \cdot \mathbf{Q}_0'' \in \mathbb{Z}_{q_s}^{n \times m_s} & \mathbf{D}_1 &= \mathbf{D}'_0 \cdot \mathbf{Q}_1 \in \mathbb{Z}_{q_s}^{n \times \bar{m}} \\ \mathbf{D}_2 &= \mathbf{D}'_0 \cdot \mathbf{Q}_2 \in \mathbb{Z}_{q_s}^{n \times \bar{m}} & \mathbf{D}_3 &= \mathbf{D}'_0 \cdot \mathbf{Q}_3 \in \mathbb{Z}_{q_s}^{n \times \bar{m}} \\ \mathbf{D}_4 &= \mathbf{D}'_0 \cdot \mathbf{Q}_4 \in \mathbb{Z}_{q_s}^{n \times \bar{m}} \end{aligned}$$

and form a commitment key $CK = \big(\mathbf{D}_0 = [\mathbf{D}'_0 \mid \mathbf{D}_0''], \mathbf{D}_1, \mathbf{D}_3, \mathbf{D}_4\big)$ made of statistically uniform matrices. The remaining components of par and $PK_{\mathcal{B}}$ are generated according to the specification of ParGen and $\mathsf{BKeygen}$. In particular, $\mathbf{A}, \mathbf{A}_0, \mathbf{A}_1, \ldots, \mathbf{A}_\ell \hookleftarrow U(\mathbb{Z}_{q_s}^{n \times m_s})$ and $\mathbf{D} \hookleftarrow U\big(\mathbb{Z}_{q_s}^{n \times (m_s/2)}\big)$ are chosen uniformly.

In all cases, the public key $PK_{\mathcal{B}} := \big(\mathbf{A},\ \{\mathbf{A}_j\}_{j=0}^{\ell},\ \mathbf{D},\ \mathbf{u}\big)$ is then given to $\mathcal{A}$.

Queries. We first note that the $\mathcal{Q}_{\mathsf{deposit}}$ oracle can be perfectly simulated by $\mathcal{B}^{\mathsf{SIS}}$ as the $\mathsf{Deposit}$ protocol does not involve the bank's secret key $SK_{\mathcal{B}}$. Depending on the value of $\vartheta \in \{0,1,2\}$, $\mathcal{B}^{\mathsf{SIS}}$ answers $\mathcal{Q}_{\mathsf{withdraw}}$-queries in different manners.

• If $\vartheta = 0$, $\mathcal{B}$ can always use the trapdoor $\mathbf{T_C} \in \mathbb{Z}^{m\times m}$ to answer $\mathcal{Q}_{\mathsf{withdraw}}$-queries given that $h_{\tau^{(i)}} \neq 0 \bmod q_s$ whenever $\tau^{(i)}_{|t^\dagger} \neq \tau^{\star}_{|t^\dagger}$. At the i-th query, the adversary $\mathcal{A}$ chooses a commitment $\mathbf{c}_{\mathcal{U}}^{(i)} \in \mathbb{Z}_{q_s}^n$ and proves knowledge of an opening $\big(\mathbf{r}_0^{(i)}, \mathbf{e}_u^{(i)}, \tilde{\mathbf{t}}^{(i)}, \tilde{\mathbf{k}_0}^{(i)}\big)$ such that $\mathbf{e}_u^{(i)}$ is consistent with the adversarially-chosen public key $PK_{\mathcal{U}}^{(i)}$. To answer the query, $\mathcal{B}^{\mathsf{SIS}}$ first samples $\boldsymbol{r}_1^{(i)} \hookleftarrow D_{\mathbb{Z}^{m_s},\sigma}$ and $\mathbf{k}_1^{(i)} \hookleftarrow U(\mathbb{Z}_q^m)$ and computes a vector $\mathbf{c}'_{\mathcal{U}}{}^{(i)} \in \mathbb{Z}_{q_s}^n$ as

$$\mathbf{c}'_{\mathcal{U}}{}^{(i)} = \mathbf{c}_{\mathcal{U}}^{(i)} + \mathbf{D}_0'' \cdot \mathbf{r}_1^{(i)} + \mathbf{D}_4 \cdot \tilde{\mathbf{k}_1}^{(i)} \in \mathbb{Z}_{q_s}^n.$$

Then, algorithm $\mathcal{B}^{\mathsf{SIS}}$ defines $\mathbf{u}_{\mathcal{U}}{}^{(i)} = \mathbf{u}^{(i)} + \mathbf{D} \cdot \mathsf{vdec}_{n,q_s-1}(\mathbf{c}'_{\mathcal{U}}{}^{(i)}) \in \mathbb{Z}_{q_s}^n$ and uses $\mathbf{T_C} \in \mathbb{Z}^{m_s\times m_s}$ to sample a short vector $\mathbf{v}^{(i)} \in \mathbb{Z}^{2m_s}$ in $D_{\Lambda_{q_s}^{\mathbf{u}_{\mathcal{U}}{}^{(i)}}(\mathbf{A}_{\tau^{(i)}}),\sigma}$ such that $(\tau^{(i)}, \mathbf{v}^{(i)}, \mathbf{r}_1^{(i)})$ satisfies

$$\mathbf{A}_{\tau^{(i)}} \cdot \mathbf{v}^{(i)} = \mathbf{u} + \mathbf{D} \cdot \mathsf{vdec}_{n,q_s-1}\Big(\mathbf{c}_{\mathcal{U}}^{(i)} + \mathbf{D}_0'' \cdot \mathbf{r}_1^{(i)} + \mathbf{D}_4 \cdot \tilde{\mathbf{k}_1}^{(i)} \bmod q_s\Big)$$

• If $\vartheta = 1$, $\mathcal{B}^{\mathsf{SIS}}$ simulates the $\mathcal{Q}_{\mathsf{withdraw}}$ oracle as follows. At the i-th query, $\mathcal{A}$ supplies a public key $PK_{\mathcal{U}}^{(i)}$, a commitment $\mathbf{c}_{\mathcal{U}}^{(i)} \in \mathbb{Z}_{q_s}^n$ and proves knowledge of an opening $\big(\mathbf{r}_0^{(i)}, \mathbf{e}_u^{(i)}, \tilde{\mathbf{t}}^{(i)}, \tilde{\mathbf{k}_0}^{(i)}\big)$ for which $\mathbf{e}_u^{(i)}$ is consistent with $PK_{\mathcal{U}}^{(i)}$. If the proof of knowledge verifies, the way that $\mathcal{B}^{\mathsf{SIS}}$ responds depends on the index $i \in [1, Q_w]$.

- If $i \neq i^\star$, $\mathcal{B}^{\mathsf{SIS}}$ proceeds as in the case $\vartheta = 0$. Namely, it recalls the ℓ-bit tag $\tau^{(i)} \in \{0,1\}^\ell$ that was chosen in the setup phase and samples vectors $\mathbf{r}_1^{(i)} \hookleftarrow D_{\mathbb{Z}^{m_s},\sigma}$, $\mathbf{k}_1 \in \mathbb{Z}_q^m$. Note that

$$\begin{aligned}\mathbf{A}_{\tau^{(i)}} &= \Big[\ \bar{\mathbf{A}}\ \Big|\ \bar{\mathbf{A}} \cdot (\mathbf{S}_0 + \textstyle\sum_{j=1}^{\ell} \mathrm{id}_i[j]\mathbf{S}_j) + h_{\tau^{(i)}}\mathbf{C}\ \Big] \\ &= \big[\ \bar{\mathbf{A}}\ \big|\ \bar{\mathbf{A}} + h_{\tau^{(i)}} \cdot \mathbf{C}\ \big] \end{aligned} \tag{45}$$

  Since $h_{\tau^{(i)}} \neq 0$, $\mathcal{B}^{\mathsf{SIS}}$ can use the trapdoor $\mathbf{T_C} \in \mathbb{Z}^{m_s\times m_s}$ of $\Lambda_{q_s}^{\perp}(\mathbf{C})$ to compute a short vector $\mathbf{v}_i = [\mathbf{v}_{i,1}^T \mid \mathbf{v}_{i,2}^T]^T \in \mathbb{Z}^{2m_s}$ such that

$$\mathbf{A}_{\tau^{(i)}} \cdot \begin{bmatrix}\mathbf{v}_{i,1}\\ \mathbf{v}_{i,2}\end{bmatrix} = \mathbf{u} + \mathbf{D} \cdot \mathsf{vdec}_{n,q_s-1}\Big(\mathbf{c}_{\mathcal{U}}^{(i)} + \mathbf{D}_0'' \cdot \mathbf{r}_1^{(i)} + \mathbf{D}_4 \cdot \tilde{\mathbf{k}_1}^{(i)}\Big),$$

- If $i = i^\star$, $\mathcal{B}^{\mathsf{SIS}}$ has to respond without using the trapdoor $\mathbf{T_C}$ which vanishes from the matrix $\mathbf{A}_{\tau^{(i^\star)}}$ (44) for the ℓ-bit tag $\tau^{(i^\star)} \in \{0,1\}^\ell$. To do this, recalls the vector $\mathbf{v}_1, \mathbf{v}_2 \in \mathbb{Z}^{m_s}$ and $\mathbf{c}'_{\mathcal{U}} \in \mathbb{Z}_{q_s}^n$ that were used to define $\mathbf{u} \in \mathbb{Z}_{q_s}^n$ in

(43). Then, $\mathcal{B}^{\mathsf{SIS}}$ picks $\mathbf{k}_1^{(i^\star)} \hookleftarrow U(\mathbb{Z}_q^m)$ and uses the trapdoor $\mathbf{T}_{\mathbf{D}_0''}$ of $\Lambda_{q_s}^{\perp}(\mathbf{D}_0'')$ to sample a short vector $\mathbf{r}_1^{(i^\star)} \in \mathbb{Z}^{m_s}$ such that

$$\mathbf{D}_0'' \cdot \mathbf{r}_1^{(i^\star)} = \mathbf{c}_{\mathcal{U}}' - \mathbf{c}_{\mathcal{U}}^{(i^\star)} - \mathbf{D}_4 \cdot \tilde{\mathbf{k}_1}^{(i^\star)} \mod q_s,$$

before returning $\left(\tau^{(i^\star)}, \mathbf{v}_{i^\star} = [\mathbf{v}_1^T \mid \mathbf{v}_2^T]^T, \mathbf{r}_1^{(i^\star)}\right)$ to $\mathcal{A}$. From the definition of $\mathbf{u} \in \mathbb{Z}_{q_s}^n$ (43), it is easy to see that $(\tau^{(i^\star)}, \mathbf{v}_{i^\star}, \mathbf{r}_1^{(i^\star)})$ satisfy

$$\mathbf{A}_{\tau^{(i^\star)}} \cdot \mathbf{v}_{i^\star} = \mathbf{u} + \mathbf{D} \cdot \mathsf{vdec}_{n,q_s-1}\left(\mathbf{c}_{\mathcal{U}}^{(i^\star)} + \mathbf{D}_0'' \cdot \mathbf{r}_1^{(i^\star)} + \mathbf{D}_4 \cdot \tilde{\mathbf{k}_1}^{(i^\star)}\right),$$

Moreover, they are statistically close to the output distribution of the real $\mathcal{Q}_{\mathsf{withdraw}}$ oracle.

• If $\vartheta = 2$, can always answer $\mathcal{Q}_{\mathsf{withdraw}}$-queries by faithfully playing the bank's side of the protocol since it knows $SK_{\mathcal{B}}$. At each query $i \in [1, Q_w]$, however, $\mathcal{A}$ sends a commitment $\mathbf{c}_{\mathcal{U}}^{(i)} \in \mathbb{Z}_{q_s}^n$ and proves knowledge of an opening $\left(\mathbf{r}_0^{(i)}, \mathbf{e}_u^{(i)}, \tilde{\mathbf{t}}^{(i)}, \tilde{\mathbf{k}_0}^{(i)}\right)$ such that $\mathbf{e}_u^{(i)} \in \{0,1\}^{m_f}$ satisfies $PK_{\mathcal{U}}^{(i)} = \mathbf{F} \cdot \mathbf{e}_u^{(i)} \mod p$. At each execution of this proof of knowledge, $\mathcal{B}^{\mathsf{SIS}}$ uses the knowledge extractor $\mathcal{E}_{\mathsf{Withdraw}}^{\mathcal{A}}$ to extract $(\mathbf{r}_0^{(i)}, \mathbf{e}_u^{(i)}, \tilde{\mathbf{t}}^{(i)}, \tilde{\mathbf{k}_0}^{(i)})$ by rewinding $\mathcal{A}$ and retains them for later use.

Output. When $\mathcal{A}$ halts, $\mathcal{B}^{\mathsf{SIS}}$ looks up the database T which must contain a coin $coin^\star = (R^\star, \mathbf{y}_S^\star, \mathbf{y}_T^\star, \pi_K^\star)$ such that $\mathbf{y}_S^\star$ is *not* the result of evaluating the PRF on any $J \in [0, 2^L - 1]$ for a key

$$\mathbf{k}^{(i)} = \mathbf{H}_{m,q-1} \cdot (\tilde{\mathbf{k}_0}^{(i)} + \tilde{\mathbf{k}_1}^{(i)}) \qquad i \in [1, Q_W]$$

extracted by $\mathcal{E}_{\mathsf{Withdraw}}^{\mathcal{A}}$. If we parse $\pi_K^\star = (\{\mathsf{Comm}_{K,j}^\star\}_{j=1}^\kappa, \mathsf{Chall}_K^\star, \{\mathsf{Resp}_{K,j}^\star\}_{j=1}^\kappa)$, with high probability, the random oracle H must have been queried on the input $(R^\star, \mathbf{y}_{\mathbf{S}}^\star, \mathbf{y}_{\mathbf{T}}^\star, \{\mathsf{Comm}_{K,j}^\star\}_{j=1}^\kappa)$. Indeed, otherwise, we could only have the equality $\mathsf{Chall}_K^\star = H(R^\star, \mathbf{y}_{\mathbf{S}}^\star, \mathbf{y}_{\mathbf{T}}^\star, \{\mathsf{Comm}_{K,j}^\star\}_{j=1}^\kappa)$ with negligible probability $\leq 3^{-\kappa}$. With probability at least $\varepsilon' := \epsilon - 3^{-\kappa}$, the tuple $(R^\star, \mathbf{y}_{\mathbf{S}}^\star, \mathbf{y}_{\mathbf{T}}^\star, \{\mathsf{Comm}_{K,j}^\star\}_{j=1}^\kappa)$ coincides with the $\mu^\star$-th random oracle query for some $\mu^\star \in [1, Q_H]$.

At this point, $\mathcal{B}^{\mathsf{SIS}}$ runs the knowledge extractor $\mathcal{E}_{\mathsf{Spend}}^{\mathcal{A}}$ which replays the adversary $\mathcal{A}$ up to $32 \cdot Q_H/(\epsilon - 3^{-\kappa})$ times with the *same* random tape and input as in the very first run. All queries are answered as previously, except for one difference in the treatment of random oracle queries from the $\mu^\star$-th query onwards. More precisely, the first $\mu^\star - 1$ hash queries – which coincide with those of the first run since $\mathcal{A}$ is always run with the same random tape – obtain the same responses $\mathsf{Chall}_1, \ldots, \mathsf{Chall}_{\mu^\star - 1}$ as in the first execution. This implies that the input of the $\mu^\star$-th hash query will be $(R^\star, \mathbf{y}_{\mathbf{S}}^\star, \mathbf{y}_{\mathbf{T}}^\star, \{\mathsf{Comm}_{K,j}^\star\}_{j=1}^\kappa)$ as in the $\mu^\star$-th query of the first run. However, from the $\mu^\star$-th query forward, $\mathcal{A}$ obtains fresh random oracle values $\mathsf{Chall}_{\mu^\star}', \ldots, r_{Q_H}'$ at each new execution. The Improved Forking Lemma of Brickell *et al.* [21] ensures that, with probability at least $1/2$, the reduction $\mathcal{B}^{\mathsf{SIS}}$ can obtain a 3-fork involving the same tuple $(R^\star, \mathbf{y}_{\mathbf{S}}^\star, \mathbf{y}_{\mathbf{T}}^\star, \{\mathsf{Comm}_{K,j}^\star\}_{j=1}^\kappa)$ with pairwise

distinct answers $\mathsf{Chall}_{\mu^\star}^{(1)}, \mathsf{Chall}_{\mu^\star}^{(2)}, \mathsf{Chall}_{\mu^\star}^{(3)} \in \{1,2,3\}^\kappa$. With probability $1-(7/9)^\kappa$ it can be shown that there exists an index $j \in \{1,\ldots,\kappa\}$ for which the j-th bits of $\mathsf{Chall}_{\mu^\star}^{(1)}, \mathsf{Chall}_{\mu^\star}^{(2)}, \mathsf{Chall}_{\kappa^\star}^{(3)}$ are $(\mathsf{Chall}_{\mu^\star,j}^{(1)}, \mathsf{Chall}_{\mu^\star,j}^{(2)}, \mathsf{Chall}_{\mu^\star,j}^{(3)}) = (1,2,3)$. From the corresponding responses $(\mathsf{Resp}_{K,j}^{\star\,(1)}, \mathsf{Resp}_{K,j}^{\star\,(2)}, \mathsf{Resp}_{K,j}^{\star\,(3)})$, $\mathcal{B}^{\mathsf{SIS}}$ can extract witnesses $\mathbf{v}^\star = (\mathbf{v}_1^\star, \mathbf{v}_2^\star) \in \mathbb{Z}^{m_s} \times \mathbb{Z}^{m_s}$, $\tau^\star \in \{0,1\}^\ell$, $\mathbf{r}^\star \in \mathbb{Z}^{2m_s}$, $\tilde{\mathbf{w}}_s{}^\star \in \{0,1\}^{n\delta_{q_s}-1}$, $\mathbf{e}_u^\star \in \{0,1\}^{m_f}$, $\tilde{\mathbf{t}}^\star, \tilde{\mathbf{k}_0}{}^\star, \tilde{\mathbf{k}_1}{}^\star \in \{0,1\}^{\bar{m}}$ from the proof of knowledge $\pi_K^\star$ such that the verification equation (39) are satisfied. At this stage, $\mathcal{B}^{\mathsf{SIS}}$ halts and reports failure in the following situations:

- $\vartheta = 0$ but $\tau^\star \in \{0,1\}^\ell$ is recycled from some wallet generated by the $\mathcal{Q}_{\mathsf{withdraw}}$ oracle.
- $\vartheta = 0$ and $\mathcal{B}^{\mathsf{SIS}}$ did not correctly guess $i^\star$ and $t^\dagger$ (and thus the longest common prefix between $\tau^\star$ and one of the $\{\tau^{(i)}\}_{i=1}^{Q_w}$).
- $\vartheta = 1$ but the string $\tau^\star \in \{0,1\}^\ell$ never appeared in a wallet created by the $\mathcal{Q}_{\mathsf{withdraw}}$ oracle.
- $\vartheta = 1$ and $\tau^\star \in \{0,1\}^\ell$ was used by the $\mathcal{Q}_{\mathsf{withdraw}}$ oracle, but in a different query than the $i^\star$-th $\mathcal{Q}_{\mathsf{withdraw}}$-query.
- $\vartheta = 1$ and the extracted vectors $\mathbf{e}_u^\star \in \{0,1\}^{m_f}$, $\tilde{\mathbf{t}}^\star$, $\tilde{\mathbf{k}_0}{}^\star$, $\tilde{\mathbf{k}_1}{}^\star \in \{0,1\}^{\bar{m}}$, $\mathbf{v}^\star \in \mathbb{Z}^{2m_s}$, $\tilde{\mathbf{w}}_s^\star \in \{0,1\}^{n\delta_{q_s}-1}$, $\mathbf{r}^\star \in \mathbb{Z}^{2m_s}$, which satisfy (39), also result in the collision

$$\begin{aligned}\mathbf{D}_0 \cdot \mathbf{r}^\star + \mathbf{D}_1 \cdot \mathbf{e}_u^\star + \mathbf{D}_2 \cdot \tilde{\mathbf{t}}^\star + \mathbf{D}_3 \cdot \tilde{\mathbf{k}_0}{}^\star + \mathbf{D}_4 \cdot \tilde{\mathbf{k}_1}{}^\star \\ = \mathbf{D}_0 \cdot \mathbf{r}^{(i^\star)} + \mathbf{D}_1 \cdot \mathbf{e}_u^{(i^\star)} + \mathbf{D}_2 \cdot \tilde{\mathbf{t}}^{(i^\star)} + \mathbf{D}_3 \cdot \tilde{\mathbf{k}_0}{}^{(i^\star)} + \mathbf{D}_4 \cdot \tilde{\mathbf{k}_1}{}^{(i^\star)} \quad (46)\end{aligned}$$

  where $\left(\mathbf{r}^{(i^\star)}, \mathbf{e}_u^{(i^\star)}, \tilde{\mathbf{t}}^{(i^\star)}, \tilde{\mathbf{k}_0}{}^{(i^\star)}, \tilde{\mathbf{k}_1}{}^{(i^\star)}\right)$ are the vectors involved in the $i^\star$-th $\mathcal{Q}_{\mathsf{withdraw}}$ query.
- $\vartheta = 2$ and $\mathcal{E}_{\mathsf{Spend}}^{\mathcal{A}}$ outputs vectors $\mathbf{e}_u^\star \in \{0,1\}^{m_f}$, $\tilde{\mathbf{t}}^\star$, $\tilde{\mathbf{k}_0}{}^\star$, $\tilde{\mathbf{k}_1}{}^\star \in \{0,1\}^{\bar{m}}$, $\mathbf{v}^\star \in \mathbb{Z}^{2m_s}$, $\mathbf{r}^\star \in \mathbb{Z}^{2m_s}$ such that the collision (46) does not occur.

Let bad be the event that one of the above situations occurs. Since $\vartheta \hookleftarrow U(\{0,1,2\})$, $i^\star \hookleftarrow U([1,Q_w])$ and $t^\dagger \hookleftarrow U([1,\ell])$ (in the case $\vartheta = 0$) are chosen independently of $\mathcal{A}$'s view, the choice of ϑ is correct with probability $1/3$. If $\vartheta = 0$, $\mathcal{B}^{\mathsf{SIS}}$ correctly guesses $i^\star \hookleftarrow U([1,Q_w])$ and $t^\dagger \hookleftarrow U([1,\ell])$ with probability $1/(\ell \cdot Q_w)$ and, when $\vartheta = 1$, $\mathcal{B}$'s correctly chooses $i^\star \in [1,Q_w]$ with probability $1/Q_w$. We find

$$\Pr[\neg\mathsf{bad}] \geq \frac{1}{3 \cdot \ell \cdot Q_w}.$$

Assuming that $\neg\mathsf{bad}$ occurs, $\mathcal{B}^{\mathsf{SIS}}$ can solve the given SIS instance as follows.
• If $\vartheta = 0$, $\mathcal{B}$ knows small-norm $\mathbf{e}_u \in \mathbb{Z}^{m_s}$ and $\mathbf{R} \in \mathbb{Z}^{m_s \times (m_s/2)}$ such that $\mathbf{u} = \bar{\mathbf{A}} \cdot \mathbf{e}_u \bmod q_s$ and $\mathbf{D} = \bar{\mathbf{A}} \cdot \mathbf{R} \bmod q_s$. From the extracted $\tilde{\mathbf{w}}_s{}^\star \in \{0,1\}^{m_s/2}$ and $\mathbf{v}^\star = (\mathbf{v}_1^{\star T} \mid \mathbf{v}_2^{\star T})^T \in \mathbb{Z}^{2m_s}$, it can thus obtain a short integer vector

$$\mathbf{z} = \mathbf{v}_1^\star + \left(\mathbf{Q}_0 + \sum_{i=1}^{\ell} \tau^\star[i] \cdot \mathbf{Q}_i\right) \cdot \mathbf{v}_2^\star - \mathbf{R} \cdot \tilde{\mathbf{w}}_s{}^\star - \mathbf{e}_u \ \in \mathbb{Z}^{m_s}$$

such that $\bar{\mathbf{A}}\cdot\mathbf{z} = \mathbf{0} \bmod q_s$ and $\|\mathbf{z}\|_2 \leq \sigma^2 m_s^{3/2}(\ell+1)+\sqrt{2}m_s^{1/2}+\sigma m_s^{3/2}$. Moreover, with overwhelming probability, $\mathbf{z} \neq \mathbf{0}$ since the syndrome $\mathbf{u} \in \mathbb{Z}_{q_s}^n$ statistically hides $\mathbf{e}_u \in \mathbb{Z}^{m_s}$ in $\Lambda_{q_s}^{\mathbf{u}}(\bar{\mathbf{A}})$.

• If $\vartheta = 1$, $\mathcal{E}_{\mathsf{Spend}}^{\mathcal{A}}$ extracted witnesses $(\tau^\star, \mathbf{v}_1^\star, \mathbf{v}_2^\star, \tilde{\mathbf{w}_s}^\star, \mathbf{r}^\star, \mathbf{e}_u^\star, \tilde{\mathbf{t}}^\star, \tilde{\mathbf{k}_0}^\star, \tilde{\mathbf{k}_1}^\star)$ satisfying $\tau^\star = \tau^{(i^\star)}$, $\tilde{\mathbf{w}_s}^\star \neq \tilde{\mathbf{w}_s}^{(i^\star)}$ and

$$\mathbf{H}_{n,q_s-1} \cdot \tilde{\mathbf{w}_s}^{(i^\star)} = \mathbf{D}_0 \cdot \mathbf{r}^{(i^\star)} + \mathbf{D}_1 \cdot \mathbf{e}_u^{(i^\star)} + \mathbf{D}_2 \cdot \tilde{\mathbf{t}}^{(i^\star)} + \mathbf{D}_3 \cdot \tilde{\mathbf{k}_0}^{(i^\star)} + \mathbf{D}_4 \cdot \tilde{\mathbf{k}_1}^{(i^\star)}$$

and

$$\left[\, \mathbf{A} \;\middle|\; \mathbf{A}_0 \;\middle|\; \mathbf{A}_1 \;\middle|\; \ldots \;\middle|\; \mathbf{A}_\ell \;\middle|\; -\mathbf{D} \,\right] \cdot \left[\begin{array}{c} \mathbf{v}_1^\star \\ \hline \mathbf{v}_2^\star \\ \hline \tau^\star[1]\mathbf{v}_2^\star \\ \hline \vdots \\ \hline \tau^\star[\ell]\mathbf{v}_2^\star \\ \hline \tilde{\mathbf{w}_s}^\star \end{array}\right] = \mathbf{u} \bmod q_s. \tag{47}$$

From the setup phase, $\mathcal{B}^{\mathsf{SIS}}$ also knows small-norm vectors $\mathbf{v}_1, \mathbf{v}_2 \in \mathbb{Z}^{2m_s}$ and $\tilde{\mathbf{w}_s}^{(i^\star)} = \mathsf{vdec}_{n,q_s-1}(\mathbf{c}'_{\mathcal{U}}) \in \{0,1\}^{m_s/2}$ satisfying (43) (and which were used to respond the $i^\star$-th $\mathcal{Q}_{\mathsf{withdraw}}$-query), such that

$$\left[\, \mathbf{A} \;\middle|\; \mathbf{A}_0 \;\middle|\; \mathbf{A}_1 \;\middle|\; \ldots \;\middle|\; \mathbf{A}_\ell \;\middle|\; -\mathbf{D} \,\right] \cdot \left[\begin{array}{c} \mathbf{v}_1 \\ \hline \mathbf{v}_2 \\ \hline \tau^\star[1]\mathbf{v}_2 \\ \hline \vdots \\ \hline \tau^\star[\ell]\mathbf{v}_2 \\ \hline \tilde{\mathbf{w}_s}^{(i^\star)} \end{array}\right] = \mathbf{u} \bmod q_s. \tag{48}$$

Hence, by subtracting (48) from (47), $\mathcal{B}^{\mathsf{SIS}}$ obtains the vector

$$\mathbf{z} = (\mathbf{v}_1^\star - \mathbf{v}_1) + (\mathbf{S}_0 + \sum_{j=1}^{\ell} \tau^\star[j] \cdot \mathbf{S}_j) \cdot (\mathbf{v}_2^\star - \mathbf{v}_2) \; + \mathbf{R} \cdot \Big(\tilde{\mathbf{w}_s}^\star - \tilde{\mathbf{w}_s}^{(i^\star)}\Big), \tag{49}$$

which is a small-norm vector $\mathbf{z} \in \mathbb{Z}^{m_s}$ in $\Lambda_{q_s}^{\perp}(\bar{\mathbf{A}})$ (its norm can be bounded by $\|\mathbf{z}\|_2 \leq \sqrt{2}\sigma^2 m^{3/2}(\ell+2)+\sigma m^{3/2}$) and we argue that $\mathbf{z} \neq \mathbf{0}$ with high probability. Indeed, since $\mathsf{vdec}_{n,q_s-1} : \mathbb{Z}_{q_s}^n \to \{0,1\}^{m_s/2}$ is an injective function, we know that $\tilde{\mathbf{w}_s}^\star \neq \tilde{\mathbf{w}_s}^{(i^\star)}$ so long as $\neg\mathsf{bad}$ occurs. In this case, the last term of (49) does not cancel out in $\mathbb{Z}^{m_s}$, so that we cannot have $(\mathbf{v}_1^\star, \mathbf{v}_2^\star) = (\mathbf{v}_1, \mathbf{v}_2)$. The claim follows from the fact that, conditionally on $PK_{\mathcal{B}}$, the columns of $\{\mathbf{S}_j\}_{j=0}^{\ell}$ still retain a large amount of entropy. This implies that we can only have $\mathbf{z} = \mathbf{0}^m$ with negligible probability.

• If $coin = 2$, the collision (46) immediately provides $\mathcal{B}^{\mathsf{SIS}}$ with a SIS solution

$$\begin{aligned}\mathbf{z} = (\mathbf{r}_0^\star - \mathbf{r}_0^{(i^\star)}) + \mathbf{Q}_0'' \cdot (\mathbf{r}_1^\star - \mathbf{r}_1^{(i^\star)}) + \mathbf{Q}_1 \cdot (\mathbf{e}_u^\star - \mathbf{e}_u^{(i^\star)}) + \mathbf{Q}_2 \cdot \big(\tilde{\mathbf{t}}^\star - \tilde{\mathbf{t}}^{(i^\star)}\big) \\ + \mathbf{Q}_3 \cdot \big(\tilde{\mathbf{k}_0}^\star - \tilde{\mathbf{k}_0}^{(i^\star)}\big) + \mathbf{Q}_4 \cdot \big(\tilde{\mathbf{k}_1}^\star - \tilde{\mathbf{k}_1}^{(i^\star)}\big) \in \mathbb{Z}^{m_s},\end{aligned} \tag{50}$$

which has norm smaller than $\|\mathbf{z}\|_2 \leq \sigma^2\sqrt{2}m_s^{3/2} + \sqrt{2m_s} + 4\sigma m_s\sqrt{2m\log q}$ and is in $\Lambda_{q_s}^{\perp}(\bar{\mathbf{A}})$. Moreover, due to the large amount of entropy retained by the columns $\mathbf{Q}_0'', \mathbf{Q}_1, \mathbf{Q}_2, \mathbf{Q}_3, \mathbf{Q}_4$ conditionally on $(\mathbf{D}_0', \mathbf{D}_0'', \mathbf{D}_1, \mathbf{D}_2, \mathbf{D}_3, \mathbf{D}_4)$, the obtained vector $\mathbf{z} \in \mathbb{Z}^{m_s}$ is non-zero with overwhelming probability since $(\tilde{\mathbf{k}_0}^{\star}, \tilde{\mathbf{k}_1}^{\star}) \neq (\tilde{\mathbf{k}_0}^{(i^\star)}, \tilde{\mathbf{k}_1}^{(i^\star)})$ by hypothesis. □

D.2 Proof of Theorem 3

The proof relies on the following lemma which rules out some undesirable event when the domain of the PRF is restricted to have polynomial size.

Lemma 6. *Let $\mathbf{k}, \mathbf{k}' \hookleftarrow U(\mathbb{Z}_q^m)$ be independently sampled PRF seeds. The probability that the functions $F_{\mathbf{k}}, F_{\mathbf{k}'} : \{0,1\}^L \to \mathbb{Z}_p^m$, which are defined as*

$$F_{\mathbf{k}}(J) = \lfloor \prod_{i=1}^{L} \mathbf{P}_{J[L+1-i]} \cdot \mathbf{k} \rfloor_p, \qquad F_{\mathbf{k}'}(J) = \lfloor \prod_{i=1}^{L} \mathbf{P}_{J[L+1-i]} \cdot \mathbf{k}' \rfloor_p,$$

have intersecting ranges is at most $2^{2L}/p^m$.

Proof. For a given value $\mathbf{y}_T = F_{\mathbf{k}}(J) \in \mathbb{Z}_p^m$, the number of vectors in $\mathbb{Z}_q^m$ that round to $\mathbf{y}_T$ is $(q/p)^m$. If we fix the first PRF key $\mathbf{k} \in \mathbb{Z}_q^m$, there are $2^L \cdot (q/p)^m$ "forbidden" values for the unrounded value $\prod_{i=1}^{L} \mathbf{P}_{J[L+1-i]} \cdot \mathbf{k}' \bmod q$ when we choose a different random key $\mathbf{k}' \hookleftarrow U(\mathbb{Z}_q^m)$. Since $\mathbf{P}_0, \mathbf{P}_1 \in \{0,1\}^{m\times m}$ are $\mathbb{Z}_q$-invertible, this implies that there are $2^{2L} \cdot (q/p)^m$ forbidden values for the vector $\mathbf{k}' \in \mathbb{Z}_q^m$. Since $\mathbf{k}' \in \mathbb{Z}_q^m$ is chosen uniformly in $\mathbb{Z}_q^m$, the probability that the two ranges intersect is at most $2^{2L}/p^m$. □

Proof (of Theorem 3). For the sake of contradiction, let us assume that a PPT adversary $\mathcal{A}$ can produce two coins for which the Identify algorithm fails after Q_w execution of the withdrawal protocol in interaction with the challenger. We build an algorithm $\mathcal{B}^{\mathsf{SIS}}$ that uses $\mathcal{A}$ to solve an instance $(\bar{\mathbf{A}}, \beta) \in \mathbb{Z}_{q_s}^{n\times m_s} \times \mathbb{R}$ of the $\mathsf{SIS}_{n,m_s,q_s,\beta}^{\infty}$ problem or find a collision for $H_0 : \{0,1\}^* \to \mathbb{Z}_p^m$.

Let $coin_1^\star = (R_1^\star, \mathbf{y}_S^\star, \mathbf{y}_{T,1}^\star, \pi_{K,1}^\star)$, $coin_2^\star = (R_2^\star, \mathbf{y}_S^\star, \mathbf{y}_{T,2}^\star, \pi_{K,2}^\star)$ be the two coins produced by the adversary $\mathcal{A}$ at the end of the game. By hypothesis, we know that these two coins contain the same serial number. If we define $\mathcal{E}_{\mathsf{Withdraw}}^{\mathcal{A}}$ to be the knowledge extractor (as defined in the proof of Theorem 2) of the argument of knowledge generated by the prover at step 1 of Withdraw, we observe that each successful witness extraction allows $\mathcal{E}_{\mathsf{Withdraw}}^{\mathcal{A}}$ to reconstruct all wallets $\mathcal{W}_f = (\mathbf{e}_{u,f}, \mathbf{t}_f, \mathbf{k}_{0,f}, \mathbf{k}_{1,f}, sig_{\mathcal{B},f}, 0)$ generated by $\mathcal{Q}_{\mathsf{withdraw}}$ (for $f \in [1, Q_w]$) and, from these, all the serial numbers $\{\mathbf{y}_{S,f,j}\}_{f\in[1,Q_w],j\in[0,2^L-1]}$ that can be legally generated from $\{\mathcal{W}_f\}_{f=1}^{Q_w}$. Let T be the database of these serial numbers, which $\mathcal{B}^{\mathsf{SIS}}$ gradually constructs via $\mathcal{E}_{\mathsf{Withdraw}}^{\mathcal{A}}$ when simulating the $\mathcal{Q}_{\mathsf{withdraw}}$ oracle.

We assume that, at the end of the game, the common serial number $\mathbf{y}_S^\star$ of $\{coin_1^\star, coin_2^\star\}$ appears in T. Otherwise, $\mathcal{A}$ would be an adversary against the

balance property (which would contradict the SIS assumption in the random oracle model, as shown by Theorem 2).

Moreover, with overwhelming probability, $\mathbf{y}_S^\star$ appears *exactly* once in T. Indeed, the only situations where $\mathbf{y}_S^\star$ can occur more than once in T is when the pseudo-random functions from which serial numbers $\mathbf{y}_S \in \mathbb{Z}_p^m$ are derived have intersecting ranges on different keys: namely, we have

$$F_{\mathbf{k}}(J) = \lfloor \prod_{i=1}^{L} \mathbf{P}_{J[L+1-i]} \cdot \mathbf{k} \rfloor_p = F_{\mathbf{k}'}(J') = \lfloor \prod_{i=1}^{L} \mathbf{P}_{J'[L+1-i]} \cdot \mathbf{k}' \rfloor_p$$

for distinct seeds $\mathbf{k} \neq \mathbf{k}' \in \mathbb{Z}_q^m$ and possibly distinct inputs J, J'. However, since the PRF keys $\mathbf{k}, \mathbf{k}' \in \mathbb{Z}_q^m$ are jointly generated by $\mathcal{U}$ and $\mathcal{B}$ during the $\mathsf{Withdraw}$ protocol, they are independent and uniformly random in $\mathbb{Z}_q^m$ as long as either $\mathcal{B}$ or $\mathcal{U}$ is honest. Since J, J' live a polynomial-size domain $[0, 2^L - 1]$, the functions $F_{\mathbf{k}}(.)$, $F_{\mathbf{k}'}(.)$ have non-disjoint ranges with negligible probability $2^{2L}/p^m$, as shown by Lemma 6.

During the game, the reduction $\mathcal{B}^{\mathsf{SIS}}$ will interpret either $coin_1^\star$ or $coin_2^\star$ as a proof of knowledge of a forgery for the signature scheme with efficient protocols. To this end, $\mathcal{B}^{\mathsf{SIS}}$ interacts with $\mathcal{A}$ *exactly* in the same way as in the proof of Theorem 2. In particular, $\mathcal{B}^{\mathsf{SIS}}$ chooses a random coin $\vartheta \hookleftarrow U(\{0,1,2\})$ and an index $i^\star \hookleftarrow U([1, Q_w])$ so as to foresee $\mathcal{A}$'s attack strategy. The only change w.r.t. the proof of Theorem 2 is that, at the end of the first execution of $\mathcal{A}$ and before running additional executions of $\mathcal{A}$ with different random oracles, $\mathcal{B}^{\mathsf{SIS}}$ must determine which one of $coin_1^\star$ or $coin_2^\star$ must be interpreted as proving knowledge of a signature forgery. To this end, $\mathcal{B}^{\mathsf{SIS}}$ uses the information revealed by the knowledge extractor $\mathcal{E}^{\mathcal{A}}_{\mathsf{Withdraw}}$ during the first execution of $\mathcal{A}$.

Output. At the end of the game, when $\mathcal{A}$ outputs $coin_1^\star = (R_1^\star, \mathbf{y}_S^\star, \mathbf{y}_{T,1}^\star, \pi_{K,1}^\star)$ and $coin_2^\star = (R_2^\star, \mathbf{y}_S^\star, \mathbf{y}_{T,2}^\star, \pi_{K,2}^\star)$, the reduction $\mathcal{B}^{\mathsf{SIS}}$ immediately halts in the event that $R_1^\star = H_0(PK_{\mathcal{M}_1}, \mathtt{info}_1) = H_0(PK_{\mathcal{M}_2}, \mathtt{info}_2) = R_2^\star$ since, in this case, it has found a collision $(PK_{\mathcal{M}_1}, \mathtt{info}_1) \neq (PK_{\mathcal{M}_2}, \mathtt{info}_2)$ for H_0. Otherwise, $\mathcal{B}^{\mathsf{SIS}}$ looks up the database T of serial numbers $\{\mathbf{y}_{S,f,j}\}_{f \in [1,Q_w], j \in [0,2^L-1]}$ generated by $\mathcal{E}^{\mathcal{A}}_{\mathsf{Withdraw}}$ to find the unique pair of indexes $(f^\star, j^\star) \in [1, Q_w] \times [0, 2^L - 1]$ for which $\mathbf{y}_{S,f^\star,j^\star} = \mathbf{y}_S^\star$ (i.e., $\mathbf{y}_S^\star \in \mathbb{Z}_p^m$ matches the $j^\star$-th serial number produced by the $f^\star$-th withdrawn wallet $\mathcal{W}_{f^\star}$). Having determined the pair $(f^\star, j^\star)$, $\mathcal{B}^{\mathsf{SIS}}$ recalls the $f^\star$-th wallet

$$\mathcal{W}_{f^\star} = \big(\mathbf{e}_{u,f^\star}, \mathbf{t}_{f^\star}, \mathbf{k}_{0,f^\star}, \mathbf{k}_{1,f^\star}, sig_{\mathcal{B},f^\star} = (\tau_{f^\star}, \mathbf{v}_{f^\star}, \mathbf{r}_{f^\star}), 0\big)$$

issued by $\mathcal{Q}_{\mathsf{withdraw}}$. At this point, $\mathcal{B}^{\mathsf{SIS}}$ parses $j^\star \in [0, 2^L - 1]$ as a string $j^\star[1] \ldots j^\star[L] \in \{0,1\}^L$, and distinguishes two situations:

- If $coin_1^\star$ contains $\mathbf{y}_{T,1}^\star \in \mathbb{Z}_p^m$ such that

$$\mathbf{y}_{T,1}^\star = \mathbf{F} \cdot \mathbf{e}_{u,f^\star} + H_{\mathrm{FRD}}(R_1^\star) \cdot \lfloor \prod_{i=1}^{L} \mathbf{P}_{j^\star[L+1-i]} \cdot \mathbf{t}_{f^\star} \rfloor_p,$$

then $coin_2^\star$ must contain $\mathbf{y}_{T,2}^\star \in \mathbb{Z}_p^m$ such that

$$\mathbf{y}_{T,2}^\star \neq \mathbf{F} \cdot \mathbf{e}_{u,f^\star} + H_{\mathrm{FRD}}(R_2^\star) \cdot \lfloor \prod_{i=1}^{L} \mathbf{P}_{j^\star[L+1-i]} \cdot \mathbf{t}_{f^\star} \rfloor_p. \tag{51}$$

(Otherwise, $(coin_1^\star, coin_2^\star)$ would not defeat the $\mathsf{Identify}$ algorithm). In this case, $\mathcal{B}^{\mathsf{SIS}}$ can consider $coin_2^\star = (R_2^\star, \mathbf{y}_S^\star, \mathbf{y}_{T,2}^\star, \pi_{K,2}^\star)$ as an argument of knowledge of a fake message-signature pair $(\mathbf{e}_u^\star, \tilde{\mathbf{t}}^\star, \tilde{\mathbf{k}_0}^\star, \tilde{\mathbf{k}_1}^\star, sig_{\mathcal{B}}^\star)$ such that

$$(\mathbf{e}_u^\star, \tilde{\mathbf{t}}^\star) \neq (\mathbf{e}_{u,f^\star}, \tilde{\mathbf{t}}_{f^\star}).$$

Indeed, if $\mathbf{t}^\star = \mathbf{H}_{m,q-1} \cdot \tilde{\mathbf{t}}^\star \bmod q$, the soundness of the argument system ensures that $\mathbf{y}_{T,2}^\star = \mathbf{F} \cdot \mathbf{e}_u^\star + H_{\mathrm{FRD}}(R_2^\star) \cdot \lfloor \prod_{i=1}^{L} \mathbf{P}_{j^\star[L+1-i]} \cdot \mathbf{t}^\star \rfloor_p$. which contradicts (51) if $(\mathbf{e}_u^\star, \tilde{\mathbf{t}}^\star) = (\mathbf{e}_{u,f^\star}, \tilde{\mathbf{t}}_{f^\star})$.

- If $coin_1^\star$ contains $\mathbf{y}_{T,1}^\star \in \mathbb{Z}_p^m$ such that

$$\mathbf{y}_{T,1}^\star \neq \mathbf{F} \cdot \mathbf{e}_{u,f^\star} + H_{\mathrm{FRD}}(R_1^\star) \cdot \lfloor \prod_{i=1}^{L} \mathbf{P}_{j^\star[L+1-i]} \cdot \mathbf{t}_{f^\star} \rfloor_p,$$

a similar reasoning allows $\mathcal{B}^{\mathsf{SIS}}$ to consider $coin_1^\star = (R_1^\star, \mathbf{y}_S^\star, \mathbf{y}_{T,1}^\star, \pi_{K,1}^\star)$ as a non-interactive argument of knowledge of forgery $(\mathbf{e}_u^\star, \tilde{\mathbf{t}}^\star, \tilde{\mathbf{k}_0}^\star, \tilde{\mathbf{k}_1}^\star, sig_{\mathcal{B}}^\star)$ such that $(\mathbf{e}_u^\star, \tilde{\mathbf{t}}^\star) \neq (\mathbf{e}_{u,f^\star}, \tilde{\mathbf{t}}_{f^\star})$.

If $\mathcal{A}$ is a successful adversary against the identification property, there thus exists $d \in \{1, 2\}$ such that $coin_d^\star = (R_d^\star, \mathbf{y}_S^\star, \mathbf{y}_{T,d}^\star, \pi_{K,d}^\star)$ can be seen as proving knowledge of a forgery for the underlying signature scheme with efficient protocols. At this point, $\mathcal{B}^{\mathsf{SIS}}$ proceeds as in the end of the proof of Theorem 2. Namely, $\mathcal{B}^{\mathsf{SIS}}$ parses the proof of knowledge $\pi_{K,d}^\star$ as $(\{\mathsf{Comm}_{K,j}^\star\}_{j=1}^\kappa, \mathsf{Chall}_K^\star, \{\mathsf{Resp}_{K,j}^\star\}_{j=1}^\kappa)$ knowing that, with overwhelming probability, the random oracle H was queried on the input $(R_d^\star, \mathbf{y}_{\mathbf{S}}^\star, \mathbf{y}_{\mathbf{T},d}^\star, \{\mathsf{Comm}_{K,j}^\star\}_{j=1}^\kappa)$. With probability at least $\varepsilon' := \epsilon - 3^{-\kappa}$, the tuple $(R_d^\star, \mathbf{y}_{\mathbf{S}}^\star, \mathbf{y}_{\mathbf{T},d}^\star, \{\mathsf{Comm}_{K,j}^\star\}_{j=1}^\kappa)$ was indeed queried to H and coincides with the $\mu^\star$-th random oracle query for some $\mu^\star \leq Q_H$.

At this stage, $\mathcal{B}^{\mathsf{SIS}}$ appeals to $\mathcal{E}_{\mathsf{Spend}}^{\mathcal{A}}$ which replays the $\mathcal{A}$ up to $32 \cdot Q_H/(\epsilon - 3^{-\kappa})$ times with the *same* random tape and input as in the very first run. All queries are answered as in the initial run, except that a forking occurs at the $\mu^\star$-th H-query. Namely, the first $\mu^\star - 1$ hash queries – which coincide with those of the first run – receive the same answers $\mathsf{Chall}_1, \ldots, \mathsf{Chall}_{\mu^\star - 1}$ as initially. For this reason, the input of the $\mu^\star$-th hash query will be $(R_d^\star, \mathbf{y}_{\mathbf{S}}^\star, \mathbf{y}_{\mathbf{T},d}^\star, \{\mathsf{Comm}_{K,j}^\star\}_{j=1}^\kappa)$ as in the $\mu^\star$-th query of the initial execution. At the $\mu^\star$-th H-query, however, $\mathcal{A}$ starts interacting with a different random oracle H which outputs $\mathsf{Chall}'_{\mu^\star}, \ldots, r'_{Q_H}$ at each new execution of $\mathcal{A}$. The Forking Lemma of Brickell *et al.* [21] tells that, with probability at least $1/2$, $\mathcal{B}^{\mathsf{SIS}}$ can infer a 3-fork involving the same $(R_d^\star, \mathbf{y}_{\mathbf{S}}^\star, \mathbf{y}_{\mathbf{T},d}^\star, \{\mathsf{Comm}_{K,j}^\star\}_{j=1}^\kappa)$ with pairwise distinct answers $\mathsf{Chall}_{\mu^\star}^{(1)}, \mathsf{Chall}_{\mu^\star}^{(2)}, \mathsf{Chall}_{\mu^\star}^{(3)} \in \{1, 2, 3\}^\kappa$. With probability $1 - (7/9)^\kappa$ it can be shown that there exists an $j \in \{1, \ldots, \kappa\}$ for which the

j-th bits of $\mathsf{Chall}^{(1)}_{\mu^\star}, \mathsf{Chall}^{(2)}_{\mu^\star}, \mathsf{Chall}^{(3)}_{\kappa^\star}$ are $(\mathsf{Chall}^{(1)}_{\mu^\star,j}, \mathsf{Chall}^{(2)}_{\mu^\star,j}, \mathsf{Chall}^{(3)}_{\mu^\star,j}) = (1,2,3)$. From the responses $(\mathsf{Resp}^{\star}_{K,j}{}^{(1)}, \mathsf{Resp}^{\star}_{K,j}{}^{(2)}, \mathsf{Resp}^{\star}_{K,j}{}^{(3)})$, $\mathcal{B}^{\mathsf{SIS}}$ can extract witnesses $(\mathbf{v}_1^\star, \mathbf{v}_2^\star) \in \mathbb{Z}^{m_s} \times \mathbb{Z}^{m_s}$, $\tau^\star \in \{0,1\}^\ell$, $\mathbf{r}^\star = [\mathbf{r}_0^{\star T} \mid \mathbf{r}_1^{\star T}]^T \in \mathbb{Z}^{2m_s}$, $\tilde{\mathbf{w}}_s{}^\star \in \{0,1\}^{m_s/2}$, $\mathbf{e}_u^\star \in \{0,1\}^{m_f}$ and $\tilde{\mathbf{t}}^\star, \tilde{\mathbf{k}_0}{}^\star, \tilde{\mathbf{k}_1}{}^\star \in \{0,1\}^{\bar{m}}$ such that

$$\mathbf{A}_{\tau^\star} \cdot \begin{bmatrix} \mathbf{v}_1^\star \\ \mathbf{v}_2^\star \end{bmatrix} = \mathbf{u} + \mathbf{D} \cdot \tilde{\mathbf{w}}_s{}^\star$$
$$\mathbf{H}_{n,q_s-1} \cdot \tilde{\mathbf{w}}_s{}^\star = \mathbf{D}_0 \cdot \mathbf{r}^\star + \mathbf{D}_1 \cdot \mathbf{e}_u^\star + \mathbf{D}_2 \cdot \tilde{\mathbf{t}}^\star + \mathbf{D}_3 \cdot \tilde{\mathbf{k}_0}{}^\star + \mathbf{D}_4 \cdot \tilde{\mathbf{k}_1}{}^\star,$$

Now, $\mathcal{B}^{\mathsf{SIS}}$ halts and reports failure in the same situations as it did in the proof of Theorem 2. If we call bad the event that one of the above situations occurs, the same analysis as in the proof of Theorem 2 shows that $\Pr[\neg\mathsf{bad}] \geq \frac{1}{3 \cdot \ell \cdot Q_w}$. If $\mathcal{B}^{\mathsf{SIS}}$ is lucky enough to have $\neg\mathsf{bad}$, the given SIS instance can be solved as follows.

• If $\vartheta \in \{0,1\}$, $\mathcal{B}^{\mathsf{SIS}}$ succeeds in the same way and under exactly the same conditions as in the proof of Theorem 2.

• If $\vartheta = 2$, $\mathcal{B}^{\mathsf{SIS}}$ obtains a collision $(\mathbf{e}_u^\star, \tilde{\mathbf{t}}^\star) \neq (\mathbf{e}_{u,f^\star}, \tilde{\mathbf{t}}_{f^\star})$ of the form

$$\begin{aligned} &\mathbf{D}_0 \cdot \mathbf{r}^\star + \mathbf{D}_1 \cdot \mathbf{e}_u^\star + \mathbf{D}_2 \cdot \tilde{\mathbf{t}}^\star + \mathbf{D}_3 \cdot \tilde{\mathbf{k}_0}{}^\star + \mathbf{D}_4 \cdot \tilde{\mathbf{k}_1}{}^\star \\ &\quad = \mathbf{D}_0 \cdot \mathbf{r}_{f^\star} + \mathbf{D}_1 \cdot \mathbf{e}_{u,f^\star} + \mathbf{D}_2 \cdot \tilde{\mathbf{t}}_{f^\star} + \mathbf{D}_3 \cdot \tilde{\mathbf{k}}_{0,f^\star} + \mathbf{D}_4 \cdot \tilde{\mathbf{k}}_{1,f^\star} \end{aligned} \tag{52}$$

where $\tilde{\mathbf{t}}_{f^\star}, \tilde{\mathbf{k}}_{0,f^\star}, \tilde{\mathbf{k}}_{1,f^\star} \in \{0,1\}^{\bar{m}}$ and $\mathbf{r}_{f^\star} \in \mathbb{Z}^{2m_s}$, $\mathbf{e}_{u,f^\star} \in \{0,1\}^{m_f}$ are the vectors involved in the $f^\star$-th $\mathcal{Q}_{\mathsf{withdraw}}$ query. This reveals the SIS solution

$$\begin{aligned} \mathbf{z} = &(\mathbf{r}_0^\star - \mathbf{r}_{0,f^\star}) + \mathbf{Q}_0'' \cdot (\mathbf{r}_1^\star - \mathbf{r}_{1,f^\star}) + \mathbf{Q}_1 \cdot (\mathbf{e}_u^\star - \mathbf{e}_{u,f^\star}) \\ &+ \mathbf{Q}_2 \cdot (\tilde{\mathbf{t}}^\star - \tilde{\mathbf{t}}_{f^\star}) + \mathbf{Q}_3 \cdot (\tilde{\mathbf{k}_0}{}^\star - \tilde{\mathbf{k}}_{0,f^\star}) + \mathbf{Q}_4 \cdot (\tilde{\mathbf{k}_1}{}^\star - \tilde{\mathbf{k}}_{1,f^\star}) \in \mathbb{Z}^{m_s}, \end{aligned} \tag{53}$$

which is shorter than $\|\mathbf{z}\|_2 \leq \sigma^2\sqrt{2}m_s^{3/2} + \sqrt{2m_s} + 4\sigma m_s\sqrt{2m\log q}$ and lies in $\Lambda_{q_s}^\perp(\bar{\mathbf{A}})$. Due to the remaining entropy in the columns $\mathbf{Q}_0'', \mathbf{Q}_1, \mathbf{Q}_2, \mathbf{Q}_3, \mathbf{Q}_4$ conditionally on $(\mathbf{D}_0', \mathbf{D}_0'', \mathbf{D}_1, \mathbf{D}_2, \mathbf{D}_3, \mathbf{D}_4)$, the obtained vector $\mathbf{z} \in \mathbb{Z}^{m_s}$ is non-zero w.h.p. since $(\tilde{\mathbf{t}}^\star, \mathbf{e}_u^\star) \neq (\tilde{\mathbf{t}}_{f^\star}, \mathbf{e}_{u,f^\star})$ by hypothesis. □

D.3 Proof of Theorem 4

Proof. Like [23], we rely on the fact that each coin contains a built-in proof of knowledge of the user's private key $SK_{\mathcal{U}}$. The only way for the adversary to falsely accuse the user of having double-spent a coin is thus to craft a coin which has the same serial number as a honestly spent coin. In turn, this requires the adversary to fake a proof of knowledge of $SK_{\mathcal{U}}$.

More formally, let us assume that a PPT adversary $\mathcal{A}$ can output two coins $(coin_1^\star, coin_2^\star)$ for which $\mathsf{Identify}(\mathsf{par}, PK_{\mathcal{B}}, coin_1^\star, coin_2^\star) = PK_{\mathcal{U}}$ for some public key $PK_{\mathcal{U}}$ such that the common serial number $\mathbf{y}_S^\star$ of $coin_1^\star$ and $coin_2^\star$ was not used more than once by user $\mathcal{U}$. In the random oracle model, we describe a SIS solver $\mathcal{B}^{\mathsf{SIS}}$ that uses $\mathcal{A}$ to solve an instance of $\mathsf{SIS}_{m,q,\beta}$ problem: $\mathcal{B}^{\mathsf{SIS}}$ takes as

input $\bar{\mathbf{A}} \in \mathbb{Z}_p^{m \times m_f}$ and finds a non-zero short vector $\mathbf{w} \in \mathbb{Z}^{m_f}$ in $\Lambda_p^{\perp}(\bar{\mathbf{A}})$.

Algorithm $\mathcal{B}^{\mathsf{SIS}}$ generates par via the actual setup algorithm except that, at step 4 of this setup algorithm, $\mathcal{B}^{\mathsf{SIS}}$ sets $\mathbf{F} = \bar{\mathbf{A}} \in \mathbb{Z}_p^{m \times m_f}$. The distribution of par, however, is the same as in the real scheme. Then, $\mathcal{B}^{\mathsf{SIS}}$ starts interacting with $\mathcal{A}$ exactly as in the real game of Definition 8. In particular, at the i-th query to the $\mathcal{Q}_{\mathsf{GetKey}}$ oracle, $\mathcal{B}^{\mathsf{SIS}}$ picks a $SK_{\mathcal{U}_i} = \mathbf{e}_{u,i} \hookleftarrow U(\{0,1\}^{m_f})$ and computes $PK_{\mathcal{U}_i} = \mathbf{F} \cdot \mathbf{e}_{u,i} \in \mathbb{Z}_p^m$, which is returned to $\mathcal{A}$. Also, in all $\mathcal{Q}_{\mathsf{withdraw}}$-queries, $\mathcal{B}^{\mathsf{SIS}}$ faithfully runs the user's part of the protocol. At step 1 of each execution of $\mathsf{Withdraw}$, $\mathcal{B}^{\mathsf{SIS}}$ thus honestly generates an argument of knowledge of witnesses $(\mathbf{e}_u, \tilde{\mathbf{t}}, \tilde{\mathbf{k}}_0) \in \{0,1\}^{m_f} \times \{0,1\}^{\bar{m}} \times \{0,1\}^{\bar{m}}$ such that

$$\mathbf{c}_{\mathcal{U}} = \mathbf{D}_0' \cdot \mathbf{r}_0 + \mathbf{D}_1 \cdot \mathbf{e}_u + \mathbf{D}_2 \cdot \tilde{\mathbf{t}} + \mathbf{D}_3 \cdot \tilde{\mathbf{k}}_0 \in \mathbb{Z}_{q_s}^n,$$

and $PK_{\mathcal{U}} = \mathbf{F} \cdot \mathbf{e}_u \in \mathbb{Z}_p^m$. If the withdrawal protocol successfully terminates, $\mathcal{B}^{\mathsf{SIS}}$ stores a wallet $\mathcal{W} := \big(\mathbf{e}_u, \mathbf{t}, \mathbf{k}_0, \mathbf{k}_1, sig_{\mathcal{B}} = (\tau, \mathbf{v}, \mathbf{r}), J = 0\big)$ for later use in executions of the $\mathcal{Q}_{\mathsf{Spend}}$ oracle. At each invocation of $\mathcal{Q}_{\mathsf{Spend}}$, $\mathcal{B}^{\mathsf{SIS}}$ proceeds exactly as the real user $\mathcal{U}$ would. Finally, all queries to the random oracle H are answered in the standard way, by returning a uniformly random value in $\{1,2,3\}^{\kappa}$ (of course, the same response is given in case a given hash query is made more than once).

When the adversary $\mathcal{A}$ halts, it outputs two $coin_1^{\star} = (R_1^{\star}, \mathbf{y}_S^{\star}, \mathbf{y}_{T,1}^{\star}, \pi_{K,1}^{\star})$ and $coin_2^{\star} = (R_2^{\star}, \mathbf{y}_S^{\star}, \mathbf{y}_{T,2}^{\star}, \pi_{K,2}^{\star})$ that share the same serial number $\mathbf{y}_S^{\star} \in \mathbb{Z}_p^m$ and for which $\mathsf{Identify}$ outputs the public key $PK_{\mathcal{U}^{\star}}$ of some user $U^{\star}$. By hypothesis, we know that $\mathbf{y}_S^{\star} \in \mathbb{Z}_p^m$ appeared in at most one output of the $\mathcal{Q}_{\mathsf{Spend}}$ oracle. Moreover, we also know that $\mathbf{y}_{T,1}^{\star} \neq \mathbf{y}_{T,2}^{\star}$ since $\mathsf{Identify}$ would output $\perp$ otherwise. There thus exists $d \in \{1,2\}$ such that $coin_d^{\star} = (R_d^{\star}, \mathbf{y}_S^{\star}, \mathbf{y}_{T,d}^{\star}, \pi_{K,d}^{\star})$ was not returned by $\mathcal{Q}_{\mathsf{Spend}}$, which means that $\mathcal{B}^{\mathsf{SIS}}$ can interpret $coin_d^{\star}$ as a fake argument of knowledge of $SK_{\mathcal{U}^{\star}}$. At this point, $\mathcal{B}^{\mathsf{SIS}}$ exploits the property that the argument systems used in $\mathcal{Q}_{\mathsf{withdraw}}$ and $\mathcal{Q}_{\mathsf{Spend}}$ are statistically witness indistinguishable. From the $\mathcal{Q}_{\mathsf{GetKey}}$-query where $PK_{\mathcal{U}^{\star}}$ was generated, $\mathcal{B}^{\mathsf{SIS}}$ has recollection of a vector $SK_{\mathcal{U}^{\star}} = \mathbf{e}_u^{\star} \in \{0,1\}^{m_f}$ such that $PK_{\mathcal{U}^{\star}} = \mathbf{F} \cdot \mathbf{e}_u^{\star} \in \mathbb{Z}_p^m$. The result of [76, Lemma 8] implies that, with overwhelming probability, there exists at least another vector $\mathbf{e}_u \in \{0,1\}^{m_f}$ such that $\mathbf{e}_u \neq \mathbf{e}_u^{\star}$ and $PK_{\mathcal{U}^{\star}} = \mathbf{F} \cdot \mathbf{e}_u \bmod p$. The strategy of $\mathcal{B}^{\mathsf{SIS}}$ is thus to use $coin_d^{\star} = (R_d^{\star}, \mathbf{y}_S^{\star}, \mathbf{y}_{T,d}^{\star}, \pi_{K,d}^{\star})$ as a fake signature of knowledge of $SK_{\mathcal{U}^{\star}}$ when replaying the adversary $\mathcal{A}$ so as to extract a vector $\mathbf{e}_u^{\diamond} \in \{0,1\}^{m_f}$ such that $PK_{\mathcal{U}^{\star}} = \mathbf{F} \cdot \mathbf{e}_u^{\diamond} \bmod p$. The statistical WI property of the argument system then ensures that $\mathbf{e}_u^{\diamond} \neq \mathbf{e}_u^{\star}$, so that $\mathbf{e}_u^{\diamond} - \mathbf{e}_u^{\star} \in \{-1,0,1\}^{m_f}$ is a short non-zero vector of $\Lambda_p^{\perp}(\mathbf{F})$.

To compute $\mathbf{e}_u^{\diamond} \in \{0,1\}^{m_f}$, $\mathcal{B}^{\mathsf{SIS}}$ proceeds by replaying $\mathcal{A}$ sufficiently many times and applying the Improved Forking Lemma of *et al.* [21]. If we parse $\pi_{K,d}^{\star}$ as $(\{\mathsf{Comm}_{K,j}^{\star}\}_{j=1}^{\kappa}, \mathsf{Chall}_K^{\star}, \{\mathsf{Resp}_{K,j}^{\star}\}_{j=1}^{\kappa})$, $\mathcal{A}$ must have queried the random oracle H on the input $(R_d^{\star}, \mathbf{y}_S^{\star}, \mathbf{y}_{T,d}^{\star}, \pi_{K,d}^{\star})$. Otherwise, we would only have $\mathsf{Chall}_K^{\star} = H(R_d^{\star}, \mathbf{y}_S^{\star}, \mathbf{y}_{T,d}^{\star}, \{\mathsf{Comm}_{K,j}^{\star}\}_{j=1}^{\kappa})$ with negligible probability $3^{-\kappa}$. With probability at least $\varepsilon' := \varepsilon - 3^{-\kappa}$, the tuple $(R_d^{\star}, \mathbf{y}_S^{\star}, \mathbf{y}_{T,d}^{\star}, \{\mathsf{Comm}_{K,j}^{\star}\}_{j=1}^{\kappa})$ was the input of the $\mu^{\star}$-th H-query for some $\mu^{\star} \leq Q_H$.

$\mathcal{B}^{\mathsf{SIS}}$ thus runs the adversary $\mathcal{A}$ up to $32 \cdot Q_H/(\varepsilon - 3^{-\kappa})$ times with the *same*

random tape and the *same* input as in the very first execution. The first $\mu^\star - 1$ H-queries obtain the same responses $\mathsf{Chall}_1, \ldots, \mathsf{Chall}_{\mu^\star-1}$ as in the first execution. The forking occurs at the $\mu^\star$-th H-query which will also be made for the same input $(R_d^\star, \mathbf{y}_S^\star, \mathbf{y}_{T,d}^\star, \{\mathsf{Comm}_{K,j}^\star\}_{j=1}^\kappa)$. At the $\mu^\star$-th query, $\mathcal{A}$ thus obtains fresh random oracle outputs $\mathsf{Chall}'_{\mu^\star}, \ldots, \mathsf{Chall}'_{Q_H}$ at each new execution. The Improved Forking Lemma *et al.* [21] implies that, with probability $> 1/2$, these repeated executions yield a 3-fork involving the tuple $(R_d^\star, \mathbf{y}_S^\star, \mathbf{y}_{T,d}^\star, \{\mathsf{Comm}_{K,j}^\star\}_{j=1}^\kappa)$ and with pairwise distinct responses $\mathsf{Chall}_{\mu^\star}^{(1)}, \mathsf{Chall}_{\mu^\star}^{(2)}, \mathsf{Chall}_{\mu^\star}^{(3)} \in \{1,2,3\}^\kappa$. Since the forgeries of the 3-fork all correspond to the same $(R_d^\star, \mathbf{y}_S^\star, \mathbf{y}_{T,d}^\star, \{\mathsf{Comm}_{K,j}^\star\}_{j=1}^\kappa)$ and since $\{\mathsf{Comm}_{K,j}^\star\}_{j=1}^\kappa$ contain commitments to the counter $J^\star \in \{0,1\}^L$ and the decompositions

$$(\tilde{\mathbf{k}_0}^\star, \tilde{\mathbf{k}_1}^\star) = \big(\mathsf{vdec}_{m,q-1}(\mathbf{k}_0^\star), \mathsf{vdec}_{m,q-1}(\mathbf{k}_1^\star)\big) \in \{0,1\}^{\bar{m}} \times \{0,1\}^{\bar{m}}$$

of the PRF key $\mathbf{k}^\star = \mathbf{k}_0^\star + \mathbf{k}_1^\star \bmod q$, they lead $\mathsf{Identify}$ to reveal the same user's public key $PK_{\mathcal{U}^\star}$. With probability $1 - (7/9)^\kappa$, it can be shown that there exists $j \in \{1, \ldots, \kappa\}$ such that the j-th bits of $\mathsf{Chall}_{\mu^\star}^{(1)}, \mathsf{Chall}_{\mu^\star}^{(2)}, \mathsf{Chall}_{\mu^\star}^{(3)}$ are $(\mathsf{Chall}_{\mu^\star,j}^{(1)}, \mathsf{Chall}_{\mu^\star,j}^{(2)}, \mathsf{Chall}_{\mu^\star,j}^{(3)}) = (1,2,3)$. From the corresponding responses $(\mathsf{Resp}_{K,j}^\star{}^{(1)}, \mathsf{Resp}_{K,j}^\star{}^{(2)}, \mathsf{Resp}_{K,j}^\star{}^{(3)})$, $\mathcal{B}^{\mathsf{SIS}}$ extracts a binary vector $\mathbf{e}_u^\diamond \in \{0,1\}^{m_f}$ such that $PK_{\mathcal{U}} = \mathbf{F} \cdot \mathbf{e}_u^\diamond \in \mathbb{Z}_p^m$.

Due to the statistical witness indistinguishability of the Stern-like proof of knowledge which is used in $\mathsf{Withdraw}$ and Spend, with probability at least $1/2$, we have $\mathbf{e}_u^\diamond \neq \mathbf{e}_u^\star$, so that the difference $\mathbf{w} = \mathbf{e}_u^\diamond - \mathbf{e}_u^\star \in \{-1,0,1\}^{m_f}$ is a valid solution of the SIS instance. □

D.4 Proof of Theorem 5

Proof. Recall that, in the game of Definition 6, the adversary plays the role of the bank that colludes with merchants in order to decide whether the $\mathcal{Q}_{\mathsf{Spend}}$ oracle runs the real Spend protocol or a simulator. We consider an adversary $\mathcal{A}$ that makes Q_U queries to the $\mathcal{Q}_{\mathsf{GetKey}}$ oracle and Q_w queries for each user's public key created by $\mathcal{Q}_{\mathsf{GetKey}}$.

The proof proceeds with a sequence of games which begins with Game 0 where the challenger's bit is $d = 1$ and ends with a game where this bit is $d = 0$. In other words, the adversary interacts with a $\mathcal{Q}_{\mathsf{Spend}}$ oracle that runs actual executions of the Spend protocol on behalf of users Game 0. In the last game, the adversary interacts with a simulator $(\mathsf{SimParGen}, \mathsf{SimSpend})$ which does not have access to users wallets nor their keys or even the value of their counters $J \in \{0, \ldots, 2^L - 1\}$.

For each i, we denote by W_i the event that the adversary outputs $d' = 1$ in Game i. We will prove that, under the LWE assumption, the adversary outputs $d' = 1$ with about the same probabilities in Game 0 and in the final game.

Game 0**:** This is the real game where the adversary $\mathcal{A}$ has access to a $\mathcal{Q}_{\mathsf{GetKey}}$ oracle which it uses to create new honest users to whom it can issue wallets

via the $\mathcal{Q}_{\mathsf{withdraw}}$ oracle. At each $\mathcal{Q}_{\mathsf{Spend}}$-query, the challenger runs the real Spend oracle on behalf of the honest user chosen by the adversary. We call W_0 the probability that $\mathcal{A}$ outputs 1.

Game 1**:** In this game, we bring a first modification to the $\mathcal{Q}_{\mathsf{Spend}}$ oracle. Namely, instead of faithfully generating the ZK argument of knowledge at step 4, the oracle generates a simulated proof via the statistical honest-verifier zero-knowledge simulator (recall that statistical HVZK is preserved by parallel repetitions of the underlying protocol with ternary challenges) and by programming the random oracle $H : \{0,1\}^* \to \{1,2,3\}^\kappa$. At each query, the simulator may have to program $H : \{0,1\}^* \to \{1,2,3\}^\kappa$ for an input where it is already defined (in which case the simulation fails), but this only occurs with probability $Q_S \cdot Q_H/3^\kappa$, if Q_S and Q_H denote the number of $\mathcal{Q}_{\mathsf{Spend}}$-queries and the number of H-queries, respectively. If the simulation never fails, the adversary's view is statistically close to that of Game 0. We have $|\Pr[W_1] - \Pr[W_0]| \leq Q_S \cdot Q_H/3^\kappa + 2^{-\lambda}$.

We now proceed with a sub-sequence of hybrid games where we gradually modify the output distribution of the $\mathcal{Q}_{\mathsf{Spend}}$ oracle. At each step, we tamper with the output distribution of all spending queries involving a different wallet. In Game $2.(k,f)$, the responses of $\mathcal{Q}_{\mathsf{Spend}}$ depend on the index $i \in \{1,\ldots,Q_U\}$ of the involved user and the index $j \in \{1,\ldots,Q_w\}$ of the wallet. For all queries involving users $\mathcal{U}_i$ (i.e., the user involved in the i-th $\mathcal{Q}_{\mathsf{GetKey}}$-query) such that $i \in \{1,\ldots,k-1\}$, the spending oracle replaces $\mathbf{y}_S$ by a random vector of $\mathbb{Z}_p^m$. For all queries involving users $i \in \{k+1,\ldots,Q_U\}$, $\mathcal{Q}_{\mathsf{Spend}}$ computes $\mathbf{y}_S$ via real PRF evaluations for $\mathcal{Q}_{\mathsf{Spend}}$-queries . As for queries involving the k-th user, their index depends on the index $f \in \{1,\ldots,Q_w\}$ of the wallet. The security tags $\mathbf{y}_T \in \mathbb{Z}_p^m$ are always faithfully computed using $\mathbf{e}_u$ and actual PRF values for each $k \in \{1,\ldots,Q_U\}$ and $f \in \{1,\ldots,Q_w\}$.

For convenience, we define Game $2.(k,0)$ to be identical to Game $2.(k-1,Q_w)$ for each $k \in \{1,\ldots,Q_U\}$ and Game $2.(0,f)$ as being identical to Game 1 for all $f \in \{0,\ldots,Q_w\}$.

Game $2.(k,f)$ $(1 \leq k \leq Q_U,\ 1 \leq f \leq Q_w)$**:** In this game, the adversary has access to a hybrid $\mathcal{Q}_{\mathsf{Spend}}$ oracle. At each $\mathcal{Q}_{\mathsf{Spend}}$-query $(PK_{\mathcal{B}}, i, j, PK_{\mathcal{M}}, \mathtt{info})$, the response is the same as in Game 2 if $i < k$ or $i = k$ and $j \leq f$ (in particular, the spent coin $(R, \mathbf{y}_S, \mathbf{y}_T, \pi_K)$ is computed using actual PRF evaluations $\mathbf{y}_S$ and $\mathbf{y}_T$). If $i > k$ or $i = k$ and $j > f$, instead of computing the serial number as $\mathbf{y}_S = \lfloor \prod_{i=1}^{L} \mathbf{P}_{J[L+1-i]} \cdot \mathbf{k} \rfloor_p \in \mathbb{Z}_p^m$, the oracle defines $\mathbf{y}_S$ to be a truly random vector $\mathbf{y}_S \hookleftarrow U(\mathbb{Z}_p^m)$ chosen uniformly in $\mathbb{Z}_p^m$.

We claim that any significant change in the output distribution of $\mathcal{A}$ between Game $2.(k,f)$ and Game $2.(k,f-1)$ (or between Game $2.(k,1)$ and Game $2.(k-1,Q_w)$) would imply a distinguisher $\mathcal{B}^{\mathsf{LWE}}$ against the PRF, which would in turn contradict the LWE assumption, as shown by [17, Theorem 5.1].

The reduction $\mathcal{B}^{\mathsf{LWE}}$ interacts with a PRF challenger which has a random key $\mathbf{k}^\star \hookleftarrow U(\mathbb{Z}_q^m)$. Depending on the value of an internally flipped random

coin $\omega \in_R \{0,1\}$, the PRF challenger either always outputs the correct PRF evaluation $F_{\mathbf{k}}(J) = \lfloor \prod_{i=1}^{L} \mathbf{P}_{J[L+1-i]} \cdot \mathbf{k}^\star \rfloor_p \in \mathbb{Z}_p^m$ when queried on an L-bit string $J = J[1] \ldots J[L] \in \{0,1\}^L$ or always outputs random elements of $\mathbb{Z}_p^m$. Our distinguisher $\mathcal{B}^{\mathsf{LWE}}$ obtains from its challenger the parameters p, q and public matrices $\mathbf{P}_0, \mathbf{P}_1 \in \{0,1\}^{m \times m}$, which it uses to define par. The remaining components of par are generated according to the specification of ParGen. At each $\mathcal{Q}_{\mathsf{withdraw}}$-query, $\mathcal{B}^{\mathsf{LWE}}$ executes the actual withdrawal protocol in interaction with the bank-executing adversary $\mathcal{A}$. When $\mathcal{A}$ queries the $\mathcal{Q}_{\mathsf{Spend}}$ oracle on the input $(PK_{\mathcal{B}}, i, j, PK_{\mathcal{M}}, \mathtt{info})$, $\mathcal{B}^{\mathsf{LWE}}$ first recalls the j-th wallet $\mathcal{W}_j = \big(\mathbf{e}_u, \mathbf{t}, \mathbf{k}_0, \mathbf{k}_1, sig_{\mathcal{B}}, J\big)$ obtained by user $\mathcal{U}_i$. If no such wallet exists or if the counter J as reached its maximal value $J = 2^L - 1$, $\mathcal{B}^{\mathsf{LWE}}$ outputs $\perp$. Otherwise, it computes $R = H_0(PK_{\mathcal{M}}, \mathtt{info}) \in \mathbb{Z}_p^m$ and does the following:

- If $i < k$ or $i = k$ and $j \leq f$, $\mathcal{B}^{\mathsf{LWE}}$ picks $\mathbf{y}_S \hookleftarrow U(\mathbb{Z}_p^m)$ at random, computes $\mathbf{y}_T = \mathbf{F} \cdot \mathbf{e}_u + H_{\mathrm{FRD}}(R) \cdot \lfloor \prod_{i=1}^{L} \mathbf{P}_{J[L+1-i]} \cdot \mathbf{t} \rfloor_p$, and simulates π_K by programming $H : \{0,1\}^* \to \{1,2,3\}^\kappa$.
- If $i > k$ or $i = k$ and $j > f$, the reduction $\mathcal{B}^{\mathsf{LWE}}$ computes

$$\mathbf{y}_S = \lfloor \prod_{i=1}^{L} \mathbf{P}_{J[L+1-i]} \cdot \mathbf{k} \rfloor_p, \qquad \mathbf{y}_T = \mathbf{F} \cdot \mathbf{e}_u + H_{\mathrm{FRD}}(R) \cdot \lfloor \prod_{i=1}^{L} \mathbf{P}_{J[L+1-i]} \cdot \mathbf{t} \rfloor_p,$$

  where $\mathbf{k} = \mathbf{k}_0 + \mathbf{k}_1 \bmod q$. Then, it simulates π_K by programming the random oracle as in Game 1.
- If $i = k$ and $j = f$, $\mathcal{B}^{\mathsf{LWE}}$ invokes its PRF challenger on the input $J \in \{0,1\}^L$. The latter challenger returns $\mathbf{y}_S = F_{\mathbf{k}}(J) = \lfloor \prod_{i=1}^{L} \mathbf{P}_{J[L+1-i]} \cdot \mathbf{k}^\star \rfloor_p$ if $\omega = 1$ and a random $\mathbf{y}_S \hookleftarrow U(\mathbb{Z}_p^m)$ if $\omega = 0$. The response $\mathbf{y}_S \in \mathbb{Z}_p^m$ is used by $\mathcal{B}^{\mathsf{LWE}}$ to build a coin in the following way: $\mathcal{B}^{\mathsf{LWE}}$ faithfully computes the security tag $\mathbf{y}_T = \mathbf{F} \cdot \mathbf{e}_u + H_{\mathrm{FRD}}(R) \cdot \lfloor \prod_{i=1}^{L} \mathbf{P}_{J[L+1-i]} \cdot \mathbf{t} \rfloor_p$ and simulates the zero-knowledge argument π_K by programming the random oracle H as in Game 1.

Having completed the generation of $coin = (R, \mathbf{y}_S, \mathbf{y}_T, \pi_K)$, $\mathcal{B}^{\mathsf{LWE}}$ returns $coin$ to $\mathcal{A}$ and increments the value J of the counter of $\mathcal{W}_j$.

At the end of the game, $\mathcal{A}$ outputs a bit $d' \in \{0,1\}$ and $\mathcal{B}^{\mathsf{LWE}}$ outputs the same d' as a guess for the value of $\omega \in \{0,1\}$.

We claim that $\mathcal{B}^{\mathsf{LWE}}$ is playing Game 2.$(k, f-1)$ with $\mathcal{A}$ if the bit ω chosen by $\mathcal{B}^{\mathsf{LWE}}$'s challenger is $\omega = 1$. Indeed, the two games only differ when $(i,j) = (k,f)$. In this case, the wallet $\mathcal{W}_f = \big(\mathbf{e}_u, \mathbf{t}, \mathbf{k}_0, \mathbf{k}_1, sig_{\mathcal{B}}, J\big)$ obtained by $\mathcal{B}^{\mathsf{LWE}}$ is such that the challenger's key $\mathbf{k}^\star$ differs from $\mathbf{k}_0 + \mathbf{k}_1 \bmod q$ with high probability. However, at step 1 of the $\mathsf{Withdraw}$ protocol, $\mathcal{A}$ can only see a statistically hiding commitment $\mathbf{c}_{\mathcal{U}} \in \mathbb{Z}_{q_s}^n$ to $(\mathbf{e}_u, \tilde{\mathbf{t}}, \tilde{\mathbf{k}}_0)$ – recall that $\mathbf{D}_0' \cdot \mathbf{r}_0 \bmod q_s$ is statistically uniform over $\mathbb{Z}_{q_s}^n$ if $\mathbf{r}_0 \hookleftarrow D_{\mathbb{Z}^{m_s}, \sigma}$ – and a statistically WI proof of knowledge of vectors $(\mathbf{r}_0, \mathbf{e}_u, \tilde{\mathbf{t}}, \tilde{\mathbf{k}}_0)$ which satisfy (19) and $PK_{\mathcal{U}} = \mathbf{F} \cdot \mathbf{e}_u \bmod p$. For this reason, if we define $\mathbf{k}_0^\star = \mathbf{k}^\star - \mathbf{k}_1 \bmod q$, $\mathbf{k}_{00} = \mathsf{vdec}_{m,q-1}(\mathbf{k}_0^\star) \in \{0,1\}^{m\delta_{q-1}}$ and

$$\mathbf{c}_{\mathcal{U}}^\star = \mathbf{c}_{\mathcal{U}} - \mathbf{D}_1 \cdot \mathbf{e}_u - \mathbf{D}_2 \cdot \tilde{\mathbf{t}} - \mathbf{D}_3 \cdot \mathbf{k}_{00},$$

the shifted lattice $\Lambda_{q_s}^{\mathbf{c}_{\mathcal{U}}^\star}(\mathbf{D}_0')$ contains a vector $\mathbf{r}_0^\star \in D_{\Lambda_{q_s}^{\mathbf{c}_{\mathcal{U}}^\star}(\mathbf{D}_0'),\sigma}$ which satisfies

$$\mathbf{c}_{\mathcal{U}} = \mathbf{D}_0' \cdot \mathbf{r}_0^\star + \mathbf{D}_1 \cdot \mathbf{e}_u + \mathbf{D}_2 \cdot \tilde{\mathbf{t}} + \mathbf{D}_3 \cdot \mathbf{k}_{00} \mod q_s.$$

The statistical witness indistinguishability of the argument system (which is preserved by its parallel repetitions) ensures that $\mathcal{A}$'s view in the Withdraw protocol is statistically close to the view that would be generated using the witnesses $(\mathbf{r}_0^\star, \mathbf{e}_u, \tilde{\mathbf{t}}, \mathbf{k}_{00})$. We conclude that $\mathcal{B}^{\mathsf{LWE}}$ is playing Game $2.(k, f-1)$ (or Game $2.(k-1, Q_w)$ if $f = 0$) with $\mathcal{A}$ if the PRF's challenger's bit is $\omega = 1$. If $\omega = 1$, $\mathcal{A}$'s view is clearly identical to its view in Game $2.(k, f)$.

We thus have $|\Pr[W_{2.(k,f)}] - \Pr[W_{2.(k,f-1)}]| \leq \mathbf{Adv}^{\mathrm{PRF}}_{\mathcal{B}^{\mathsf{LWE}}}(\lambda)$. The result of [17, Theorem 5.1] implies that Game $2.(k, f)$ and Game $2.(k, f-1)$ are computationally indistinguishable under the LWE assumption.

We now proceed with another sub-series of hybrid games where we gradually tamper with the security tags $\mathbf{y}_T \in \mathbb{Z}_p^m$ produced by the $\mathcal{Q}_{\mathsf{Spend}}$ oracle. For convenience, we define Game $3.(0, f)$, for each $f \in \{1, \ldots, Q_w\}$, to be identical to Game $2.(Q_U, Q_w)$ and Game $3.(k, 0)$ to be the same as Game $3.(k-1, Q_w)$.

Game $3.(k, f)$ $(1 \leq k \leq Q_U,\ 1 \leq f \leq Q_w)$**:** The adversary is presented with a hybrid $\mathcal{Q}_{\mathsf{Spend}}$ oracle. At each $\mathcal{Q}_{\mathsf{Spend}}$-query $(PK_{\mathcal{B}}, i, j, PK_{\mathcal{M}}, \mathtt{info})$, the response is the same as in Game $2.(Q_U, Q_w)$ if $i < k$ or $i = k$ and $j \leq f$: namely, the spent coin $(R, \mathbf{y}_S, \mathbf{y}_T, \pi_K)$ is computed using actual PRF evaluations for $\mathbf{y}_T$ and randomly sampled $\mathbf{y}_S \hookleftarrow U(\mathbb{Z}_q^m)$. If $i > k$ or $i = k$ and $j > f$, instead of computing the security tag as $\mathbf{y}_T = PK_{\mathcal{U}} + H_0(R) \cdot \lfloor \prod_{i=1}^{L} \mathbf{P}_{J[L+1-i]} \cdot \mathbf{t} \rfloor_p$, the oracle picks $\mathbf{y}_T \hookleftarrow U(\mathbb{Z}_p^m)$ uniformly at random.

The same arguments as in the first sub-series of hybrids show that any noticeable behavioral change on behalf of $\mathcal{A}$ between Game $3.(k, f-1)$ and Game $3.(k, f)$ implies a distinguisher $\mathcal{B}^{\mathsf{LWE}}$ against the PRF of [17] and thus contradicts the LWE assumption.

When we reach Game $3.(Q_U, Q_w)$, we find that $\mathcal{A}$ is interacting with a simulator $(\mathsf{SimParGen}, \mathsf{SimSpend})$ that emulates the $\mathcal{Q}_{\mathsf{Spend}}$ oracle without using any user' key nor any wallet. Specifically, SimParGen generates par as the actual ParGen algorithm does. At each $\mathcal{Q}_{\mathsf{Spend}}$-query, SimSpend chooses $\mathbf{y}_S$ and $\mathbf{y}_T$ as uniformly random vectors in $\mathbb{Z}_p^m$ and simulates π_K by programming $H : \{0,1\}^* \to \{1,2,3\}^\kappa$ and using the statistical HVZK simulator of the argument system (note that the simulation trapdoor τ_{sim} produced by SimParGen is the empty string ε since we are in the random oracle model).

Moreover, we have shown that Game $3.(Q_U, Q_w)$ is computationally indistinguishable from Game 0) under the LWE assumption. □

E Extension with Coin Traceability

Like the construction of Camenisch *et al.* [23], our construction extends so as to provide a mechanism whereby all the coins of a cheating user when a double-spending is detected. To this end, we adapt the method of [23] which relies on

the verifiable encryption paradigm.

The latter method consists in augmenting the withdrawal protocol with a step where the user verifiably encrypts the PRF seed $\mathbf{k} = \mathbf{k}_0 + \mathbf{k}_1 \bmod q$ under his own key and signs the resulting ciphertext using his long-term digital signature key. The spending phase is then adapted so as to make sure that any double-spending uncovers the decryption key that undoes the verifiable encryption. Having identified the double-spender and recovered the key that decrypts the verifiably encrypted seed, the bank is able to recompute the serial numbers of all the coins belonging to the misbehaving user.

To adapt the technique of [23] in the lattice setting, a first idea is to have the user verifiably encrypt the seed $\mathbf{k} \in \mathbb{Z}_q^m$ under his own public key $PK_{\mathcal{U}}$ (for example, using the dual Regev cryptosystem [54]) during the withdrawal phase. In turn, the Spend protocol must be modified so that $\mathbf{y}_T \in \mathbb{Z}_p^m$ conceals $SK_{\mathcal{U}} = \mathbf{e}_u$ rather than $PK_{\mathcal{U}}$. One advantage of this approach is to provide stronger incentives not to overspend since any cheating attempt exposes the user's private key[6] (and not just his public key). On the other hand, the entire protocol becomes much more expensive since each bit of $\mathbf{k} \in \mathbb{Z}_q^m$ has to be encrypted. If we use an LWE-based public-key encryption scheme with plaintext space $\{0,1\}^{m\lceil \log q \rceil}$ for this purpose, the decryption key must be a matrix $\mathbf{E}_u \in \mathbb{Z}^{m \times m \cdot \lceil \log q \rceil}$ with the multi-bit dual Regev cryptosystem [54] whereas the primal Regev system [88] requires $\widetilde{\mathcal{O}}(nm \log q)$ bits in the decryption key. Not only does it require equally large PRF outputs (which expand from $\widetilde{\mathcal{O}}(m \log q)$ to $\widetilde{\mathcal{O}}(nm \log q)$ bits), it also inflates the commitment key CK by a factor n.

To improve the efficiency, we suggest a different method based on a secret-key LWE-based encryption scheme. Let us consider the following private-key scheme, where secret keys only take $\widetilde{\mathcal{O}}(n \log q_e)$ bits, for some modulus q_e.

Parameters$(1^n, 1^\delta)$: Given a security parameter n and a desired message length $\delta = \mathsf{poly}(n)$, choose a modulus q_e, a noise distribution χ, a plaintext space $\{0,1\}^\delta$. Output $\mathsf{pp} := (n, q_e, \delta, \chi)$

Keygen(pp) : Choose a secret key $SK = \mathbf{x} \xleftarrow{\$} \mathbb{Z}_{q_e}^n$.

Encrypt(pp, SK, μ) : To encrypt $\mu \in \{0,1\}^\delta$, sample $\mathbf{A}_E \xleftarrow{\$} \mathbb{Z}_{q_e}^{n \times \delta}$; $\mathbf{r}_E \hookleftarrow \chi^\delta$, and output

$$\mathbf{c} = (\mathbf{A}_E, \mathbf{b}_E = \mathbf{A}_E^\top \cdot \mathbf{x} + 2 \cdot \mathbf{r}_E) + \mu) \in \mathbb{Z}_{q_e}^{n \times \delta} \times \mathbb{Z}_{q_e}^\delta.$$

Decrypt$(\mathsf{pp}, SK, \mathbf{c})$: To decrypt $\mathbf{c} = (\mathbf{A}_E, \mathbf{b}_E)$ using $SK = \mathbf{x}$, compute

$$\mu' = \big((\mathbf{b}_E - \mathbf{A}_E^\top \cdot \mathbf{x}) \bmod q_e\big) \bmod 2 \in \{0,1\}^\delta.$$

This construction is known to be secure under chose-plaintext attacks assuming the hardness of LWE.

At the first step of the $\mathsf{Withdraw}$ protocol of our modified system, the user also commits to a symmetric encryption key $\mathbf{x} \hookleftarrow \mathbb{Z}_{q_e}^n$. At step 2, the bank returns its

[6] For this reason, this approach is limited to only achieve weak exculpability since any double-spending reveals $SK_{\mathcal{U}}$.

own contribution $\mathbf{k}_1 \in \mathbb{Z}_q^n$ to the PRF seed $\mathbf{k}$ and expects to receive a verifiable secret-key encryption of $\mathbf{k} = \mathbf{k}_0 + \mathbf{k}_1 \bmod q$ and a ZK argument showing that this encryption correctly encrypts $\mathbf{k}$ under the committed secret key $\mathbf{x} \in \mathbb{Z}_{q_e}^n$.

At each execution of the Spend protocol, the user verifiably computes $\mathbf{y}_T$ by blinding a concatenation of $\mathbf{F} \cdot \mathbf{e}_u \mod q_s$ and the secret $\mathbf{x} \in \mathbb{Z}_{q_e}^n$.

In more details, our modifications require the following parameters:

- n is the security security parameter.
- For the PRF of [17], we use an odd prime $p = \mathsf{poly}(n)$ and choose q super-polynomial in n so that p divides q and $m = n\lceil \log q \rceil$.
- For the SIS-based signature scheme, we use $q_s = p^{t_s}$; $m_s = \Omega(n \log q_s)$.
- For LWE-based encryption layer, we pick $q_e = p^{t_e}$ for some $t_e \in \mathbb{N}$ and use $\delta = \bar{m}/2 = m\delta_{q-1}$.

Note that we need $\lceil \log q \rceil = \omega(\log n)$ while t_s, t_e can be constant. We explicitly require that $\lceil \log q \rceil \geq t_s + t_e$, and let $t = \lceil \log q \rceil - (t_s + t_e) \geq 0$.

In the modified construction, users' public keys now consist of a syndrome $\mathbf{v}_u = \mathbf{F} \cdot \mathbf{e}_u \bmod q_s$ and a commitment $\mathbf{com}_x$ to $\mathbf{x} \in \mathbb{Z}_{q_e}^n$. The reason to include $\mathbf{com}_x$ in $PK_{\mathcal{U}}$ is to force the user to use the same secret key $\mathbf{x}$ in all withdrawals and make sure that his coins will all be traceable (regardless of which wallet they belong to), should he overspend.

Following [23], we augment our e-cash system with an algorithm Trace which takes as input the database $\{\mathsf{T}_{\mathcal{W}}\}$ of withdrawn wallets and a piece of tracing information $\zeta_{\mathcal{U}}$ associated with user $\mathcal{U}$. It outputs the list of all serial numbers of coins belonging to user $\mathcal{U}$.

ParGen: is as in Section 5 except that $\mathbf{F}$ is chosen over $\mathbb{Z}_{q_s}$ instead of $\mathbb{Z}_p$. Namely, we choose $\mathbf{F} \xleftarrow{\$} \mathbb{Z}_{q_s}^{n \times m_s}$. Also, an additional key CK' is chosen for a statistically hiding commitment with message space $\{0,1\}^{n\delta_{q_e-1}}$ (in order to commit to the decryption key $\mathbf{x} \in \mathbb{Z}_{q_e}^n$). We finally need one more matrix $\mathbf{D}_5 \hookleftarrow U\big(\mathbb{Z}_{q_s}^{n \times n\delta_{q_e-1}}\big)$ in CK since the user's secret key component $\mathbf{x} \in \mathbb{Z}_{q_e}^n$ must be certified by the bank in executions of Withdraw.

UKeygen(par)**:** Pick $\mathbf{e}_u \hookleftarrow U(\{0,1\}^{m_s})$ and compute $\mathbf{v}_u = \mathbf{F} \cdot \mathbf{e}_u \in \mathbb{Z}_{q_s}^n$. Pick $\mathbf{x} \hookleftarrow U(\mathbb{Z}_{q_e}^n)$ and generate a commitment $\mathbf{com}_x = \mathsf{COM}_{CK'}(\mathsf{vdec}_{n,q_e-1}(\mathbf{x}), \mathbf{r}_x)$. The user's public key is $PK_{\mathcal{U}} := (\mathbf{v}_u, \mathbf{com}_x)$ and the underlying secret key is $SK_{\mathcal{U}} = (\mathbf{e}_u, \mathbf{x}, \mathbf{r}_x)$.

Withdraw: Is as before with the following changes. First, we let the user prove that $\mathbf{com}_x$ is consistent with $\mathbf{c}_{\mathcal{U}}$ at step 1 (i.e., they both open to the same $\mathsf{bin}(\mathbf{x})$). At step 2, the user receives $\mathbf{k}_1 \in \mathbb{Z}_q^m$ for the bank $\mathcal{B}$ and verifiably encrypts $\mathbf{k} = \mathbf{k}_0 + \mathbf{k}_1 \in \mathbb{Z}_q^m$. Namely, the user sends back a ciphertext $\mathbf{c_k}$ of $\mathsf{vdec}_{m,q-1}(\mathbf{k}) \in \{0,1\}^{\bar{m}}$ together with a signature of $\mathbf{c_k}$ – which $\mathcal{U}$ computes using a long-term signature key – and an interactive ZK argument that $\mathbf{c}$ is an encryption under the secret key $\mathbf{x} \in \mathbb{Z}_{q_e}^n$ (for which $\mathsf{vdec}(\mathbf{x})$ is committed in $\mathbf{com}_x$) of $\mathsf{vdec}_{m,q-1}(\mathbf{k}) \in \{0,1\}^{\bar{m}}$, where $\mathbf{k} = \mathbf{k}_0 + \mathbf{k}_1 \bmod q$. If the ZK argument verifies, the bank sends the signature $(\tau, \mathbf{v}, \mathbf{r}_1) \in \{0,1\}^{\ell} \times \mathbb{Z}^{2m_s} \times \mathbb{Z}^{m_s}$ and additionally stores $\mathsf{T}_{\mathcal{W}} = (PK_{\mathcal{U}}, \mathbf{c_k})$, together with the user's signature on $\mathbf{c_k}$.

Spend: The serial number $\mathbf{y}_S \in \mathbb{Z}_p^m$ is as before but $\mathbf{y}_T$ is computed in a different way. Since $q_s = p^{t_s}$, the first component $\mathbf{v}_u \in \mathbb{Z}_{q_s}^n$ of $PK_{\mathcal{U}} = (\mathbf{v}_u, \mathbf{com}_{\mathbf{x}})$ can be bijectively encoded as a vector $\bar{\mathbf{v}}_u \in \mathbb{Z}_p^{nt_s}$. Similarly, since $q_e = p^{t_e}$, $\mathbf{x} \in \mathbb{Z}_{q_e}^n$ can be encoded as $\bar{\mathbf{x}} \in \mathbb{Z}_p^{nt_e}$. Recalling that $m = nt_s + nt_e + nt$, we define the vector

$$\mathsf{token}_{\mathcal{U}} = \left[\begin{array}{c} \bar{\mathbf{v}}_u \\ \hline \bar{\mathbf{x}} \\ \hline \mathbf{0}^{nt} \end{array}\right] \in \mathbb{Z}_p^m.$$

Then, the security tag $\mathbf{y}_T$ is obtained as

$$\mathbf{y}_T = \mathsf{token}_{\mathcal{U}} + H_{\mathrm{FRD}}(R) \cdot \lfloor \prod_{i=1}^{L} \mathbf{P}_{J[L+1-i]} \cdot \mathbf{t} \rfloor_p \in \mathbb{Z}_p^m,$$

where $R = H_0(PK_{\mathcal{M}}, \mathtt{info}) \in \mathbb{Z}_p^m$. The well-formedness of $\mathbf{y}_T$ can be proved in zero-knowledge with Stern-like protocols.

Identify: The algorithm recovers $\mathsf{token}_{\mathcal{U}}$ (instead of $PK_{\mathcal{U}}$) as in Section 5. From $\mathsf{token}_{\mathcal{U}}$, the bank $\mathcal{B}$ recovers $\mathbf{v}_u \in \mathbb{Z}_{q_s}^n$ and $\mathbf{x} \in \mathbb{Z}_{q_e}^n$. Based on $\mathbf{v}_u \in \mathbb{Z}_{q_s}^n$, $\mathcal{B}$ looks up the wallet database to find $\mathsf{T}_{\mathcal{W}} = (PK_{\mathcal{U}}, \mathbf{c}_k)$ such that the first part $PK_{\mathcal{U}}$ matches $\mathbf{v}_u \in \mathbb{Z}_{q_s}^n$. It outputs the tracing information $\zeta_{\mathcal{U}} = (PK_{\mathcal{U}}, \mathbf{x})$.

Trace: Since $\mathcal{B}$ stores all the ciphertexts $\mathbf{c}_{\mathbf{k}_f}$ submitted by $\mathcal{U}$ for all withdrawn wallets, it can decrypt the PRF seeds $\mathbf{k}_f \in \mathbb{Z}_q^m$ using the decryption key $\mathbf{x}$.

The ZK argument showing the well-formedness of $\mathbf{y}_T$ with respect to the committed $\mathbf{x}$ and $\mathbf{e}_u$ relies on the fact that the transformation $\mathbf{v}_u \to \bar{\mathbf{v}}_u$ uses a linear bijection associated with a matrix $\mathbf{H}_{q_s,p,n}$ of dimension $n \times nt_s$ and a similar linear encoding works for $\mathbf{x} \to \bar{\mathbf{x}}$. We can prove that $\mathbf{v}_u = \mathbf{F} \cdot \mathbf{e}_u \bmod q_s$ and that $\mathbf{x}$ has decomposition $\mathsf{vdec}_{n,q_e-1}(\mathbf{x})$, where $\mathbf{e}_u \in \{0,1\}^{m_s}$ and $\mathsf{vdec}(\mathbf{x})$ certified by the bank when $\mathcal{B}$ obliviously signed the content of $\mathbf{c}_{\mathcal{U}}$ during the Withdraw protocol.

In the security analysis, all proofs go through in the same way as in Section 6. The only change is that, in the proof of anonymity, we need to rely on the statistical ZK property of the zero-knowledge argument generated by the prover at step 1 of the withdrawal protocol. The reason is that, in order to meaningfully rely on the pseudo-randomness of the PRF of [17], we need to make sure that the seed $\mathbf{k}$ is statistically independent of the adversary's view. To this end, we have to rely on the chosen-plaintext security of the LWE-based symmetric encryption scheme as the bank can see encryption of $\mathbf{k}$ in Withdraw. In turn, relying on the hardness of $\mathsf{LWE}_{n,q_e,\chi}$ at this step requires that the ZK argument at step 1 of Withdraw (which shows the consistency of $\mathbf{c}_k$ with $\mathbf{com}_x$ and $\mathbf{c}_{\mathcal{U}}$) be simulatable without witnesses. Since we assume trusted public parameters, however, Damgård's technique [44] can be used to simulate proofs without any additional round.