Black-Box Accumulation Based on Lattices

Sebastian H. Faller[1], Pascal Baumer[2], Michael Klooß[3], Alexander Koch[3], Astrid Ottenhues[3], and Markus Raiber[3]

Institute of Information Security and Dependability (KASTEL)
Karlsruhe Institute of Technology
Karlsruhe, Germany
[1]sebastian.faller@mailbox.org, [2]ueeap@student.kit.edu,
[3]firstname.lastname@kit.edu

Abstract. Black-box accumulation (BBA) is a cryptographic protocol that allows users to accumulate and redeem points, e.g. in payment systems, and offers provable security and privacy guarantees. Loosely speaking, the transactions of users remain unlinkable, while adversaries cannot claim a false amount of points or use points from other users. Attempts to spend the same points multiple times (double spending) reveal the identity of the misbehaving user and an undeniable proof of guilt. Known instantiations of BBA rely on classical number-theoretic assumptions, which are not post-quantum secure. In this work, we propose the first lattice-based instantiation of BBA, which is plausibly post-quantum secure. It relies on the hardness of the Learning with Errors (LWE) and Short Integer Solution (SIS) assumptions and is secure in the Random Oracle Model (ROM).
Our work shows that a lattice-based instantiation of BBA can be realized with a communication cost per transaction of about 199 MB if built on the zero-knowledge protocol by Yang et al. (CRYPTO 2019) and the CL-type signature of Libert et al. (ASIACRYPT 2017). Without any zero-knowledge overhead, our protocol requires 1.8 MB communication.

Keywords: Lattice-based Cryptography · Black-box Accumulation (BBA) · Electronic Funds Transfer · Security and Privacy · Learning with Errors (LWE) · Short Integer Solution (SIS)

1 Introduction

Black-box accumulation (BBA), introduced in [24], allows the anonymous collection and redemption of points. BBA protocols feature two roles: users and operators. The users can accumulate and spend points on a cryptographic token issued by the operators, via the respective interactive protocols. In real-world scenarios like loyalty programs in shops or prepayment systems for public transport, users can collect incentives or bonus points. For the operators, the secure transfer of points is of paramount importance, whereas users want to protect their privacy. BBA offers a provably secure solution to both concerns. It allows users to collect and redeem points in an unlinkable manner and it protects operators from malicious users trying to claim more points than collected.

Several works have extended the framework of BBA. BBA+ [21] added stronger notions of both security and privacy as well as offline-transactions (in the sense that no permanent connection to a central database is required). More recently, [6, 7, 22] improved several aspects of BBA+. However, all of the proposed instantiations are based on classical cryptographic building blocks whose security guarantees rely on number-theoretic assumptions which are broken by Shor's algorithms [40], rendering them insecure against quantum adversaries.

In contrast, lattice-based hardness assumptions have so-far withstood attempts to break them with quantum algorithms and allow to construct an extensive variety of cryptographic primitives, including commitments, public-key encryption [19, 25, 38, 39] and fully homomorphic encryption (FHE) [18], and are hence considered an ideal candidate to achieve post-quantum (PQ) security. Moreover, lattice-based protocols usually feature good asymptotic efficiency, parallelism, and security under worst-case intractability assumptions. The downside is an increase in communication costs for certain important building blocks, such as zero-knowledge (ZK) proofs. As all known BBA constructions are heavily based on ZK proofs, it gives rise to the difficult question of how to instantiate BBA from lattice-based assumptions, while remaining relatively efficient.

Contribution. In this work, we propose the first lattice-based instantiation of BBA, called BABL (**B**lack-Box **A**ccumulation **B**ased on **L**attices). It relies on the LWE and SIS problems, and is proven secure in the ROM. We follow the security framework of [22], refered to as BBW in the following.

Moreover, we give a concrete instantiation, together with a suitable choice of lattice parameters, and evaluate the scheme's communication complexity. Without any zero-knowledge overhead, our protocol requires 1.8 MB communication, which shows that the efficiency baseline of our general approach/construction is low. When using the popular ZK proof system by Yang et al. [43], an optimized version requires 199 MB communication, too much to be practically usable. However, lattice-based ZK proofs are improving rapidly, see e.g. [42] for some performance comparisons. Thus, it is plausible that the added computation and communication cost shrinks to an actually practical level in the near future. Our construction is the most efficient lattice-based payment system (BBA or E-Cash, cf. Section 1) to date. The closest competitor, E-Cash, needs 262 MB per transaction, using the same ZK protocol. While this does not yet make our protocol fit for practice, it places lattice-based BBA schemes into the range of practicality, where a further round of optimizations could likely allow its real-world use.

Our Construction in a Nutshell. On a high level, our construction follows the approach of BBW [22], but it requires care to translate it to lattices, without reaching a giga-/terabyte range of communication cost per transaction. In BBW, the user holds a token which is basically a commitment whose contents are signed by the operator. This commitment contains a serial number for double-spending detection, a secret key uniquely identifying the user, and the amount of points. (For simplicity, we omit double-spending tags for now.) An update of the number of points (e.g. in a purchase) works as follows. The user sends a

fresh (rerandomized) commitment to the operator, reveals the serial number, and proves in zero-knowledge that the commitment's contents are signed and that the new balance lies within admitted bounds. Using a property of the commitment scheme, the operator updates the balance while keeping the committed content intact. A serial number is chosen by a two-party coin-toss, to ensure it cannot be used to track users. Finally, the operator provides a signature for the new token.

To implement this strategy, [22] uses group-based commitments, ZK proofs, and so-called CL-type signatures [10, 11], which have practically efficient ZK proofs for proving possession of a signature on a commitment or committed value. To replace Multi-Pedersen commitments, we use the lattice-based multi-block commitment scheme of [25], called KTX commitment in the following. We make use of a structural property of these commitments, which allows to "add blocks" to the committed message later, without knowing the messages in the other blocks, similar to Multi-Pedersen commitments. Finally, to replace the group-based CL-type signatures and ZK proofs, we rely on the ZK protocol of Yang et al. [43] combined with the CL-type signatures of Libert et al. [28]. We explicitly define the security guarantee offered by the signature-schemes, which was not explicitly given in [28].

Related Work. The previous BBA protocols by Blömer et al. [6], Bobolz et al. [7], Hartung et al. [21], Hoffmann et al. [22], and Jager and Rupp [24] are all based on number-theoretic hardness assumptions. While this allows them to be much more efficient, it also makes them insecure when quantum computers become available. We think that future privacy issues regarding payments made today, and the security of a users' collected points in the future are reason enough to switch to post-quantum payment systems in the long term. The closest relatives to BBA are Electronic Cash (E-Cash) cryptosystems [14]. The first *compact* E-Cash scheme was given in [9], where compact means that the complexity of withdrawal and spending is logarithmic in the size of an (electronic) wallet. In (compact) E-Cash, there exist three parties, namely a bank, a user, and a merchant. The bank allows withdrawing a wallet containing coins and depositing coins. The wallet is signed by the bank, to make it possible for a user to prove the legitimacy of their wallet. Further, the user can spend the coins from their withdrawn wallet in a privacy-preserving way at a merchant. This is achieved by proving in zero-knowledge the legitimacy of the origin of the coin. The merchant can then deposit the received coin at the bank. The bank can detect double-spenders and prove their guilt, if and only if they are guilty. There are two lattice-based versions of compact E-Cash in the literature: The work of Libert et al. [28] – which propose (implicit) CL-type signatures, and an abstraction of Stern-type ZK protocols [41] – and Yang et al. [43]'s system, which applies their ZK argument system and further optimizations to construct a more efficient system similar to [28]. [8] showed some major issues with the double-spending in compact E-Cash. [15] solved these problems with their lattice-based version of E-Cash.

We stress that E-Cash and BBA are quite different. Most importantly, BBA allows payments in both directions, i.e., points can be accumulated and spent. E-Cash does allow to deposit points but only at the bank, which means that

a wallet cannot be "refilled". Instead, a new wallet with fresh coins has to be generated. Additionally, E-Cash and BBA have different assumptions on the involved parties. On the one hand, it is not possible in BBA to separate issuer and accumulator – such as when the issuer is an E-Cash bank, and the accumulator a merchant. Issuer and accumulator have the same secret key. On the other hand, a merchant and bank must not collude in E-Cash, as this can break privacy. A BBA issuer and an accumulator can collude without breaking privacy. This is necessary due to an impossibility result, cf. [12]. Further, E-Cash only allows the transferal of a single coin per transaction. The transaction value in BBA is an integer in a certain range (e.g. 32-bit integers). Hence, when payments for products with different prices are made, BBA requires just one transaction, whereas many E-Cash transactions would be necessary.

1.1 Outline

In Section 2 we introduce formal definitions of BBA protocols, and the necessary lattice-based hardness assumptions. Furthermore, we introduce the basic cryptographic building blocks and their respective instantiations, which we will use in the following. Our construction of a lattice-based BBA scheme is given in Section 3. The concrete choice of parameters and the efficiency evaluation are in Section 4. Finally, Section 5 discusses open problems and future work.

Acknowledgements. We thank the anonymous reviewers for their feedback. The work presented in this paper has been funded by the German Federal Ministry of Education and Research (BMBF) under the project "PQC4MED" (ID 16KIS1044) and the topic Engineering Secure Systems of the Helmholtz Association (HGF) and by KASTEL Security Research Labs.

2 Preliminaries

Notation. We use $\lambda \in \mathbb{N}$ as security parameter. Vectors and matrices are in bold. For $n \in \mathbb{N}$ we write $\mathbf{I}_n$ for the identity matrix of dimension n. We denote by log the binary logarithm. $\|\cdot\|$ denotes the Euclidean norm and $\|\cdot\|_\infty$ the maximum norm. For $q \in \mathbb{N}$ we denote by $\mathbb{Z}_q = \{-\lfloor (q-1)/2 \rfloor, \ldots, \lceil (q-1)/2 \rceil\}$ the ring of congruence classes of integers modulo q. We denote by $x \leftarrow S$ that x is drawn uniformly at random from set S and by $y \leftarrow D$ that y is drawn according to distribution D. We denote by $\cdot\|\cdot$ the concatenation of vectors, i.e., for $\mathbf{x}, \mathbf{y} \in \mathbb{Z}_q^n$ we have $(\mathbf{x}\|\mathbf{y}) \in \mathbb{Z}_q^{2n}$. For $\mathbf{x} \in \mathbb{Z}_q^m$, we denote by $\mathsf{bin}(\mathbf{x}) \in \mathbb{Z}_2^{m\lceil \log q \rceil}$ the binary decomposite of $\mathbf{x} \in \mathbb{Z}_q^m$, i.e. $\mathbf{x}_j = \sum_{i=0}^{\lceil \log q \rceil - 1} \mathsf{bin}(\mathbf{x})_{\lceil \log q \rceil \cdot j + i} \cdot 2^i$. Inversely, for a $\mathbf{y} \in \mathbb{Z}_2^{\lceil \log q \rceil}$, we denote by $\mathsf{toInt}(\mathbf{y}) := \sum_{i=0}^{\lceil \log q \rceil - 1} \mathbf{y}_i \cdot 2^i \in \mathbb{Z}_q$ the integer (modulo q) represented by $\mathbf{y}$. For a full-rank matrix $\mathbf{M} \in \mathbb{Z}_q^{n \times m}$, we denote by $\widetilde{\mathbf{M}}$ the Gram–Schmidt orthogonalization of $\mathbf{M}$'s columns.

2.1 Black-Box Accumulation

In this section, we give an overview of the BBW framework defined in [22], which we base this work on. It allows a user to anonymously collect (and redeem) points from the operators, which cover the following three roles: i) the *issuer* issues new tokens to the users of the system, ii) the *accumulator* adds points to a token, and iii) the *verifier* subtracts points from a token and verifies that a user's balance is large enough to perform that transaction. As these roles share the same key pair, we do not distinguish them within the paper and refer to them as the *operator*.

The protocols are *offline*, meaning transactions can be performed without a permanent connection of the operator to a database. Nonetheless, regular access to a shared database to store the double-spending tags is needed. We require a common reference string which is set-up by a Trusted Third Party (TTP).[1] Next, let us give the formal definition of a BBW scheme.

Definition 2.1 (BBW Scheme). *A* black-box wallet *scheme* $\mathsf{BBW} = (\mathsf{Setup}, \mathsf{OGen}, \mathsf{UGen}, \mathsf{Issue}, \mathsf{Update}, \mathsf{UVer}, \mathsf{IdentDS}, \mathsf{VerifyGuilt})$ *consists of probabilistic polynomial time (PPT) algorithms* Setup, OGen *and* UGen*, interactive protocols* Issue *and* Update *and deterministic polynomial time algorithms* UVer, IdentDS *and* VerifyGuilt*:*

- $\mathsf{CRS} \leftarrow \mathsf{Setup}(1^\lambda)$*: On input* 1^λ*, returns a common reference string* CRS*. All following algorithms always receive* CRS *(implicitly) as input.*
- $(\mathsf{pk}_\mathcal{O}, \mathsf{sk}_\mathcal{O}) \leftarrow \mathsf{OGen}(\mathsf{CRS})$*: Returns a public and secret key for operator* $\mathcal{O}$*.*
- $(\mathsf{pk}_\mathcal{U}, \mathsf{sk}_\mathcal{U}) \leftarrow \mathsf{UGen}(\mathsf{CRS})$*: Returns a public and secret key for user* $\mathcal{U}$*.*
- $((\mathcal{T}, b_\mathcal{U}), b_\mathcal{O}) \leftarrow \mathsf{Issue}\langle \mathcal{U}(\mathsf{pk}_\mathcal{O}, \mathsf{pk}_\mathcal{U}, \mathsf{sk}_\mathcal{U}), \mathcal{O}(\mathsf{pk}_\mathcal{O}, \mathsf{sk}_\mathcal{O}, \mathsf{pk}_\mathcal{U})\rangle$*: User* $\mathcal{U}$ *communicates with operator* $\mathcal{O}$*, who produces a new token* $\mathcal{T}$ *for* $\mathcal{U}$ *with balance* 0*. The user's input is their key pair* $(\mathsf{pk}_\mathcal{U}, \mathsf{sk}_\mathcal{U})$*, and* $\mathcal{O}$*'s public key* $\mathsf{pk}_\mathcal{O}$*, while* $\mathcal{O}$*'s input is its key pair* $(\mathsf{pk}_\mathcal{O}, \mathsf{sk}_\mathcal{O})$*, and the user's public key* $\mathsf{pk}_\mathcal{U}$*. The bits* $b_\mathcal{O}$ *and* $b_\mathcal{U}$ *indicate whether* $\mathcal{O}$ *and* $\mathcal{U}$ *"accept" the protocol run, respectively.*
- $((\mathcal{T}^*, b_\mathcal{U}), (\mathsf{dstag}, b_\mathcal{O})) \leftarrow \mathsf{Update}\langle \mathcal{U}(\mathsf{pk}_\mathcal{O}, \mathsf{pk}_\mathcal{U}, \mathsf{sk}_\mathcal{U}, \mathcal{T}, v), \mathcal{O}(\mathsf{pk}_\mathcal{U}, \mathsf{sk}_\mathcal{O}, v)\rangle$*: User* $\mathcal{U}$ *updates the token by interacting with the operator* $\mathcal{O}$*. Both get as inputs the public keys* $\mathsf{pk}_\mathcal{O}$ *and* $\mathsf{pk}_\mathcal{U}$ *and their respective secret key, and the (possibly negative) value* v *to be added to the token's balance.* $\mathcal{U}$ *additionally gets their token* $\mathcal{T}$ *(with balance* w*) as input. In the end,* $\mathcal{U}$ *outputs an updated token* $\mathcal{T}^*$ *with balance* $w + v$*, and a bit* $b_\mathcal{U}$ *indicating acceptance of the execution. The operator outputs an acceptance bit* $b_\mathcal{O}$ *and a so-called double-spending tag* dstag *(which later allows detection of reuses of the same token).*[2]
- $b \leftarrow \mathsf{UVer}(\mathsf{pk}_\mathcal{O}, \mathsf{pk}_\mathcal{U}, \mathsf{sk}_\mathcal{U}, \mathcal{T}, w)$*: User* $\mathcal{U}$ *verifies a token* $\mathcal{T}$*, given the operator's public key* $\mathsf{pk}_\mathcal{O}$*, the user's key pair* $(\mathsf{pk}_\mathcal{U}, \mathsf{sk}_\mathcal{U})$*, and a value* w*, and outputs* 1 *if* $\mathcal{T}$ *is a valid token of* $\mathcal{U}$ *with balance* w*, or* 0 *otherwise.*

[1] Our setup only requires a *uniform random string (URS)*, also called *transparent setup*. In practice, it can be heuristically chosen, e.g. as a hash image.

[2] Note, that [21, 22, 24] distinguish between an Add and a Sub transaction for updating the token, where the first one hides the user's balance and the latter one reveals it (or hides it via expensive range proofs). As we will discuss in Section 3 there is no need for us to distinguish those cases, as the balance is always hidden in our construction.

– $(\mathsf{pk}_{\mathcal{U}}, \Pi) \leftarrow \mathsf{IdentDS}(\mathsf{pk}_{\mathcal{O}}, \mathsf{dstag}_1, \mathsf{dstag}_2)$*: Takes as input the operator's public key* $\mathsf{pk}_{\mathcal{O}}$ *and two double-spending tags* $\mathsf{dstag}_1, \mathsf{dstag}_2$*. If* $\mathsf{dstag}_1, \mathsf{dstag}_2$ *come from a transaction with the same token, then* IdentDS *outputs the public key* $\mathsf{pk}_{\mathcal{U}}$ *of the user* $\mathcal{U}$ *that "double-spent" their token and a proof of guilt* Π*. (*Π *can later be verified by a third party, using the* VerifyGuilt *algorithm described next.) Otherwise,it outputs an error symbol* $\perp$*.*
– $b \leftarrow \mathsf{VerifyGuilt}(\mathsf{pk}_{\mathcal{O}}, \mathsf{pk}_{\mathcal{U}}, \Pi)$*: Given a proof of guilt* Π*,* $\mathcal{O}$*'s and* $\mathcal{U}$*'s public keys* $\mathsf{pk}_{\mathcal{O}}$*, and* $\mathsf{pk}_{\mathcal{U}}$*, it outputs* 1 *if* $\mathcal{U}$ *is guilty of double-spending,* 0 *otherwise.*

We say a BBW scheme is *correct* if the two interactive protocols Issue and Update and the algorithms UVer, IdentDS, and VerifyGuilt are correct. For Issue this means, if both parties follow the protocol, Issue outputs a valid token $\mathcal{T}$ (as verified by UVer) and both parties accept the execution. Similarly, Update is correct if both parties accept the execution and the output is a valid (as above) updated token (with new balance $w + v$) if the parties follow the protocol. Correctness of UVer, IdentDS, and VerifyGuilt are defined in the canonical way.

Privacy and Security Properties. We give an informal description of the security properties and refer to App. E for the full definitions.

On the system side we formalize security by three properties: i) a scheme is *owner-binding* if a token is bound to a unique user, and can only be used by it, ii) a scheme is *balance-binding* if no false balance can be claimed, i.e., one can only claim a certain (overall) balance for a token if this balance equals the exact amount of points that have been legitimately collected with this token up to this point in time, and iii) a scheme features *double-spending detection* if a user that presents an already used token in a transaction can be (provably) identified.

For the privacy of the user, we demand the following properties: i) the scheme is *privacy preserving*, i.e., an adversary is not able to link any transactions of the user, even with corrupt operators, ii) the scheme offers *false-accusation protection*, if no malicious operator can falsely produce a proof of guilt for an honest user, and iii) a scheme should feature *post-compromise security*, i.e., that after a temporary compromise of the user, the unlinkability (but not the false-accusation property) can be recovered (by introducing new randomness into the token).

A difference in the description of BBW and our framework is that BBW allows embedding attributes in the token, i.e., the token's expiration date or data for age verification. Including such attributes is direct, but omitted for simplicity.

2.2 Lattices

We recall the basics of lattice-based cryptography required for our construction.

Definition 2.2. *A* lattice $\mathcal{L}$ *is the group of all integer linear combinations of* k *linearly independent vectors* $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_k\} \subseteq \mathbb{R}^n$*, for* $k \in \mathbb{N}$*:* $\mathcal{L} = \mathcal{L}(\mathbf{B}) := \left\{\sum_{i=1}^{k} z_i \cdot \mathbf{b}_i \mid z_i \in \mathbb{Z}\right\}$*. Let* $m \geq n \geq 1$*, a prime number* $q > 2$*,* $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ *and* $\mathbf{u} \in \mathbb{Z}_q^n$*. We write:*

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

Definition 2.3 (Discrete Gaussian Distribution). *For a lattice $\mathcal{L}$, a vector $\mathbf{c} \in \mathbb{R}^m$, and a real number $\sigma > 0$, define $\rho_{\sigma,\mathbf{c}}(\mathbf{x}) = \exp(-\pi\|\mathbf{x}-\mathbf{c}\|^2/\sigma^2)$. The* discrete Gaussian distribution *of support $\mathcal{L}$, center $\mathbf{c}$ and parameter σ is defined as $D_{\mathcal{L},\sigma,\mathbf{c}}(\mathbf{y}) = \rho_{\sigma,\mathbf{c}}(\mathbf{y})/\rho_{\sigma,\mathbf{c}}(\mathcal{L})$ for any $\mathbf{y} \in \mathcal{L}$, where $\rho_{\sigma,\mathbf{c}}(\mathcal{L}) = \sum_{x\in\mathcal{L}} \rho_{\sigma,\mathbf{c}}(x)$. We denote by $D_{\mathcal{L},\sigma}(\mathbf{y})$ the distribution centered in $\mathbf{c} = \mathbf{0}^m$ and exploit the fact that samples from $D_{\mathcal{L},\sigma}$ have small maximum norm with high probability.*

Lemma 1 ([5, Lemma 1.5]). *For any lattice $\mathcal{L} \subset \mathbb{R}^n$ and positive real number $\sigma > 0$, we have $\Pr_{b\leftarrow D_{\mathcal{L},\sigma}}[\|\mathbf{b}\| \leq \sqrt{n}\sigma] \geq 1 - 2^{\Omega(n)}$.*

The following lemmas specify how one can sample an (almost) random lattice basis of $\Lambda_q^\perp(\mathbf{A})$, together with a short trapdoor basis, and how to extend a basis:

Lemma 2 ([4, Theorem 3.2]). *There is a PPT algorithm* $\mathsf{TrapGen}$*, that takes as input $1^n, 1^m$ and an integer $q > 2$ 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 the uniform distribution over $\mathbb{Z}_q^{n\times m}$ and $\|\widetilde{\mathbf{T_A}}\| \leq \mathcal{O}(\sqrt{n \log q})$.*

Lemma 3 ([13, Lemma 3.2]). *For $m' > m$, there exists a PPT algorithm* $\mathsf{ExtBasis}$ *that takes as 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 the left $n \times m$ submatrix of $\mathbf{B}$, and outputs a basis $\mathbf{T_B}$ of $\Lambda_q^\perp(\mathbf{B})$ with $\|\widetilde{\mathbf{T_B}}\| \leq \|\widetilde{\mathbf{T_A}}\|$.*

2.3 Instantiation of Building Blocks

KTX-Commitments. In our construction, we use the commitment scheme of [25]. Let $n \in \mathcal{O}(\lambda)$, $q \in \mathcal{O}(n^4)$, $m_0, m_1 \in \Theta(n \log q)$, $0 < \sigma_{\mathsf{Com}} \in \mathbb{R}$, where m_0 is the size of the randomness vector $\mathbf{r}$, m_1 is the size of the message vector $\mathbf{m}$ and σ_{Com} is the parameter of the Gaussian distribution for the randomness. In the simplest case, one can commit to one message block $\mathbf{m} \in \mathbb{Z}_2^{m_1}$ by computing:

$$\mathsf{Gen}(1^\lambda)\colon \mathbf{D}_0 \leftarrow \mathbb{Z}_q^{n\times m_0}, \mathbf{D}_1 \leftarrow \mathbb{Z}_q^{n\times m_1}, \text{ output } (\mathbf{D}_0, \mathbf{D}_1, \sigma_{\mathsf{Com}}).$$
$$\mathsf{Com}(\mathsf{params}, \mathbf{m}; \mathbf{r}) := \mathbf{D}_0 \cdot \mathbf{r} + \mathbf{D}_1 \cdot \mathbf{m} \in \mathbb{Z}_q^n,$$

where $\mathsf{params} = (\mathbf{D}_0, \mathbf{D}_1, \sigma_{\mathsf{Com}})$ are the public parameters.

The matrices $\mathbf{D}_0, \mathbf{D}_1$ are drawn uniformly at random. For each new commitment the randomness $\mathbf{r} \leftarrow D_{\mathbb{Z}_q^{m_0},\sigma_{\mathsf{Com}}}$ is chosen according to the discrete Gaussian distribution $D_{\mathbb{Z}_q^{m_0},\sigma_{\mathsf{Com}}}$. Usually, we set $m_0 = 2m_1$. A commitment $\mathbf{c}$ can be opened by showing $\mathbf{m}$ and $\mathbf{r}$. If it holds that $\mathbf{m} \in \mathbb{Z}_2^{m_1}$, $\|\mathbf{r}\| \leq \sigma_{\mathsf{Com}}\sqrt{m_0}$ and $\mathbf{c} = \mathbf{D}_0 \cdot \mathbf{r} + \mathbf{D}_1 \cdot \mathbf{m}$, the commitment is valid. This scheme is statistically hiding. It is computationally binding, which can be seen by a straightforward reduction on $SIS_{n,q,2\sigma_{\mathsf{Com}}\sqrt{m_0},m_0+m_1}$. For $N \in \mathbb{N}$, the scheme can be extended to a commitment scheme on N messages by using N matrices. For the security proof of our construction, we require our commitment scheme to be equivocal. The scheme presented above can easily be turned into an equivocal commitment scheme, by using lattice trapdoor gadgets, as discussed in App. B. It is necessary in the construction of the trapdoor to have $m_0 > n\lceil \log q \rceil$.

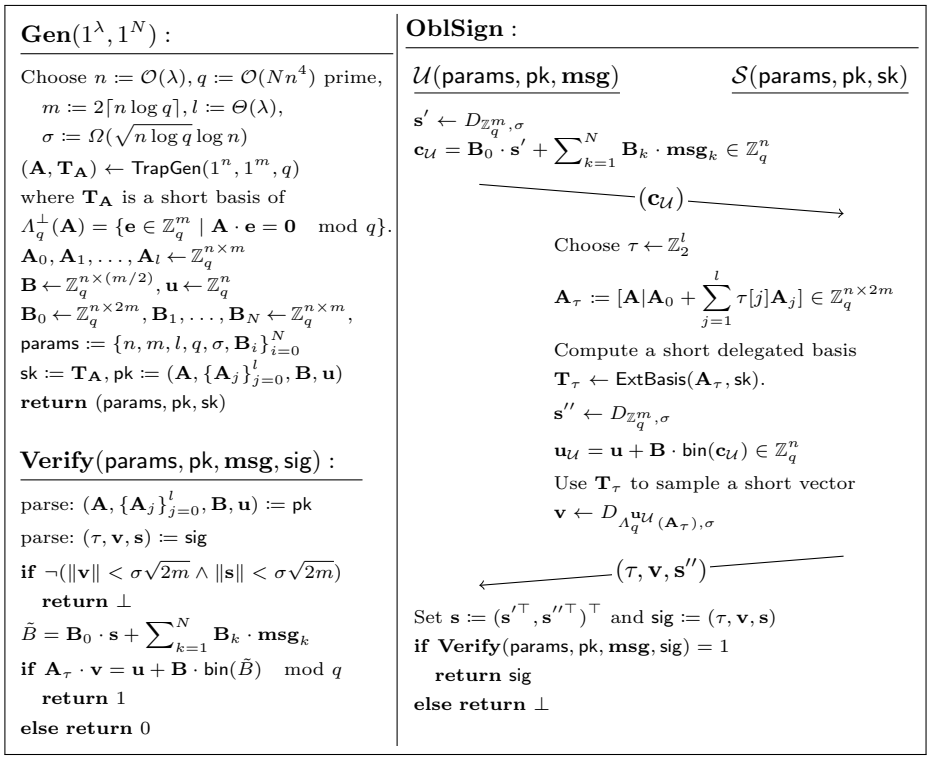

Fig. 1: Gen, Verify and OblSign algorithms of the signature scheme.

Signature Scheme by Libert et al. The scheme for obliviously signing committed messages by Libert et al. [27] consists of the two algorithms $\mathsf{Gen}, \mathsf{Vfy}$, and the interactive protocol $\mathsf{OblSign}$ (described in Fig. 1). It allows the signing of N-block messages $\mathbf{msg} = (\mathbf{m}_1, \ldots, \mathbf{m}_N)$, for $N = \mathsf{poly}(\lambda)$. In our construction we will use the notation $\mathsf{OblSign}.\mathcal{S}(\mathsf{pk}, \mathsf{sk}, \mathbf{c}_\mathcal{U}) \to (\tau, \mathbf{v}, \mathbf{s}'')$ to denote the part of the protocol, which is executed by the signer, where params is derived from the relevant parts of the implicitly given CRS. The algorithm takes a key pair $(\mathsf{pk}, \mathsf{sk})$ and a commitment $\mathbf{c}_\mathcal{U}$ and outputs the signer's part of the signature $(\tau, \mathbf{v}, \mathbf{s}'')$ on the content of the commitment $\mathbf{c}_\mathcal{U}$. See App. C for a proof sketch or [28, Theorem 2] for a full proof of Lemma 4. Additionally, we give in App. C a formal definition of the security, which was only implicit in [28][27].

Lemma 4 ([28, Theorem 2]). *Let* $\beta' := \sigma^2 m\sqrt{2m}(l+2) + \sigma m\sqrt{m}$ *and* $\beta'' = \sigma^2 m\sqrt{2m} + \sqrt{2m} + 4\sigma m\sqrt{2m \log q}$. *Then the above scheme is secure if the* $SIS_{n,q,\beta',m}$ *and* $SIS_{n,q,\beta'',m}$ *assumptions hold.*

3 Our Construction of BABL

We denote by S the signature scheme of [28] (cf. Section 2.3) and by C the commitment scheme of [25] used in Issue and Update (cf. Section 2.3). The two

$\mathbf{OGen}(\mathsf{CRS})$	$\mathbf{UGen}(\mathsf{CRS})$
Generate $(\mathsf{params}', \mathsf{pk}_\mathsf{S}, \mathsf{sk}_\mathsf{S}) \leftarrow \mathsf{S.Gen}(1^\lambda, 1^5)$	Draw $\mathsf{sk}_\mathcal{U} \leftarrow \mathbb{Z}_2^{m_\mathsf{sk}}$
return $(\mathsf{sk}_\mathcal{O} := \mathsf{sk}_\mathsf{S}, \mathsf{pk}_\mathcal{O} := \mathsf{pk}_\mathsf{S})$	**return** $(\mathsf{pk}_\mathcal{U} := \mathbf{F} \cdot \mathsf{sk}_\mathcal{U}, \mathsf{sk}_\mathcal{U})$

Fig. 2: Generation algorithms **OGen**, **UGen** for operators and user, respectively

zero-knowledge proof systems P1 and P2 are instantiations of the ZK scheme from [43] and are used in Issue and Update, respectively. (For a general description of the building blocks and their security notions, see App. A.2).

System Setup and Key Generation. We describe the choice of moduli, (matrix) dimensions, and their relation for the building blocks (as defined in Section 2.3). The parameter generation for both parties is described in Fig. 2.

Setup(1^λ)**:**

- Choose a prime modulus $q_0 = \mathcal{O}(\lambda^4)$.
- For the signature scheme, set the modulus to $q = q_0^e$ for some $e > 0$. Let dimensions $n_\mathsf{S} = \mathcal{O}(\lambda)$ and $m_\mathsf{S} = 2n_\mathsf{S}\lceil \log q \rceil$, and tag length $l_\tau = \Theta(\lambda)$.
- The Gaussian parameter is set to $\sigma = \Omega(\sqrt{n_\mathsf{S} \log q} \log n_\mathsf{S})$.
- Choose $n_\mathsf{sk} = \mathcal{O}(\lambda)$, $m_\mathsf{sk} = n_\mathsf{sk} \log q_0$ and draw $\mathbf{F} \leftarrow \mathbb{Z}_{q_0}^{n_\mathsf{sk} \times m_\mathsf{sk}}$ (for $\mathsf{pk}_\mathcal{U}$ later).
- For the commitment scheme, let dimension $n_\mathsf{C} = n_\mathsf{S}$. Let $m_\mathbf{r} = 2n_\mathsf{C}\lceil \log q \rceil$ be the size of commitment randomness, let $m_\mathsf{nr} = \mathcal{O}(\lambda)$ be the size of serial numbers, and let $m_\mathbf{b} = \mathcal{O}(\lambda)$ be the size of balance vectors. We require $2^{m_\mathbf{b}} < q/4$ for the balance space $\mathbb{V}$, and choose $\mathbb{V} = \{0, \ldots, 2^{m_\mathbf{b}} - 1\}$. Thus, for all $x, y \in \mathbb{V}$, there is no wrap-around for $x + y$. Draw $\mathbf{D}_0 = (\mathbf{D}_0^0, \mathbf{D}_0^1) \leftarrow \mathbb{Z}_q^{n_\mathsf{C} \times 2m_\mathbf{r}}$, $\mathbf{D}_1 \leftarrow \mathbb{Z}_q^{n_\mathsf{C} \times m_\mathsf{sk}}$, $\mathbf{D}_2 \leftarrow \mathbb{Z}_q^{n_\mathsf{C} \times m_\mathbf{b}}$, $\mathbf{D}_3, \mathbf{D}_4 \leftarrow \mathbb{Z}_q^{n_\mathsf{C} \times m_\mathsf{nr}\lceil \log q_0 \rceil}$, $\mathbf{D}_5 \leftarrow \mathbb{Z}_q^{n_\mathsf{C} \times m_\mathsf{sk}\lceil \log q_0 \rceil}$ and set $N = 5$.[3] Choose a Gaussian parameter $\sigma_\mathsf{Com} > 0$ and use the same modulus q, as for the signature scheme.
- Let $H_{FRD} \colon \mathbb{Z}_{q_0}^{m_\mathsf{sk}} \to \mathbb{Z}_{q_0}^{m_\mathsf{sk} \times m_\mathsf{sk}}$ be a full-rank difference function, (see App. A.2).
- **return** $\mathsf{CRS} := (1^\lambda, q_0, q, n_\mathsf{S}, m_\mathsf{S}, n_\mathsf{C}, m_\mathbf{r}, n_\mathsf{sk}, m_\mathsf{sk}, m_\mathbf{b}, m_\mathsf{nr}, \sigma, \sigma_\mathsf{Com}, \mathbf{F}, H_{FRD}, \{\mathbf{D}_i\}_{i=0}^5)$.

Issuing a New Token. In this protocol, the user $\mathcal{U}$ interacts with the operator $\mathcal{O}$ (in the issuer role) to get a fresh token with balance $\mathbf{b} = \mathbf{0}$. The token is a tuple of the form $\mathcal{T} = (\mathbf{c}, \mathbf{r}, \mathsf{sk}_\mathcal{U}, \mathbf{b}, \mathbf{s}_\mathcal{U}, \mathbf{s}_\mathcal{O}, \mathbf{u}_\mathcal{U}, \mathsf{sig})$, where $\mathbf{c}$ is a multi-block commitment to the values $\mathsf{sk}_\mathcal{U}, \mathbf{b}, \mathbf{s}_\mathcal{U}, \mathbf{s}_\mathcal{O}$, and $\mathbf{u}_\mathcal{U}$ with randomness $\mathbf{r}$. Here, $\mathsf{sk}_\mathcal{U}$ is the user's secret key, and the vectors $\mathbf{s}_\mathcal{U}$ and $\mathbf{s}_\mathcal{O}$ are the two shares of the token's serial number, chosen by $\mathcal{U}$ and $\mathcal{O}$, respectively. The vector $\mathbf{u}_\mathcal{U}$ is randomly drawn by the user and is used in the generation of the $\mathbf{t}$-part of the double-spending tag (to be explained below). Finally, sig is a signature on the commitment $\mathbf{c}$.

The Issue protocol is the only protocol in which the operator sees $\mathsf{pk}_\mathcal{U}$, the public key of the user. In subsequent transactions of Update, possession of $\mathsf{sk}_\mathcal{U}$ (and thus, of $\mathsf{pk}_\mathcal{U}$) is proven via ZK proof.

Fig. 3 (left) shows the Issue protocol in detail. First, the user $\mathcal{U}$ draws their part of the serial number $\mathbf{s}_\mathcal{U} \leftarrow \mathbb{Z}_{q_0}^{m_\mathsf{nr}}$, the vector $\mathbf{u}_\mathcal{U} \leftarrow \mathbb{Z}_{q_0}^{m_\mathsf{sk}}$ for the computation of the

[3] We will use these matrices for the signature, too. We ignore params', output by S.Gen.

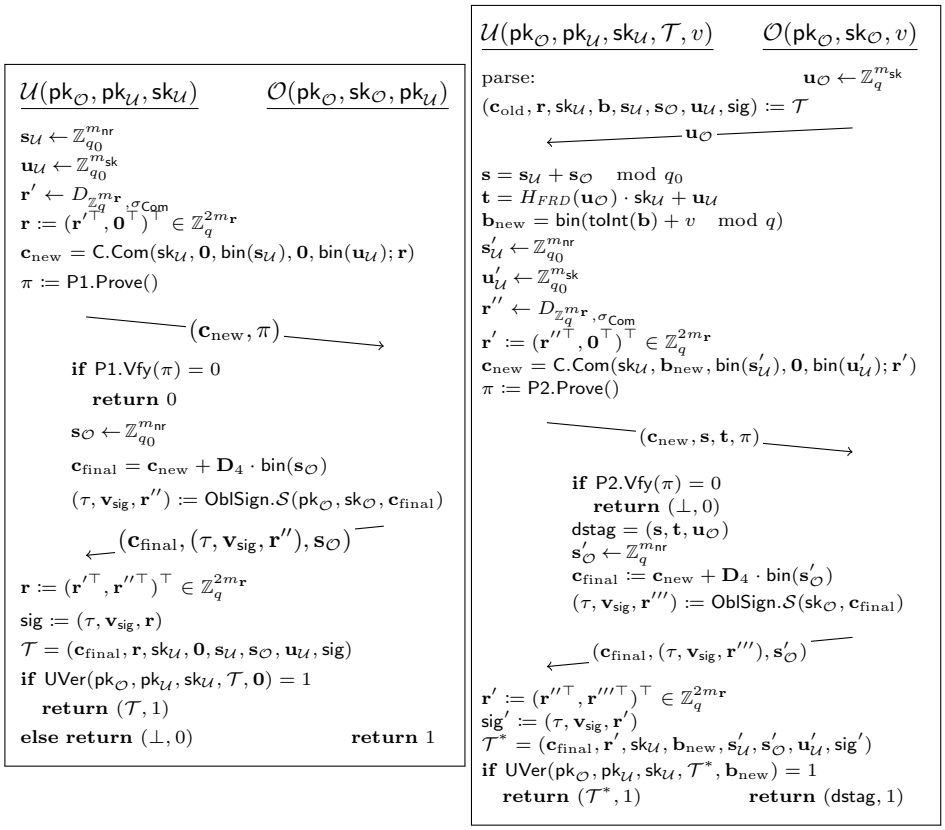

Fig. 3: Protocols for issuing (left) and updating (right) a token $\mathcal{T}$.

double-spending tag, and a random vector for the commitment $\mathbf{r}' \leftarrow D_{\mathbb{Z}_q^{m_\mathbf{r}}, \sigma_{\mathsf{Com}}}$. The other half of the randomness vector $\mathbf{r} \in \mathbb{Z}_q^{2m_\mathbf{r}}$ is set to 0, so the randomness chosen by $\mathcal{O}$ can later be added (after $\mathcal{U}$ received $\mathbf{r}''$ from $\mathcal{O}$). $\mathcal{U}$ then commits (using randomness $\mathbf{r}'$) on a five-block message, containing the secret key $\mathsf{sk}_\mathcal{U}$, $\mathbf{s}_\mathcal{U}$ and $\mathbf{u}_\mathcal{U}$, with the second and fourth message block of the commitment being initialized to $\mathbf{0}$. This is because the second block represents the balance of the token, and is supposed to be $\mathbf{0}$ after issuance of the token, and the fourth block is zero for the operator to later add their share $\mathbf{s}_\mathcal{O}$ of the serial number to the block. Afterwards, the user sends the commitment to $\mathcal{O}$, together with a ZK proof π that ensures that the commitment contains the secret key which belongs to the user's public key.

After verification of π, $\mathcal{O}$ proceeds by adding their share $\mathbf{s}_\mathcal{O}$ of the serial number $(\mathbf{s}_\mathcal{U}, \mathbf{s}_\mathcal{O})$ to the commitment. Then, the operator signs the committed message obliviously and sends the final commitment and the signature back to the user. The user verifies if the token is correct and accepts if this is the case.

We denote the used ZK proof system by $\mathsf{P1}$. With $\mathsf{P1}$, the user proves the following relations to the operator:

1. $\mathsf{pk}_{\mathcal{U}} = \mathbf{F} \cdot \mathsf{sk}_{\mathcal{U}} \mod q_0$
2. $\mathbf{c}_{\text{new}} = \mathbf{D}_0^0 \cdot \mathbf{r}' + \mathbf{D}_0^1 \cdot \mathbf{0} + \mathbf{D}_1 \cdot \mathsf{sk}_{\mathcal{U}} + \mathbf{D}_2 \cdot \mathbf{0} + \mathbf{D}_3 \cdot \bar{\mathbf{s}}_{\mathcal{U}} + \mathbf{D}_4 \cdot \mathbf{0} + \mathbf{D}_5 \cdot \bar{\mathbf{u}}_{\mathcal{U}}$
3. $\|\mathbf{r}'\| \leq \sigma_{\mathsf{Com}} \sqrt{m_{\mathbf{r}}}$
4. $\mathsf{sk}_{\mathcal{U}}$, $\bar{\mathbf{s}}_{\mathcal{U}}$, $\bar{\mathbf{u}}_{\mathcal{U}}$ are binary and $\mathsf{bin}(\mathbf{s}_{\mathcal{U}}) = \bar{\mathbf{s}}_{\mathcal{U}}$, $\mathsf{bin}(\mathbf{u}_{\mathcal{U}}) = \bar{\mathbf{u}}_{\mathcal{U}}$

In App. D, we show how these equations can be proven via the ZK protocol of [43]. We denote this proof by $\pi \coloneqq \mathsf{P1.Prove}()$. We note here, that the "actual" serial number of the token will be $\mathbf{s} = \mathbf{s}_{\mathcal{U}} + \mathbf{s}_{\mathcal{O}} \mod q_0 \in \mathbb{Z}_{q_0}^{m_{\text{nr}}}$. Thus, $\mathbf{s}$ is uniformly random in $\mathbb{Z}_{q_0}^{m_{\text{nr}}}$ if one of the parties was honest. Hence, the collision probability is negligible if $q_0^{-m_{\text{nr}}}$ is negligible, which must be ensured by parameter choices.

Updating the Balance of a Token. We start with an overview of the update protocol, and then a detailed explanation. See Fig. 3 (right). The user starts with a token from a run of Issue or Update. Showing the token to the operator in the plain would make transactions linkable – we hence use ZK proofs for this.

First, the operator sends a "challenge" $\mathbf{u}_{\mathcal{O}}$. From this, the user generates a double-spending tag $\mathsf{dstag} = (\mathbf{s}, \mathbf{t}, \mathbf{u}_{\mathcal{O}})$, where $\mathbf{s}$ is the serial number, computed as $\mathbf{s} = \mathbf{s}_{\mathcal{U}} + \mathbf{s}_{\mathcal{O}}$, and $\mathbf{t}$ is a masking of $\mathsf{sk}_{\mathcal{U}}$, effectively a one-time pad encryption of $\mathsf{sk}_{\mathcal{U}}$ with $\mathbf{u}_{\mathcal{U}}$. Given two values $\mathbf{t}$, $\mathbf{t}'$ with distinct "challenges" $\mathbf{u}_{\mathcal{O}}$, $\mathbf{u}'_{\mathcal{O}}$, one can easily compute $\mathsf{sk}_{\mathcal{U}}$. Since the user is bound to $\mathbf{s}$ and $\mathsf{sk}_{\mathcal{U}}$ by the token (and the commitment), this ensures that a double-spending user must reuse a serial number $\mathbf{s}$, and, since with overwhelming probability $\mathbf{u}_{\mathcal{O}} \neq \mathbf{u}'_{\mathcal{O}}$, is caught and deanonymized when doing so. This implements double spending detection. (Note, that a benign user can not be deanonymized).

The user also sets up a new (partial) token $\mathbf{c}_{\text{new}}$, analogous to the Issue protocol, but with balance set $\mathsf{toInt}(\mathbf{b}) + v$, according to the transaction. Then it proves that $\mathbf{c}_{\text{new}}$ is a valid token (analogous to Issue) with correct balance, its connection and the validity of the "old" token (and its balance), and also, that the double spending tag $\mathsf{dstag} = (\mathbf{s}, \mathbf{t}, \mathbf{u}_{\mathcal{O}})$ was correctly computed.

As in Issue, the operator first verifies the proof. It then chooses its part $\mathbf{s}'_{\mathcal{O}}$ of the new serial number and obliviously signs the adapted commitment $\mathbf{c}_{\text{final}}$. Also as in Issue, the user verifies the commitment and signature. After the transaction, the user has a freshly updated token, and the operator a double-spending tag.

Now, we describe the relevant parts in more detail. Update is defined as an interactive protocol between user $\mathcal{U}$ and operator $\mathcal{O}$. Both parties take as inputs their key pairs, and the transaction value $v \in \mathbb{V}$. Additionally, the user gets as input the operator's public key and the old token $\mathcal{T}$ containing their current balance $\mathsf{toInt}(\mathbf{b}) \in \mathbb{V}$. The protocol outputs for $\mathcal{U}$ and $\mathcal{O}$ consist of a new token $\mathcal{T}^*$ and a double-spending tag dstag, respectively, as well as the output bits $b_{\mathcal{U}}, b_{\mathcal{O}}$, respectively, indicating a party accepts the execution of the protocol.

The user's token is $\mathcal{T} = (\mathbf{c}_{\text{old}}, \mathbf{r}, \mathsf{sk}_{\mathcal{U}}, \mathbf{b}, \mathbf{s}_{\mathcal{U}}, \mathbf{s}_{\mathcal{O}}, \mathbf{u}_{\mathcal{U}}, \mathsf{sig})$. As explained above, the protocol generates a new token, but with a different balance and additional consistency proofs and double-spending detection. This is reflected by the variables

with an additional prime, corresponding to those of Issue. The updated balance is $\mathbf{b}_{\text{new}} \coloneqq \mathsf{bin}(\mathsf{toInt}(\mathbf{b}) + v \mod q)$. The partial commitment is of the form

$$\mathbf{c}_{\text{new}} = \mathbf{D}_0^0 \cdot \mathbf{r}' + \mathbf{D}_1 \cdot \mathsf{sk}_{\mathcal{U}} + \mathbf{D}_2 \cdot \mathbf{b}_{\text{new}} + \mathbf{D}_3 \cdot \mathsf{bin}(\mathbf{s}'_{\mathcal{U}}) + \mathbf{D}_5 \cdot \mathsf{bin}(\mathbf{u}'_{\mathcal{U}}), \quad (1)$$

which differs from Issue in the term $\mathbf{D}_2 \cdot \mathbf{b}_{\text{new}}$, where $\mathbf{b}_{\text{new}} = \mathbf{0}$. As in Issue, the new serial number is calculated from $(\mathbf{s}'_{\mathcal{U}}, \mathbf{s}'_{\mathcal{O}})$ chosen by user and operator, respectively. Also note the user only adds by $\mathbf{D}_0^0 \cdot \mathbf{r}'$ the first half of the randomness to the commitment. The second half will be chosen by the operator. For double-spending detection, $\mathcal{O}$ sends $\mathbf{u}_{\mathcal{O}} \leftarrow \mathbb{Z}_{q_0}^{m_{\mathsf{sk}}}$. The user calculates the vector $\mathbf{s} \coloneqq \mathbf{s}_{\mathcal{U}} + \mathbf{s}_{\mathcal{O}} \mod q_0$, and $\mathbf{t} \coloneqq H_{FRD}(\mathbf{u}_{\mathcal{O}}) \cdot \mathsf{sk}_{\mathcal{U}} + \mathbf{u}_{\mathcal{U}}$. Recall that H_{FRD} denotes the full-rank difference function from App. A.2. Also note, that $\mathbf{u}_{\mathcal{U}}$ perfectly masks $H_{FRD}(\mathbf{u}_{\mathcal{O}}) \cdot \mathsf{sk}_{\mathcal{U}}$, as it is a uniformly random value (chosen when $\mathbf{c}_{\text{old}}$ was issued). Hence, $\mathbf{t}$ reveals nothing about $\mathsf{sk}_{\mathcal{U}}$. However, if the user double-spends by reusing an old token, given $\mathbf{u}_{\mathcal{O}} \neq \mathbf{u}'_{\mathcal{O}}$ in these two executions (which happens with overwhelming probability), then $(H_{FRD}(\mathbf{u}_{\mathcal{O}}) - H_{FRD}(\mathbf{u}'_{\mathcal{O}}))^{-1}(\mathbf{t}' - \mathbf{t}'') = \mathsf{sk}_{\mathcal{U}}$. Thus, the identity of the misbehaving user is revealed. We denote by $\mathsf{P2}$ a non-interactive ZK proof system, and by $\pi \coloneqq \mathsf{P2.Prove}()$ its output. $\mathsf{P2}$ proves following equations:

1. $\mathsf{Vfy}(\mathsf{pk}_{\mathcal{O}}, \mathsf{sig}, (\mathsf{sk}_{\mathcal{U}}, \mathbf{b}, \bar{\mathbf{s}}_{\mathcal{U}}, \bar{\mathbf{s}}_{\mathcal{O}}, \bar{\mathbf{u}}_{\mathcal{U}})) = 1$
2. $\mathbf{s} = \mathbf{s}_{\mathcal{U}} + \mathbf{s}_{\mathcal{O}} \mod q_0$
3. $\mathbf{t} = H_{FRD}(\mathbf{u}_{\mathcal{O}}) \cdot \mathsf{sk}_{\mathcal{U}} + \mathbf{u}_{\mathcal{U}} \mod q_0$
4. $\mathbf{c}_{\text{new}} = \mathbf{D}_0^0 \cdot \mathbf{r}'' + \mathbf{D}_0^1 \cdot \mathbf{0} + \mathbf{D}_1 \cdot \mathsf{sk}_{\mathcal{U}} + \mathbf{D}_2 \cdot \mathbf{b}_{\text{new}} + \mathbf{D}_3 \cdot \bar{\mathbf{s}}'_{\mathcal{U}} + \mathbf{D}_5 \cdot \bar{\mathbf{u}}'_{\mathcal{U}}$
5. $\mathbf{b}_{\text{new}} = \mathsf{bin}(\mathsf{toInt}(\mathbf{b}) + v \mod q))$
6. $\|\mathbf{r}'\| \leq \sigma_{\mathsf{Com}} \sqrt{m_{\mathbf{r}}}$
7. $\mathsf{sk}_{\mathcal{U}}$, $\bar{\mathbf{s}}_{\mathcal{U}}$, $\bar{\mathbf{s}}_{\mathcal{O}}$, $\bar{\mathbf{u}}_{\mathcal{U}}$, $\bar{\mathbf{u}}'_{\mathcal{U}}$, $\mathbf{b}_{\text{new}}$, $\bar{\mathbf{s}}'_{\mathcal{U}}$ are binary, $\mathsf{bin}(\mathbf{s}_{\mathcal{U}}) = \bar{\mathbf{s}}_{\mathcal{U}}$, $\mathsf{bin}(\mathbf{s}_{\mathcal{O}}) = \bar{\mathbf{s}}_{\mathcal{O}}$, $\mathsf{bin}(\mathbf{u}_{\mathcal{U}}) = \bar{\mathbf{u}}_{\mathcal{U}}$, $\mathsf{bin}(\mathbf{u}'_{\mathcal{U}}) = \bar{\mathbf{u}}'_{\mathcal{U}}$, and $\mathsf{bin}(\mathbf{s}'_{\mathcal{U}}) = \bar{\mathbf{s}}'_{\mathcal{U}}$

The equations prove that the old token was valid, and its contents were used to compute $\mathbf{b}_{\text{new}}$, dstag and $\mathbf{c}_{\text{new}}$. Items 2 and 3 are for showing that the serial number $\mathbf{s}$ and tag $\mathbf{t}$ were computed correctly (from these values). The remaining equations prove the well-formedness of the new token, similar to Issue. In App. D, we show how the equations can be transformed into the generic form $\mathbf{A} \cdot \mathbf{x} = \mathbf{y} \mod q$, where $\mathbf{A}$ is a public matrix, $\mathbf{y}$ is a public vector and $\mathbf{x}$ is the secret witness. Once the equations are in this form, the ZK protocol from [43] can be leveraged.

The user sends $(\mathbf{c}_{\text{new}}, \mathbf{s}, \mathbf{t}, \pi)$ to $\mathcal{O}$, who checks the validity of the proof. The remainder of the protocol is essentially as in Issue, i.e. $\mathcal{O}$ picks their share $\mathbf{s}_{\mathcal{O}}$ of the serial number, obliviously signs the extended commitment $\mathbf{c}_{\text{final}}$, and sends the respective values to $\mathcal{U}$. The double-spending tag $\mathsf{dstag} = (\mathbf{s}, \mathbf{t}, \mathbf{u}_{\mathcal{O}})$ is stored in a database of the operator, after the transaction ended successfully. If an entry $(\mathbf{s}, \mathbf{t}')$ is already recorded, $\mathsf{IdentDS}$ can be used to identify the offending user.

Security of the Construction. We give an intuition on why the protocol is secure (see App. E.3 for the formal version of the argument): the commitment binds the user to the values in a token. The ZK property of the proof system protects the user's actions from being linked between executions. Its soundness ensures that the user cannot cheat. Thus, at the end of the protocol, the user has a new token with an updated balance. The operator only learns that the old and new token are valid, the value v of the transaction, and the double-spending tag.

$\mathsf{UVer}(\mathsf{pk}_{\mathcal{O}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, \mathcal{T}, \mathbf{b})$

parse: $(\mathbf{c}, \mathbf{r}, \mathsf{sk}_{\mathcal{U}}, \mathbf{b}, \mathbf{s}_{\mathcal{U}}, \mathbf{s}_{\mathcal{O}}, \mathbf{u}_{\mathcal{U}}, \mathsf{sig}) \coloneqq \mathcal{T}$
parse: $(\tau_{\mathsf{sig}}, \mathbf{v}_{\mathsf{sig}}, \mathbf{r}) \coloneqq \mathsf{sig}$
$\mathsf{msg} \coloneqq \mathsf{sk}_{\mathcal{U}}, \mathbf{b}, \mathsf{bin}(\mathbf{s}_{\mathcal{U}}), \mathsf{bin}(\mathbf{s}_{\mathcal{O}}), \mathsf{bin}(\mathbf{u}_{\mathcal{U}})$
if $\mathbf{c} = \mathsf{C.Com}(\mathsf{msg}; \mathbf{r})$
$\quad \wedge\, \mathsf{S.Verify}(\mathsf{pk}_{\mathcal{O}}, \mathsf{sig}, \mathsf{msg}) = 1$
return 1
else return 0

$\mathsf{IdentDS}(\mathsf{pk}_{\mathcal{O}}, (\mathbf{s}, z_1), (\mathbf{s}', z_2))$

parse: $(\mathbf{t}, \mathbf{u}_{\mathcal{O}}) \coloneqq z_1, (\mathbf{t}', \mathbf{u}'_{\mathcal{O}}) \coloneqq z_2$
if $\mathbf{s} \neq \mathbf{s}' \vee \mathbf{u}_{\mathcal{O}} = \mathbf{u}'_{\mathcal{O}}$ **return** $\perp$
else
$\mathsf{sk}_{\mathcal{U}} \coloneqq (\mathbf{t} - \mathbf{t}')$
$\quad \cdot (H_{FRD}(\mathbf{u}_{\mathcal{O}}) - H_{FRD}(\mathbf{u}'_{\mathcal{O}}))^{-1} \bmod q_0$
$\mathsf{pk}_{\mathcal{U}} \coloneqq \mathbf{F} \cdot \mathsf{sk}_{\mathcal{U}}$
return $(\mathsf{pk}_{\mathcal{U}}, \Pi = \mathsf{sk}_{\mathcal{U}})$

Fig. 4: UVer for token verification, and IdentDS to handle double-spending.

Discussion. We point out a difference between our construction of BBA and previous ones [21, 22, 24]. There, range proofs were expensive and therefore optional. However, going without range proofs was only possible if the user revealed the balance when spending points, leaking a lot of information. Due to the lattice-based setting, our ZK proofs implicitly ensure that the balance is within the allowed range $\mathbb{V}$. This is because our proofs rely on bit decomposition. That is, we prove $\mathsf{toInt}(\mathbf{x}) = \sum_{i=0}^{m_{\mathfrak{b}}-1} \mathbf{x}_i \cdot 2^i \in \{0, \ldots, 2^{m_{\mathfrak{b}}} - 1\} = \mathbb{V}$ and $\mathbf{x} \in \{0,1\}^{m_{\mathfrak{b}}}$. Consequently, $\mathsf{toInt}(\mathbf{x})$ is a positive integer. (Recall that $2^{m_{\mathfrak{b}}} < q/4$, so the unique representative of the congruence class in $\mathbb{Z}_q$ is positive.) More precisely, we prove that $\mathbf{b}_{\text{new}}$ is of the form $\mathbf{b}_{\text{new}} = \mathsf{bin}(\mathsf{toInt}(\mathbf{b}) + v \mod q)$. Now, if $\mathsf{toInt}(\mathbf{b}) + v \mod q$ would be negative or bigger than $2^{m_{\mathfrak{b}}} - 1$, the user could not generate a ZK proof which would be accepted by the operator.

Detecting Double-Spending. To identify a double-spender, in other words, a user who tries to spend the same token twice, the operator runs the IdentDS algorithm (see Fig. 4). For double-spending detection, the operator requires access to the database of double-spending tags. When the user did double-spend, the operator can calculate the user's secret-key $\mathsf{sk}_{\mathcal{U}}$ from the double-spending tags.

The VerifyGuilt algorithm takes the users public key $\mathsf{pk}_{\mathcal{U}}$ and a proof Π of double-spending. The algorithm outputs 1 if $\mathbf{F} \cdot \Pi = \mathsf{pk}_{\mathcal{U}} \mod q_0 \wedge \|\Pi\|_\infty \leq 1$. Note, that it is not possible to generate a proof of guilt for benign user, as our construction offers *false-accusation protection.*

User-Verify Algorithm. The UVer algorithm (see Fig. 4) checks if the commitment $\mathbf{c}$ contained in the token is truly a commitment on the messages $\mathsf{sk}_{\mathcal{U}}, \mathbf{b}, \mathsf{bin}(\mathbf{s}_{\mathcal{U}}), \mathsf{bin}(\mathbf{s}_{\mathcal{O}})$ and $\mathsf{bin}(\mathbf{u}_{\mathcal{U}})$ with randomness $\mathbf{r}$. Further the algorithm checks if sig is a valid signature on the commitment $\mathbf{c}$ under the operator's secret key. The algorithm outputs 1 if both conditions are fulfilled, otherwise 0.

4 Efficiency Evaluation

In this section, we evaluate the efficiency of our construction. We concentrate on the communication cost, as this is the main bottleneck for mobile payments. Therefore, we first analyze the communication cost of an Update transaction.

Next, we briefly explain our choice of parameters and calculate the concrete communication cost of Update and Issue given those parameters. We also compare the efficiency of our instantiation to similar protocols.

Communication Cost. From Fig. 3 (right) we can derive the cost of Update. We denote the communication cost by $\mathcal{C}_{\mathsf{Update}}$ and we write $|\mathbf{vec}|$ for the number of bits needed to represent a vector $\mathbf{vec}$.

$$\mathcal{C}_{\mathsf{Update}} = |\mathbf{u}_{\mathcal{O}}| + |\mathbf{c}_{\text{new}}| + |\mathbf{s}| + |\mathbf{t}| + |\pi| + |\mathbf{c}_{\text{final}}| + |(\tau, \mathbf{v}_{\mathsf{sig}}, \mathbf{r}'')| + |\mathbf{s}'_{\mathcal{O}}|$$

The biggest part of this sum is $|\pi|$. Therefore, we will further analyze the size of the proof. According to Yang et al. [43] the size of this proof is

$$|\pi| = (\log(2p+1) + \kappa + (3l_1 + 2l_2 + 2m_{\mathsf{Update}} + 2l_{\mathsf{Update}}) \cdot \log q) \cdot N + (l_1 + m_{\mathsf{Update}}) \cdot \log q, \quad (2)$$

where p, κ, l_1, l_2, N are parameters of the zero-knowledge protocol, m_{Update} is the length of the witness and l_{Update} is the size of the set $\mathcal{M}$.

Using the fast mode (cf. [43, Sec. 1.2]) to prove the norm bounds on $\mathbf{r}$ and $\mathbf{v}$ we arrive at the size $m_{\mathsf{Update}} \coloneqq |\tau| + 2n_{\mathsf{S}}|\tau| + 2m_{\mathsf{S}} + n_{\mathsf{C}} \log q + 2m_{\mathbf{r}} + m_{\mathsf{sk}} + 2m_{\mathbf{b}} + 3m_{\mathsf{nr}} \log q_0 + 2m_{\mathsf{sk}} \log q_0 + m_{\mathbf{r}} + b\lambda(\log(10m_{\mathsf{S}}\beta/b) + 1)$ for the witness and $l_{\mathsf{Update}} \coloneqq |\tau| + n_{\mathsf{S}}|\tau| + n_{\mathsf{C}} \log q + 2m_{\mathbf{b}} \log q_0 + 3m_{\mathsf{nr}} \log q_0 + 2m_{\mathsf{sk}} + b\lambda(\log(10m_{\mathsf{S}}\beta/b) + 1)$ for $\mathcal{M}$, where b allows for a trade-off between proof size and tightness of the proven bound.

Next, we look at the signature $|\mathsf{sig}|$. By definition, we have $\mathsf{sig} = (\tau, \mathbf{v}, \mathbf{s})$, where $\tau \in \mathbb{Z}_2^l, \mathbf{v} \in \mathbb{Z}_q^{2m_{\mathsf{S}}}$ and $\mathbf{s} \in \mathbb{Z}_q^{2m_{\mathsf{S}}}$. Hence, we get $|\mathsf{sig}| = l + 4m_{\mathsf{S}} \cdot \log(\beta)$ bits.

Choice of Parameters. To provide practical parameters for our scheme, we follow the heuristic approach of setting parameters high enough to withstand best-known attacks instead of deriving them from a reduction to a hard lattice problem such as SVP. For the sake of comparison we choose parameters for 80-bit security. In real-world scenarios, a higher level of security is desirable. To do so, we examine the root Hermite factor (RHF) [17] of our SIS/LWE problems. According to [43], to achieve 80-bit security a RHF of at most 1.0048 is required. We follow [43] and estimate the required RHF as

$$\text{RHF}(SIS_{n,q,\beta}) \approx 2^{\frac{\log^2 \beta}{4n \log q}} \quad \text{resp.} \quad \text{RHF}(LWE_{n,q,\alpha}) \approx 2^{\frac{\log^2 \frac{\alpha}{5.31}}{4n \log q}}$$

see [26], resp. [3]. BABL relies on the the following assumptions:

- $SIS_{n_{\mathsf{sk}},q,\sqrt{m_{\mathsf{sk}}}}$ so it is infeasible to derive the secret key from the public key.
- $SIS_{n_{\mathsf{C}},q,2\sigma_{\mathsf{Com}}\sqrt{2m_{\mathbf{r}}}\frac{m_{\mathbf{r}}}{b}}$ for the commitment scheme to be binding.
- $SIS_{n_{\mathsf{S}},q,\beta'\frac{m_{\mathsf{S}}}{b}}$, $SIS_{n_{\mathsf{S}},q,\beta''\frac{m_{\mathsf{S}}}{b}}$ for the signature to be secure.
- SIS_{l_1,q,β_1}, SIS_{l_2,q,β_2} and $LWE_{l_2,q,\alpha}$ for the zero-knowledge protocol [43].

where factors $\frac{m_{\mathbf{r}}}{b}$, $\frac{m_{\mathsf{S}}}{b}$ are due to the soundness loss from the fast mode of [43]. From the ZK argument we have the constraints that $q_0 > p$ and $q > \max(\beta_1, \beta_2)$, where we repeat the proof $\lceil 2^{80}/p \rceil$ times to achieve a soundness error of 2^{-80}.

Param.	Value	Param.	Value	Param.	Value
p	2^{80}	$\log q_0$	100	$\log q$	200
l_1	7050	n_{S}	880	n_{C}	880
l_2	7000	m_{S}	352 000	$m_{\mathbf{r}}$	352 000
σ_3	67	l_τ	80	σ_{Com}	4195.0
σ_4	5.96×10^{30}	σ	671.0	m_{nr}	1
β_1	1.75×10^{59}	β	8055.0	$m_{\mathbf{b}}$	32
β_2	8×10^{58}	β'	9.6×10^{20}		
α	1.05×10^{-58}	β''	1.29×10^{19}		
b	16			n_{sk}	9
κ	128			m_{sk}	900
	RHF		RHF		RHF
SIS_{l_1,q,β_1}	1.0048	$SIS_{n_{\mathsf{S}},q,\beta'}$	1.0048	$SIS_{n_{\mathsf{C}},q,2\sqrt{2m_{\mathbf{r}}}\sigma_{\mathsf{Com}}\frac{m_{\mathbf{r}}}{b}}$	1.0014
SIS_{l_2,q,β_2}	1.0048	$SIS_{n_{\mathsf{S}},q,\beta''}$	1.004	$SIS_{n_{\mathsf{sk}},q,\sqrt{m_{\mathsf{sk}}}}$	1.0046
$LWE_{l_2,q,\alpha}$	1.0047				

Table 1: Concrete choices of parameters and resulting values for the underlying assumptions of the zero-knowledge proof (left), the signature scheme (middle), the commitment scheme (right) and for the secret keys (bottom right).

We tested for values of $p = 2^{10}$ up to $p = 2^{80}$ and arrived at the smallest proof size for $p = 2^{80}$. Then we set q_0 such that $q = q_0^2$ is just big enough. Finally we set all dimensions n just high enough to achieve the desired RHF of 1.0048. This resulted in the parameters shown in Table 1. The size of the proof π is 197 MB. Overall, the communication cost for Update is 199 MB and for Issue 70 MB.

Comparison with Similar Protocols. In Table 2, we compare the result for our instantiation with other protocols. The given values for our construction, and E-Cash of [43] and of [28] are theoretical estimations, while the values for BBA+ [24] and BBW [21] are results of empirical experiments on a software implementation (the benchmarks are described in [21]). Therefore, the comparison should be taken with a grain of salt. However, it suffices to illustrate the efficiency gap between the lattice-based constructions and the elliptic curve-based ones. Note, that our construction is slightly more efficient than the construction of [43]. Even though we used the same zero-knowledge proof, we do not need to prove statements about the correct evaluation of weak pseudorandom functions.

5 Future Work

Post-Quantum Security. Despite recent progress in proving Fiat–Shamir transformations of Σ-protocols secure in the quantum random oracle model [16, 32] none of the results seems to apply to our setting. That is, even if we assume that the results apply to [43], the resulting notion of security is not sufficient for our proofs. We essentially require witness-extended emulation (WEE) [20, 29],

i.e., except with negligible probability an accepting proof can be extracted. This is a stronger notion than the knowledge soundness, which [16, 32] use. In the classical setting, the difference is small since amplification (via rewinding) can be used to obtain WEE from knowledge soundness [29]. In the quantum setting, this is unclear. A possible remedy would be a transformation which allows online extraction of the witness, e.g. by (additionally) committing to the witness with an extractable commitment scheme or a dual-mode commitment scheme. Knowledge of the extraction trapdoor allows to prove "operator soundness", while the hiding property still ensures the "privacy notions". However, this would further increase proof sizes and introduce a global system trapdoor (which is undesirable).

Efficiency. As seen in Section 4, our construction requires a high amount of network traffic. For real-world scenarios this cost is still unacceptably high. Basing a construction on the stronger assumptions of Ring-LWE and Ring-SIS should allow more efficient schemes. However, while more efficient zero-knowledge proofs are known in the ring setting, we are not aware of more efficient CL-type signatures. Thus, this remains the most important open question.

References

[1] S. Agrawal, D. Boneh, and X. Boyen. "Efficient Lattice (H)IBE in the Standard Model". In: *EUROCRYPT 2010*. Ed. by H. Gilbert. Vol. 6110. LNCS. Springer, Heidelberg, 2010, pp. 553–572. DOI: 10.1007/978-3-642-13190-5_28.

[2] M. Ajtai. "Generating Hard Instances of Lattice Problems (Extended Abstract)". In: *28th ACM STOC*. ACM Press, 1996, pp. 99–108. DOI: 10.1145/237814.237838.

[3] M. R. Albrecht, R. Player, and S. Scott. *On The Concrete Hardness Of Learning With Errors*. Cryptology ePrint Archive, Report 2015/046. https://eprint.iacr.org/2015/046.

[4] J. Alwen and C. Peikert. *Generating Shorter Bases for Hard Random Lattices*. Cryptology ePrint Archive, Report 2008/521. https://eprint.iacr.org/2008/521.

Protocol	**Issuance**	**Transaction**	**Token/Wallet**	**Based on**
Our work	70 MB	199 MB	11 MB	Lattices
E-Cash [28]	33 TB	720 TB	4 MB	Lattices
E-Cash [43]	53 MB	262 MB	4 MB	Lattices
BBA+ [24]	1 kB	14 kB	<1 kB	Elliptic Curves
BBW [21]	1 kB	5 kB	<1 kB	Elliptic Curves

Table 2: Comparison of the efficiency of similar protocols with our work

[5] W. Banaszczyk. “New bounds in some transference theorems in the geometry of numbers”. In: *Mathematische Annalen* 296.1 (1993), pp. 625–635. DOI: 10.1007/BF01445125.

[6] J. Blömer, J. Bobolz, D. Diemert, and F. Eidens. “Updatable Anonymous Credentials and Applications to Incentive Systems”. In: *ACM CCS 2019*. Ed. by L. Cavallaro, J. Kinder, X. Wang, and J. Katz. ACM Press, 2019, pp. 1671–1685. DOI: 10.1145/3319535.3354223.

[7] J. Bobolz, F. Eidens, S. Krenn, D. Slamanig, and C. Striecks. “Privacy-Preserving Incentive Systems with Highly Efficient Point-Collection”. In: *ASIACCS 20*. Ed. by H.-M. Sun, S.-P. Shieh, G. Gu, and G. Ateniese. ACM Press, 2020, pp. 319–333. DOI: 10.1145/3320269.3384769.

[8] F. Bourse, D. Pointcheval, and O. Sanders. “Divisible E-Cash from Constrained Pseudo-Random Functions”. In: *ASIACRYPT 2019, Part I*. Ed. by S. D. Galbraith and S. Moriai. Vol. 11921. LNCS. Springer, Heidelberg, 2019, pp. 679–708. DOI: 10.1007/978-3-030-34578-5_24.

[9] J. Camenisch, S. Hohenberger, and A. Lysyanskaya. “Compact E-Cash”. In: *EUROCRYPT 2005*. Ed. by R. Cramer. Vol. 3494. LNCS. Springer, Heidelberg, 2005, pp. 302–321. DOI: 10.1007/11426639_18.

[10] J. Camenisch and A. Lysyanskaya. “A Signature Scheme with Efficient Protocols”. In: *SCN 02*. Ed. by S. Cimato, C. Galdi, and G. Persiano. Vol. 2576. LNCS. Springer, Heidelberg, 2003, pp. 268–289. DOI: 10.1007/3-540-36413-7_20.

[11] J. Camenisch and A. Lysyanskaya. “Signature Schemes and Anonymous Credentials from Bilinear Maps”. In: *CRYPTO 2004*. Ed. by M. Franklin. Vol. 3152. LNCS. Springer, Heidelberg, 2004, pp. 56–72. DOI: 10.1007/978-3-540-28628-8_4.

[12] S. Canard and A. Gouget. “Anonymity in Transferable E-cash”. In: *ACNS 08*. Ed. by S. M. Bellovin, R. Gennaro, A. D. Keromytis, and M. Yung. Vol. 5037. LNCS. Springer, Heidelberg, 2008, pp. 207–223. DOI: 10.1007/978-3-540-68914-0_13.

[13] D. Cash, D. Hofheinz, E. Kiltz, and C. Peikert. “Bonsai Trees, or How to Delegate a Lattice Basis”. In: *Journal of Cryptology* 25.4 (2012), pp. 601–639. DOI: 10.1007/s00145-011-9105-2.

[14] D. Chaum. “Blind Signatures for Untraceable Payments”. In: *Advances in Cryptology* (1983), pp. 199–203. DOI: 10.1007/978-1-4757-0602-4_18.

[15] A. Deo, B. Libert, K. Nguyen, and O. Sanders. “Lattice-Based E-Cash, Revisited”. In: *ASIACRYPT 2020, Part II*. Ed. by S. Moriai and H. Wang. Vol. 12492. LNCS. Springer, Heidelberg, 2020, pp. 318–348. DOI: 10.1007/978-3-030-64834-3_11.

[16] J. Don, S. Fehr, C. Majenz, and C. Schaffner. “Security of the Fiat-Shamir Transformation in the Quantum Random-Oracle Model”. In: *CRYPTO 2019, Part II*. Ed. by A. Boldyreva and D. Micciancio. Vol. 11693. LNCS. Springer, Heidelberg, 2019, pp. 356–383. DOI: 10.1007/978-3-030-26951-7_13.

[17] N. Gama and P. Q. Nguyen. "Predicting Lattice Reduction". In: *EUROCRYPT 2008*. Ed. by N. P. Smart. Vol. 4965. LNCS. Springer, Heidelberg, 2008, pp. 31–51. DOI: 10.1007/978-3-540-78967-3_3.

[18] C. Gentry. "Fully homomorphic encryption using ideal lattices". In: *41st ACM STOC*. Ed. by M. Mitzenmacher. ACM Press, 2009, pp. 169–178. DOI: 10.1145/1536414.1536440.

[19] C. Gentry, C. Peikert, and V. Vaikuntanathan. "Trapdoors for hard lattices and new cryptographic constructions". In: *40th ACM STOC*. Ed. by R. E. Ladner and C. Dwork. ACM Press, 2008, pp. 197–206. DOI: 10.1145/1374376.1374407.

[20] J. Groth and Y. Ishai. "Sub-linear Zero-Knowledge Argument for Correctness of a Shuffle". In: *EUROCRYPT 2008*. Ed. by N. P. Smart. Vol. 4965. LNCS. Springer, Heidelberg, 2008, pp. 379–396. DOI: 10.1007/978-3-540-78967-3_22.

[21] G. Hartung, M. Hoffmann, M. Nagel, and A. Rupp. "BBA+: Improving the Security and Applicability of Privacy-Preserving Point Collection". In: *ACM CCS 2017*. Ed. by B. M. Thuraisingham, D. Evans, T. Malkin, and D. Xu. ACM Press, 2017, pp. 1925–1942. DOI: 10.1145/3133956.3134071.

[22] M. Hoffmann, M. Klooß, M. Raiber, and A. Rupp. "Black-Box Wallets: Fast Anonymous Two-Way Payments for Constrained Devices". In: *PoPETs* 2020.1 (2020), pp. 165–194. DOI: 10.2478/popets-2020-0010.

[23] S. Hohenberger and B. Waters. "Short and Stateless Signatures from the RSA Assumption". In: *CRYPTO 2009*. Ed. by S. Halevi. Vol. 5677. LNCS. Springer, Heidelberg, 2009, pp. 654–670. DOI: 10.1007/978-3-642-03356-8_38.

[24] T. Jager and A. Rupp. "Black-Box Accumulation: Collecting Incentives in a Privacy-Preserving Way". In: *PoPETs* 2016.3 (2016), pp. 62–82. DOI: 10.1515/popets-2016-0016.

[25] A. Kawachi, K. Tanaka, and K. Xagawa. "Concurrently Secure Identification Schemes Based on the Worst-Case Hardness of Lattice Problems". In: *ASIACRYPT 2008*. Ed. by J. Pieprzyk. Vol. 5350. LNCS. Springer, Heidelberg, 2008, pp. 372–389. DOI: 10.1007/978-3-540-89255-7_23.

[26] A. Kosba, Z. Zhao, A. Miller, Y. Qian, H. Chan, C. Papamanthou, R. Pass, abhi shelat, and E. Shi. *C∅C∅: A Framework for Building Composable Zero-Knowledge Proofs*. Cryptology ePrint Archive, Report 2015/1093. https://eprint.iacr.org/2015/1093.

[27] 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, Part II*. Ed. by J. H. Cheon and T. Takagi. Vol. 10032. LNCS. Springer, Heidelberg, 2016, pp. 373–403. DOI: 10.1007/978-3-662-53890-6_13.

[28] B. Libert, S. Ling, K. Nguyen, and H. Wang. "Zero-Knowledge Arguments for Lattice-Based PRFs and Applications to E-Cash". In: *ASIACRYPT 2017, Part III*. Ed. by T. Takagi and T. Peyrin. Vol. 10626.

LNCS. Springer, Heidelberg, 2017, pp. 304–335. DOI: 10.1007/978-3-319-70700-6_11.

[29] Y. Lindell. "Parallel Coin-Tossing and Constant-Round Secure Two-Party Computation". In: *CRYPTO 2001*. Ed. by J. Kilian. Vol. 2139. LNCS. Springer, Heidelberg, 2001, pp. 171–189. DOI: 10.1007/3-540-44647-8_10.

[30] S. Ling, K. Nguyen, D. Stehlé, and H. Wang. "Improved Zero-Knowledge Proofs of Knowledge for the ISIS Problem, and Applications". In: *PKC 2013*. Ed. by K. Kurosawa and G. Hanaoka. Vol. 7778. LNCS. Springer, Heidelberg, 2013, pp. 107–124. DOI: 10.1007/978-3-642-36362-7_8.

[31] H. Lipmaa, N. Asokan, and V. Niemi. "Secure Vickrey Auctions without Threshold Trust". In: *FC 2002*. Ed. by M. Blaze. Vol. 2357. LNCS. Springer, Heidelberg, 2003, pp. 87–101. DOI: 10.1007/3-540-36504-4_7.

[32] Q. Liu and M. Zhandry. "Revisiting Post-quantum Fiat-Shamir". In: *CRYPTO 2019, Part II*. Ed. by A. Boldyreva and D. Micciancio. Vol. 11693. LNCS. Springer, Heidelberg, 2019, pp. 326–355. DOI: 10.1007/978-3-030-26951-7_12.

[33] V. Lyubashevsky. "Fiat-Shamir with Aborts: Applications to Lattice and Factoring-Based Signatures". In: *ASIACRYPT 2009*. Ed. by M. Matsui. Vol. 5912. LNCS. Springer, Heidelberg, 2009, pp. 598–616. DOI: 10.1007/978-3-642-10366-7_35.

[34] V. Lyubashevsky. "Lattice Signatures without Trapdoors". In: *EUROCRYPT 2012*. Ed. by D. Pointcheval and T. Johansson. Vol. 7237. LNCS. Springer, Heidelberg, 2012, pp. 738–755. DOI: 10.1007/978-3-642-29011-4_43.

[35] V. Lyubashevsky. "Lattice-Based Identification Schemes Secure Under Active Attacks". In: *PKC 2008*. Ed. by R. Cramer. Vol. 4939. LNCS. Springer, Heidelberg, 2008, pp. 162–179. DOI: 10.1007/978-3-540-78440-1_10.

[36] C. Peikert. *A Decade of Lattice Cryptography*. Cryptology ePrint Archive, Report 2015/939. https://eprint.iacr.org/2015/939.

[37] C. Peikert. "An Efficient and Parallel Gaussian Sampler for Lattices". In: *CRYPTO 2010*. Ed. by T. Rabin. Vol. 6223. LNCS. Springer, Heidelberg, 2010, pp. 80–97. DOI: 10.1007/978-3-642-14623-7_5.

[38] C. Peikert and B. Waters. "Lossy trapdoor functions and their applications". In: *40th ACM STOC*. Ed. by R. E. Ladner and C. Dwork. ACM Press, 2008, pp. 187–196. DOI: 10.1145/1374376.1374406.

[39] O. Regev. "On lattices, learning with errors, random linear codes, and cryptography". In: *37th ACM STOC*. Ed. by H. N. Gabow and R. Fagin. ACM Press, 2005, pp. 84–93. DOI: 10.1145/1060590.1060603.

[40] P. W. Shor. "Algorithms for Quantum Computation: Discrete Logarithms and Factoring". In: *35th FOCS*. IEEE Computer Society Press, 1994, pp. 124–134. DOI: 10.1109/SFCS.1994.365700.

[41] J. Stern. "A new paradigm for public key identification". In: *IEEE Transactions on Information Theory* 42.6 (1996), pp. 1757–1768. DOI: 10.1109/18.556672.

[42] C. Weng, K. Yang, J. Katz, and X. Wang. *Wolverine: Fast, Scalable, and Communication-Efficient Zero-Knowledge Proofs for Boolean and Arithmetic Circuits*. Cryptology ePrint Archive, Report 2020/925. https://eprint.iacr.org/2020/925.

[43] R. Yang, M. H. Au, Z. Zhang, Q. Xu, Z. Yu, and W. Whyte. "Efficient Lattice-Based Zero-Knowledge Arguments with Standard Soundness: Construction and Applications". In: *CRYPTO 2019, Part I*. Ed. by A. Boldyreva and D. Micciancio. Vol. 11692. LNCS. Springer, Heidelberg, 2019, pp. 147–175. DOI: 10.1007/978-3-030-26948-7_6.

A Hardness Assumptions and Cryptographic Building Blocks

A.1 Lattice-based Hardness Assumptions

Definition A.1 (Short Integer Solution). *Given a modulus $q \in \mathbb{N}$, $m \in \mathbb{N}$ uniformly random vectors $\mathbf{a}_i \in \mathbb{Z}_q^n$ (written as a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$), and a uniformly random vector $\mathbf{u} \in \mathbb{Z}_q^n$, the* Inhomogeneous Short Integer Solution *(ISIS) problem is to find a non-zero integer vector $\mathbf{z} \in \mathbb{Z}^m$ of norm $\|\mathbf{z}\| \leq \beta$ such that*

$$\mathbf{Az} = \sum_{i=1}^{m} \mathbf{a}_i \cdot z_i = \mathbf{u} \in \mathbb{Z}_q^n,$$

where $\beta \in \mathbb{R}$ is a parameter with $\beta < q$.

In the case where $\mathbf{u}$ is not uniform but fixed to $\mathbf{0}$, the problem is called Short Integer Solution *(SIS). We write $SIS_{n,q,\beta,m}$, if we want to emphasize the respective parameters.*

For typical parameter choices, SIS and ISIS are equivalent. Ajtai showed in his seminal work [2] that the average-case SIS problem can be reduced in polynomial time to the short integer vector problem (SIVP), a worst-case problem on lattices.

Regev [39] introduced the LWE problem and gave a quantum reduction to SIVP. We define the decisional variant of the respective hardness assumption.

Definition A.2 (Learning with Errors). *$LWE_{n,q,\chi,m}$: For a secret vector $\mathbf{s} \in \mathbb{Z}_q^n$ and a probability distribution χ over $\mathbb{Z}_q^m$, sample a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ uniformly at random and a vector $\mathbf{e} \leftarrow \chi$. Given $(\mathbf{A}, \mathbf{b}^*)$ where $\mathbf{b}^*$ is either $\mathbf{b}_0$ or $\mathbf{b}_1$, where $\mathbf{b}_0^\top = \mathbf{s}^\top \mathbf{A} + \mathbf{e} \mod q$ and $\mathbf{b}_1$ is chosen uniformly at random. Decide whether $\mathbf{b}^* = \mathbf{b}_0$ or $\mathbf{b}^* = \mathbf{b}_1$.*

A.2 Building Blocks

We give a brief overview of the used building blocks. In Section 2.3, we give instantiations of these building blocks based on the hardness of SIS and LWE.

Commitments. A *commitment scheme* allows one to *commit* to (i.e., fix) a value, without revealing it immediately. At a later point, the commitment can be *opened* and the committed value is revealed. A commitment scheme consists of two PPT algorithms: The parameter generation $\mathsf{Gen}(1^\lambda)$ outputs public parameters params, and the commitment algorithm $\mathsf{Com}(\mathsf{params}, m; r) \to c$ outputs, for a given message m, some explicit randomness r and the public parameters params, a commitment c on that value m. We often omit the input of the public parameters.

To open the commitment c, one can reveal the randomness r, to check, if $\mathsf{Com}(\mathsf{params}, m; r) = c$ holds. Informally, we want a commitment scheme to be *hiding*, i.e. no (efficient) adversary can learn anything about the message m in a commitment, prior to opening. Furthermore, a commitment should be *binding* which means it should be (computationally) infeasible to open a commitment on m to any other value than m. A commitment scheme is *equivocal*, if there exist an additional trapdoor generation algorithm $\mathsf{EqGen}(1^\lambda)$ that outputs the public parameters params, together with a trapdoor td. With the trapdoor td, it is possible to open a commitment c on the value m to another value m' with $m' \neq m$, by running a second algorithm, called $\mathsf{Equiv}(\mathsf{td}, c, m')$ that outputs a randomness value r' for opening c to m'. We require the two setups via Gen and EqGen to be computationally indistinguishable.

Oblivious Signing of Committed Messages. In our construction, we use the signature scheme of Libert et al. [28]; in particular, their protocol for obliviously signing a committed message (see Section 2.3). A *signature scheme for oblivious signing of committed messages* consists of the following algorithms/protocols:

- A key-generation algorithm $\mathsf{Gen}(1^\lambda)$ that outputs $(\mathsf{params}, \mathsf{pk}, \mathsf{sk})$, namely public parameters params, and a pair consisting of a public and a secret key.
- $\mathsf{OblSign}\langle\mathcal{U}(\mathsf{params}, \mathsf{pk}, m), \mathcal{S}(\mathsf{params}, \mathsf{pk}, \mathsf{sk})\rangle$, an interactive protocol, where a user $\mathcal{U}$ interacts with a signer $\mathcal{S}$ to obtain a signature on a message m inside of a commitment. In this protocol, $\mathcal{U}$ sends a commitment $c \leftarrow \mathsf{Com}(m; r)$ on m to the signer $\mathcal{S}$ and eventually $\mathcal{U}$ outputs a valid signature on m.
- a verification algorithm $\mathsf{Vfy}(\mathsf{params}, \mathsf{pk}, m, \mathsf{sig}) \to b$ that allows to check, whether sig is a valid signature on message m public key pk.

The signer does not learn anything about m, as the commitment scheme is hiding. This protocol offers a security notion that is almost identical to common EUF-CMA security but takes into account that the user sends commitments and not plain messages. Libert et al. forgo an abstract definition of the signature's security as they directly apply the signature scheme to their E-Cash. We give a formal definition in App. C.

Zero-Knowledge Proofs. A proof system allows a party, called *prover*, to prove to another party, called *verifier*, that some statement is true. It is a *zero-knowledge (ZK)* protocol, if (informally) the verifier gains no additional knowledge, except for the truth of the statement. More precisely, the prover can convince the verifier that a word x belongs to a certain $\mathcal{NP}$-language L, while even a malicious verifier learns nothing about x except for the truth of $x \in L$. The protocol is a *proof of*

knowledge (PoK), or *extractable*, if a convincing prover must know an $\mathcal{NP}$-witness w. For example, if $x = m$, $w = \sigma$, and the language is "I know a signature σ on message m", then a ZK-PoK guarantees that a convincing prover knows a signature σ, yet, the verifier learns nothing about σ. A ZK-PoK is correct, if an honest execution with correct statement always accepts. It has *soundness error* $p \in [0, 1]$, if the probability that the verifier accepts a false statement is at most p.

Full-Rank Difference Function. We define full-rank differences as introduced by Agrawal, Boneh, and Boyen [1], and refer to them for a concrete instantiation. We use this in the calculation of the double-spending tag. Let $q \in \mathbb{N}$ be a prime and $n \in \mathbb{N}$. A *full-rank difference function* is an efficiently computable function $H_{FRD} \colon \mathbb{Z}_q^n \to \mathbb{Z}_q^{n \times n}$ satisfying that for all distinct $u, v \in \mathbb{Z}_q^n$, the matrix $H_{FRD}(u) - H_{FRD}(v) \in \mathbb{Z}_q^{n \times n}$ is full rank.

B Equivocality of the Commitment Scheme

For the proof that our system is privacy-preserving (cf. Theorem E.6), we need an equivocal commitment scheme. In the proof, we replace all personal information of the user with random values and show that an adversary will (with high probability) not notice the difference. However, as the adversary can adaptively corrupt parties, we need to be able to open commitments of users to "plausible" values, once the users are corrupted. We thus extend the commitment scheme from [25] to get equivocality by using lattice trapdoor gadgets.

Instead of choosing, a uniformly random matrix $\mathbf{D}_0 \in \mathbb{Z}_q^{n \times m_0}$ for the commitment scheme, we use the trapdoor generation algorithm described in [36]. Thus, our matrix $\mathbf{D}_0' \in \mathbb{Z}_q^{n \times m_0}$ is output alongside with a trapdoor matrix $\mathbf{R}$, which we will describe later in more detail. This $\mathbf{R}$ allows to compute short (I)SIS solutions. $\mathbf{D}_0'$ is close to a uniformly random matrix. We denote this by $(\mathbf{D}_0', \mathbf{R}) \leftarrow \mathsf{TrapKeyGen}(1^n, m, q)$. This is done as follows: First, we write $\mathbf{G}$ for the "powers-of-two" matrix

$$\mathbf{G} := \begin{pmatrix} \mathbf{g}^\top & & \\ & \ddots & \\ & & \mathbf{g}^\top \end{pmatrix} \in \mathbb{Z}_q^{n \times nl},$$

where $\mathbf{g} = (2^{l-1}, \ldots, 4, 2, 1)^\top \in \mathbb{Z}_q^l$ and $l = \lceil \log q \rceil$. Choose $\bar{\mathbf{A}} \leftarrow \mathbb{Z}_q^{n \times \bar{m}}$ uniformly at random for a sufficiently large $\bar{m}$, such that $\bar{m} + nl = m$ and a random integer matrix $\bar{\mathbf{R}} \in \mathbb{Z}_q^{\bar{m} \times nl}$ of small norm, i.e., a matrix having discrete Gaussian entries. Let $\mathbf{D}_0' := [\bar{\mathbf{A}} \mid \mathbf{G} - \bar{\mathbf{A}}\bar{\mathbf{R}}] \in \mathbb{Z}_q^{n \times \bar{m} + nl}$. Define $\mathbf{R} = \begin{bmatrix} \bar{\mathbf{R}} \\ \mathbf{I} \end{bmatrix} \in \mathbb{Z}_q^{\bar{m} + nl \times nl}$. Then, it is clear that

$$\mathbf{D}_0'\mathbf{R} = \bar{\mathbf{A}}\bar{\mathbf{R}} + \mathbf{G} - \bar{\mathbf{A}}\bar{\mathbf{R}} = \mathbf{G} \in \mathbb{Z}_q^{n \times nl}.$$

The interesting point is that solving the SIS problem with respect to $\mathbf{G}$ is easy. Let's say we want to find a short $\mathbf{e} \in \mathbb{Z}_q^{nl}$ (i.e., $\|e\|_\infty \leq \beta, \beta \in \mathbb{R}$) with

$\mathbf{Ge} = \mathbf{v} \mod q$, for some $\mathbf{v} \in \mathbb{Z}_q^n$. One can simply use the binary representation $\mathsf{bin}(\mathbf{v}) = \mathbf{e} \in \mathbb{Z}_2^{nl}$. Next, if one gets our $\mathbf{D}'_0$, it is easy to find a short $\mathbf{x}$ with $\mathbf{D}'_0\mathbf{x} = \mathbf{v} \mod q$ given the trapdoor matrix $\mathbf{R}$. Let $\mathbf{x} = \mathbf{Re} \in \mathbb{Z}_q^{\bar{m}+nl}$ and $\mathbf{e} = \mathsf{bin}(\mathbf{v}) \in \mathbb{Z}_q^{nl}$, then

$$\mathbf{D}'_0\mathbf{x} = \mathbf{D}'_0\mathbf{Re} = \mathbf{Ge} = \mathbf{v} \mod q$$

and $\mathbf{x}$ is small, as $\mathbf{e}$ consists of zeroes and ones and $\mathbf{R}$ has small norm. Note, that $\mathbf{x} = \mathbf{Re}$ is not Gaussian-distributed and could leak information about the trapdoor $\mathbf{R}$. We therefore choose an appropriately distributed perturbation vector $\mathbf{p} \in \mathbb{Z}_q^{\bar{m}+nl}$ to "correct" the skewed distribution. Then, we sample $\mathbf{w} \leftarrow \mathsf{bin}(\mathbf{v} - \mathbf{D}'_0\mathbf{p})$ and finally set $\mathbf{x} := \mathbf{p} + \mathbf{Rw}$. This "convolution" technique was introduced in [37]. Now, the trapdoor parameter generation and the equivocation of a commitment $\mathbf{c} \in \mathbb{Z}_q^n$ work as follows:
$\mathsf{EqGen}(1^\lambda)$: Generate a close-to-uniform matrix $\mathbf{D}'_0 \in \mathbb{Z}_q^{n \times m_0}$ alongside with a trapdoor as described above via $(\mathbf{D}'_0, \mathbf{R}) \leftarrow \mathsf{TrapGen}(1^\lambda, n, m_0, q)$ and a uniformly random matrix $\mathbf{D}_1 \leftarrow \mathbb{Z}_q^{n \times m_1}$. Output $((\mathbf{D}'_0, \mathbf{D}_1), \mathbf{R})$.
$\mathsf{Equiv}(\mathbf{R}, \mathbf{c}, \mathbf{m}')$: Let $\mathbf{c}' = \mathbf{c} - \mathbf{D}_1 \cdot \mathbf{m}' \in \mathbb{Z}_q^n$. Now use $\mathbf{R}$, to find a short solution $\mathbf{x}$ to $\mathbf{D}'_0 \cdot \mathbf{x} = \mathbf{c}' \mod q$. We get

$$\mathbf{D}'_0 \cdot \mathbf{x} + \mathbf{D}_1 \cdot \mathbf{m}' = \mathbf{c}' + \mathbf{D}_1 \cdot \mathbf{m}' = \mathbf{c} \mod q.$$

Output the opening information $(\mathbf{x}, \mathbf{m}') \in \mathbb{Z}_q^{m_0} \times \mathbb{Z}_q^{m_1}$.

C Security of the Signatures

In this section we define the security of the signature scheme from [28]. The precise notion of security for the oblivious signing protocol was not given by [28] and the security proof is incorporated in the proof of the *balance* property of their e-cash system. Therefore we also give a short proof sketch.

Definition C.1. *We say a signature scheme for obliviously signing committed messages $\Pi = (\mathsf{Gen}, \mathsf{OblSign}, \mathsf{Verify})$ (formally defined in [28]) is* secure*, if for all PPT adversaries $\mathcal{A}$, there is a negligible function $\mathsf{negl}(\lambda)$, s.t.:*

$$\mathsf{Pr}\left[\mathsf{Exp}^{\mathsf{OblSign\text{-}forge}}_{\mathcal{A},\Pi}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda).$$

In the experiment in Fig. 5 the adversary $\mathcal{A}$ can execute the $\mathsf{OblSign}$ protocol polynomial many times with an oracle $\mathsf{HonSign}$ acting as an honest signer. In these executions, $\mathcal{A}$ sends commitments $\mathbf{c}_1, \ldots, \mathbf{c}_n$ to $\mathsf{HonSign}$ and receives signatures on the committed messages ($n = \mathsf{poly}(\lambda)$). Finally, $\mathcal{A}$ must output a message signature pair $(\mathbf{m}^*, \mathsf{sig}^*)$ and openings $(\mathbf{r}_1, \mathbf{m}_1), \ldots, (\mathbf{r}_n, \mathbf{m}_n)$ to $\mathbf{c}_1, \ldots, \mathbf{c}_n$. This is exactly the notion of security we require for our proofs. Our proofs rely on the extractability of the zero-knowledge proof, and therefore it will suffice for the simulator to have commitments on certain messages as the messages can be extracted from the zero-knowledge proof.

Experiment $\mathsf{Exp}^{\mathsf{OblSign\text{-}forge}}_{\mathcal{A},\Pi}(\lambda)$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$
$(\mathbf{m}^*, \mathsf{sig}^*, st) \leftarrow \mathcal{A}^{\mathsf{HonSign}}(\mathsf{pk})$
$(\mathbf{r}_1, \mathbf{m}_1), \ldots, (\mathbf{r}_n, \mathbf{m}_n) \leftarrow \mathcal{A}(st)$
The experiment returns 1 iff
- $\mathsf{Verify}(\mathsf{pk}, \mathbf{m}^*, \mathsf{sig}^*) = 1$,
- $\forall i \in \{1, \ldots, n\} : \mathbf{m}^* \neq \mathbf{m}_i$,
- $(\mathbf{r}_1, \mathbf{m}_1), \ldots, (\mathbf{r}_n, \mathbf{m}_n)$ are valid openings to $\mathbf{c}_1, \ldots, \mathbf{c}_n$.

Fig. 5: Signature experiment

Proof sketch (Proof sketch for Lemma 4). Assume there is an adversary $\mathcal{A}$, winning the above experiment with non-negligible probability. We sketch the construction of an adversary $\mathcal{B}$, breaking $SIS_{n,q,\beta',m}$ or $SIS_{n,q,\beta'',m}$, with β', β'' like in Section 2.3. In the beginning $\mathcal{B}$ guesses the strategy, which $\mathcal{A}$ will use to win the game. There are three strategies and $\mathcal{B}$ chooses uniformly at random between them. In case 1, $\mathcal{A}$ outputs $\mathsf{sig}^* = (\tau^*, \mathbf{v}^*, \mathbf{r}^*)$, where τ^* was never output by the oracle before. In case 2, $\mathcal{A}$ outputs $\mathsf{sig}^* = (\tau^*, \mathbf{v}^*, \mathbf{r}^*)$, where $\mathcal{A}$ reused one of the τ_i from an interaction with the oracle, but $\mathbf{c}^* = \mathbf{D}_0 \cdot \mathbf{r}^* + \mathbf{D}_1 \cdot \mathbf{m}^*$ was never signed by the oracle. In case 3, $\mathcal{A}$ outputs $\mathsf{sig}^* = (\tau^*, \mathbf{v}^*, \mathbf{r}^*)$, where $\mathcal{A}$ reused one of the τ_i and $\mathbf{c}^* = \mathbf{D}_0 \cdot \mathbf{r}^* + \mathbf{D}_1 \cdot \mathbf{m}^*$ from an interaction with the oracle.

In case 1, the prefix-guessing technique from [23] is used. Concretely, $\mathcal{B}$ guesses the length of the shortest prefix of τ^* that differs from the prefixes of $\tau_i, i \in \{1, \ldots, n\}$ of the same length. Then, $\mathcal{B}$ uses the trapdoor generation algorithm from Lemma 2 to hide a trapdoor in the public key. The matrices $\mathbf{A}_0, \ldots, \mathbf{A}_l$ in the public key are crafted, such that the trapdoor can be used to answer all signing queries, if the prefix guess was right. Further, if the guess was right, the trapdoor vanishes for τ^* and $\mathbf{m}^*, \mathsf{sig}^*$ yield a direct solution to a $SIS_{n,q,\beta',m}$ instance. The probability to guess the prefix correctly is $1/(\mathcal{Q} \cdot l)$, where $\mathcal{Q}$ is the number of $\mathsf{OblSign}$ executions and l the length of the tags.

In case 2, $\mathcal{B}$ guesses the index $\hat{i}$ of the execution of $\mathsf{OblSign}$ from which τ^* will be reused. Like before, a trapdoor matrix is hidden in the public key, which can be used to answer all $\mathsf{OblSign}$ queries, where $i \neq \hat{i}$. To answer the $\hat{i}$-th query, $\mathcal{B}$ chooses $\hat{\mathbf{c}} \leftarrow \mathbb{Z}_q^n, \mathbf{v} \leftarrow D_{\mathbb{Z}^m, \sigma_{\mathsf{Com}}}$ and sets $\mathbf{u} = \mathbf{A}_{\tau_{\hat{i}}} \mathbf{v} - \mathbf{B} \cdot \mathsf{vdec}(\hat{\mathbf{c}})$ beforehand. Again, the matrices $\mathbf{A}_0, \ldots, \mathbf{A}_l$ are crafted such that the trapdoor vanishes for τ^* and $\mathcal{B}$ can calculate a $SIS_{n,q,\beta',m}$ solution, if the adversary outputs a forgery.

Case 3 is a straightforward reduction on the binding property of the commitment scheme from [25], and therefore on $SIS_{n,q,\beta'',m}$.

D Zero-Knowledge Proofs for Issue and Update

This section describes the zero-knowledge arguments for our construction. We first sketch the general protocol. Then we show in detail how it is used to prove the necessary statements for proofs P1 and P2 from Section 3.

We employ the protocol of Yang et al. [43]. Roughly speaking, the protocol is a so-called Σ-protocol, and uses the Fiat–Shamir with abort techniques [33–35], to improve efficiency.

Let $l, m, n \in \mathbb{N}$ be positive integers and $q \in \mathbb{N}$ be a power-of-prime. The protocol allows to prove the following relation:

$$\mathcal{R}^* := \{((\mathbf{A}, \mathbf{y}, \mathcal{M}), \mathbf{x}) \in \mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m \times ([1, n]^3)^l \times \mathbb{Z}_q^n \mid \mathbf{A} \cdot \mathbf{x} = \mathbf{y} \wedge \forall (h, i, j) \in \mathcal{M} : \mathbf{x}[h] = \mathbf{x}[i] \cdot \mathbf{x}[j]\}. \quad (3)$$

This allows to prove linear equations, which occur naturally in lattice-based settings. The quadratic constraints allow, e.g., to prove that the vector $\mathbf{x}$ is binary.

Yang et al. [43] show that their proof system for the relation $\mathcal{R}^*$ is complete, a proof of knowledge under the SIS_{l_1,q,β_1} and SIS_{l_2,q,β_2} assumptions, and zero-knowledge under the $LWE_{l_2,q,\alpha}$ assumption, where q is as in $\mathcal{R}^*$. Here, $l = |\mathcal{M}|$ with $\mathcal{M}$ from $\mathcal{R}^*$, $l_1, l_2, p \in \mathsf{poly}(\lambda)$ are positive integers, where p is the size of the challenge space, cf. [43]. Moreover, σ_3 and σ_4 are parameters for two discrete Gaussian distributions, where σ_3 is an integer with $\sigma_3 \geq \sqrt{2l_2/\pi}$, and $\sigma_4 = 2p\sqrt{2l_1 + 2l_2 + n + l} \cdot \sigma_3 \log l$. Finally, $\beta_1 = 16p\sqrt{l_1 + l_2 + n}(\sigma_3 + p\sigma_4)$, and $\beta_2 = 16p\sqrt{l_1 + l_2 + l}(\sigma_3 + p\sigma_4)$, and $\alpha = \sqrt{2\pi}\sigma_3/q$ are parameters following from the reductions on SIS/LWE.

Zero-Knowledge for Issue. Recall that in Issue (Section 3), the following equations have to be proven:

1. $\mathsf{pk}_{\mathcal{U}} = \mathbf{F} \cdot \mathsf{sk}_{\mathcal{U}} \mod q_0$
2. $\mathbf{c}_{\text{new}} = \mathbf{D}_0^0 \cdot \mathbf{r}' + \mathbf{D}_0^1 \cdot \mathbf{0} + \mathbf{D}_1 \cdot \mathsf{sk}_{\mathcal{U}} + \mathbf{D}_2 \cdot \mathbf{0} + \mathbf{D}_3 \cdot \bar{\mathbf{s}}_{\mathcal{U}} + \mathbf{D}_4 \cdot \mathbf{0} + \mathbf{D}_5 \cdot \bar{\mathbf{u}}_{\mathcal{U}}$
3. $\|\mathbf{r}'\| \leq \sigma_{\mathsf{Com}}\sqrt{m_{\mathbf{r}}}$
4. $\mathsf{sk}_{\mathcal{U}}$, $\bar{\mathbf{s}}_{\mathcal{U}}$, $\bar{\mathbf{u}}_{\mathcal{U}}$ are binary and $\mathsf{bin}(\mathbf{s}_{\mathcal{U}}) = \bar{\mathbf{s}}_{\mathcal{U}}$, $\mathsf{bin}(\mathbf{u}_{\mathcal{U}}) = \bar{\mathbf{u}}_{\mathcal{U}}$

Note that all vectors in the witness $\mathbf{x}_{\mathsf{Issue}} := (\mathbf{r}' \| \mathsf{sk}_{\mathcal{U}} \| \mathsf{bin}(\mathbf{s}'_{\mathcal{U}}) \| \mathsf{bin}(\mathbf{u}'_{\mathcal{U}}))$, except $\mathbf{r}'$, are binary vectors. Like [43], we prove the norm inequality $\|\mathbf{r}'\| \leq \sigma_{\mathsf{Com}}\sqrt{m_{\mathbf{r}}}$ in the *maximum* norm instead of the euclidean norm. This simplifies reductions. Also like [43], we let $\beta := 12\sigma_{\mathsf{Com}}$ be the maximum norm bound for $\mathbf{r}'$. To prove this norm bound on $\mathbf{r}'$, we resort to a binary decomposition, i.e. we consider $\mathsf{bin}(\mathbf{r}')$. However, since we want positive vectors, we consider $\mathbf{r}_1 := \mathbf{r}' + \boldsymbol{\beta}_{\mathsf{Com}}$ with $\boldsymbol{\beta}_{\mathsf{Com}} := (\beta, \ldots, \beta)^\top \in \mathbb{Z}_q^{m_{\mathbf{r}}}$ instead. The norm bound for $\mathbf{r}_1$ is then 2β. More precisely, using bin only works if β is a power of 2. Hence, we use a different decomposition technique, described in [28, 30, 31].

We recall it for completeness. For any positive integer $B \in \mathbb{N}$ this technique can decompose an integer $b \in \{0, \ldots, B\}$ into a binary vector $\mathbf{c}$ of size $\delta_B := \lfloor \log B \rfloor + 1$. For that, let $B_i = \lfloor (B + 2^{i-1})/2^i \rfloor$ for $i \in \{1, \ldots, \delta_B\}$, and note that $\sum_{j=1}^{\delta_B} B_j = B$. Now, we decompose b as follows:

1. Set $c' := b, \mathbf{c} := \mathbf{0}$
2. For $j = 1$ to δ_B:

(a) If $c' \geq B_j$, then $\mathbf{c}[j] \coloneqq 1$ else $\mathbf{c}[j] \coloneqq 0$
(b) $c' \coloneqq c' - B_j \cdot \mathbf{c}[j]$

3. output $\mathbf{c}$.

Now, let m be some integer. By doing this on every component of a vector $\mathbf{x} \in \mathbb{Z}^m$ with $\|\mathbf{x}\|_\infty \leq B$ we get a vector decomposition function $\mathsf{vdec}_{m,B} : \mathbb{Z}^m \to \mathbb{Z}_2^{m\delta_B}$. This decomposition can be reversed by using the matrix $\mathbf{H}_{m,B} \coloneqq \mathbf{I}_m \otimes [B_1 \cdots B_{\delta_B}] \in \mathbb{Z}^{m \times m\delta_B}$. By construction, for all $\mathbf{x} \in \mathbb{Z}^m$ with $\|\mathbf{x}\|_\infty \leq B$ we have $\mathbf{H}_{m,B} \cdot \mathsf{vdec}_{m,B}(\mathbf{x}) = \mathbf{x}$.

We let $\bar{\mathbf{r}} \coloneqq \mathsf{vdec}_{m_\mathbf{r},2\beta}(\mathbf{r}_1) \in \mathbb{Z}_2^{m_\mathbf{r}\lceil \log 2\beta \rceil}$. Consequently, we have $\mathbf{H}_{m_\mathbf{r},2\beta} \cdot \bar{\mathbf{r}} = \mathbf{r}_1$. For efficiency, we prove some equations modulo q_0, while our zero-knowledge proof works modulo q. We use the fact that for $a, b \in \mathbb{Z}_{q_0}$, we have $a = b \mod q_0 \Leftrightarrow \frac{q}{q_0} a = \frac{q}{q_0} b \mod q$, so we can simply multiply both sides of the equation by $\frac{q}{q_0}$.

All in all, we set $\mathbf{x}_{\mathsf{Issue}} \coloneqq (\bar{\mathbf{r}} \| \mathsf{sk}_\mathcal{U} \| \mathsf{bin}(\mathbf{s}_\mathcal{U}) \| \mathsf{bin}(\mathbf{u}_\mathcal{U}))$ and $\mathbf{y}_{\mathsf{Issue}} \coloneqq (\frac{q}{q_0}\mathsf{pk}_\mathcal{U} \| (\mathbf{c}_{\text{new}} + \mathbf{D}_0 \boldsymbol{\beta}_{\mathsf{Com}}))$. The vector $\mathbf{x}_{\mathsf{Issue}}$ has $m_\mathbf{r} \cdot \lceil \log 2\beta \rceil + m_{\mathsf{sk}} + m_{\mathsf{nr}} \lceil \log q \rceil + m_{\mathsf{sk}} \lceil \log q \rceil =: m_\mathcal{M}$ entries. We can ensure that each entry is either 0 or 1 by setting $\mathcal{M}_{\mathsf{Issue}} \coloneqq \{(i,i,i) \mid 0 \leq i < m_{\mathsf{Issue}}\}$. Since q is a power-of-prime we have that $\forall x \in \mathbb{Z}_q : x = x^2 \Leftrightarrow \{0,1\}$. Finally, we set

$$\mathbf{A}_{\mathsf{Issue}} \coloneqq \begin{pmatrix} \mathbf{0} & \frac{q}{q_0}\mathbf{F} & \mathbf{0} & \mathbf{0} \\ \mathbf{D}_0 \mathbf{H}_{m_\mathbf{r},2\beta} & \mathbf{D}_1 & \mathbf{D}_3 & \mathbf{D}_5 \end{pmatrix} \in \mathbb{Z}_q^{n_{\mathsf{sk}} + n_\mathsf{C} \times m_{\mathsf{Issue}}}.$$

Zero-Knowledge for Update. For Update (see Section 3), the user proves the following statements to the operator:

1. $\mathsf{Vfy}(\mathsf{pk}_\mathcal{O}, \mathsf{sig}, (\mathsf{sk}_\mathcal{U}, \mathbf{b}, \bar{\mathbf{s}}_\mathcal{U}, \bar{\mathbf{s}}_\mathcal{O}, \bar{\mathbf{u}}_\mathcal{U})) = 1$
2. $\mathbf{s} = \mathbf{s}_\mathcal{U} + \mathbf{s}_\mathcal{O} \mod q_0$
3. $\mathbf{t} = H_{FRD}(\mathbf{u}_\mathcal{O}) \cdot \mathsf{sk}_\mathcal{U} + \mathbf{u}_\mathcal{U} \mod q_0$
4. $\mathbf{c}_{\text{new}} = \mathbf{D}_0^0 \cdot \mathbf{r}'' + \mathbf{D}_0^1 \cdot \mathbf{0} + \mathbf{D}_1 \cdot \mathsf{sk}_\mathcal{U} + \mathbf{D}_2 \cdot \mathbf{b}_{\text{new}} + \mathbf{D}_3 \cdot \bar{\mathbf{s}}'_\mathcal{U} + \mathbf{D}_5 \cdot \bar{\mathbf{u}}'_\mathcal{U}$
5. $\mathbf{b}_{\text{new}} = \mathsf{bin}(\mathsf{toInt}(\mathbf{b}) + v \mod q))$
6. $\|\mathbf{r}'\| \leq \sigma_{\mathsf{Com}} \sqrt{m_\mathbf{r}}$
7. $\mathsf{sk}_\mathcal{U}$, $\bar{\mathbf{s}}_\mathcal{U}$, $\bar{\mathbf{s}}_\mathcal{O}$, $\bar{\mathbf{u}}_\mathcal{U}$, $\bar{\mathbf{u}}'_\mathcal{U}$, $\mathbf{b}_{\text{new}}$, $\bar{\mathbf{s}}'_\mathcal{U}$ are binary, $\mathsf{bin}(\mathbf{s}_\mathcal{U}) = \bar{\mathbf{s}}_\mathcal{U}$, $\mathsf{bin}(\mathbf{s}_\mathcal{O}) = \bar{\mathbf{s}}_\mathcal{O}$, $\mathsf{bin}(\mathbf{u}_\mathcal{U}) = \bar{\mathbf{u}}_\mathcal{U}$, $\mathsf{bin}(\mathbf{u}'_\mathcal{U}) = \bar{\mathbf{u}}'_\mathcal{U}$, and $\mathsf{bin}(\mathbf{s}'_\mathcal{U}) = \bar{\mathbf{s}}'_\mathcal{U}$

where $\mathbf{s}, \mathsf{pk}_\mathcal{O}, \mathbf{c}_{\text{new}}, \mathbf{D}_0, \ldots, \mathbf{D}_5, \mathbf{t}, \mathbf{u}_\mathcal{O}, v$ are known to both parties, either through the communication in Update or through the CRS and sig, $\mathbf{r}'$,$\mathbf{b}$, $\mathbf{b}_{\text{new}}$, $\mathbf{s}'_\mathcal{U}$, $\mathbf{s}_\mathcal{U}$, $\mathbf{s}_\mathcal{O}$, $\mathsf{sk}_\mathcal{U}$, $\mathbf{u}_\mathcal{U}$, $\mathbf{u}'_\mathcal{U}$ are the witnesses.

The first equation, i.e. signature verification, is by far the most complex. The other equations are quite similar to the ones from Section 3. The proof of possession of a valid message-signature pair for the signature scheme is easily expressed in terms of $\mathcal{R}^*$. Yang et al. [43] give one such translation, we reuse it here. In the next step, we extend it to a zero-knowledge argument for all the equations we have to show in Update.

In the signature scheme from [28], the public key is of the form $\mathsf{pk} = (\mathbf{A}, \{\mathbf{A}_j\}_{j=0}^{l_\tau}, \mathbf{B}, \mathbf{u}) \in \mathbb{Z}_q^{n_\mathsf{S} \times m_\mathsf{S}} \times (\mathbb{Z}_q^{n_\mathsf{S} \times m_\mathsf{S}})^{l_\tau} \times \mathbb{Z}_q^{n_\mathsf{S} \times (m_\mathsf{S}/2)} \times \mathbb{Z}_q^{n_\mathsf{S}}$ and the signature is of the form $\mathsf{sig} = (\tau_\mathsf{sig}, \mathbf{v}_\mathsf{sig}, \mathbf{s}_\mathsf{sig}) \in \mathbb{Z}_2^{l_\tau} \times \mathbb{Z}^{2m_\mathsf{S}} \times \mathbb{Z}^{2m_\mathsf{S}}$. Let $\mathsf{msg} \coloneqq (\mathbf{r}, \mathsf{sk}_\mathcal{U}, \mathbf{b}, \mathsf{bin}(\mathbf{s}_\mathcal{U}), \mathsf{bin}(\mathbf{s}_\mathcal{O}), \mathsf{bin}(\mathbf{u}_\mathcal{U}))$.

We know that $\mathsf{Vfy}(\mathsf{pk}_\mathcal{O}, \mathsf{sig}, \mathsf{msg}) = 1$ holds if $\mathbf{v}_\mathsf{sig}$ and $\mathbf{s}_\mathsf{sig}$ have bounded norm and the verification equation holds. As in Yang et al. [43], we use $\|\mathbf{v}_\mathsf{sig}\|_\infty < \beta$ and $\|\mathbf{s}_\mathsf{sig}\|_\infty < \beta$, where $\beta \coloneqq 12\sigma$.

The verification equation is:

$$\mathbf{u} + \mathbf{B} \cdot \mathsf{bin}(\mathbf{D}_0 \cdot \mathbf{s}_\mathsf{sig} + \sum_{k=1}^{5} \mathbf{D}_k \cdot \mathsf{msg}_k) = \mathbf{A}_{\tau_\mathsf{sig}} \cdot \mathbf{v}_\mathsf{sig} \mod q, \tag{4}$$

where $\mathbf{A}_{\tau_\mathsf{sig}} = [\mathbf{A} | \mathbf{A}_0 + \sum_{j=1}^{l_\tau} \tau_\mathsf{sig}[j]\mathbf{A}_j] \in \mathbb{Z}_q^{n_\mathsf{S} \times 2m_\mathsf{S}}$ (cf. Section 2.3). As we want to describe this verification equation via the relation $\mathcal{R}^*$, we want to circumvent the $\mathsf{bin}(\cdot)$ function. Therefore we split Equation (4) into two equations and we introduce a new vector $\mathbf{w} \in \mathbb{Z}_2^{n_\mathsf{S} \lceil \log q \rceil}$. We use the powers-of-two matrix to say that $\mathbf{w}$ is the binary composite of $\mathbf{D}_0 \cdot \mathbf{s}_\mathsf{sig} + \sum_{k=1}^{5} \mathbf{D}_k \cdot \mathsf{msg}_k$.

For some $m \in \mathbb{N}$, let $\mathbf{G}_m$ be the "powers-of-two" matrix with m rows: $\mathbf{G}_m \coloneqq \mathbf{I}_m \otimes \mathbf{g}^\top \in \mathbb{Z}_q^{m \times m\lceil \log q \rceil}$, where $\mathbf{g} = (2^{\lceil \log q \rceil - 1}, \ldots, 4, 2, 1)^\top \in \mathbb{Z}_q^{\lceil \log q \rceil}$. Then, we get

$$\mathbf{u} + \mathbf{B} \cdot \mathbf{w} = \mathbf{A}_{\tau_\mathsf{sig}} \cdot \mathbf{v}_\mathsf{sig} \mod q, \text{ and}$$

$$\mathbf{G}_{n_\mathsf{S}} \cdot \mathbf{w} = \mathbf{D}_0 \cdot \mathbf{s}_\mathsf{sig} + \sum_{k=1}^{5} \mathbf{D}_k \cdot \mathsf{msg}_k \mod q$$

Now, if we write $\mathbf{v}_\mathsf{sig} = (\mathbf{v}_1 \| \mathbf{v}_2) \in \mathbb{Z}_q^{m_\mathsf{S}} \times \mathbb{Z}_q^{m_\mathsf{S}}$, we get

$$\mathbf{u} + \mathbf{B} \cdot \mathbf{w} = \mathbf{A} \cdot \mathbf{v}_1 + (\mathbf{A}_0 + \sum_{j=1}^{l} \tau_\mathsf{sig}[j]\mathbf{A}_j) \cdot \mathbf{v}_2 \mod q, \tag{5}$$

$$\mathbf{G}_{n_\mathsf{S}} \cdot \mathbf{w} = \mathbf{D}_0 \cdot \mathbf{s}_\mathsf{sig} + \sum_{k=1}^{5} \mathbf{D}_k \cdot \mathsf{msg}_k \mod q, \tag{6}$$

and $\|\mathbf{v}_1\|_\infty < \beta, \|\mathbf{v}_2\|_\infty < \beta, \|\mathbf{s}_\mathsf{sig}\|_\infty < \beta$. The vectors $\mathbf{v}_1, \mathbf{v}_2, \mathbf{s}_\mathsf{sig}$ have bounded maximum norm. Like we already saw for Issue, we want them to be non-negative. Thus, we define $\boldsymbol{\beta}_1 \coloneqq (\beta, \ldots, \beta)^\top \in \mathbb{Z}_q^{m_\mathsf{S}}, \boldsymbol{\beta}_2 \coloneqq (\beta, \ldots, \beta)^\top \in \mathbb{Z}_q^{2m_\mathsf{S}}$ and set $\mathbf{v}_1' \coloneqq \mathbf{v}_1 + \boldsymbol{\beta}_1, \mathbf{v}_2' \coloneqq \mathbf{v}_2 + \boldsymbol{\beta}_1$ and $\mathbf{s}_\mathsf{sig}' \coloneqq \mathbf{s}_\mathsf{sig} + \boldsymbol{\beta}_2$. The next step is to decompose these vectors to obtain binary vectors. We let $\bar{\mathbf{v}}_1 \coloneqq \mathsf{vdec}_{m_\mathsf{S}, 2\beta}(\mathbf{v}_1') \in \mathbb{Z}_2^{m_\mathsf{S} \lceil \log 2\beta \rceil}, \bar{\mathbf{v}}_2 \coloneqq \mathsf{vdec}_{m_\mathsf{S}, 2\beta}(\mathbf{v}_2') \in \mathbb{Z}_2^{m_\mathsf{S} \lceil \log 2\beta \rceil}$ and $\bar{\mathbf{s}}_\mathsf{sig} \coloneqq \mathsf{vdec}_{2m_\mathsf{S}, 2\beta}(\mathbf{s}_\mathsf{sig}') \in \mathbb{Z}_2^{2m_\mathsf{S} \lceil \log 2\beta \rceil}$. Consequently, we have $\mathbf{H}_{m_\mathsf{S}, 2\beta} \cdot \bar{\mathbf{v}}_1 = \mathbf{v}_1', \mathbf{H}_{m_\mathsf{S}, 2\beta} \cdot \bar{\mathbf{v}}_2 = \mathbf{v}_2'$ and $\mathbf{H}_{2m_\mathsf{S}, 2\beta} \cdot \bar{\mathbf{s}}_\mathsf{sig} = \mathbf{s}_\mathsf{sig}'$.

Next, we deal with the subset-sum-like behavior of the tag $\tau_\mathsf{sig} \in \mathbb{Z}_2^{l_\tau}$. We define vectors $\mathbf{u}_i \coloneqq \mathbf{A}_i \mathbf{v}_2$ and $\mathbf{u}_i' \coloneqq \tau_\mathsf{sig}[i] \cdot \mathbf{u}_i$ for $i \in \{1, \ldots, l_\tau\}$. Moreover, we let $\hat{\mathbf{u}} \coloneqq (\mathbf{u}_1 \| \ldots \| \mathbf{u}_{l_\tau})$ and $\hat{\mathbf{u}}' \coloneqq (\mathbf{u}_1' \| \ldots \| \mathbf{u}_{l_\tau}')$. Finally, we set for convenience $\mathbf{J} \coloneqq (\mathbf{I}_{n_\mathsf{S}}, \ldots, \mathbf{I}_{n_\mathsf{S}}) \in \mathbb{Z}_q^{n_\mathsf{S} \times n_\mathsf{S} l_\tau}$ and $\bar{\mathbf{A}} \coloneqq (\mathbf{A}_1, \ldots, \mathbf{A}_{l_\tau})$.

Now we can express Equation (4) as linear equations

$$-\mathbf{I}_{n_{\mathsf{S}} l_\tau} \cdot \hat{\mathbf{u}} + \bar{\mathbf{A}} \cdot \mathbf{H}_{m_{\mathsf{S}},2\beta} \cdot \bar{\mathbf{v}}_2 = \bar{\mathbf{A}} \cdot \boldsymbol{\beta}_1, \tag{7}$$

$$\mathbf{J} \cdot \hat{\mathbf{u}}' + \mathbf{A}_0 \cdot \mathbf{H}_{m_{\mathsf{S}},2\beta} \cdot \bar{\mathbf{v}}_2 + \mathbf{A} \cdot \mathbf{H}_{m_{\mathsf{S}},2\beta} \bar{\mathbf{v}}_1 - \mathbf{B} \cdot \mathbf{w} = \mathbf{u} + \mathbf{A}_0 \cdot \boldsymbol{\beta}_1 + \mathbf{A} \cdot \boldsymbol{\beta}_1, \tag{8}$$

$$-\mathbf{G}_{n_{\mathsf{S}}} \cdot \mathbf{w} + \mathbf{D}_0 \cdot \mathbf{H}_{2m_{\mathsf{S}},2\beta} \cdot \bar{\mathbf{s}}_{\mathsf{sig}} + \sum_{k=1}^{5} \mathbf{D}_k \cdot \mathsf{msg}_k = \mathbf{D}_0 \cdot \boldsymbol{\beta}_2, \tag{9}$$

under the additional constraint that $\mathbf{u}'_i = \tau_{\mathsf{sig}}[i] \cdot \mathbf{u}_i$ for $i \in \{1, \ldots, l_\tau\}$. The last constraint is ensured by a suitable index set $\mathcal{M}$. Equation (7) ensures that the vectors $\mathbf{u}_i$ are calculated correctly, Equation (8) ensures that Equation (5) holds, and Equation (9) ensures that Equation (6) holds. All in all, we can prove the possession of a valid message-signature pair. For the balance, we rewrite $\mathbf{b}_{\text{new}} = \mathsf{bin}(\mathsf{tolnt}(\mathbf{b}) + v \mod q))$ to $\mathsf{tolnt}(\mathbf{b}_{\text{new}}) - \mathsf{tolnt}(\mathbf{b}) = v \mod q$.

Now it remains to combine the equations we derived above with the other necessary equations for our zero-knowledge proof. We again multiply both sides with $\frac{q}{q_0}$ if we want to show equality modulo q_0. We set

$$\begin{aligned}\mathbf{x}_{\mathsf{Update}} \coloneqq (\tau \| \hat{\mathbf{u}}' \| \hat{\mathbf{u}} \| \bar{\mathbf{v}}_2 \| \bar{\mathbf{v}}_1 \| \mathbf{w} \| \bar{\mathbf{s}}_{\mathsf{sig}} \| \mathsf{sk}_{\mathcal{U}} \| \mathbf{b} \| \mathsf{bin}(\mathbf{s}_{\mathcal{U}}) \\ \| \mathsf{bin}(\mathbf{s}_{\mathcal{O}}) \| \mathsf{bin}(\mathbf{u}_{\mathcal{U}}) \| \mathbf{r}' \| \mathbf{b}_{\text{new}} \| \mathsf{bin}(\mathbf{s}'_{\mathcal{U}}) \| \mathsf{bin}(\mathbf{u}'_{\mathcal{U}}))\end{aligned}$$

and

$$\mathbf{y}_{\mathsf{Update}} \coloneqq (\bar{\mathbf{A}} \cdot \boldsymbol{\beta}_1 \| \mathbf{u} + \mathbf{A}_0 \cdot \boldsymbol{\beta}_1 + \mathbf{A} \cdot \boldsymbol{\beta}_1 \| \mathbf{D}_0 \cdot \boldsymbol{\beta}_2 \| \mathbf{c}_{\text{new}} \| \frac{q}{q_0}\mathbf{s} \| v \| \frac{q}{q_0}\mathbf{t}).$$

Finally, we set $\mathbf{A}_{\mathsf{Update}}$ as in Fig. 6.

$$\mathbf{A}_{\mathsf{Update}} = \begin{pmatrix} \mathbf{0}^n & -\mathbf{I}_{n_{\mathsf{S}} l_\tau} & \bar{\mathbf{A}} \cdot \mathbf{H}_{m_{\mathsf{S}},2\beta} & & & & & & & & & & & & \\ \vdots & \mathbf{J} & \mathbf{A}_0 \cdot \mathbf{H}_{m_{\mathsf{S}},2\beta} & \mathbf{A} \cdot \mathbf{H}_{m_{\mathsf{S}},2\beta} & -\mathbf{B} & & & & & & & & & & \\ & & & & -\mathbf{G}_{n_{\mathsf{S}}} & \mathbf{D}_0 \cdot \mathbf{H}_{2m_{\mathsf{S}},2\beta} & \mathbf{D}_1 & \mathbf{D}_2 & \mathbf{D}_3 & \mathbf{D}_4 & \mathbf{D}_5 & & & & \\ & & & & & & \mathbf{D}_1 & & & & & \mathbf{D}_0 & \mathbf{D}_2 & \mathbf{D}_3 & \mathbf{D}_5 \\ & & & & & & & & \frac{q}{q_0}\mathbf{G}_1 & \frac{q}{q_0}\mathbf{G}_1 & & & & & \\ & & & & & & & -\mathbf{G}_1 & & & & & \mathbf{G}_1 & & \\ & & & & & & \frac{q}{q_0} H_{FRD}(\mathbf{u}_{\mathcal{O}}) & & & & \frac{q}{q_0}\mathbf{G}_m & & & & \end{pmatrix}.$$

Fig. 6: The matrix $\mathbf{A}_{\mathsf{Update}}$.

E Security and Privacy Notions

In this section we render the precise definitions of the security and privacy notions as defined by [22].

Definition E.1 (Correctness of BABL). *Similar to [21] the* BABL *scheme is called correct if the following holds: If the system is set up by the* Setup *algorithm, the keys are generated by* UGen *and* OGen*, all parties follow the protocol honestly,*

then, the following properties hold: (1) Correctness of the Issue *protocol: Both parties return as acceptance bit* 1*. (2) Correctness of the* Update *protocol: For all valid tokens and balances, after adding a value the user always returns as acceptance bit* 1*.*

Definition E.2 (Oracles, from [22, Def. 3.2]). MalIssue *lets the adversary initiate the* Issue *protocol with an honest issuer* $\mathcal{O}$ *provided that there is no pending* MalIssue *call for* $\mathsf{pk}_{\mathcal{U}}$ *and* $\mathsf{pk}_{\mathcal{U}}$ *has also not been used in a successful call to* MalIssue *before.* MalUpdate *lets the adversary initiate the* Update *protocol with an honest operator* $\mathcal{O}$ *for an input value* v*. We say that a call to an oracle is successful if the honest party represented by the oracle accepts the run.*

E.1 System Security

We denote by $\mathcal{T}^{\mathsf{Update}}_{\lambda,\mathsf{CRS}}$ the set of all transcripts of Update transactions, meaning all exchanged messages from the beginning, until both parties terminate.

Definition E.3. *A scheme is called* simulation-linkable *if it satisfies the following conditions:*

Completeness: *Let* $n \in \mathbb{N}, \mathsf{CRS} \leftarrow \mathsf{Setup}(1^\lambda)$ *and* $tr \in \mathcal{T}^{\mathsf{Update}}_{\lambda,\mathsf{CRS}}$ *be a transcript. Then there exist inputs* $\mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, \mathcal{T}, \mathbf{b}$ *and random choices for an honest user* $\mathcal{U}$ *and honest operator* $\mathcal{O}$ *such that a run of the* Update *protocol between* $\mathcal{U}$ *and* $\mathcal{O}$ *with those inputs leads to the same transcript* tr*.*

Extractability: *There exists a PPT algorithm* ExtractUID *that, given two related transcripts* $tr_1, tr_2 \in \mathcal{T}^{\mathsf{Update}}_{\lambda,\mathsf{CRS}}$ *produced by the interaction of a honest user* $\mathcal{U}$ *with public key* $\mathsf{pk}_{\mathcal{U}}$ *with a honest operator* $\mathcal{O}$ *outputs the public key* $\mathsf{pk}_{\mathcal{U}}$*. Two transcripts* tr_1, tr_2 *are called related if they are identical except for the zero-knowledge challenges, output by the Random Oracle.*

Additionally, there exists an expected PPT algorithm **GenerateTranscripts** *that, given access to a transcript oracle* $\mathcal{O} = \langle \mathcal{U}, \mathcal{O} \rangle$ *which outputs transcripts between a user and an operator, outputs two related transcripts* $tr_1, tr_2 \in \mathcal{T}^{\mathsf{Update}}_{\lambda,\mathsf{CRS}}$ *with overwhelming probability.* **GenerateTranscripts** *is allowed to rewind* $\mathcal{O}$ *and reprogram the Random Oracle.*

Definition E.4. *A simulation-linkable scheme is called* owner-binding *if for any PPT adversary* $\mathcal{A}$ *in the experiments* $\mathsf{Exp}^{\mathsf{ob\text{-}issue}}_{\mathsf{BABL},\mathcal{A}}(\lambda)$ *and* $\mathsf{Exp}^{\mathsf{ob\text{-}update}}_{\mathsf{BABL},\mathcal{A}}(\lambda)$ *from Fig. 7 the advantages of* $\mathcal{A}$ *defined by*

$$\mathsf{Adv}^{\mathsf{ob\text{-}issue}}_{\mathsf{BABL},\mathcal{A}}(\lambda) = \mathsf{Pr}\Big[\mathsf{Exp}^{\mathsf{ob\text{-}issue}}_{\mathsf{BABL},\mathcal{A}}(\lambda) = 1\Big]$$

$$\mathsf{Adv}^{\mathsf{ob\text{-}update}}_{\mathsf{BABL},\mathcal{A}}(\lambda) = \mathsf{Pr}\Big[\mathsf{Exp}^{\mathsf{ob\text{-}update}}_{\mathsf{BABL},\mathcal{A}}(\lambda) = 1\Big]$$

are negligible in λ*.*

Definition E.5. *A simulation-linkable scheme ensures* doubles-spending detection *if for any PPT adversary* $\mathcal{A}$ *in the experiments* $\mathsf{Exp}^{\mathsf{dsd}}_{\mathsf{BABL},\mathcal{A}}(\lambda)$ *from Fig. 8 the advantage of* $\mathcal{A}$ *defined by*

$$\mathsf{Adv}^{\mathsf{dsd}}_{\mathsf{BABL},\mathcal{A}}(\lambda) = \mathsf{Pr}\Big[\mathsf{Exp}^{\mathsf{dsd}}_{\mathsf{BABL},\mathcal{A}}(\lambda) = 1\Big]$$

Experiment $\mathsf{Exp}^{\mathsf{ob\text{-}issue}}_{\mathsf{BABL},\mathcal{A}}(\lambda)$
$\mathsf{CRS} \leftarrow \mathsf{Setup}(1^\lambda)$
$(\mathsf{pk}_\mathcal{O}, \mathsf{sk}_\mathcal{O}) \leftarrow \mathsf{OGen}(\mathsf{CRS})$
$(\mathsf{pk}_\mathcal{U}, \mathsf{sk}_\mathcal{U}) \leftarrow \mathsf{UGen}(\mathsf{CRS})$
$b \leftarrow \mathcal{A}^{\mathsf{MalIssue},\mathsf{MalUpdate}}(\mathsf{CRS}, \mathsf{pk}_\mathcal{O}, \mathsf{pk}_\mathcal{U})$
The experiment outputs 1 iff $\mathcal{A}$ made a successful call to $\mathsf{MalIssue}(\mathsf{pk}_\mathcal{U})$.

Experiment $\mathsf{Exp}^{\mathsf{ob\text{-}update}}_{\mathsf{BABL},\mathcal{A}}(\lambda)$
$\mathsf{CRS} \leftarrow \mathsf{Setup}(1^\lambda)$
$(\mathsf{pk}_\mathcal{O}, \mathsf{sk}_\mathcal{O}) \leftarrow \mathsf{OGen}(\mathsf{CRS})$
$b \leftarrow \mathcal{A}^{\mathsf{MalIssue},\mathsf{MalUpdate}}(\mathsf{pk}_\mathcal{O})$
The experiment outputs 1 iff $\mathcal{A}$ made a successful call to MalUpdate such that ExtractUID applied to that call outputs a public key $\mathsf{pk}_\mathcal{U}$, for which MalIssue has never been called before.

Fig. 7: Owner-binding experiment

Experiment $\mathsf{Exp}^{\mathsf{dsd}}_{\mathsf{BABL},\mathcal{A}}(\lambda)$
$\mathsf{CRS} \leftarrow \mathsf{Setup}(1^\lambda)$
$(\mathsf{pk}_\mathcal{O}, \mathsf{sk}_\mathcal{O}) \leftarrow \mathsf{OGen}(\mathsf{CRS})$
$b \leftarrow \mathcal{A}^{\mathsf{MalIssue},\mathsf{MalUpdate}}(\mathsf{pk}_\mathcal{O})$
The experiment outputs 1 iff $\mathcal{A}$ did two successful MalUpdate calls resulting in two double-spending tags $dstag_1 = (\mathbf{s}, z_1)$ and $dstag_2 = (\mathbf{s}, z_2)$ with extracted public keys $\mathsf{pk}^{(1)}_\mathcal{U}$ and $\mathsf{pk}^{(2)}_\mathcal{U}$ such that at least one of the following conditions is satisfied:

- $\mathsf{pk}^{(1)}_\mathcal{U} \neq \mathsf{pk}^{(2)}_\mathcal{U}$ or
- $\mathsf{IdentDS}(\mathsf{pk}_\mathcal{O}, dstag_1, dstag_2) \neq (\mathsf{pk}^{(1)}_\mathcal{U}, \pi)$ or
- $\mathsf{IdentDS}(\mathsf{pk}_\mathcal{O}, dstag_1, dstag_2) = (\mathsf{pk}^{(1)}_\mathcal{U}, \pi)$ but $\mathsf{VerifyGuilt}(\mathsf{pk}_\mathcal{O}, \mathsf{pk}^{(1)}_\mathcal{U}, \pi) \neq 0$.

Fig. 8: Double-spending experiment

is negligible in λ.

Definition E.6. *A simulation-linkable scheme is called* balance-binding *if for any PPT adversary* $\mathcal{A}$ *in the experiments* $\mathsf{Exp}^{\mathsf{bb}}_{\mathsf{BABL},\mathcal{A}}(\lambda)$ *from Fig. 9 the advantage of* $\mathcal{A}$ *defined by*

$$\mathsf{Adv}^{\mathsf{bb}}_{\mathsf{BABL},\mathcal{A}}(\lambda) = \mathsf{Pr}\Big[\mathsf{Exp}^{\mathsf{bb}}_{\mathsf{BABL},\mathcal{A}}(\lambda) = 1\Big]$$

is negligible in λ.

E.2 User Security and Privacy

User security is defined using the real/ideal world paradigm. The adversary can query the HonUser oracle to spawn new users. In the real world, the adversary

Experiment $\mathsf{Exp}^{\mathsf{bb}}_{\mathsf{BABL},\mathcal{A}}(\lambda)$
$\mathsf{CRS} \leftarrow \mathsf{Setup}(1^\lambda)$
$(\mathsf{pk}_\mathcal{O}, \mathsf{sk}_\mathcal{O}) \leftarrow \mathsf{OGen}(\mathsf{CRS})$
$b \leftarrow \mathcal{A}^{\mathsf{MalIssue},\mathsf{MalUpdate}}(\mathsf{pk}_\mathcal{O})$
The experiment outputs 1 iff $\mathcal{A}$ made a successful call to MalUpdate with extracted user public-key $\mathsf{pk}_\mathcal{U}$, s.t.

- all successful MalIssue / MalUpdate calls produce unique token version numbers,
- the claimed balance $w \in \mathbb{V}$ does not equal the sum of previously collected accumulation values v for $\mathsf{pk}_\mathcal{U}$, i.e. $w \neq \sum_{v \in V_{\mathsf{pk}_\mathcal{U}}} v$ where $V_{\mathsf{pk}_\mathcal{U}}$ is the list of all accumulation values $v \in \mathbb{V}$ that appeared in previous successful calls to MalUpdate for which $\mathsf{pk}_\mathcal{U}$ could be extracted using ExtractUID.

Fig. 9: Balance-binding experiment

interacts with oracles RHonIssue and RHonUpdate implementing the real user protocols. In the ideal world, the adversary interacts with a simulator. The simulator has to play the role of the oracles, but without receiving any private user information. We denote this by SHonIssue, SHonUpdate. In both worlds, the adversary can query RCorrupt or SCorrupt, respectively, to corrupt a user. By this, they learn all private information of the respective user.

Definition E.7. *A scheme is called* privacy-preserving *if there exist PPT algorithms* SimSetup *and* SCorrupt *as well as PPT protocols* SHonIssue, SHonUpdate *that receive no private user information, such that for all PPT adversaries* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *in the experiment depicted in Fig. 10, the advantage* $\mathsf{Adv}^{\mathsf{priv}}_{\mathsf{BABL},\mathcal{A}}(\lambda)$ *of* $\mathcal{A}$ *defined by*

$$\left|\mathsf{Pr}\left[\mathsf{Exp}^{\mathsf{priv\text{-}real}}_{\mathsf{BABL},\mathcal{A}}(\lambda) = 1\right] - \mathsf{Pr}\left[\mathsf{Exp}^{\mathsf{priv\text{-}ideal}}_{\mathsf{BABL},\mathcal{A}}(\lambda) = 1\right]\right|$$

is negligible in λ.

Definition E.8. *A simulation-linkable scheme ensures* false-accusation protection *if for any PPT adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *in the experiments* $\mathsf{Exp}^{\mathsf{facp}}_{\mathsf{BABL},\mathcal{A}}(\lambda)$ *from Fig. 11 the advantage of* $\mathcal{A}$ *defined by*

$$\mathsf{Adv}^{\mathsf{facp}}_{\mathsf{BABL},\mathcal{A}}(\lambda) = \mathsf{Pr}\left[\mathsf{Exp}^{\mathsf{facp}}_{\mathsf{BABL},\mathcal{A}}(\lambda) = 1\right]$$

is negligible in λ. *(Note, that this does not guarantee anything, once the user was compromised.)*

E.3 Security and Privacy

Our construction fulfills the desired security and privacy properties mentioned in Section Section 2.1. We formulate the theorems and give proof sketches. Note, that the proofs follow closely the proofs from [22]. Only small changes were necessary to adopt the proofs to the lattice-setting:

Experiment $\mathsf{Exp}^{\mathsf{priv\text{-}real}}_{\mathsf{BABL},\mathcal{A}}(\lambda)$
$\mathsf{CRS} \leftarrow \mathsf{Setup}(1^\lambda)$
$(\mathsf{pk}_\mathcal{O}, state) \leftarrow \mathcal{A}_1(\mathsf{CRS})$
$b \leftarrow \mathcal{A}_2^{\mathsf{HonUser},\mathsf{RHonIssue},\mathsf{RHonUpdate},\mathsf{RCorrupt}}(\mathsf{pk}_\mathcal{O}, state)$
return b

Experiment $\mathsf{Exp}^{\mathsf{priv\text{-}ideal}}_{\mathsf{BABL},\mathcal{A}}(\lambda)$
$(\mathsf{CRS}, td_{sim}) \leftarrow \mathsf{SimSetup}(1^\lambda)$
$(\mathsf{pk}_\mathcal{O}, state) \leftarrow \mathcal{A}_1(\mathsf{CRS})$
$b \leftarrow \mathcal{A}_2^{\mathsf{HonUser},\mathsf{SHonIssue},\mathsf{SHonUpdate},\mathsf{SCorrupt}}(\mathsf{pk}_\mathcal{O}, state)$
return b

Fig. 10: Real/Ideal world privacy experiment

Experiment $\mathsf{Exp}^{\mathsf{facp}}_{\mathsf{BABL},\mathcal{A}}(\lambda)$
$\mathsf{CRS} \leftarrow \mathsf{Setup}(1^\lambda)$
$(\mathsf{pk}_\mathcal{O}, \mathsf{sk}_\mathcal{O}) \leftarrow \mathcal{A}_1(\mathsf{CRS})$
$(\mathsf{pk}_\mathcal{U}, \mathsf{sk}_\mathcal{U}) \leftarrow \mathsf{UGen}(\mathsf{CRS})$
$\pi \leftarrow \mathcal{A}_2^{\mathsf{RHonIssue},\mathsf{RHonUpdate}}(\mathsf{pk}_\mathcal{O}, \mathsf{pk}_\mathcal{U})$
Return 1 iff $\mathsf{VerifyGuilt}(\mathsf{pk}_\mathcal{O}, \mathsf{pk}_\mathcal{U}, \pi) = 1$.

Fig. 11: False-accusation experiment

Theorem E.1 (Simulation-Linkability). *Suppose* BABL *is correct,* S *is secure and* P1, P2 *are sound. Then* BABL *is simulation-linkable.*

Proof sketch (Simulation-Linkability (Theorem E.1)). As by definition, a scheme is called simulation-linkable when it is complete and extractable, we have to show that BABL fulfills these properties.

Completeness requires, that for every accepted transcript, there is a choice of parameters, such that the transcript is the result of an honest protocol run. This is given in our case, as the sum of the serial number and the **t**-part of the double-spending tag are indistinguishable from random values. Further, as the transcript is accepted, the soundness property of the zero-knowledge protocol from [43] guarantees that a commitment is well-formed. It remains to show that the signature is honestly generated, but as the token is accepted by the user and the signature is secure, this is given.

To prove the extractability property we can rely on the fact that the protocol from [43] is extractable, because it is a proof of knowledge.

Theorem E.2 (Owner-Binding w.r.t. Issue). *Suppose the* $SIS_{n_{\mathsf{sk}},q,\sqrt{m_{\mathsf{sk}}}}$ *assumption holds and* P1 *is extractable. Then* BABL *is owner-binding w.r.t.* Issue.

Proof sketch (Owner-Binding wrt. Issue *(Theorem E.2)).* Proving this property is a straightforward reduction on $SIS_{n_{\mathsf{sk}},q,\sqrt{m_{\mathsf{sk}}}}$.

Theorem E.3 (Owner-Binding w.r.t. Update). *Suppose* BABL *is simulation-linkable,* P2 *is extractable and* S *is secure. Then* BABL *is owner-binding w.r.t.* Update.

Proof sketch (Owner-Binding wrt. Update *(Theorem E.3)).* To prove the owner-binding property for the Update protocol we define a series of games for a hybrid argument. In these games, we test the required properties for the owner-binding property step by step. Finally, we show that the advantage of the adversary to win in the original game differs only negligibly from the other games.

First, we show that it is indeed possible to extract the user's secret key by reducing this problem on the already proven simulation-linkability property of our scheme. Then it is possible to extract witnesses for all occurred zero-knowledge proofs as the zero-knowledge argument from [43] is extractable. Finally, as there are extracted witnesses, the only way left for the adversary to win the owner-binding game is to forge a signature. This is prevented by the security of our signature. Note here, that the security of the signature is not exactly the usual EUF-CMA security, as the user sends commitments and the signer signs the committed messages (and not the commitment itself). However, the notion of security given in Definition C.1 suffices for our proofs.

Theorem E.4 (Double-Spending Detection). *Suppose* BABL *is simulation-linkable, the* $SIS_{n_{\mathsf{sk}},q,\sqrt{m_{\mathsf{sk}}}}$ *assumption holds,* P2 *is extractable and* S *is secure. Then* BABL *ensures double-spending detection.*

Proof sketch (Double-Spending Detection (Theorem E.4)). Similar to the last theorem, we prove this property with a hybrid argument. In particular, we consider all ways in which IdentDS (Fig. 4) could be tricked into not recognizing an actual act of double-spending. We show that an adversary would therefore either be able to find a collision on the serial number, or they were was able to manipulate the double-spending tag in a specific way. The former happens only with negligible probability, as the serial number is chosen in a coin-toss-like manner. The latter happens only with negligible probability, as the double-spending certainly includes values that the operators drew at random, and as the zero-knowledge proof is sound.

Theorem E.5 (Balance-Binding). *Suppose* P1 *and* P2 *are extractable and sound and* C *is statistically hiding, and* S *is secure Then* BABL *is balance-binding.*

Proof sketch (Balance-Binding (Theorem E.5)). Similar to the proofs for the previous properties, we prove the balance-binding property by defining several games, where we show step by step, that the probability for an adversary to break the balance-binding property of BABL is negligible. More precisely, we interpret every transaction as a node in a graph. Two nodes are connected if the output serial number of the first transaction is the input serial number of the last. We ensure through the game hops, that all nodes have an indegree of exactly one (except for the issuance of the token) and outdegrees of at most one. If this was not the case, there would be collisions on the serial number, double-spendings, or

forged signatures. Additionally, every such chain of transactions must be started by the issuance of a token and the balance must only change according to the transaction values of the nodes.

Theorem E.6 (Privacy-Preserving). *Suppose* $\mathsf{P1}$ *and* $\mathsf{P2}$ *are zero-knowledge and* C *is equivocal. Then* BABL *is privacy-preserving.*

Proof sketch (Privacy-Preserving (Theorem E.6)). We prove this property by defining several games, where the oracles of the real experiment are step-by-step replaced by oracles that hold no personal information of the user, called the ideal world. By showing that an adversary is only with negligible probability able to tell apart the real from the ideal world, we prove that BABL is indeed privacy-preserving. In more detail, we make use of the fact that the zero-knowledge proof from [43] is indeed zero-knowledge and the commitment scheme is equivocal. We use the equivocality property to replace the real values in the token with random ones, which makes it impossible to extract personal information from the user in the ideal world.

Theorem E.7 (False-Accusation Protection). *Suppose the* $SIS_{n_{\mathsf{sk}},q,\sqrt{m_{\mathsf{sk}}}}$ *assumption holds and the scheme ensures double-spending detection. Then* BABL *ensures false-accusation protection.*

Proof sketch (False-Accusation Protection (Theorem E.7)). Just like in the privacy-preserving proof we use the real/ideal world paradigm. Now, if the adversary is able to output a false proof of guilt for an honest user, one can directly construct an adversary breaking the $SIS_{n_{\mathsf{sk}},q,\sqrt{m_{\mathsf{sk}}}}$ assumption. If the adversary is not able to output a proof of guilt for a guilty user, this can be leveraged to distinguish the real world from the ideal world.