Black-Box Wallets: Fast Anonymous Two-Way Payments for Constrained Devices*

Max Hoffmann[†]
max.hoffmann@rub.de

Michael Klooß[‡]
michael.klooss@kit.edu

Markus Raiber[‡]
markus.raiber@kit.edu

Andy Rupp[‡]
andy.rupp@kit.edu

October 1, 2019

Abstract

Black-box accumulation (BBA) is a building block which enables a privacy-preserving implementation of point collection and redemption, a functionality required in a variety of user-centric applications including loyalty programs, incentive systems, and mobile payments. By definition, BBA+ schemes (Hartung et al. CCS '17) offer strong privacy and security guarantees, such as unlinkability of transactions and correctness of the balance flows of all (even malicious) users. Unfortunately, the instantiation of BBA+ presented at CCS '17 is, on modern smartphones, just fast enough for comfortable use. It is too slow for wearables, let alone smart-cards. Moreover, it lacks a crucial property: For the sake of efficiency, the user's balance is presented in the clear when points are deducted. This may allow to track owners by just observing revealed balances, even though privacy is otherwise guaranteed. The authors intentionally forgo the use of costly range proofs, which would remedy this problem.
We present an instantiation of BBA+ with some extensions following a different technical approach which significantly improves efficiency. To this end, we get rid of pairing groups, rely on different zero-knowledge and fast range proofs, along with a slightly modified version of Baldimtsi-Lysyanskaya blind signatures (CCS '13). Our prototype implementation *with* range proofs (for 16 bit balances) outperforms BBA+ *without* range proofs by a factor of 2.5. Moreover, we give estimates showing that smart-card implementations are within reach.

1 Introduction

A variety of applications such as mobile payments, toll collection, or loyalty and incentive programs follow the same basic principle: certain types of points are collected and redeemed. From a user's perspective this should be done in a privacy-preserving way. The operator requires protection against any kind of frauds. This basic functionality has been formally captured by [23, 20], where a general building block called BBA(+) for privacy-preserving point collection is presented. BBA+ is a framework where users create a token (a wallet) and then use it to collect or spend points. It offers strong privacy guarantees. For example, transactions are unlinkable, even those before and after a user was corrupted and all secrets were leaked.

The protocols for collecting and spending points are offline in the sense that no connection to a central database or entity is required. Thus, double-spending detection is needed to prevent reuse of old token states. This ensures that a user cannot skip the collection of negative points (i.e. spent points) by simply reusing an old token, as this will reveal his identity.

Unfortunately, the instantiation of BBA+ given in [20] comes with some drawbacks: one is its reliance on Groth–Sahai zero-knowledge proofs [19], which require a large amount of computations in pairing groups and all other building blocks to also work in this setting. This leads to slow performance on weak hardware and makes it unsuited for usage on smart cards, where even with a dedicated co-processor for pairing-based operations [35], evaluation of a pairing takes about 160 ms and multiplication in $\mathbb{G}_2$ about 100 ms (for the Fp254BNb and Fp254n2BNb curves). For point collection/redemption the user has to compute $\approx$ 100 multiplications in $\mathbb{G}_2$ alone, which already takes 10 s.

BBA+ also requires spending operations to reveal the current balance. This may allow the system operator to track a (anonymous) user by just observing balances in spending operations. In the end, this may even help to identify this user. So, while the privacy guarantees ensure that nothing but the balance is leaked, this may not be enough. This attack on user privacy is especially relevant in scenarios where a user spends points (much) more often than he collects them, such as mobile payments. To protect from this attack, range proofs, e.g. [8], could be used. That is, instead of revealing the balance,

*Please cite the conference version of this work to appear in PoPETS'20(1). This work is licensed under the CC BY-NC license.
[†]Ruhr-University Bochum
[‡]Karlsruhe Institute of Technology

a ZK proof of sufficient funds is performed. Since range proofs are expensive and would make BBA+ impractical, especially with the techniques used in [20], the authors refrained from using them.

1.1 Our Contribution

Using the Commit-then-Blind-Sign Principle. In BBA+, tokens (wallets) consist essentially of (a commitment to) balance, serial number, and a signature (on the commitment). In an interaction, the serial number is revealed, the signature verified, the balance updated, and a new signature (on the updated commitment) is issued. Reuse of a serial number indicates double spending. To achieve authenticity of wallets and prevent tampering, the signature is used. To attain unlinkability, BBA+ cannot send the signature in the plain, as this would link the interaction with the one which issued the signature. The solution in [20] is to *prove possession* of a valid signature (on a randomized commitment) in zero-knowledge. One may call this the "commit-sign-prove" principle. Our approach is different: We use a *blind signature*, so that a newly issued signature is unknown to the issuer, and can later be shown in the plain. This "commit-then-blind-sign" principle also simplifies our construction and security proofs. To this end, a suitable signature scheme without pairings is needed. To the best of our knowledge the only existing scheme [3] satisfying our requirements is only known to be secure under *sequential* executions. We need security under *concurrent* executions, and thus present a slight modification of [3] and prove EUF-CMA security in the generic group model. Under *sequential* security, only one protocol may run at any time. This is a nonsensical assumption for a distributed system such as BBW, so we allow malicious users to attack by running several executions *concurrently*, arbitrarily interleaved. Security in this setting is harder to achieve.

Pairing-free. Our protocol does not rely on pairing groups. While pairing groups are used to construct many primitives, most notably the non-interactive zero-knowledge proofs [19] used in [20], they require extra structure. This extra structure has led to recent improvements in finding discrete logarithms in those groups (c.f. [25, 5]), necessitating the use of larger parameters to maintain ≈128 bit security, resulting in slower algorithms. Additionally, the fastest currently known elliptic curves (for example Four$\mathbb{Q}$ [13]) use techniques for fast scalar-point multiplication that are not applicable to pairing groups. Moreover, evaluating pairings is expensive. Thus, when targeting constrained devices, it is useful to avoid the need for pairing groups.

Range Proofs. When a user spends points, BBA+ discloses the current balance in the clear, which opens possible attacks on privacy. We implement efficient range proofs (Bulletproofs [8]), which show in zero-knowledge that a user has sufficient funds.

Removing the Trapdoor. Defining the legitimate balance of a wallet/token requires a way to link transactions to that wallet, but transactions need to be unlinkable. To solve this dilemma, Hartung et al. [20] introduce a trapdoor only known to a trusted third party which allows to track users. This trapdoor is only intended for definitional purposes but knowledge of it eliminates any privacy guarantees. We define a new property called simulation-linkability, and show how to circumvent the need for such a trapdoor by relying on interactive proof systems, where standard rewinding techniques replace the trapdoor. Thus, linkability is possible only within the security proof, but not in reality where rewinding users is not possible.

Attributes. Our construction allows embedding of attributes into tokens. This can be used for different purposes. For example, it is possible to embed an expiration date into all tokens. If tokens are valid for, say 6 months, then double-spending information for older tokens can safely be deleted, keeping the database small. This may also be used to limit the amount of damage that can be inflicted through double-spending a stolen token. Other use cases include embedding of discounts or age verification, for example children or senior pricing, in pre-payment systems.

Implementation. To evaluate the suitability of our construction for use on constrained devices, we implemented our BBW instantiation and measured execution times on a smartphone. For 16 bit balances, we estimate a run-time of 122 ms on the user side (without communication). For the system side, we measure a run-time of 182 ms, which, considering that the system side may use more powerful hardware, is good enough. The total communication is about 4 kB. We also provide estimations for smart cards based on the number of multiplications required, where only 68 are needed for 16 bit ranges. This results in approximately 4 s for 256 bit curves on the MultOS 4.3.1 card and 1.5 s for a card with co-processor for ECC operations.

Compared to BBA+, point collection is 5.6 times faster while transmitting half as much data. Spending points is 6 times faster when performing range proofs for 16 bit values, and needs roughly a quarter of the communication of BBA+ (with 16 bit range proofs). This makes our scheme well suited for privacy-preserving prepayment systems on constrained devices such as smartphones or wearables and provides a step towards privacy-preserving wallets on smart cards.

Finally, we note that 16 bit are already enough for many applications. For example, the EU limited anonymous payments to 150 EUR [24].

1.2 Further Related Work

Intuitively, the problem of anonymous point collection seems to be solvable using (transferable) e-cash. Hartung et al. [20] already argued why this does not work though: Roughly speaking, in traditional offline e-cash, e.g. [9], withdrawing a coin is identifying and coins cannot be transferred, thus the accumulator can neither execute the withdraw protocol with the user nor transfer coins it withdrew itself from the bank. Using transferable e-cash such as [4] where anonymous and unlinkable transfers of coins is possible (under certain assumptions), the accumulator could withdraw coins and transfer them to the user. However, an impossibility result by Canard and Gouget [12] implies, that if the issuer, accumulator, and verifier collude, transactions can be linked. Additionally, users would be free to transfer coins arbitrarily among each other, allowing for pooling of points which is not desired in certain applications.

In [31] a privacy-preserving prepaid system for public transportation is constructed. However, payment is done by first paying the maximum possible transportation fee and then getting a refund based on the actual ride taken, where the possible refund values are encoded in the system parameters. While this might be practical in the case of public transportation with only a small amount of different prices, it cannot be extended to a general two-way payment system.

Besides BBA+, which we already discussed, [29] implements an anonymous and unlinkable incentive scheme called uCentive. However, the scheme targets a simpler scenario than we do: points are not accumulatable and double-spending detection is done online. Moreover, security and privacy protocols are stated only informally and no proofs are given.

In [3] Anonymous Credentials Light (ACL) are presented, a building block for anonymous credentials equipped with attributes. ACL does not provide a two-way payment system, but may be used as building block for one. However, security is only guaranteed for sequential interactions, a limitation not practical for real-world scenarios. Their scheme has been implemented on a smart card [21] and shown to be fairly practical.

Dimitriou [16] constructed an efficient privacy-preserving point accumulation scheme based on ACL. However, security is based on informal claims which do not reflect the capabilities of real-world adversaries. Problems stemming from concurrent executions are ignored, as made evident by the use of the signature scheme of [3] which is only sequentially secure. Additionally, spending points reveals the user's identity.

In [22] a privacy-preserving system for toll collection is built based on BBA+. In their system, users accumulate debt while driving and pay at regular intervals. The authors extend BBA+ to address the challenges posed by real-world interactions in this scenario, such as broken or lost user hardware. Interestingly, they avoid the need for range proofs by following a post-payment instead of a pre-payment approach. Their scheme is still based on Groth–Sahai proofs and thus pairing groups.

Lastly, privacy-preserving cryptocurrencies such as Dash [15] and ZCash [32, 36] provide another way of anonymous two-way payments. However, these require a reliable online connection, while our scheme explicitly allows offline transactions. Furthermore, Dash comes without formal security and privacy definitions, while in ZCash transactions are only confirmed after several minutes, which makes them unsuitable for time-sensitive scenarios, such as paying in a store using one's smartphone. Additionally, using cryptocurrencies it is again possible for users to transfer and pool funds among themselves, which is not desired in certain applications.

1.3 Paper Organization

The rest of this paper is organized as follows. In Section 2 we introduce some cryptographic assumptions and informally describe cryptographic building blocks used. Section 3 describes the BBA+ scheme as introduced by Hartung et al. [20], and Section 4 describes our modifications to address weaknesses of BBA+. In Section 5 we detail how we instantiated our construction, while Section 6 explains the signature scheme used. Finally, in Section 7 we provide and discuss performance metrics obtained by implementing our construction on a smartphone.

2 Preliminaries

We use the common notation to describe cryptographic protocols and define their security. By n, we denote the security parameter. For two functions $f(n)$ and $g(n)$ we write $f \approx g$ if $|f - g| \leq \mathsf{negl}(n)$ for a negligible function $\mathsf{negl}(n)$. We use additive notation for (commutative) groups of prime order p. Group elements are always denoted by upper case letters, scalars (i.e. elements of $\mathbb{Z}_p$) by lower case letters.

2.1 Assumptions

Definition 2.1 (Group Generator) *A group generator is a PPT algorithm $gp := (\mathbb{G}, G, p) \leftarrow \mathsf{SetupGrp}(1^n)$ that on input of a security parameter 1^n outputs the description of a cyclic group $\mathbb{G}$ of prime order p with $|p| = n$ and generator G.*

Definition 2.2 (Discrete Logarithm Assumption) *We say the DLOG assumption holds with regard to $\mathsf{SetupGrp}$ if the advantage $\mathsf{Adv}^{\mathrm{dlog}}_{\mathsf{SetupGrp},\mathcal{A}}(n)$ defined as*

$$\Pr\left[\begin{array}{l}(\mathbb{G}, G, p) \leftarrow \mathsf{SetupGrp}(1^n); x \leftarrow \mathbb{Z}_p; X = xG;\\ x' \leftarrow \mathcal{A}(\mathbb{G}, G, X) \colon x'G = X\end{array}\right]$$

is negligible in n for all PPT algorithms $\mathcal{A}$.

Definition 2.3 (Decisional Diffie-Hellman Assumption) *We say the DDH assumption holds with regard to $\mathsf{SetupGrp}$ if the advantage $\mathsf{Adv}^{\mathrm{ddh}}_{\mathsf{SetupGrp},\mathcal{A}}(n)$ defined as*

$$\Pr\left[\begin{array}{l}(\mathbb{G}, G, p) \leftarrow \mathsf{SetupGrp}(1^n); x, y, z \leftarrow \mathbb{Z}_p;\\ X = xG, Y = yG, Z_0 = xyG, Z_1 = zG;\\ b \leftarrow \{0,1\}; b' \leftarrow \mathcal{A}(\mathbb{G}, G, X, Y, Z_b) \colon b = b'\end{array}\right] - \frac{1}{2}$$

is negligible in n for all PPT algorithms $\mathcal{A}$.

2.2 Building Blocks

For formal definitions of the building blocks used, see Appendix A.

Homomorphic Commitments A commitment scheme allows the committing party to commit (using Com) to some message m from some message space $\mathcal{M}$ while keeping it secret. At a later point, m can then be revealed together with a "proof" that it has indeed been fixed during the committing step (verifiable using Open). Homomorphic commitments allow adding of commitments in a way that allows opening of the result to the sum of the original messages.

Signatures A signature scheme allows a signer (with a public and a secret key) to sign (using Sign) a message m with his secret key so that anyone with knowledge of his public key can verify (using Verify) that m was indeed signed by the signer and has not been modified since. For privacy-preserving protocols, signatures with additional protocols $\mathsf{BlindSign}$ and $\mathsf{BlindVerify}$ for blind signature issuance and verification can be used. Recall that EUF-CMA security is the standard security notion for signatures. It asserts an adversary, which is given a signing oracle which signs any messages of its choice, cannot produce a signature on a "fresh" message (of its choice).

Interactive Zero-Knowledge Proofs A zero-knowledge (ZK) proof allows a prover to convince someone (called the verifier) of the truth of a statement (usually membership in some NP language) without revealing any information beside the validity of the statement. A ZK proof of knowledge (PoK) convinces the verifier that not only the statement is true, but also the prover "knows" a witness for it (again without revealing any information about the witness). We use the notion of witness-extended emulation introduced by Groth and Ishai [18] to formalize knowledge of the prover.

3 BBA+ Description

Hartung et al. [20] introduced a generic building block for privacy-preserving point collection called $\mathsf{BBA+}$. $\mathsf{BBA+}$ allows anonymous and unlinkable collection and redemption of points, while ensuring users can't over- or underclaim their balance, and preventing users from trading points among themselves. It also supports offline double-spending detection, avoiding the need of a permanent connection to some centralized database.

On a high level, the $\mathsf{BBA+}$ framework involves three types of parties:

A Trusted Third Party (TTP) generates the common reference string, together with a trapdoor that allows linking of transactions, once to setup the system. It does not need to interact with anyone afterwards.

Operators of the system can be divided into three roles: the Issuer ($\mathcal{I}$) responsible for issuing new tokens to users, the Accumulator ($\mathcal{AC}$) from which points are collected and the Verifier ($\mathcal{V}$) to whom points can be spent. All operators have to trust each other and in particular share the same secret key. Additionally, the operators operate a shared central database in which they store double-spending information. Connection to this database does not have to be permanently available, as double-spending can be detected after-the-fact when inserting values into this database: spending the same token twice results in two transactions with the same serial number, and if two such transactions are detected, it is possible to retrieve the identity of the misbehaving user (together with a proof thereof).

Users have a token (piggy-bank) with which they want to collect points. They are in possession of a public and secret key. The public key is used to identify the user in the system, and is assumed to be bound to some kind of physical ID (e.g. passport, social security number, etc.). It is assumed that ensuring uniqueness of keys and verifying the physical ID is done out-of-band before interacting with the $\mathsf{BBA+}$ protocols. To become a participant of the scheme, a user performs the token issuing protocol with the issuer, during which knowledge of the secret key corresponding to the claimed identity is proven. Then, a user can use this token to accumulate points by running the Add protocol with the accumulator and spend these points by running the Sub protocol with the verifier. (For more intuitive notation, we use Add and Sub instead of Acc and Ver as in [20].)

Definition 3.1 ($\mathsf{BBA+}$ Scheme [20]) *A $\mathsf{BBA+}$ scheme consists of PPT algorithms Setup, IGen, UGen, deterministic polynomial time algorithms UVer, $\mathsf{IdentDS}$, $\mathsf{VerifyGuilt}$ and interactive protocols Issue, Add and Sub. At the core of the scheme lies a token τ, containing a unique serial number s and the balance $w \in \mathbb{V}$. It is bound to the owner's identity (i.e. the user's public key).*

Setup: *The Setup algorithm takes as input the security parameter n and outputs a common reference string crs and a trapdoor td. $\mathsf{IGen}(crs)$ generates a key pair $(\mathsf{PK}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}})$ for the issuer, which is also shared with the accumulator and verifier. $\mathsf{UGen}(crs)$ generates a user key pair $(\mathsf{PK}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}})$. UVer, given a user key pair $(\mathsf{PK}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}})$, a token τ and a balance w outputs a bit b indicating whether τ is a valid token with balance w owned by the user with public key $\mathsf{PK}_{\mathcal{U}}$.*

Token generation: *The protocol Issue is executed between a user $\mathcal{U}$, given his own key pair $(\mathsf{PK}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}})$ and $\mathsf{PK}_{\mathcal{I}}$; and the issuer $\mathcal{I}$, given $\mathsf{PK}_{\mathcal{U}}$ and the issuer key pair $(\mathsf{PK}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}})$. Upon successful execution, $\mathcal{U}$ outputs a token τ with initial balance 0 for the key pair $(\mathsf{PK}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}})$.*

Both $\mathcal{U}$ and $\mathcal{I}$ additionally output a bit $b_{\mathcal{U}}/b_{\mathcal{I}}$ that indicates whether they accept the run.

Collecting points: *To collect points, the protocol* Add *is executed between user $\mathcal{U}$ with token τ containing serial number s and balance w and the accumulator $\mathcal{AC}$ on common input v (the amount of points to be collected). At the end of the protocol, $\mathcal{U}$ outputs an updated token τ^* (with balance $w+v$) while $\mathcal{AC}$ outputs a double-spending tag dstag containing s. They additionally again output a bit $b_{\mathcal{U}}/b_{\mathcal{AC}}$ indicating whether they accepted the run.*

Redeeming points: *To spend points, the* Sub *protocol is executed between a user $\mathcal{U}$ and a verifier $\mathcal{V}$. It has the same inputs and outputs as* Add*, with the exception that the current balance is also part of the common input (i.e. the verifier learns the user's balance in the clear). This allows the verifier to ensure that the user has a sufficient balance to spend the desired amount of points, or can be used in a post-payment system to clear the accumulated debt.*

Detection of double-spending: *Using the same token twice results in two transactions with the same serial number. At regular intervals (or whenever an online connection is available), operators insert the double-spending tags generated through the* Add *and* Sub *protocols into the central database. While doing so, for each inserted tag dstag they check whether the serial number s of it is already contained in the database. If this is the case, the corresponding double-spending tag $dstag'$ is retrieved from the database and* $\mathsf{IdentDS}(dstag, dstag')$ *is used to identify the offending user:* IdentDS*, given two double-spending tags, returns the public key* $\mathsf{PK}_{\mathcal{U}}$ *of a user together with a proof of guilt π or returns an error $\perp$. In following legal action,* VerifyGuilt *can then be used to convince a judge (or anyone else) of the misbehavior:* VerifyGuilt*, given a user public key* $\mathsf{PK}_{\mathcal{U}}$ *and a proof of guilt π outputs a bit b indicating whether the user with public key* $\mathsf{PK}_{\mathcal{U}}$ *is guilty of double-spending.*

The BBA+ scheme comes with strong security and privacy guarantees, which can be divided in the system side and the user side: Security for the system side consists of three goals:

1. Tokens can only be created and used in the name of their legitimate owners (owner-binding).
2. A given user cannot claim more (or less) points than he has collected up to that point, unless an old version of a token is presented (balance-binding)
3. Users presenting old versions of their token can be identified after the fact (double-spending detection).

Security for the user side consists of two goals:

1. Users should have the privacy guarantee that their transactions are anonymous and cannot be used for tracking (privacy-preserving)
2. Nobody should be able to forge a proof that a user has committed a double-spending (false-accusation protection)

To formally define what it means to use a token in the name of someone and what the amount of points a user has collected is, a way to link individual transactions to the owner of the token is required. In BBA+, this is achieved by having the TTP generate a trapdoor which allows linking of transactions to individual users. More formally, they defined a property called "trapdoor linkability" which requires that all transactions result in some linking information which can be used, together with the linking trapdoor, to identify the corresponding user via a deterministic algorithm ExtractUID.

In their security experiments formalizing above goals, an adversary $\mathcal{A}$ may concurrently interact with an honest issuer, accumulator, and verifier an arbitrary (bounded by the polynomial run-time) number of times. The adversary, playing the role of the user, may behave dishonestly and not follow the corresponding protocols. In order to formalize this adversarial setting, Hartung et al. [20] defined oracles the adversary may query:

Definition 3.2 (Oracles)

$\mathsf{MalIssue}(\mathsf{PK}_{\mathcal{U}})$ *lets the adversary initiate the* Issue *protocol with an honest issuer $\mathcal{I}$, 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.*

$\mathsf{MalAcc}(v)$ *is used by the adversary to initiate the* Add *protocol with $\mathcal{AC}$ for input v.*

$\mathsf{MalVer}(v, w)$ *is used by the adversary to initiate the* Sub *protocol with $\mathcal{V}$ for inputs w and v.*

We say that a call to an oracle is successful *if the honest party represented by the oracle accepts the run (i.e. the accept bit $b_{\mathcal{I}}/b_{\mathcal{AC}}/b_{\mathcal{V}}$ is 1).*

Then they defined owner-binding, which states that a token bound to some user public key $\mathsf{PK}_{\mathcal{U}}$ can only be obtained with knowledge of the corresponding $\mathsf{sk}_{\mathcal{U}}$ and that tokens used during Add/Sub must be bound to a public key for which the Issue protocol has been performed.

Next, they require that, unless double-spending occurred, a user cannot claim a different balance than the exact sum of points collected up to that point:

Definition 3.3 (Balance-Binding) *A trapdoor-linkable* BBA+ *scheme is called balance-binding if for any PPT adversary $\mathcal{A}$ in the experiment* $\mathsf{Exp}^{bb}_{\mathsf{BBA+},\mathcal{A}}(n)$ *from Figure 1 the advantage of $\mathcal{A}$ defined by*

$$\mathsf{Adv}^{bb}_{\mathsf{BBA+},\mathcal{A}}(n) := \Pr\left[\mathsf{Exp}^{bb}_{\mathsf{BBA+},\mathcal{A}}(n) = 1\right] \qquad (1)$$

is negligible in n.

Since they desired their system to be offline (i.e. without the need of a permanent connection to some centralized database), users cannot be prevented from using the same token twice. Thus, they require that a user doing so is detected, revealing his identity and providing a proof that he did cheat which can be verified by anyone.

Experiment $\mathsf{Exp}^{\mathsf{bb}}_{\mathsf{BBA+},\mathcal{A}}$

$(crs, td) \leftarrow \mathsf{Setup}(1^n)$, $(\mathsf{PK}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}) \leftarrow \mathsf{IGen}(crs)$

$b \leftarrow \mathcal{A}^{\mathsf{MalIssue},\mathsf{MalAcc},\mathsf{MalVer}}(\mathsf{PK}_{\mathcal{I}})$

The experiment outputs 1 iff $\mathcal{A}$ made a successful call to MalVer with extracted user public-key $\mathsf{PK}_{\mathcal{U}}$ such that

- all successful MalIssue/MalAcc/MalVer calls produced 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 MalAcc or MalVer for which $\mathsf{PK}_{\mathcal{U}}$ could be extracted using ExtractUID.

Figure 1. Balance-binding experiment.

User security is defined using the real/ideal world paradigm. In the real world, the adversary interacts with oracles 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. To this end, it is given access to a simulation trapdoor for the crs. It is then demanded that no PPT adversary can distinguish between the real and the ideal world. For an in-depth explanation of the different oracles used see [20].

Definition 3.4 (Privacy-Preserving) *A* BBA+ *scheme is* privacy-preserving *if there exist PPT algorithms* SimSetup *and* SimCorrupt *as well as interactive PPT algorithms* SimHonIssue, SimHonAdd *and* SimHonSub *that receive no private user input, such that for all PPT adversaries* $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ *in the experiments from Figure 2, the advantage* $\mathsf{Adv}^{priv}_{\mathsf{BBA+},\mathcal{A}}(n)$ *of* $\mathcal{A}$ *defined by*

$$\left|\Pr[\mathsf{Exp}^{priv\text{-}real}_{\mathsf{BBA+},\mathcal{A}}(n) = 1] - \Pr[\mathsf{Exp}^{priv\text{-}ideal}_{\mathsf{BBA+},\mathcal{A}}(n) = 1]\right| \qquad (2)$$

is negligible in n.

Lastly, false-accusation protection guarantees that users cannot be framed for a double-spending they did not commit.

For formal definitions of owner-binding, double-spending detection and false-accusation protection see Appendix B.

4 From BBA+ to BBW

Several shortcomings of the BBA+ scheme, as described in Section 3, were discussed in the introduction. To recall, it relies on Groth–Sahai (GS) zero-knowledge proofs [19] together with pairing-based cryptographic primitives. Since the protocol is interactive anyway, we can replace GS proofs with more efficient interactive proofs, namely Sigma protocols [28]. Now we are not bound by the constraints of GS proofs anymore, in particular, pairing-free instantiations become possible.

Another drawback of BBA+ was the need to reveal the current balance to spend points, as this can impact real-world privacy guarantees. Hartung et al. consider range-proofs to ensure a sufficient balance as a solution, but refrain from using them due to poor performance. We introduce a balance space $\mathbb{V} = \{0, \ldots, 2^{l-1}\}$ (which is sufficiently small to avoid problems caused by overflows, e.g. $2|\mathbb{V}| < |\mathbb{Z}_p|$). We employ Bulletproofs [8], which are very efficient range-proofs compatible with our existing zero-knowledge proofs.

The security guarantees of BBA+ require a method to define the correct balance of a user for the security proof, while still ensuring user privacy. Hartung et al. [20] solve this by introducing a trapdoor that allows to completely abolish privacy, which is only known to a trusted third party (and thus available in the security proofs, but assumed to not be used in the real world). Through our use of interactive zero-knowledge protocols, we can make use of a technique called *rewinding* to achieve the same result, without the need of a trapdoor. Rewinding denotes the process of running an adversary $\mathcal{A}$ to receive some output and then rewind $\mathcal{A}$ to some previous state to let it re-run with the same randomness but different input (e.g. zero-knowledge challenge). While this is possible during security proofs (where black-box access to the program of an adversary is assumed), it is not possible in the real world, where it is not possible for a system operator to reset the user's hardware. Formally, we introduce a new property for BBA+ schemes, called *simulation linkability* which replaces *trapdoor linkability* introduced in [20]. As a side effect, since our common reference string is indeed only a common *random* string, and there is no trapdoor anymore, it is possible to set up our scheme without a trusted third party by generating the crs for example through evaluating a cryptographic hash function on e.g. the timestamp during initialization. Note also that, unlike in BBA+, a misbehaving TTP that leaks information on how the crs was generated (including e.g. the discrete logarithms of the elements in the crs) to the operator does not impact user privacy (though leaking this information to users allows them to claim arbitrary balances and attributes).

Another drawback of BBA+ that hinders deployment in the real world is the need to store double-spending tags forever. We solve this problem by adding attributes to the token that can be revealed during Add and Sub interactions. Their main use is to define validity periods for tokens, which allows to discard double-spending tags after the validity period expired. Additionally, they allow for efficient verification of other attributes the user may have, such as discounted prices for students or age-verification for purchases (such as alcohol). This allows checking of the required attributes once during token generation (by e.g. checking a student card or passport) instead of for each transaction, removing the need to show identifying documents. We want to note though that care has to be taken when choosing the possible attributes, as they might allow linking of users if chosen

```
Experiment $\mathsf{Exp}^{\mathsf{priv\text{-}real}}_{\mathsf{BBA+},\mathcal{A}}(1^n)$
  $(crs, td) \leftarrow \mathsf{Setup}(1^n)$, $(\mathsf{PK}_{\mathcal{I}}, state_0) \leftarrow \mathcal{A}_0(crs)$
  $b \leftarrow \mathcal{A}_1^{\mathsf{HonUser},\mathsf{RealHonIssue},\mathsf{RealHonAdd},\mathsf{RealHonSub},\mathsf{RealCorrupt}}(\mathsf{PK}_{\mathcal{I}}, state_0)$
  return $b$
```

```
Experiment $\mathsf{Exp}^{\mathsf{priv\text{-}ideal}}_{\mathsf{BBA+},\mathcal{A}}(1^n)$
  $(crs, td_{\mathsf{sim}}) \leftarrow \mathsf{SimSetup}(1^n)$, $(\mathsf{PK}_{\mathcal{I}}, state_0) \leftarrow \mathcal{A}_0(crs)$
  $b \leftarrow \mathcal{A}_1^{\mathsf{HonUser},\mathsf{SimHonIssue},\mathsf{SimHonAdd},\mathsf{SimHonSub},\mathsf{SimCorrupt}}(\mathsf{PK}_{\mathcal{I}}, state_0)$
  return $b$
```

Figure 2. Real/Ideal world privacy experiments

poorly. In our envisioned scenarios, either the majority of users share the same attributes (e.g. there are only 2 validity periods in use at any given time) or the attributes are already clear from the context (e.g. to be able to buy alcohol, the user must be above the legal age). The user learns (during token creation) which attributes will be part of his token and during each interaction chooses which attributes will be revealed (outside of our protocol), and needs to not accept attributes that would violate his privacy. As a concrete example, we propose validity periods of six months, ending every three months, so new tokens are always valid for between three and six months (depending on how close to the end of a quarter the token is issued). This allows to keep the database of double-spending tags small while also ensuring that approximately half the users share the same attribute. We want to stress that only attributes required for a given transaction are revealed (e.g. when no age-restricted goods are to be purchased, there is no need to reveal the age-verification attribute).

Definition 4.1 (BBW Scheme) *A* BBW *Scheme consists of PPT algorithms* Setup, IGen, UGen, *deterministic polynomial time algorithms* UVer, IdentDS, VerifyGuilt *and interactive protocols* Issue, Add *and* Sub. *At the core of the scheme lies a token τ that contains a unique serial number s, the tokens balance $w \in \mathbb{V}$ and some attribute(s) $attr$ and is bound to the owner's identity (i.e. the user's public key).*

Setup: *The* Setup *algorithm takes as input the security parameter n and outputs a common reference string crs.* $\mathsf{IGen}(crs)$ *generates a key pair $(\mathsf{PK}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}})$ for the issuer, which is also shared with the accumulator and verifier.* $\mathsf{UGen}(crs)$ *generates a user key pair $(\mathsf{PK}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}})$.* UVer *given a user key pair $(\mathsf{PK}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}})$, a token τ, attribute $attr$ and a balance w outputs a bit b indicating whether τ is a valid token with public attribute $attr$ and balance w owned by the user with public key $\mathsf{PK}_{\mathcal{U}}$.*

Token generation: *The protocol* Issue *is executed between a user $\mathcal{U}$, given some attribute $attr$, his own key pair $(\mathsf{PK}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}})$ and $\mathsf{PK}_{\mathcal{I}}$; and the issuer $\mathcal{I}$, given $\mathsf{PK}_{\mathcal{U}}$, the same attribute $attr$ and the issuer key pair $(\mathsf{PK}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}})$. Upon successful execution, $\mathcal{U}$ outputs a token τ with initial balance 0 and public attribute $attr$ for the key pair $(\mathsf{PK}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}})$. Both $\mathcal{U}$ and $\mathcal{I}$ additionally output a bit $b_{\mathcal{U}}/b_{\mathcal{I}}$ that indicates whether they accept the run.*

Collecting points: *To collect points, the protocol* Add *is executed between a user $\mathcal{U}$ with token τ containing serial number s, balance w and public attribute $attr$ and the accumulator $\mathcal{AC}$ on common input $(attr, v)$ (the public attribute of the token and the amount of points to be collected). At the end of the protocol, $\mathcal{U}$ outputs an updated token τ^* (with balance $w + v$) while $\mathcal{AC}$ outputs a double-spending tag $dstag$ containing s. They additionally again output a bit $b_{\mathcal{U}}/b_{\mathcal{AC}}$ indicating whether they accepted the run.*

Spending points: *To spend points, the* Sub *protocol is executed between a user $\mathcal{U}$ and a verifier $\mathcal{V}$. It has the same inputs and outputs as* Add*, except that the new balance w^* is computed as $w^* = w - v$. Additionally, the user proves (in zero-knowledge) that $w \geq v$ holds.*

Detection of double-spending: IdentDS*, given two double-spending tags, returns the public key $\mathsf{PK}_{\mathcal{U}}$ of a user together with a proof of guilt π or returns an error $\perp$.* VerifyGuilt *given a user public key $\mathsf{PK}_{\mathcal{U}}$ and a proof of guilt π outputs a bit b indicating whether the user with public key $\mathsf{PK}_{\mathcal{U}}$ is guilty of double-spending.*

To remove the need of the trapdoor for ExtractUID, we formally define what it means for a scheme to be *simulation-linkable*. The main difference to *trapdoor-linkable* is that there is no need for a trapdoor but the algorithm is allowed to fail with negligible probability.

For some fixed $n \in \mathbb{N}$, $crs \leftarrow \mathsf{Setup}(1^n)$ $\mathcal{T}^{\mathsf{Add}}_{n,crs}$ denotes the set of transcripts (including $\mathcal{AC}$'s output) resulting from any Add protocol run accepted by $\mathcal{AC}$ with any (potentially malicious) party, and $\mathcal{T}^{\mathsf{Sub}}_{n,crs}$ analog for $\mathcal{V}$.

Definition 4.2 (Simulation-Linkability) *A* BBW *scheme is called simulation-linkable if it satisfies the following conditions:*

Completeness: *Let $n \in \mathbb{N}$, $(crs, td) \leftarrow \mathsf{Setup}(1^n)$ and $tr \in \mathcal{T}^{\mathsf{Add}}_{n,crs}$. Then there exist inputs $\mathsf{PK}_{\mathcal{U}}$, $\mathsf{sk}_{\mathcal{U}}$, τ, w and random choices for an honest user $\mathcal{U}$ and honest accumulator $\mathcal{AC}$ such that an* Add *protocol run between $\mathcal{U}$ and $\mathcal{AC}$ with these inputs and random choices leads to the same transcript tr. The same holds for all $tr \in \mathcal{T}^{\mathsf{Sub}}_{n,crs}$ with respect to* Sub.

Extractability: *There exists a PPT algorithm* ExtractUID *that, given two related transcripts $tr_1, tr_2 \in \mathcal{T}^{\mathsf{Add}}_{n,crs}$ (or $\mathcal{T}^{\mathsf{Sub}}_{n,crs}$ resp.) produced by the interaction of a honest user $\mathcal{U}$ with public key $\mathsf{PK}_{\mathcal{U}}$ and the honest accumulator $\mathcal{AC}$ (or honest verifier $\mathcal{V}$ resp.) outputs the public key $\mathsf{PK}_{\mathcal{U}}$. Two transcripts tr_1, tr_2 are called related if they are identical up to the point where the zero-knowledge challenge* *is sent, and then contain different challenges and different responses.*

Additionally, there exists an expected PPT algorithm $\mathsf{GenerateTranscripts}$ *that, given access to a transcript oracle* $\mathcal{O} = \langle \mathcal{U}, \mathcal{AC} \rangle$ *(or* $\mathcal{O} = \langle \mathcal{U}, \mathcal{V} \rangle$ *resp.) outputs two related transcripts* $tr_1, tr_2 \in \mathcal{T}^{\mathsf{Add}}_{n,crs}$ *(resp.* $tr_1, tr_2 \in \mathcal{T}^{\mathsf{Sub}}_{n,crs}$*) with overwhelming probability.* $\mathsf{GenerateTranscripts}$ *is allowed to rewind* $\mathcal{O}$ *and resume with fresh randomness for* $\mathcal{AC}$ *(resp.* $\mathcal{V}$*).*

Remark
Since $\mathsf{ExtractUID}$ *only receives transcripts* $tr_1, tr_2 \in \mathcal{T}^{\mathsf{Add}}_{n,crs}$ *(resp.* $\mathcal{T}^{\mathsf{Sub}}_{n,crs}$*), and completeness in above definition requires that any such transcript can be produced by an honest user, the output of* $\mathsf{ExtractUID}$ *is well defined even for malicious users* $\mathcal{U}^*$*.*

We note that our version of $\mathsf{ExtractUID}$ only outputs the user's public key with overwhelming probability (instead of always, as is the case for trapdoor linkability). Thus, it is necessary to extend all security experiments: the adversary additionally also wins if he is able to perform a successful interaction for which $\mathsf{ExtractUID}$ fails to output the public key. Additionally, for user privacy, the ideal world oracles now need access to the user attributes where needed, but the oracle for Sub does not receive the balance anymore. The formal definitions are lengthy and can be found in Appendices D.3 and D.4.

Remark (Modeling privacy for attributes)
Recall that modeling user security via the real/ideal paradigm ensures that a real-world adversary learns no *more than an ideal-world adversary. However, if a (desired) privacy property does not hold in the ideal world, then neither need it hold in the real. Thus, we must assume that attributes are used in a privacy-preserving manner. Uniquely identifying attributes do* not *contradict* real/ideal *world security. However, due it is sufficient to study the impact of attributes on privacy in the ideal world. The impact on privacy is* highly dependent on the setting. *Thus, it has to be carefully analyzed individually for every application of* BBW*. In Section 4, we outlined some acceptable usage, such as children, adult, senior pricing earlier where a "physical world" adversary must learn the attributes anyway. Similarly, sensible implementations of expiration dates should preserve privacy, because at any time, a large portion of users share the same expiration date.*

5 Instantiation

Recall that a token contains a signature on (a commitment to) the balance, serial number and some other values, which is verified in a transaction. Then to modify the balance, a new signature on (a commitment to) the updated balance and a new serial number is formed. To provide correctness on the one hand, and user privacy on the other hand, ZK proofs are used to hide sensible information while proving that the new commitment is formed correctly. Thus, to instantiate our scheme, we need to find a suitable and efficient zero-knowledge proof system and a compatible signature and commitment scheme. We have one algorithm $\mathsf{SetupGrp}$ that outputs the description of an elliptic curve group $\mathbb{G}$ of prime order p with generator G which is then used by all building blocks.

As already mentioned, for efficiency reasons we make use of sigma protocols for our ZK proofs. We make use of the generalized definition of a sigma protocol by Maurer [28] which is perfect special honest verifier zero-knowledge. We extend it by replacing the challenge with a Blum coin-toss to achieve perfect composable zero-knowledge as proposed by Damgård [14]. To do so, we use the Pedersen commitment scheme for messages of length 1 (a single element of $\mathbb{Z}_p$). We refer to this commitment scheme as C_{ZK} in the following. Maurer's scheme allows proofs of knowledge for preimages of group-homomorphisms (where the homomorphism may depend on the statement being proven). We consider homomorphisms $\Phi\colon \mathbb{A} \to \mathbb{B}$, where $\mathbb{A}$ and $\mathbb{B}$ are of the form $\mathbb{G}^{n_1} \times \mathbb{Z}_p^{n_2}$.

To prove that $w \geq v$ in the Sub protocol, we use range proofs. A range proof is a ZK proof that the content of a commitment lies within a given interval. We prove that $w - v \in \mathbb{V}$, which is equivalent to $w \geq v$ as long as $|\mathbb{Z}_p| > 2|\mathbb{V}|$. We use Bulletproofs [8] to implement the range proof, referred to as RP in the following.

For commitments, we make use of the Pedersen multi-commitment scheme, referred to as PC in the following, which allows to commit to a vector m^l of messages in a single group element.

Choosing a signature scheme poses the greatest challenge: the running time of the scheme is dominated by verifying signatures (both in zero-knowledge and on the user side to ensure correct token creation). We need a signature scheme that allows signing of committed messages (i.e. signing a group element) and verification of such signatures in an unlinkable way. Additionally, both during the process of obtaining a signature as well as during verification, it must be possible to prove, in zero-knowledge, statements involving the content of the signed message. The latter requirement severely restricts the class of possible signature schemes. While structure-preserving signatures obviously fulfill our requirements, they rely on pairings. Normal blind signature schemes (without pairings) usually do not allow to (efficiently) prove statements about the content of the signed message (for example due to the use of a cryptographic hash function whose output is then signed). The same problem arises when using normal signatures to get a signature on a commitment and then prove knowledge of both the commitment and the signature in zero-knowledge (which satisfies unlinkability), as again the use of a hash-function on the message prevents efficient zero-knowledge proofs containing the content of the message. Camenisch and Lysyanskaya [10, 11] introduced a class of signatures commonly referred to as CL-type signatures intended for usage in privacy-preserving protocols. CL-type signatures consist of a commitment scheme and a signature scheme and come with efficient (interactive) protocols to obtain signatures on committed messages and to prove knowledge of a signature in zero-knowledge, which are exactly the properties we require. Unfortunately, known CL-type signatures rely on either RSA groups ([10]) or pairings ([11, 30]). Thus, we decided to adapt the blind signature scheme of Baldimtsi and Lysyanskaya [3] into a CL-type signature scheme. Since the authors only provided a proof for sequential security of their blind signature protocol (where the adversary must finish one interaction with the signer before he is allowed to start a new one), while we require concurrent security for our scheme (as it is implausible to assume an adversary cannot interleave protocol executions in a distributed setting as targeted by us), we also provide a proof of concurrent security in the generic group model which may be of independent interest. The resulting signature scheme is described in detail in Section 6.

5.1 Instantiating BBW

The scheme is summarized in Figures 3 and 5 to 7. In the following, we explain the details of the algorithms and protocols. Whenever a ZKPoK is conducted between parties, the verifying party aborts and outputs $\perp$ if it does not accept the proof.

Setup: Setup, run by the trusted third party, generates a suitable group and the common reference string crs for the scheme, which consists of a common reference string crs_{com} for the commitment scheme PC and a common reference string crs_{pok} for the zero-knowledge scheme Z that is used in Issue, Add and Sub.

IGen generates a key pair for the system operator, which simply consists of a key pair for the signature scheme S.

UGen generates a key pair for a user, consisting of a secret key $\mathsf{sk}_{\mathcal{U}}$ (a random element in $\mathbb{Z}_p$), and a public key $\mathsf{PK}_{\mathcal{U}} = \mathsf{sk}_{\mathcal{U}} G$, which is used as the users identity.

Token generation: The protocol to issue new tokens is shown in Figure 5. A token consists of a random serial number $s \in \mathbb{Z}_p$, a balance $w \in \mathbb{V}$, the users secret key $\mathsf{sk}_{\mathcal{U}} \in \mathbb{Z}_p$, and the double-spending information $u_1 \in \mathbb{Z}_p$, together with a valid signature σ (under the issuers public key $\mathsf{PK}_{\mathcal{I}}$) on the message $m := (s, w, \mathsf{sk}_{\mathcal{U}}, u_1)$. To ensure the serial number s is indeed random, it is chosen jointly by the user and the issuer (making use of the homomorphic property of the commitment scheme PC).

To obtain an initial token, the user forms a commitment C' on $(s' \in \mathbb{Z}_p, 0, \mathsf{sk}_{\mathcal{U}}, u_1)$, where $s' \leftarrow \mathbb{Z}_p$ is his random share of the serial number and $u_1 \leftarrow \mathbb{Z}_p$, with initial balance 0. He then performs a zero-knowledge proof P1 with the issuer to prove knowledge of the secret key corresponding to his public key $\mathsf{PK}_{\mathcal{U}}$, knowledge of the content of the commitment C' and to ensure that the balance is 0. The issuer verifies the zero-knowledge proof, then chooses $s'' \leftarrow \mathbb{Z}_p$, uses the homomorphic property of PC to generate a commitment C on $(s' + s'', 0, \mathsf{sk}_{\mathcal{U}}, u_1)$ and then engages in an interaction of the protocol BlindSign of S to sign the tuple $(s' + s'', 0, \mathsf{sk}_{\mathcal{U}}, u_1)$ with his private key $\mathsf{sk}_{\mathcal{I}}$. Finally, after receiving the signature through BlindSign, the user verifies that it is valid.

```
Setup(1^n, l)
    $gp := (\mathbb{G}, p, G) \leftarrow \mathsf{SetupGrp}(1^n)$
    $(crs_{\mathsf{sig}}, crs_{\mathsf{com}}) \leftarrow \mathsf{S.Setup}(gp, 5)$
    $crs_{\mathsf{pok}} \leftarrow \mathsf{Z.Setup}(gp)$
    $crs_{\mathsf{rp}} \leftarrow \mathsf{RP.Setup}(gp, l)$
    $crs := (gp, crs_{\mathsf{sig}}, crs_{\mathsf{com}}, crs_{\mathsf{pok}}, crs_{\mathsf{rp}})$
    return $crs$

IGen(crs)
    $(\mathsf{PK}_{\mathrm{sig}}, \mathsf{sk}_{\mathrm{sig}}) \leftarrow \mathsf{S.KeyGen}(crs)$
    return $(\mathsf{PK}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}) := (\mathsf{PK}_{\mathrm{sig}}, \mathsf{sk}_{\mathrm{sig}})$

UGen(crs)
    $y \leftarrow \mathbb{Z}_p$
    $(\mathsf{PK}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}) := (yG, y)$
    return $(\mathsf{PK}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}})$
```

Figure 3. Setup and Key Generation

Proof P1 (for the Issue protocol)
Statement: $crs_{\mathsf{com}}, C', \mathsf{PK}_{\mathcal{U}}$
Witness $s', \mathsf{sk}_{\mathcal{U}}, u_1, d'$ so that

$$\begin{aligned} \mathsf{PC.Open}(crs_{\mathsf{com}}, (s', 0, \mathsf{sk}_{\mathcal{U}}, u_1, 0), C', d') &= 1 \\ \mathsf{sk}_{\mathcal{U}} G &= \mathsf{PK}_{\mathcal{U}} \end{aligned}$$

Proof P2 (for the Add protocol)
Statement: $crs_{\mathsf{com}}, s, t, u_2, C', attr, \sigma_1$
Witness $s', w, \mathsf{sk}_{\mathcal{U}}, u_1', d', C, u_1, \sigma_2$ so that

$$\begin{aligned} \mathsf{PC.Open}(crs_{\mathsf{com}}, (s', w, \mathsf{sk}_{\mathcal{U}}, u_1', attr), C', d') &= 1 \\ \mathsf{PCB.Open}(crs_{\mathsf{com}}, (s, w, \mathsf{sk}_{\mathcal{U}}, u_1, attr), (\zeta, \zeta_1), \sigma_2) &= 1 \\ \mathsf{sk}_{\mathcal{U}} u_2 + u_1 &= t \end{aligned}$$

Proof P3 (for the Sub protocol)
Statement: $crs_{\mathsf{com}}, crs_{\mathsf{rp}}, s, t, u_2, C', attr, \sigma_1, C_{\mathsf{RP}}, v$
Witness $s', w, \mathsf{sk}_{\mathcal{U}}, u_1', d', C, u_1, \sigma_2, r$ so that

$$\begin{aligned} \mathsf{PC.Open}(crs_{\mathsf{com}}, (s', w, \mathsf{sk}_{\mathcal{U}}, u_1', attr), C', d') &= 1 \\ \mathsf{PCB.Open}(crs_{\mathsf{com}}, (s, w, \mathsf{sk}_{\mathcal{U}}, u_1, attr), (\zeta, \zeta_1), \sigma_2) &= 1 \\ \mathsf{sk}_{\mathcal{U}} u_2 + u_1 &= t \\ \mathsf{PC.Open}(crs_{\mathsf{com}}, (w - v), C_{\mathsf{RP}}, r) &= 1 \end{aligned}$$

Additionally, a range proof that $w - v \in \mathbb{V}$ (using C_{RP}) is performed.

Figure 4. Zero-Knowledge Proofs

```
$\mathcal{U}(\mathsf{PK}_\mathcal{I}, \mathsf{PK}_\mathcal{U}, \mathsf{sk}_\mathcal{U}, attr)$                    $\mathcal{I}(\mathsf{PK}_\mathcal{U}, \mathsf{PK}_\mathcal{I}, \mathsf{sk}_\mathcal{I}, attr)$
$s', u_1 \leftarrow \mathbb{Z}_p$
$m' := (s', 0, \mathsf{sk}_\mathcal{U}, u_1, 0)$
$(C', d) \leftarrow \mathsf{PC.Com}(m')$
                        $\xrightarrow{C', \pi_1}$
                        $\boxed{\text{ZKPoK P1}}$
                                                   $s'' \leftarrow \mathbb{Z}_p$
                                                   $m'' := (s'', 0, 0, 0, attr)$
                                                   $(C'', 0) \leftarrow \mathsf{PC.Com}(m''; 0)$
                        $\xleftarrow{C'', s''}$
$C = C' + C''; s = s' + s''$                       $C = C' + C''$
$m := (s, 0, \mathsf{sk}_\mathcal{U}, u_1, attr)$
$\sigma \longleftarrow$ $\boxed{\mathsf{BlindSign}(C, m)}$
$\tau = (\sigma, s, u_1)$
if $\mathsf{UVer}(\mathsf{sk}_\mathcal{U}, \tau, 0, attr) = 0$
    return $(\bot, 0)$
else return $(\tau, 1)$                            return $1$
```

Figure 5. Issue Protocol

Collecting points: To collect points, the Add protocol described in Figure 6 is used. In the protocol, $\mathcal{AC}$ starts by choosing a random value $u_2 \leftarrow \mathbb{Z}_p$ and sending it to $\mathcal{U}$, who then computes the second part of the double-spending token (s, t) as $t = \mathsf{sk}_\mathcal{U} u_2 + u_1$. The user then prepares a new token by again forming a commitment C' on $(s', w, \mathsf{sk}_\mathcal{U}, u_1')$, where $s', u_1' \leftarrow \mathbb{Z}_p$ are new random values, while w is the same as in his existing token. The user then sends C' and t together with the serial number s contained in his token to the accumulator and performs the ZKPoK P2 with $\mathcal{AC}$ to prove knowledge of a valid token and correct computation of t and C'. After verifying the proof, $\mathcal{AC}$ adds his share $s'' \leftarrow \mathbb{Z}_p$ to the new serial number as well as the amount of points collected v to the balance in C' (again using the homomorphic property of PC). Then the user and $\mathcal{AC}$ again perform the BlindSign protocol to sign the new tuple $(s' + s'', w + v, \mathsf{sk}_\mathcal{U}, u_1')$. Finally, the user again checks that the resulting token is valid and contains the correct new balance.

Spending points: The protocol Sub works mostly the same as Add, with the difference that v is subtracted from the balance and P3 contains a range proof that $w - v \in \mathbb{V}$.

Detection of double-spending: Algorithms IdentDS and VerifyGuilt are shown in Figure 7. IdentDS exploits that the two double-spending tags are blinded by the same value u_1 to extract the users secret key. The users identity is then revealed by computing his public key. VerifyGuilt checks that the value contained in the proof of guilt π is indeed the secret key corresponding to $\mathsf{PK}_\mathcal{U}$.

5.2 Security of BBW

In the security games (see Section 3 as well as appendix B), the adversary engages in multiple, possibly concurrent and arbitrarily interleaved ZK arguments. We show by hybrid argument that this does not affect the possibility to extract witnesses from those arguments.

Theorem 5.1 (Extraction) *If* P1, P2 *and* P3 *have witness-extended emulation, there exists an expected PPT extractor $\mathcal{E}^*$ that has overwhelming probability for extracting witnesses for all successful calls to* MalIssue, MalAcc *and* MalVer *made by an PPT adversary $\mathcal{A}$ in the security games.*

```
$\mathcal{U}(\mathsf{PK}_\mathcal{I}, \mathsf{PK}_\mathcal{U}, \mathsf{sk}_\mathcal{U}, w, v, attr)$              $\mathcal{AC}(\mathsf{PK}_\mathcal{U}, \mathsf{PK}_\mathcal{I}, \mathsf{sk}_\mathcal{I}, v, attr)$
parse: $(\sigma, s, u_1) := \tau$                                   $u_2 \leftarrow \mathbb{Z}_p$
parse: $(\sigma_1, (d, \gamma)) := \sigma$
                        $\xleftarrow{u_2}$
$t = \mathsf{sk}_\mathcal{U} u_2 + u_1$
$s', u_1' \leftarrow \mathbb{Z}_p$
$m' := (s', w, \mathsf{sk}_\mathcal{U}, u_1', attr)$
$(C', d^*) \leftarrow \mathsf{PC.Com}(m')$
                        $\xrightarrow{s, t, C', \sigma_1, \pi_1}$
                        $\boxed{\text{ZKPoK P2}}$
                                                   if $\neg\mathsf{S.Verify}_1(\sigma_1)$
                                                       return $(\bot, 0)$
                                                   $dstag = (s, t, u_2)$
                                                   $s'' \leftarrow \mathbb{Z}_p$
                                                   $m'' := (s'', v, 0, 0, 0)$
                                                   $(C'', 0) \leftarrow \mathsf{PC.Com}(m''; 0)$
                                                   $C^* = C' + C''$
                        $\xleftarrow{C'', s''}$
$C^* = C' + C''$
$s^* = s' + s''; w^* = w + v$
$m^* := (s^*, w^*, \mathsf{sk}_\mathcal{U}, u_1', attr)$
$\sigma^* \longleftarrow$ $\boxed{\mathsf{BlindSign}(C^*, m^*)}$
$\tau^* = (\sigma^*, s^*, u_1')$
if $\mathsf{UVer}(\mathsf{sk}_\mathcal{U}, \tau^*, w^*, attr) = 0$
    return $(\bot, 0)$
else return $(\tau, 1)$                            return $(dstag, 1)$
```

Figure 6. Add Protocol

```
$\mathsf{UVer}(\mathsf{PK}_\mathcal{I}, \mathsf{PK}_\mathcal{U}, \mathsf{sk}_\mathcal{U}, \tau, w, attr)$
parse: $(\sigma, s, u_1) := \tau$
if $(\mathsf{PK}_\mathcal{U} = \mathsf{sk}_\mathcal{U} G \wedge \mathsf{S.Verify}(\mathsf{PK}_\mathcal{I}, (s, w, \mathsf{sk}_\mathcal{U}, u_1, attr), \sigma) = 1)$
    return 1
else return 0

$\mathsf{IdentDS}(\mathsf{PK}_\mathcal{I}, (s_1, z_1), (s_2, z_w))$
parse: $(t, u_2) := z_1, (t', u_2') := z_2$
if $\left(s_1 \neq s_2 \vee u_2 = u_2'\right)$ return $\bot$
else $\mathsf{sk}_\mathcal{U} = (t - t')(u_2 - u_2')^{-1} \mod p, \mathsf{PK}_\mathcal{U} = \mathsf{sk}_\mathcal{U} G$
return $(\mathsf{PK}_\mathcal{U}, \pi := \mathsf{sk}_\mathcal{U})$

$\mathsf{VerifyGuilt}(\mathsf{PK}_\mathcal{I}, \mathsf{PK}_\mathcal{U}, \pi)$
if $\pi G = \mathsf{PK}_\mathcal{U}$ return 1
else return 0
```

Figure 7. User verification of tokens and double-spending algorithms

The idea in the proof is to extract one witness at a time and view all oracle calls except the one for which the witness is being extracted as part of the adversary. This is possible since we do not rely on extracted information during interaction with the adversary, but only to determine whether the adversary was successful. For the full proof, see Proof 4.

In the following, we refer to an interactive protocol between an honest party and a (possibly) malicious party as complete if for every transcript *tr* of a transaction that results in the honest party accepting, there exists inputs and random choices for honest parties so that an interaction between them results in the same transcript.

Theorem 5.2 (Simulation-Linkability) *If* BBW *and* S *are correct,* P2*,* P3 *and* $\mathsf{BlindSign}$ *are complete and* P2 *and* P3 *have witness-extended emulation,* BBW *is simulation-linkable (Definition 4.2).*

Theorem 5.3 (System Security) *If* BBW *is simulation-linkable, the premises of Theorem 5.1 and the DLOG assumption in* $\mathbb{G}$ *hold,* PC *is additively homomorphic and binding and* S *is* EUF-CMA *secure, then* BBW *is owner-binding (Definition B.1), balance-binding (Definition 3.3) and ensures double-spending detection (Definition B.2).*

For the proofs, see Appendix D.3. Owner-binding wrt. Issue is a straight-forward reduction to CDH. For owner-binding wrt. $\mathsf{Add}/\mathsf{Sub}$, we show that any adversary able to win the game with non-negligible probability can be used to break EUF-CMA security of S. The proof for balance-binding closely follows the one in [20]. Equivocation of Pedersen commitments does not have to be considered explicitly anymore as it is covered by EUF-CMA security. Instead of building a graph based on signed commitments, a graph based on the unique serial numbers can be built. Then it can be shown that an adversary successfully attacking balance-binding can be used to break EUF-CMA security of S.

Theorem 5.4 (User Security) *If* P1*,* P2 *and* P3 *are composable zero-knowledge and* PC *is equivocable, then* BBW *is privacy-preserving, and if additionally the DLOG assumption holds in* $\mathbb{G}$*, it is secure against false accusation of double-spending.*

The proof is largely the same as in [20]. For details, see Appendix D.4. We make use of the simulation trapdoor (i.e. the equivocality trapdoors of the commitment scheme) to step-by-step remove private input from the oracles until we are left with the ideal-world oracles that do not receive any private input. The main difference to [20] is that the signature σ_1 is sent to the adversary during the protocol, and replacing it by a simulated one relies on the DDH assumption to be indistinguishable (see Appendix C.2).

The proof for false-accusation protection is straightforward and the same as in [20].

6 Signature Scheme

Starting from the signature scheme of Baldimtsi and Lysyanskaya [3], we construct a CL-type signature scheme: Messages of arbitrary but fixed length l are first committed into a single group element using Pedersen commitments and then signed.

The scheme of Baldimtsi and Lysyanskaya (called ACL) is a blind signature *with attributes*, where efficient ZK proofs regarding the attributes are possible. It is based on the blind signature scheme of Abe [1] and a blinded variant of the Pedersen commitment scheme: Given a Pedersen commitment C on m (with opening information d) and an additional generator Z (that is part of the crs), $C' = (\gamma Z, \gamma C)$ is also a commitment on m with opening information (d, γ). Additionally, C' and C are unlinkable (without knowledge of γ) under the DDH assumption. Below, we refer to this variant of Pedersen commitments as PCB.

We interpret these "attributes" as the message, which then allows for efficient ZK proofs regarding the message. The resulting scheme works roughly as follows. The signer has a real public key PK (of which he knows the corresponding secret key sk) and a "tag" public key Z (generated by a trusted party or by applying a cryptographic hash function to his public key, so that nobody knows the corresponding secret key). To sign a message m, a blinded Pedersen commitment $C' = (Z_1, Z_2)$ is created (using the tag public key Z). A signature then consists of a non-interactive (via the Fiat–Shamir transformation) ZKPoK, proving knowledge of either the secret key sk or the discrete logarithms of Z_1 and Z_2 (which are not known to anyone). This construction allows for efficient ZK proofs involving m. To obtain blind signatures, a user sends C (together with a ZKPoK of the opening to m) to the signer, who then interactively performs the ZKPoK with the user. To obtain the challenge, the user queries the hash function (after applying appropriate blinding factors). This results in a signature that is unlinkable to any information received by the signer during signing.

Definition 6.1 *Our CL-type signature scheme consists of algorithms* Setup, $\mathsf{SimSetup}$, KeyGen, Sign *and* Verify*:*

$\mathsf{Setup}(gp, l)$**:** *Generates a common reference string* $crs_{\mathsf{com}} := (H_0, \ldots, H_l) \leftarrow \mathsf{PC.Setup}(gp, l)$*, samples random* $Z, H \leftarrow \mathbb{G}$ *and chooses a hash function* $\mathcal{H} : \{0,1\}^* \to \mathbb{Z}_p$*. Outputs* $crs = (gp, \mathcal{H}, Z, H, crs_{\mathsf{com}})$*.*

$\mathsf{SimSetup}(gp, l)$**:** *Generates a common reference string* $(crs_{\mathsf{com}}, td_{\mathsf{com}}) \leftarrow \mathsf{PC.SimSetup}(gp, l)$*, samples random* $z, h \leftarrow \mathbb{Z}_p$*, computes* $Z = zG, H = hG$ *and chooses a hash function* $\mathcal{H} : \{0,1\}^* \to \mathbb{Z}_p$*. Outputs* $(crs, td) = ((gp, \mathcal{H}, Z, H, crs_{\mathsf{com}}), (z, h, td_{\mathsf{com}}))$*.*

$\mathsf{KeyGen}(crs)$**:** *Samples random* $\mathsf{sk} \leftarrow \mathbb{Z}_p$ *and computes* $\mathsf{PK} = \mathsf{sk}G$*. Outputs* $(\mathsf{PK}, \mathsf{sk})$*.*

Sign(crs, sk, m): *Sets* $C = \mathsf{PC.Com}(crs_{\mathsf{com}}, m; 0)$. *Then samples random* $u, r_1', r_2', c', u_3' \leftarrow \mathbb{Z}_p$ *and computes* $A = uG$, $B_1 = r_1'G + c'C$, $B_2 = r_2'H + c'(Z - C)$, $B_3 = u_3'Z$ *and sets* $e = \mathcal{H}(Z, Z_1, A, B_1, B_2, B_3)$. *Then computes* $c = e - c'$, $r = u - cx$, $r_3' = u_3' - c'$ *and outputs* $\sigma = (\sigma_1, \sigma_2)$ *where* $\sigma_1 := (Z, C, r, c, r_1', r_2', c', r_3)$ *and* $\sigma_2 := (0, 1)$.

Verify$(crs, \mathsf{PK}, m, \sigma)$: *Parse* $\sigma := (\sigma_1, \sigma_2)$ *where* $\sigma_1 := (\tilde{Z}, \tilde{C}, \tilde{r}, \tilde{c}, \tilde{r}_1', \tilde{r}_2', \tilde{c}', r_3')$ *and* $\sigma_2 := (d, \gamma)$. *Check that* $\tilde{Z} \neq 0$, $\mathsf{PCB.Open}(crs_{\mathsf{com}}, m, (\tilde{Z}, \tilde{C}), \sigma_2) = 1$ *and* $\mathsf{Verify}_1(crs, \mathsf{PK}, \sigma_1) = 1$, *where* Verify_1 *outputs* 1 *if for* $\tilde{A} = \tilde{r}G + \tilde{c}\mathsf{PK}$, $\tilde{B}_1 = \tilde{r}_1'G + \tilde{c}'\tilde{C}$, $\tilde{B}_2 = \tilde{r}_2'H + \tilde{c}'(\tilde{Z} - \tilde{C})$ *and* $B_3 = r_3'Z + \tilde{c}'\tilde{Z}$ *it holds that*

$$\tilde{c} + \tilde{c}' = \mathcal{H}(\tilde{Z}, \tilde{C}, \tilde{A}, \tilde{B}_1, \tilde{B}_2, B_3)$$

and 0 *otherwise.*

The scheme additionally consists of two interactive protocols for obtaining blind signatures and verification of signatures in zero-knowledge:

BlindSign: *The user first computes a commitment* $(C, d) \leftarrow \mathsf{PC.Com}(crs_{\mathsf{com}}, m)$ *on his message* m. *He then sends* C *to the signer and performs a ZKPoK* Π_1 *of* m *and* d *such that* $\mathsf{PC.Open}(crs_{\mathsf{com}}, m, C, d) = 1$. *If the signer accepts the proof, they then engage in the interactive signing protocol to generate a blind signature.*

See Figure 8 for the description of the interactive signing protocol.

BlindVerify: *The user with signature* $\sigma := (\sigma_1, \sigma_2)$ *on a message* m, *where* $\sigma_1 = (\tilde{Z}, \tilde{C}, \tilde{r}, \tilde{c}, \tilde{r}_1', \tilde{r}_2', \tilde{c}', r_3')$ *and* $\sigma_2 = (d, \gamma)$, *sends* σ_1 *to the verifier and performs a ZKPoK* Π_2 *of* m *and* σ_2 *such that* $\mathsf{PCB.Open}(crs_{\mathsf{com}}, m, (\tilde{Z}, \tilde{C}), \sigma_2) = 1$. *The verifier checks that* $\mathsf{Verify}_1(crs, \mathsf{PK}, \sigma_1) = 1$ *and* Π_2 *is valid. To implement* Π_2 *as a* Σ*-protocol, the equations to be shown can be rewritten as*

$$\mathsf{Com}(m; d) - \gamma'\tilde{C} = 0 \tag{3}$$
$$\gamma'\tilde{Z} = Z \tag{4}$$

where $\gamma' = \gamma^{-1}$.

Remark
Modifications detailed above are mainly regarding notation and to remove the blind message and thus allow the EUF-CMA *security definition and do not change ACL on a technical level.*

Theorem 6.2 *Above signature scheme is* EUF-CMA *secure in the combined generic group and random oracle model.*

User(PK, C, m, d)		**Signer**(sk, C)
ensure $\mathsf{PC.Open}(m, C, d) = 1$		
	$\boxed{\text{ZKPoK } \Pi_1}$	
$\gamma \leftarrow \mathbb{Z}_p^*$		$C' = Z - C$
$\tilde{Z} = \gamma Z,\ \tilde{C} = \gamma Z_1$		$u, r_1', r_2', c' \leftarrow \mathbb{Z}_p$
$\tilde{C}' = \tilde{Z} - \tilde{C}$		$A = uG$
$u_3' \leftarrow \mathbb{Z}_p$		$B_1 = r_1'G + c'C$
$B_3 = u_3'Z$		$B_2 = r_2'H + c'C'$
	$\xleftarrow{A, B_1, B_2}$	
check that $A, B_1, B_2 \in \mathbb{G}$		
$t_1, t_2, t_3, t_4, t_5 \leftarrow \mathbb{Z}_p$		
$\tilde{A} = A + t_1G + t_2\mathsf{PK}$		
$\tilde{B}_1 = \gamma B_1 + t_3G + t_4\tilde{C}$		
$\tilde{B}_2 = \gamma B_2 + t_5H + t_4\tilde{C}'$		
$\varepsilon = \mathcal{H}(\tilde{Z}, \tilde{C}, \tilde{A}, \tilde{B}_1, \tilde{B}_2, B_3)$		
$e = \varepsilon - t_2 - t_4$		
	$\xrightarrow{e}$	
		$c = e - c'$
		$r = u - c\mathsf{sk}$
	$\xleftarrow{c, r, c', r_1', r_2'}$	
$\tilde{c} = c + t_2; \tilde{c}' = c' + t_4$		
$\tilde{r} = r + t_1; r_3' = u_3' - \tilde{c}'\gamma$		
$\tilde{r}_1' = \gamma r_1' + t_3; \tilde{r}_2' = \gamma r_2' + t_5$		
$\sigma_1 = (\tilde{Z}, \tilde{C}, \tilde{r}, \tilde{c}, \tilde{r}_1', \tilde{r}_2', \tilde{c}', r_3')$		
$\sigma_2 = (d, \gamma)$		
return $\sigma := (\sigma_1, \sigma_2)$		

Figure 8. Protocol for obtaining blind signatures

Note that since BlindSign contains a ZKPoK of the message m, the set of messages that have been signed using the oracle for BlindSign is well defined, and thus EUF-CMA security is applicable. In the generic group model introduced by Shoup [34], the adversary operates on the group through a group oracle that allows computation of the group law, inversion and checking for equality. The adversary only receives random identifiers of group elements he either receives as input or computes through use of the oracles. See Appendix C.1 for the proof.

We additionally require that issuing and verification of signatures through BlindSign and BlindVerify is unlinkable. More precisely, the signer/verifier should learn nothing from the interactions that allows him to link the signature verified during BlindVerify to a particular interaction of BlindSign. Note that unlinkability only holds between signing and verifying, engaging the verifying protocol twice for the same signature is linkable.

Theorem 6.3 *The protocol* BlindSign *is perfectly blind if* Π_1 *is composable perfect zero-knowledge,* PC *is perfectly hiding and* $\mathcal{H}$ *is a random oracle. It additionally allows for extraction of the signed message if* Π_1 *has witness-extended emulation.*

Theorem 6.4 *The protocol* BlindVerify *is composable zero-knowledge if* Π_2 *is composable zero-knowledge,* PCB *is hiding, the DDH assumption holds in* $\mathbb{G}$ *and* $\mathcal{H}$ *is a random oracle. It additionally has witness-extended emulation if* Π_2 *has witness-extended emulation.*

For the proofs see Appendix C.2.

7 Performance Evaluation

We evaluate the performance of BBW by measuring execution times and network payload of a practical implementation. Since data transmission times largely depend on external factors and the instantiation scenario, we exclude them from our measurements and instead provide estimates based on NFC with its maximum transmission speed of 424 kbit/s.

Evaluation is done on a modern smartphone featuring a Snapdragon 845 (4×2.8 GHz & 4×1.77 GHz) and running Android 9. The implementation was done in C++17 and employs the RELIC toolkit v0.5.0, an open source library with support for elliptic curve arithmetic [2], for underlying group operations.

In order to compare the performance of BBW with BBA+ we also implemented BBA+ with support for 16-bit range proofs as described in [20] and measured execution times on the same platform.

Note that a modern smartphone represents a particularly strong device in embedded computing. To obtain performance estimations for smart cards (or other low-end micro controllers), we count the number of multiplications done and estimate the execution time based on the performance of a dedicated ECC coprocessor [35] and the MultOS card. Note that this coprocessor was designed for fast pairing execution and was thus optimized for Barreto-Naehrig (BN) curves. Although the authors also supplied performance estimates for Curve25519, their processor does not make use of the special form of Curve25519 that would allow for faster arithmetic. A dedicated coprocessor for either Curve25519 or Four$\mathbb{Q}$ curves [13] should result in much faster implementations, especially as Four$\mathbb{Q}$ curves can achieve significantly higher performance than Curve25519 [27].

7.1 Implementation Details

To obtain small group elements (and thus minimize network payload) and fast computation times, we chose to use the elliptic curve Curve25519 ([26, 6, 7]) which provides 128 bit of security. For the range proofs in the Sub protocol we employ Bulletproofs by Bünz et al. [8]. The authors first construct a protocol with linear data transmission size and then extend it to provide logarithmic data size at the cost of additional prover computation time. We implemented both variants for both 16 and 32 bit ranges, referring to the variant with linear data size as $\mathsf{Sub}_{l,\mathrm{lin}}$ and to the logarithmic variant as $\mathsf{Sub}_{l,\mathrm{log}}$ in the following, where l denotes the size of the range. We suggest to use 16 bit on weak devices and 32 bit on smartphones (if needed).

7.2 Evaluation

Table 1 shows our measurement results. The evaluation was performed utilizing a single CPU core and results are averaged over 1000 individual executions.

While Issue and Add are highly efficient with approximately 50 ms and 60 ms respectively, the performance of Sub varies with different parameters and algorithm choices. As expected, there is a runtime-communication tradeoff when employing the logarithmic range proofs: the amount of data that has to be transmitted in total is reduced (5039 B to 3607 B for $l = 32$) while the computational complexity is increased tremendously (170 ms to 870 ms for $l = 32$). Note that, using NFC with 424 kbit/s, 3607 B are transmitted in approx. 70 ms and 5039 B are transmitted in approx. 95 ms. Even when using NFC at only 106 kbit/s (e.g. when using smart cards), the difference amounts to only about 110 ms, which is still faster than the additional computation.

Using the number of multiplications in $\mathbb{G}$, we can estimate a lower bound for the execution time on other platforms. Note that we restrict our estimations to the user side, as the backend is expected to be notably more powerful. Furthermore, as these estimations are solely based on the number of point multiplications, execution times of concrete instantiations would be higher due to the remaining computations and I/O. The dedicated coprocessor by Unterluggauer and Wenger [35] takes 23 ms for a multiplication on Curve25519. On a MultOS 4.3.1 card, point multiplication on $\approx$250 bit curves takes 61 ms [17]. The resulting estimated lower bounds for execution time of BBW are shown in Table 1. It is worth noting that, although the MultOS card does not employ a dedicated coprocessor, the execution time is only increased by a factor of less than 3.

7.3 Discussion

Our evaluation shows that, although the logarithmic variant of Sub results in less transmitted data, the overhead in execution time is tremendous. While this may not be an issue on the smartphone, as 1 s for 32 bit ranges might be an acceptable waiting period for users, more than 5.8 s or even 15.4 s for just 16-bit ranges as in our estimations for smart cards is unacceptable. Therefore, we favor the linear variant, as the penalty of transmitting the additional data is easily outweighed by the processing time saved. Regarding the smartphone, this reduces the computation time of the user to only 170 ms for $\mathsf{Sub}_{32,\mathrm{lin}}$. On more constrained devices, using the linear variant and $l = 16$ results in promising execution times, i.e. close to 1.5 s for Sub on [35], which may be acceptable depending on the scenario. Note that Sub without range proofs takes approximately as long as Add, and thus less than 1 s on [35].

Comparing our implementation with the performance of BBA+, we see improvement in all areas. Regarding transmitted data, the Issue protocols are roughly equal, while for Add the transmitted data was reduced from 4208 B to 1745 B. The most notable difference is in Sub, where the amount of transmitted data was reduced from 14368 B when supporting only a 16-bit range proof, to 5039 B for $\mathsf{Sub}_{32,\mathrm{lin}}$, i.e., supporting a 32-bit range proof.

	Execution Time [ms]				Transmitted Data [B]				Smart card estimations		
	User		System		User	System	Combined		#multiplications	Estimated Runtime [ms]	
Algorithm	BBW	BBA+	BBW	BBA+	BBW	BBW	BBW	BBA+	in $\mathbb{G}$	[35]	[17]
Issue	52	102	20	70	625	380	1005	1024	40	920	2440
Add	62	284	45	262	1330	415	1745	4208	30	690	1830
$\mathsf{Sub}_{16,\mathrm{lin}}$	122	707	182	605	2876	1045	3921	14368	68	1564	4148
$\mathsf{Sub}_{16,\mathrm{log}}$	475	-	234	-	2143	1359	3502	-	253	5819	15433
$\mathsf{Sub}_{32,\mathrm{lin}}$	170	-	302	-	3994	1045	5039	-			
$\mathsf{Sub}_{32,\mathrm{log}}$	870	-	389	-	2213	1394	3607	-			

Table 1. Performance measurements of BBW and BBA+ on our smartphone platform and lower-bound estimations for smart cards

Regarding execution time and the favorable linear variant of Sub, BBW is faster in every protocol. Especially $\mathsf{Sub}_{32,\mathrm{lin}}$ is more than four times as fast than Sub in BBA+ despite supporting 32-bit range proofs, and still twice as fast as Sub without any range proofs (where execution time is the same as in Add). Additionally, our scheme also offers a significantly faster system-side, enabling the use of weaker hardware in operator terminals (such as subway turnstiles).

Regarding practical implementations on smart cards, possible problems might arise from the amount of RAM required (which we did not optimize for on our smartphone implementation) and bad connectivity using NFC in passive mode. As almost all multiplications are fixed-point, storing pre-computation tables would allow for more efficient algorithms. However, this requires such tables for every element in the common reference string, which requires somewhat large amounts of permanent storage.

Acknowledgements

We thank the anonymous reviewers for their helpful comments. This research received funding from the German Research Foundation (DFG) within grants RU 1664/3-1 and PA 587/10-1, the Helmholtz Security Research, program Supercomputing & Big Data, Data Security and the Competence Center for Applied Security Technology (KASTEL).

References

[1] Masayuki Abe. “A Secure Three-Move Blind Signature Scheme for Polynomially Many Signatures.” en. In: *Advances in Cryptology — EUROCRYPT 2001*. Ed. by Gerhard Goos et al. Vol. 2045. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 136–151. DOI: 10.1007/3-540-44987-6_9. URL: http://link.springer.com/10.1007/3-540-44987-6_9 (visited on 01/29/2019).

[2] D. F. Aranha and C. P. L. Gouvêa. *RELIC Is an Efficient LIbrary for Cryptography*. URL: https://github.com/relic-toolkit/relic.

[3] Foteini Baldimtsi and Anna Lysyanskaya. “Anonymous Credentials Light.” en. In: *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security - CCS '13*. Berlin, Germany: ACM Press, 2013, pp. 1087–1098. DOI: 10.1145/2508859.2516687.

[4] Foteini Baldimtsi et al. “Anonymous Transferable E-Cash.” en. In: *Public-Key Cryptography – PKC 2015*. Ed. by Jonathan Katz. Springer Berlin Heidelberg, 2015, pp. 101–124.

[5] Razvan Barbulescu and Sylvain Duquesne. “Updating Key Size Estimations for Pairings.” en. In: *Journal of Cryptology* (Jan. 2018). ISSN: 1432-1378. DOI: 10.1007/s00145-018-9280-5.

[6] Daniel J. Bernstein. “Curve25519: New Diffie-Hellman Speed Records.” en. In: *Public Key Cryptography - PKC 2006*. Ed. by Moti Yung et al. Springer Berlin Heidelberg, 2006, pp. 207–228.

[7] Daniel J. Bernstein et al. “High-Speed High-Security Signatures.” en. In: *Journal of Cryptographic Engineering* 2.2 (Sept. 2012), pp. 77–89. ISSN: 2190-8516. DOI: 10.1007/s13389-012-0027-1.

[8] B. Bünz et al. “Bulletproofs: Short Proofs for Confidential Transactions and More.” In: *2018 IEEE Symposium on Security and Privacy (SP)*. May 2018, pp. 315–334. DOI: 10.1109/SP.2018.00020.

[9] Jan Camenisch, Susan Hohenberger, and Anna Lysyanskaya. “Compact E-Cash.” en. In: *Advances in Cryptology – EUROCRYPT 2005*. Ed. by Ronald Cramer. Springer Berlin Heidelberg, 2005, pp. 302–321.

[10] Jan Camenisch and Anna Lysyanskaya. “A Signature Scheme with Efficient Protocols.” en. In: *Security in Communication Networks*. Ed. by Stelvio Cimato, Giuseppe Persiano, and Clemente Galdi. Springer Berlin Heidelberg, 2003, pp. 268–289.

[11] Jan Camenisch and Anna Lysyanskaya. “Signature Schemes and Anonymous Credentials from Bilinear Maps.” en. In: *Advances in Cryptology – CRYPTO 2004*. Ed. by Matt Franklin. Springer Berlin Heidelberg, 2004, pp. 56–72.

[12] Sébastien Canard and Aline Gouget. “Anonymity in Transferable E-Cash.” en. In: *Applied Cryptography and Network Security*. Ed. by Steven M. Bellovin et al. Springer Berlin Heidelberg, 2008, pp. 207–223.

[13] Craig Costello and Patrick Longa. “Four$$\mathbb {Q}$$: Four-Dimensional Decompositions on a $$\mathbb {Q}$$-Curve over the Mersenne Prime.” en. In: *Advances in Cryptology – ASIACRYPT 2015*. Ed. by Tetsu Iwata and Jung Hee Cheon. Springer Berlin Heidelberg, 2015, pp. 214–235.

[14] Ivan Damgård. *Concurrent Zero-Knowledge Is Easy in Practice*. Tech. rep. 014. 1999. URL: https://eprint.iacr.org/1999/014 (visited on 02/26/2019).

[15] *Dash - Dash Is Digital Cash You Can Spend Anywhere*. en-US. URL: https://www.dash.org/ (visited on 08/02/2019).

[16] T. Dimitriou. “Privacy-Respecting Rewards for Participatory Sensing Applications.” In: *2018 IEEE Wireless Communications and Networking Conference (WCNC)*. Apr. 2018, pp. 1–6. DOI: 10.1109/WCNC.2018.8377269.

[17] P. Dzurenda et al. “Performance Analysis and Comparison of Different Elliptic Curves on Smart Cards.” In: *2017 15th Annual Conference on Privacy, Security and Trust (PST)*. Aug. 2017, pp. 365–36509. DOI: 10.1109/PST.2017.00050.

[18] Jens Groth and Yuval Ishai. “Sub-Linear Zero-Knowledge Argument for Correctness of a Shuffle.” en. In: *Advances in Cryptology – EUROCRYPT 2008*. Ed. by Nigel Smart. Springer Berlin Heidelberg, 2008, pp. 379–396.

[19] Jens Groth and Amit Sahai. “Efficient Non-Interactive Proof Systems for Bilinear Groups.” In: *IACR Cryptology ePrint Archive* 2007 (Jan. 2007), p. 155. URL: http://eprint.iacr.org/2007/155.

[20] Gunnar Hartung et al. “BBA+: Improving the Security and Applicability of Privacy-Preserving Point Collection.” In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. New York, NY, USA: ACM, 2017, pp. 1925–1942. DOI: 10.1145/3133956.3134071.

[21] Gesine Hinterwälder, Felix Riek, and Christof Paar. “Efficient E-Cash with Attributes on MULTOS Smartcards.” en. In: *Radio Frequency Identification*. Ed. by Stefan Mangard and Patrick Schaumont. Springer International Publishing, 2015, pp. 141–155.

[22] Max Hoffmann et al. *P4TC—Provably-Secure yet Practical Privacy-Preserving Toll Collection*. Tech. rep. 1106. 2018. URL: https://eprint.iacr.org/2018/1106 (visited on 02/28/2019).

[23] Tibor Jager and Andy Rupp. “Black-Box Accumulation: Collecting Incentives in a Privacy-Preserving Way.” en. In: *Proceedings on Privacy Enhancing Technologies* 2016.3 (July 2016), pp. 62–82. DOI: 10.1515/popets-2016-0016.

[24] Věra Jourová. *Strengthened EU Rules to Prevent Money Laundering and Terrorism Financing*. en. July 2018. URL: https://ec.europa.eu/info/files/factsheet-main-changes-5th-anti-money-laundering-directive_en (visited on 04/11/2019).

[25] Taechan Kim and Razvan Barbulescu. “Extended Tower Number Field Sieve: A New Complexity for the Medium Prime Case.” en. In: *Advances in Cryptology – CRYPTO 2016*. Ed. by Matthew Robshaw and Jonathan Katz. Springer Berlin Heidelberg, 2016, pp. 543–571.

[26] A. Langley, M. Hamburg, and S. Turner. *Elliptic Curves for Security*. en. Tech. rep. RFC7748. RFC Editor, Jan. 2016. DOI: 10.17487/RFC7748. URL: https://www.rfc-editor.org/info/rfc7748 (visited on 02/18/2019).

[27] Zhe Liu et al. “Four$$\mathbb {Q}$$on Embedded Devices with Strong Countermeasures Against Side-Channel Attacks.” en. In: *Cryptographic Hardware and Embedded Systems – CHES 2017*. Ed. by Wieland Fischer and Naofumi Homma. Springer International Publishing, 2017, pp. 665–686.

[28] Ueli Maurer. “Zero-Knowledge Proofs of Knowledge for Group Homomorphisms.” en. In: *Designs, Codes and Cryptography* 77.2-3 (Dec. 2015), pp. 663–676. ISSN: 0925-1022, 1573-7586. DOI: 10.1007/s10623-015-0103-5.

[29] M. Milutinovic et al. “uCentive: An Efficient, Anonymous and Unlinkable Incentives Scheme.” In: *2015 IEEE Trustcom/BigDataSE/ISPA*. Vol. 1. Aug. 2015, pp. 588–595. DOI: 10.1109/Trustcom.2015.423.

[30] David Pointcheval and Olivier Sanders. “Short Randomizable Signatures.” en. In: *Topics in Cryptology - CT-RSA 2016*. Ed. by Kazue Sako. Springer International Publishing, 2016, pp. 111–126.

[31] Andy Rupp et al. “P4R: Privacy-Preserving Pre-Payments with Refunds for Transportation Systems.” en. In: *Financial Cryptography and Data Security*. Ed. by Ahmad-Reza Sadeghi. Springer Berlin Heidelberg, 2013, pp. 205–212.

[32] E. B. Sasson et al. “Zerocash: Decentralized Anonymous Payments from Bitcoin.” In: *2014 IEEE Symposium on Security and Privacy*. May 2014, pp. 459–474. DOI: 10.1109/SP.2014.36.

[33] J. T. Schwartz. “Fast Probabilistic Algorithms for Verification of Polynomial Identities.” In: *J. ACM* 27.4 (Oct. 1980), pp. 701–717. ISSN: 0004-5411. DOI: 10.1145/322217.322225.

[34] Victor Shoup. “Lower Bounds for Discrete Logarithms and Related Problems.” en. In: *Advances in Cryptology — EUROCRYPT ’97*. Ed. by Walter Fumy. Springer Berlin Heidelberg, 1997, pp. 256–266.

[35] Thomas Unterluggauer and Erich Wenger. “Efficient Pairings and ECC for Embedded Systems.” en. In: *Cryptographic Hardware and Embedded Systems – CHES 2014*. Ed. by Lejla Batina and Matthew Robshaw. Springer Berlin Heidelberg, 2014, pp. 298–315.

[36] *Zcash – Privacy-Protecting Digital Currency*. en-US. URL: https : / / z . cash/ (visited on 08/02/2019).

A Formal Instantiations of Building Blocks

A.1 Commitment Scheme

Definition A.1 (Non-interactive commitment scheme) *The triple* (Setup, Com, Open) *is called a non-interactive commitment scheme for message space $\mathcal{M}$ with group setup* SetupGrp *if the following holds:*

Completeness: *For all* $gp \leftarrow \mathsf{SetupGrp}(1^k)$, *all* $crs \leftarrow \mathsf{Setup}(gp)$ *and all* $m \in \mathcal{M}$ *and* $(C, d) \leftarrow \mathsf{Com}(crs, m)$ *it holds that* $\mathsf{Open}(crs, m, C, d) = 1$

Perfectly Hiding: *For all (unbounded) adversaries $\mathcal{A}$*

$$\Pr\left[\begin{array}{l} gp \leftarrow \mathsf{SetupGrp}(1^k);\, crs \leftarrow \mathsf{Setup}(gp); \\ (m_0, m_1) \leftarrow \mathcal{A}(crs);\, b \leftarrow \{0,1\} \\ (C, d) \leftarrow \mathsf{Com}(crs, m_b) \colon \mathcal{A}(crs, C) = b \end{array}\right] = \frac{1}{2}$$

Binding: *For all PPT adversaries $\mathcal{A}$*

$$\Pr\left[\begin{array}{l} gp \leftarrow \mathsf{SetupGrp}(1^k);\, crs \leftarrow \mathsf{Setup}(gp) \\ (C, m, d, m', d') \leftarrow \mathcal{A}(crs)\colon \\ m \neq m' \wedge \mathsf{Open}(crs, m, C, d) = 1 \\ \wedge\ \mathsf{Open}(crs, m', C, d') = 1 \end{array}\right] \approx 0$$

The commitment scheme is called ***additively homomorphic*** *if the message space $\mathcal{M}$ is an additive group and it has an algorithm for adding commitments such that the resulting commitment can be opened to the sum of the messages.*
It is called ***equivocable****, if knowledge of a trapdoor allows opening of commitments to arbitrary messages and existence of the trapdoor cannot be detected.*

We use Pedersen multi-commitments:

Definition A.2 (Pedersen Multicommitments) *The Pedersen multi-commitment scheme is given by:*

$\mathsf{Setup}(gp, l)$ *samples* $x_1, \ldots, x_l \leftarrow \mathbb{Z}_p^*$ *and returns* $crs := (H, H_1, \ldots, H_l) := (G_1, x_1G_1, \ldots, x_lG_1)$.

$\mathsf{SimSetup}(gp, l)$ *works as* $\mathsf{Setup}(gp, l)$, *but sets* $td := (x_1, \ldots, x_l)$ *as trapdoor, and returns* (crs, td).

$\mathsf{Com}(crs, m)$ *where* $m = (m_1, \ldots, m_l)$ *samples* $d \leftarrow \mathbb{Z}_p^*$ *and returns* $(C, d) := (dH + \sum_{i=1}^{l} m_iH_i, d)$.

$\mathsf{SimCom}(crs)$ *returns* $(C, d) := (dH, d)$, *where* $d \leftarrow \mathbb{Z}_p^*$.

$\mathsf{Open}(crs, m, C, d)$ *where* $m = (m_1, \ldots, m_l)$ *returns* 1 *if* $dH + \sum_{i=1}^{l} m_iH_i = C$ *and* 0 *otherwise.*

$\mathsf{CAdd}(C_1, C_2)$ *returns* $C_1 + C_2$.

$\mathsf{DAdd}(d_1, d_2)$ *returns* $d_1 + d_2$.

$\mathsf{Equiv}(C, d, m, td)$ *returns* $d' = d - \sum_{i=1}^{l} m_lx_l$ *where* $m = (m_1, \ldots, m_l)$ *and* $td = (x_1, \ldots, x_l)$.

Definition A.3 (Blinded Pedersen Commitment) *For the blinded Pedersen commitment,* Setup *adds another element* $Z \leftarrow \mathbb{G} \setminus \{0\}$ *to the common reference string, and* Com *calculates the commitment* $(\gamma Z, \gamma C)$ *as described above. The unveil information consists of* (d, γ).

Theorem A.4 *If the DLOG assumption holds with regard to* SetupGrp*, the Pedersen multi-commitment scheme is an equivocable, additively homomorphic non-interactive commitment scheme and the blinded variant is an equivocable non-interactive commitment scheme.*

A.2 Signatures

Definition A.5 (Signature Scheme) *A signature scheme (with setup* SetupGrp *and message space $\mathcal{M}$) is a triple* (KeyGen, Sign, Verify) *which satisfies correctness:* $\forall gp \leftarrow \mathsf{SetupGrp}(1^n), \forall (\mathsf{PK}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(gp)$, $\forall m \in \mathcal{M}$: $\Pr[\mathsf{Verify}(\mathsf{PK}, m, \mathsf{Sign}(\mathsf{sk}, m)) = 1] = 1$
It is called EUF-CMA *secure if for all PPT adversaries $\mathcal{A}$ with access to a signing oracle $\mathcal{S}_{\mathsf{sk}}$ we have*

$$\Pr\left[\begin{array}{l} gp \leftarrow \mathsf{SetupGrp}(1^n);\, (\mathsf{PK}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(gp); \\ (m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{S}_{\mathsf{sk}}(\cdot)}(\mathsf{PK})\colon \\ m^* \notin Q_{\mathcal{A}}^{\mathcal{S}_{\mathsf{sk}}} \wedge \mathsf{Verify}(\mathsf{PK}, m^*, \sigma^*) = 1 \end{array}\right] \approx 0$$

where $Q_{\mathcal{A}}^{\mathcal{S}_{\mathsf{sk}}}$ *is the set of all messages m for which $\mathcal{A}$ did an oracle query.*

A.3 Zero-Knowledge

Definition A.6 (ZKPoK) *For a polynomial time decidable ternary relation R we define the group-dependent language L_{gp} as the set of x for which there exists w with $(gp, x, w) \in R$. We call w a witness for x.*
An interactive zero-knowledge proof of knowledge consists of PPT algorithms Setup, SimSetup, ExtSetup *and*

interactive PPT algorithms $(\mathcal{P}, \mathcal{V})$ called the prover and the verifier. By $tr = \langle \mathcal{P}(x), \mathcal{V}(y) \rangle$ we denote the public transcript produced by the interaction between $\mathcal{P}$ and $\mathcal{V}$, ending with $\mathcal{V}$ either accepting or rejecting.
Moreover, we require the following properties:

Perfect completeness: *For all $gp \leftarrow \mathsf{SetupGrp}(1^n)$, $crs \leftarrow \mathsf{Setup}(gp)$, $(x, w) \in L_{gp}$: $\langle \mathcal{P}(x, w), \mathcal{V}(x) \rangle = 1$.*

Composable Zero-Knowledge: *There exists a simulation trapdoor and a PPT algorithm called the simulator that, when given the simulation trapdoor, can simulate the prover without access to a witness for any PPT verifier $\mathcal{V}^*$ when given black-box access to $\mathcal{V}^*$ in a way that even an adversary that generates the statement to be proven and is also given the simulation trapdoor has only negligible advantage in deciding whether transcripts are produced by the simulator or an actual interaction between $\mathcal{P}$ and $\mathcal{V}^*$. Additionally, existence of the trapdoor must be undetectable from the crs.*

Witness-Extended Emulation *There exists an expected PPT extractor $\mathcal{E}$ that when given access to a transcript oracle of the interaction of a prover $\mathcal{P}^*$ with the verifier $\mathcal{V}$ produces a transcript indistinguishable from one obtained from a real interaction between $\mathcal{P}^*$ and $\mathcal{V}$, and if the transcript is accepting also outputs a valid witness w. Note that this also implies soundness.*

We extend the generic sigma protocol introduced by Maurer [28] with a Blum coin-toss to achieve concurrent zero-knowledge:

Definition A.7 (Modified Zero-Knowledge Protocol) *Our modified ZK scheme is given by:*

- $\mathsf{Setup}(gp)$ *generates $crs \leftarrow \mathsf{C}_{ZK}.\mathsf{Setup}(gp)$ and outputs $crs_{\mathsf{pok}} = crs$.*
- $\mathsf{SimSetup}(gp)$ *generates $(crs, td) \leftarrow \mathsf{C}_{ZK}.\mathsf{SimSetup}(gp)$ and outputs $(crs_{\mathsf{pok}}, td_{\mathsf{pok}}) = (crs, td)$.*
- *The interactive ZKPoK for x: $\Phi(x) = Z$ is performed roughly as follows:*

  $\mathcal{P}$ picks a random $k \leftarrow \mathbb{A}$, computes $T = \Phi(k)$ and sends T together with a commitment on his half of the challenge to $\mathcal{V}$
  $\mathcal{V}$ randomly picks his half of the challenge and sends it to $\mathcal{P}$
  $\mathcal{P}$ computes the challenge c from the two halves, sets $r = k + cx$ and sends r together with the opening of his challenge half to $\mathcal{V}$
  $\mathcal{V}$ computes c and checks that $\Phi(r) = T + cZ$ holds.

Theorem A.8 *The protocol in Definition A.7 is an argument for the relation R_{gp} with $(gp, Z, x) \in R \Leftrightarrow \Phi(x) = Z$. It has composable perfect zero-knowledge if C_{ZK} is equivocable and witness-extended emulation if C_{ZK} is binding.*

Composable perfect zero-knowledge follows from equivocability of C_{ZK} (see [14]), the proof for witness-extended emulation is straightforward.

Definition A.9 (Bulletproof Range Proof) *The Bulletproof range proof scheme consists of an PPT algorithm* Setup *and an interactive protocol* $\mathsf{RangeProof}$ *between a prover $\mathcal{P}$ and a verifier $\mathcal{V}$.*

$\mathsf{Setup}(gp, l)$ *generates a common reference string crs_{rp} for the range $[0, 2^l]$.*

$\mathsf{RangeProof}$ *The prover $\mathcal{P}$ with inputs $(crs_{\mathsf{rp}}, v, r, C)$ convinces the verifier $\mathcal{V}$ with inputs (crs_{rp}, C) that $C = \mathsf{Com}(v; r)$ and that $v \in [0, 2^l - 1]$.*

For details on how the interactive proof is performed, see [8].

Theorem A.10 *The range proof in Definition A.9 has perfect completeness, perfect special honest verifier zero-knowledge and witness extended emulation. If the challenge is replaced by a Blum coin toss, it has perfect composable zero-knowledge.*

For the proof see [8, Cor. 2] and [14] for composable zero-knowledge.

B Formal Definitions of BBA+

Definition B.1 (Owner-Binding) *A trapdoor-linkable* BBA+ *scheme is called owner-binding if for any PPT adversary $\mathcal{A}$ in the experiments $\mathsf{Exp}^{\textsf{ob-issue}}_{\mathsf{BBA+},\mathcal{A}}(n)$ and $\mathsf{Exp}^{\textsf{ob-acc-ver}}_{\mathsf{BBA+},\mathcal{A}}(n)$ from Figure 9 the advantages of $\mathcal{A}$ defined by*

$$\mathsf{Adv}^{\textsf{ob-issue}}_{\mathsf{BBA+},\mathcal{A}}(n) := \Pr\Big[\mathsf{Exp}^{\textsf{ob-issue}}_{\mathsf{BBA+},\mathcal{A}}(n) = 1\Big]$$
$$\mathsf{Adv}^{\textsf{ob-acc-ver}}_{\mathsf{BBA+},\mathcal{A}}(n) := \Pr\Big[\mathsf{Exp}^{\textsf{ob-acc-ver}}_{\mathsf{BBA+},\mathcal{A}}(n) = 1\Big]$$

are negligible in n.

Experiment $\mathsf{Exp}^{\textsf{ob-issue}}_{\mathsf{BBA+},\mathcal{A}}(n)$
$(crs, td) \leftarrow \mathsf{Setup}(1^n)$
$(\mathsf{PK}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}) \leftarrow \mathsf{IGen}(crs)$, $(\mathsf{PK}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}) \leftarrow \mathsf{UGen}(crs)$
$b \leftarrow \mathcal{A}^{\mathsf{MalIssue},\mathsf{MalAcc},\mathsf{MalVer}}(crs, \mathsf{PK}_{\mathcal{I}}, \mathsf{PK}_{\mathcal{U}})$
The experiment returns 1 iff $\mathcal{A}$ made a successful call to $\mathsf{MalIssue}(\mathsf{PK}_{\mathcal{U}})$.

Experiment $\mathsf{Exp}^{\textsf{ob-acc-ver}}_{\mathsf{BBA+},\mathcal{A}}(n)$
$(crs, td) \leftarrow \mathsf{Setup}(1^n)$, $(\mathsf{PK}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}) \leftarrow \mathsf{IGen}(crs)$
$b \leftarrow \mathcal{A}^{\mathsf{MalIssue},\mathsf{MalAcc},\mathsf{MalVer}}(\mathsf{PK}_{\mathcal{I}})$
The experiment returns 1 iff $\mathcal{A}$ made a successful call to MalAcc or MalVer such that $\mathsf{ExtractUID}$ applied to that call outputs a public key $\mathsf{PK}_{\mathcal{U}}$ for which $\mathsf{MalIssue}$ has never been called before.

Figure 9. Owner-binding experiments for Issue and Add/Sub

Definition B.2 (Double-Spending Detection) *A trapdoor-linkable* BBA+ *scheme ensures double-spending detection if for any PPT adversary $\mathcal{A}$ in the experiment*

$\mathsf{Exp}^{dsd}_{\mathsf{BBA+},\mathcal{A}}(n)$ *from* Figure 10 *the advantage of* $\mathcal{A}$ *defined by*

$$\mathsf{Adv}^{dsd}_{\mathsf{BBA+},\mathcal{A}}(n) := \Pr\Big[\mathsf{Exp}^{dsd}_{\mathsf{BBA+},\mathcal{A}}(n) = 1\Big] \qquad (5)$$

is negligible in n.

Experiment $\mathsf{Exp}^{\mathsf{dsd}}_{\mathsf{BBA+},\mathcal{A}}(n)$

$(crs, td) \leftarrow \mathsf{Setup}(1^n)$, $(\mathsf{PK}_\mathcal{I}, \mathsf{sk}_\mathcal{I}) \leftarrow \mathsf{IGen}(crs)$
$b \leftarrow \mathcal{A}^{\mathsf{MalIssue},\mathsf{MalAcc},\mathsf{MalVer}}(\mathsf{PK}_\mathcal{I})$
The experiment returns 1 iff $\mathcal{A}$ did two successful $\mathsf{MalAcc}/\mathsf{MalVer}$ calls resulting in two double-spending tags $dstag_1 = (s, z_1)$ and $dstag_2 = (s, z_2)$ 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
- $\mathsf{IdentDS}(\mathsf{PK}_\mathcal{I}, dstag_1, dstag_2) \neq (\mathsf{PK}^{(1)}_\mathcal{U}, \pi)$ or
- $\mathsf{IdentDS}(\mathsf{PK}_\mathcal{I}, dstag_1, dstag_2) = (\mathsf{PK}^{(1)}_\mathcal{U}, \pi)$ but $\mathsf{VerifyGuilt}(\mathsf{PK}_\mathcal{I}, \mathsf{PK}^{(1)}_\mathcal{U}, \pi) = 0$

Figure 10. Double-spending detection experiment.

Definition B.3 (False-Accusation Protection) *A trapdoor-linkable* BBA+ *scheme ensures false-accusation protection if for any PPT adversary* $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ *in the experiment* $\mathsf{Exp}^{facp}_{\mathsf{BBA+},\mathcal{A}}(n)$ *from* Figure 11 *the advantage of* $\mathcal{A}$ *defined by*

$$\mathsf{Adv}^{facp}_{\mathsf{BBA+},\mathcal{A}}(n) := \Pr[\mathsf{Exp}^{facp}_{\mathsf{BBA+},\mathcal{A}}(n) = 1] \qquad (6)$$

is negligible in n.

Experiment $\mathsf{Exp}^{\mathsf{facp}}_{\mathsf{BBA+},\mathcal{A}}(n)$

$(crs, td) \leftarrow \mathsf{Setup}(1^n)$
$(\mathsf{PK}_\mathcal{I}, \mathsf{sk}_\mathcal{I}) \leftarrow \mathcal{A}_0(crs)$, $(\mathsf{PK}_\mathcal{U}, \mathsf{sk}_\mathcal{U}) \leftarrow \mathsf{UGen}(crs)$
$\pi \leftarrow \mathcal{A}_1^{\mathsf{RealHonIssue},\mathsf{RealHonAdd},\mathsf{RealHonSub}}(\mathsf{PK}_\mathcal{I}, \mathsf{PK}_\mathcal{U})$
The experiment returns 1 iff $\mathsf{VerifyGuilt}(\mathsf{PK}_\mathcal{I}, \mathsf{PK}_\mathcal{U}, \pi) = 1$.

Figure 11. False accusation protection experiment

C Proofs for the Signature Scheme

C.1 Proof of EUF-CMA Security

We make use of the following Lemma:

Lemma C.1 (Schwartz [33]) *Let* p *be prime and let* $t \geq 1$. *Let* $F(X_1, \ldots, X_k) \in \mathbb{Z}_{p^t}[X_1, \ldots, X_k]$ *be a nonzero polynomial of total degree* d. *Then for random* $x_1, \ldots, x_k \in \mathbb{Z}_{p^t}$, *the probability that* $F(x_1, \ldots, x_k) = 0$ *is at most* $\frac{d}{p}$.

Proof (Proof of Theorem 6.2)
First note that because the Pedersen commitment scheme is binding and the user performed a proof of knowledge of m, *the message being signed during* BlindSign *is well-defined.*[1]

[1] And since the adversary had to compute the commitment, the message m can be obtained from the queries to the group oracle.

In the generic group model, the adversary is only allowed black-box access to the group. More precisely, group elements are identified by uniformly random chosen handles from $\{0,1\}^n$. *The adversary receives handles for the group elements in his input and is given access to oracles to compute the group law and inversion that take identifiers and return an identifier for the result.*

As a first step, we modify the group oracle as follows: The oracle keeps a list of the handles (random bitstrings) the adversary receives and the internal representation of that group element. Internally, group elements are represented by polynomials $F_i \in \mathbb{Z}_p[X_1, \ldots, X_k]$. *The generator* G *is represented by the constant polynomial* 1 *and the public key* PK *is represented by the constant polynomial* sk *(where* $\mathsf{sk}G = \mathsf{PK}$*). For each element* $H, Z, H_0, \ldots, H_l$ *of the common reference string a new indeterminate* $X_H, X_Z, X_{H_0}, \ldots X_{H_l}$ *is introduced and the element is represented by the polynomial* X_i *for the correspondent indeterminate. When the adversary queries the group law oracle on two elements* F_i, F_j *the resulting group element is represented by* $F_k = F_i + F_j$, *and when he queries the group inversion oracle on* F_i, *the resulting group element is represented by* $F_k = -F_i$. *If* F_k *is already in the list, the corresponding handle is returned, otherwise a new one is chosen at random. After the adversary finished and outputted his solution, a random* $x := (x_1, \ldots, x_k) \in \mathbb{Z}_p^k$ *is chosen and all polynomials are evaluated at* x. *During interaction, whenever a random group element is chosen (including cases where a random scalar* s *is chosen and the group element is computed by* sY *for some group element* Y*) a new indeterminate* X_i *is introduced and the element is represented as above. Whenever non-random group elements are computed, the oracle addition/inversion is used.*

More precisely, during the BlindSign *protocol, the adversary receives group elements* $A_i, B_{1,i}, B_{2,i}$. *The* A_i *are again represented by a polynomial in a newly introduced indeterminate* X_i, *while* $B_{1,i}, B_{2,i}$ *are represented by polynomials in existing indeterminates (and depend on the group element* C_i *the adversary supplied).*

If two distinct polynomials F_i, F_j *evaluate to the same group element, a simulation failure occurs and we abort, otherwise this is indistinguishable to the real experiment. We call this event* $\mathcal{F}_\mathbb{G}$. *Note that for an adversary performing* $q_\mathbb{G}$ *queries to the group oracle and engaging in* q_{sign} *signing protocols, there exist at most* $5 + l + q_\mathbb{G} + 3q_{sign} \approx q_\mathbb{G} + 3q_{sign}$ *polynomials* F_i *in at most* $3 + l + q_{sign} \approx q_{sign}$ *indeterminates* X_j. *Note also that the polynomials* F_i *are of degree either* 0 *or* 1, *as for* $F_k = F_i + F_j$ *it holds that the degree of* F_k *is at most the larger of the degrees of* F_i *and* F_j, *and polynomials inserted otherwise are of degree* 0 *or* 1. *Then it follows from* Lemma C.1 *that the probability of two polynomials* F_i *and* F_j *in* q_{sign} *indeterminates to be equal on ran-*

dom $x \in \mathbb{Z}_p^{q_{sign}}$ is at most $\frac{1}{p}$, and as there are at most $\binom{q_{\mathbb{G}}+3q_{sign}}{2}$ pairs of polynomials, we have

$$\Pr[\mathcal{F}_{\mathbb{G}}] \approx \binom{q_{\mathbb{G}} + 3q_{sign}}{2}\frac{1}{p} \tag{7}$$

Then observe that an adversary has two options to come up with a valid signature:

1. *For a given signature on message m obtained through interaction with the signing oracle, open the commitment $(\tilde{Z}, \tilde{C})$ part of the signature to a different message m'*
2. *Generate a fresh signature by computing some group elements via the group oracle and some non-group data that arbitrarily depends on the responses to the group and random oracles*

For simplicity, we first assume that no hash collisions occur, i.e. there are no queries q_1, q_2 to the random oracle $\mathcal{H}$ so that $\mathcal{H}(q_1) = \mathcal{H}(q_2)$. Should two such queries occur, we abort. We call this event $\mathcal{F}_{\mathcal{H}}$ (hash collision). Since $\mathcal{H}$ is a random oracle, for an adversary performing $q_{\mathcal{H}}$ queries to $\mathcal{H}$, we have

$$\Pr[\mathcal{F}_{\mathcal{H}}] \approx \binom{q_{\mathcal{H}}}{2}\frac{1}{p} \tag{8}$$

Regarding (1), note that $(\tilde{Z}, \tilde{C})$ uniquely defines C: For the adversary to be able to open the blinded commitment, he must supply some $\gamma \in \mathbb{Z}_p$ so that $\gamma Z = \tilde{Z}$ and $\gamma C = \tilde{C}$. Two different openings d and d' to different messages m and m' for C result in two different polynomials F and F' with coefficients m, d and m', d' that both evaluate to C. This constitutes a simulation failure as described above and is thus already contained in $\mathcal{F}_{\mathbb{G}}$.

For (2), the adversary needs to forge the ZKPoK, i.e. come up with group elements $\tilde{Z}, \tilde{C}$ and values $\tilde{c}, \tilde{r}, \tilde{c}', \tilde{r}'_1, \tilde{r}'_2, r'_3 \in \mathbb{Z}_p$ for which $\tilde{c} + \tilde{c}' = \mathcal{H}(\tilde{Z}, \tilde{C}, \tilde{A}, \tilde{B}_1, \tilde{B}_2, B_3)$ holds for $\tilde{A} = \tilde{r}G + \tilde{c}\mathsf{PK}$, $\tilde{B}_1 = \tilde{r}'_1 G + \tilde{c}'\tilde{C}$, $\tilde{B}_2 = \tilde{r}'_2 H + \tilde{c}'(\tilde{Z} - \tilde{C})$ and $B_3 = r'_3 Z + \tilde{c}'\tilde{Z}$.

We call the first part of the ZKPoK (knows sk such that $\mathsf{sk}G = \mathsf{PK}$, corresponding to $\tilde{c}$) the key side and the second part (knows (w_1, w_2, γ) such that $w_1 G = \tilde{C}, w_2 H = (\tilde{Z} - \tilde{C}), \gamma Z = \tilde{Z}$, corresponding to $\tilde{c}'$) the message side.

Now for the key side, observe that the adversary only receives non-group elements of the form $r_i = a_i + c_i\mathsf{sk}$ with new random a_i each time, and a_i and sk otherwise are only accessed through the group oracle, whose output is statistically independent of a_i and sk, and thus r_i is statistically independent of sk. Thus, the probability to output some $\tilde{A} \in \mathbb{G}$ and then for $\tilde{c}$ dependent on $\tilde{A}$ successfully compute $\tilde{r}$ for which it holds that $\tilde{A} = \tilde{r}G + \tilde{c}\mathsf{PK}$ is at most $\frac{q_{\mathbb{G}}}{2^{H(\tilde{c})}}$, where $H(\tilde{c})$ is the entropy of $\tilde{c}$.

For the message side, following equations must hold:

$$\tilde{B}_1 = \tilde{r}'_1 G + \tilde{c}'\tilde{C} \tag{9}$$

$$\tilde{B}_2 = \tilde{r}'_2 H + \tilde{c}'(\tilde{Z} - \tilde{C}) \tag{10}$$

$$B_3 = r'_3 Z + \tilde{c}'\tilde{Z} \tag{11}$$

Now, we view the group elements $\mathcal{A}$ uses as multivariate polynomials $F_i(X_1, \ldots, X_l)$ and rewrite the equations as

$$\tilde{r}'_1 = F_{\tilde{B}_1} - \tilde{c}' F_{\tilde{C}} \tag{12}$$

$$\tilde{r}'_2 X_H = F_{\tilde{B}_2} - \tilde{c}'(F_{\tilde{Z}} - F_{\tilde{C}}) \tag{13}$$

$$r'_3 X_Z = F_{B_3} - \tilde{c}' F_{\tilde{Z}} \tag{14}$$

So, after evaluating the polynomials at x chosen uniformly random as mentioned above, equation (12) holds if either $F_{\tilde{B}_1}$ and $F_{\tilde{C}}$ are both constant or only with probability $\frac{1}{2^{H(\tilde{c}')}}$, where $H(\tilde{c}')$ is the entropy of $\tilde{c}'$. Similarly, equation (14) holds if $F_{\tilde{Z}}$ and F_{B_3} are both of the form $r_i X_Z$ or again only with probability $\frac{1}{2^{H(\tilde{c}')}}$. For equation (13) to hold, both $F_{\tilde{B}_2}$ as well as $(F_{\tilde{Z}} - F_{\tilde{C}})$ need to be of the form $r_i X_H$, otherwise it again only holds with probability $\frac{1}{2^{H(\tilde{c}')}}$. But as $F_{\tilde{Z}} = r_i X_Z$ and $F_{\tilde{C}} = r_j$, it cannot hold that $(F_{\tilde{Z}} - F_{\tilde{C}}) = r_k X_H$. Thus, the probability that equations (12) to (14) are simultaneously satisfied is only $\frac{1}{2^{H(\tilde{c}')}}$, and the probability that both the key-side and the message-side are simultaneously satisfied is at most

$$\frac{q_{\mathbb{G}}}{2^{H(\tilde{c})}}\frac{q_{\mathbb{G}}}{2^{H(\tilde{c}')}} = \frac{q_{\mathbb{G}}^2}{2^{H(\tilde{c})+H(\tilde{c}')}}$$

. Since it has to hold that $\tilde{c}+\tilde{c}' = c$ and $H(c) \approx n$ since $\mathcal{H}$ is a random oracle and it holds that $H(c) = H(\tilde{c}+\tilde{c}') \leq H(\tilde{c}, \tilde{c}') \leq H(\tilde{c}) + H(\tilde{c}')$ we have that $H(\tilde{c}) + H(\tilde{c}') \geq n$ and thus the probability of (2) is at most

$$\frac{q_{\mathbb{G}}}{2^n} \tag{15}$$

So to conclude, the probability of an adversary to succeed in forging a signature is at most the sum of 1 and 2 and the failure events $\mathcal{F}_{\mathbb{G}}$ and $\mathcal{F}_{\mathcal{H}}$. So from (7), (8) and (15) we have

$$\mathsf{Adv}_{\mathcal{A}}^{\text{euf-cma}}(n) \leq \frac{q_{\mathbb{G}}^2}{2^n} + \binom{q_{\mathbb{G}} + 3q_{sign}}{2}\frac{1}{p} + \binom{q_{\mathcal{H}}}{2}\frac{1}{p}$$

which is negligible in n since $q_{\mathbb{G}}, q_{\mathcal{H}}$ and q_{sign} are polynomials in n, while $p \approx 2^n$. ∎

C.2 Proofs of Blindness/Zero-Knowledge

Proof (Proof of Theorem 6.3)
The only information that is sent outside the composable perfect zero-knowledge proof Π_1 is $C = \mathsf{PC.Com}(m; d)$ which is statistically independent of m as PC is perfectly hiding and e, which is statistically independent of m because $\mathcal{H}$ is a random oracle. Extraction of the signed message directly follows from witness-extended emulation of Pi_1. ∎

Zero-Knowledge Verification First note that because C is uniquely defined by $(\gamma Z, \gamma C)$ and C was sent to the signer, the protocol only achieves computational zero-knowledge (learning C during this interaction allows linking the signature to a signing interaction, thus we need to treat C as part of the witness). We follow the proof of blindness in [3].

Proof (Proof of Theorem 6.4)
The user sends the following information to the verifier: $\tilde{Z} = \gamma Z, \tilde{C} = \gamma C, \tilde{r} = r + t_1, \tilde{c} = c + t_2, \tilde{r}'_1 = \gamma r'_1 + t_3, \tilde{r}'_2 = \gamma r'_2 + t_5, \tilde{c}' = c' + t_4, r'_3 = u'_3 + c'\gamma$. The t_i have all been chosen uniformly at random by the user, thus $\tilde{r}, \tilde{r}'_1, \tilde{r}'_2, \tilde{c}, \tilde{c}'$ and r'_3 are all statistically independent of the values r, r'_1, r'_2, c, c' sent by the signer. Additionally, it holds that $\tilde{r}G + \tilde{c}X = \tilde{A}$, $\tilde{r}'_1 G + \tilde{c}'\tilde{C} = \tilde{B}_1$ and $\tilde{r}'_2 H + \tilde{c}'(\tilde{Z} - \tilde{C}) = \tilde{B}_2$, but $\tilde{A}, \tilde{B}_1$ and $\tilde{B}_2$ are also statistically independent of A, B_1 and B_2. Finally, $\tilde{c} + \tilde{c}' = \epsilon$ is statistically independent of $e = \epsilon - t_2 - t_4$. Thus, it remains to show that any adversary that can link $\tilde{Z}, \tilde{C}$ to C can be used to solve the DDH problem.

First, note that there are two types of signatures a simulator might output: $\sigma = (\tilde{Z}, \tilde{C}, \tilde{r}, \tilde{c}, \tilde{r}'_1, \tilde{r}'_2, \tilde{c}', r'_3)$ is correct *if there exists γ such that $\tilde{Z} = \gamma Z$ and $\tilde{C} = \gamma C_i$ for some interaction i with the signer and the signature verifies. The signature σ is* fake *if no such γ exists (but it still verifies). It is easy for the simulator to output a fake signature with control over the random oracle: it picks random $\tilde{Z}, \tilde{C} \leftarrow \mathbb{G}$, $\tilde{r}, \tilde{r}'_1, \tilde{r}'_2, \tilde{c}, \tilde{c}', r'_3 \leftarrow \mathbb{Z}_p$ and programs the random oracle so that the signature verifies.*

Now, observe that an adversary able to successfully distinguish between a correct *and a* fake *signature (with non-negligible advantage ϵ) can be used to solve the DDH problem. We construct a reduction $\mathcal{B}$ as follows:*

The reduction $\mathcal{B}$ gets as input a DDH instance (gp, A, B, D). $\mathcal{B}$ then sets $Z = A$ and $H = eG$ and $H_0, \dots, H_l$ to $e_0 A, \dots, e_l A$ for randomly chosen $e_i \in \mathbb{Z}_p$. Then $\mathcal{B}$ constructs the signature as $\tilde{Z} = D$ and $\tilde{C} = kD$ for random $k \leftarrow \mathbb{Z}_p$, chooses all other values at random and programs the random oracle as explained above. (Note that as $\mathcal{B}$ knows the discrete logarithms of $H_0, \dots, H_l$ and thus knows $k \in \mathbb{Z}_p$ for honestly generated commitments C so that $C = kA$.) If (gp, A, B, D) is a DH-tuple, then it holds that $A = aG, B = bG$ and $D = abG$, and thus $\tilde{Z} = D = bA = bZ$ and $\tilde{C} = kD = kbA = bC$ is identically distributed as a correct *signature. Conversely, if (gp, A, B, D) is not a DH-tuple, then $(\tilde{Z}, \tilde{C})$ is a* fake *signature. Thus, $\mathcal{B}$ outputs "DH" if $\mathcal{A}$ outputs "real" and "random" otherwise.* ■

D Proofs for BBW

D.1 Extraction

Proof (Proof of Theorem 5.1)
We consider an adversary $\mathcal{A}^{\mathsf{MalIssue},\mathsf{MalAdd},\mathsf{MalSub}}$ that stopped after interacting q times with the three oracles, where q is bounded by a polynomial in the security parameter.

For any successful interaction with one of the oracles (from the oracle's point of view) we extract a witness for the argument used in the interaction in the following way: We fix the randomness of $\mathcal{A}$ and that of all oracles. Then, starting with the interaction for which the ZKPoK was completed last, we iteratively extract witnesses one-by-one: For the interaction we want to extract, we supply the extractor $\mathcal{E}$ of the corresponding ZKPoK with a transcript oracle $\mathcal{O}$ of all messages part of the ZKPoK. Due to the witness-extended emulation property and since the interaction was successful and thus the first transcript produced is accepting, $\mathcal{E}$ outputs a witness with overwhelming probability in expected polynomial time.

We repeat this step for every successful oracle interaction, of which there are at most q. This results in an extraction algorithm $\mathcal{E}^*$ with expected runtime $q \cdot t$ where t is the expected runtime of $\mathcal{E}$. Since $\mathcal{E}$ has expected polynomial runtime and q is bounded by a polynomial, the resulting runtime is expected polynomial. As the probability for successful extraction is overwhelming for each interaction and the amount of interactions is bounded by q, the probability that all witnesses are successfully extracted is overwhelming. ■

D.2 Simulation-Linkability

We start with the requirement of P2, P3 and $\mathsf{BlindSign}$ being complete:

Lemma D.1 P2 *and* P3 *are complete.*

Lemma D.2 *The protocol* $\mathsf{BlindSign}$ *to obtain blind signatures is complete.*

Proof (Proof of Lemma D.1)
We prove Lemma D.1 for P2*, the proof for* P3 *works analog. A transcript for* P2 *is of the form $tr =: (\pi_1, c, \pi_2)$ where $\pi_1 =: (T, C_{\mathsf{Z}})$ and $\pi_2 = (c', d_{\mathsf{Z}}, r)$. Since tr is accepting, it holds for the group homomorphism $\Phi\colon \mathbb{A} \to \mathbb{B}$ used in* P2 *and the statement Z that $\Phi(r) = T + (c + c')Z$, as well as $\mathsf{C}_{ZK}.\mathsf{Open}(c', C_{\mathsf{Z}}, d_{\mathsf{Z}}) = 1$.*

c', d_{Z} and C_{Z} are part of the protocol, and the honest prover $\mathcal{P}$ can choose the same values. Since Z is in the image of Φ and Φ is a homomorphism, it follows that T is also in its image. Thus, there exists (at least one) $x \in \mathbb{A}$ for which it holds that $\Phi(x) = Z$. Then, for fixed c, c' and x, there exists $k \in \mathbb{A}$ such that $r = k + (c + c')x$ and $\Phi(k) = T$. Hence, the honest prover $\mathcal{P}$ with witness x and randomness k, c', d_{Z} interacting with the honest verifier $\mathcal{V}$ with randomness c produce the same transcript tr. ■

Proof (Proof of Lemma D.2)
A transcript for $\mathsf{BlindSign}$ *is of the form $tr =: (tr_{\mathsf{Z}}, (A, B_1, B_2), e, (c, r, c', r'_1, r'_2))$ where tr_{Z} is a transcript of the zero-knowledge proof of knowledge for the opening of the commitment C given as common input to the user and signer. With the same argument as in the proof of Lemma D.1 this zero-knowledge protocol is complete and there exist suitable inputs and random choices for the honest user $\mathcal{U}$ that lead to the same transcript tr_{Z}. All messages sent by the signer only depend on the common input C, his own random choices and the message e, so as long as there are inputs for the honest user*

that lead to the same value of e, the resulting interaction leads to the same transcript tr. e is calculated as $\epsilon - t_2 - t_4$, where ϵ is the output of $\mathcal{H}$ on values depending on t_2 and t_4, so there need to exist values for which $\mathcal{H}$ outputs $e + t_2 + t_4$. As $\mathbb{G}$ is cyclic with prime order, every group element is a generator. Thus, independent of the choices of t_2 and t_4, there exist values for t_1, t_3 and t_5 so that α, β_1 and β_2 can be any element from $\mathbb{G}$. Since $\mathcal{H}$ is a random oracle, for some fixed values of ζ, ζ_1 and μ, the output of $\mathcal{H}(\zeta, \zeta_1, \alpha, \beta_1, \beta_2, \mu)$ is still uniformly random, and thus there exist suitable choices for $t_1, \ldots, t_5$ so that $\mathcal{H}(\ldots) = e + t_2 + t_4$ (as there are p^3 possible values for $(\alpha, \beta_1, \beta_2)$ but only p possible values for $\mathcal{H}(\ldots)$). ■

Now we can show that our scheme is indeed simulation-linkable:

Proof (Proof of Theorem 5.2)
Completeness: *We need to show that any accepting transcript could be produced by interaction of an honest user $\mathcal{U}$ with an honest accumulator $\mathcal{AC}$/verifier $\mathcal{V}$ when given suitable inputs and randomness. More precisely, we need to show that for all values part of an accepting transcript, there exist inputs and random choices for $\mathcal{U}$ and $\mathcal{AC}/\mathcal{V}$ that result in these values being part of the transcript.*

A transcript for Add *has the form $tr =: (u_2, (s, t, C', \sigma_1,), tr_{P2}(C'', d'', s''), tr_{\mathsf{BlindSign}})$. For tr_{P2} by Lemma D.1, and for $tr_{\mathsf{BlindSign}}$ there exist suitable inputs and random choices by Lemma D.2.*

s is part of the token and chosen uniformly at random during honest token generation, so any value of s can be part of a token given as input to $\mathcal{U}$. t is calculated as $t = \mathsf{sk}_{\mathcal{U}} u_2 + u_1$, so for any user secret key $\mathsf{sk}_{\mathcal{U}}$ and any value u_2 sent by $\mathcal{AC}/\mathcal{V}$, there exists a value u_1 that results in t, which is also chosen uniformly at random during honest token generation and can thus be part of the input token. C' is a Pedersen commitment on some message m'. As Pedersen commitments are perfectly hiding, there exists a random choice d' for any message m' so that C' is a commitment on m' with opening information d'. $\sigma_1 =: (\zeta, \zeta_1, \rho, \omega, \rho'_1, \rho'_2, \omega', \mu)$ is part of a valid signature $\sigma := (\sigma_1, \sigma_2 := (d, \gamma))$. Since P2/P3 have witness-extended emulation (and are thus sound), we have that $\zeta = \gamma Z$ and $\zeta_1 = \gamma C$ for some commitment C (note that this fixes the choice of γ). Then, as above, there exists a random choice d for any message m so that C is a commitment on m with opening information d. Now, as the signature basically consists of a (non-interactive via Fiat-Shamir transformation) sigma-protocol zero-knowledge proof, with the same argument as in the proof of Lemma D.1 there exist suitable inputs and random choices for which σ_1 is produced.

Extractability: *Since* P2 *and* P3 *have witness-extended emulation, there exists an extractor $\mathcal{E}$ for them. Extraction for Σ-protocols is achieved by first getting enough related transcripts and then supplying them to an algorithm* $\mathsf{KnowledgeExtractor}$ *that outputs the witness. It is thus possible to construct* $\mathsf{GenerateTranscripts}$ *and* $\mathsf{KnowledgeExtractor}$ *from $\mathcal{E}$ and implement* $\mathsf{ExtractUID}$ *by running* $\mathsf{KnowledgeExtractor}$ *to receive a witness x containing the user secret key $\mathsf{sk}_{\mathcal{U}}$ and then computing and outputting $\mathsf{PK}_{\mathcal{U}} = \mathsf{sk}_{\mathcal{U}} G$.* ■

D.3 System Security

We first state the slightly modified formal definitions for security.

Definition D.3 (Oracles) *The adversary in the following security games is given access to the following oracles.*

$\mathsf{MalIssue}(\mathsf{PK}_{\mathcal{U}})$ *lets the adversary initiate the* Issue *protocol with an honest issuer $\mathcal{I}$ provided that there is no pending* $\mathsf{MalIssue}$ *call for $\mathsf{PK}_{\mathcal{U}}$ and $\mathsf{PK}_{\mathcal{U}}$ has also not been used in a successful call to* $\mathsf{MalIssue}$ *before.*

$\mathsf{MalAdd}(attr, v)$ *is used by the adversary to initiate the* Add *protocol with honest $\mathcal{AC}$ for input $v \in \mathbb{V}$.*

$\mathsf{MalSub}(attr, v)$ *is used by the adversary to initiate the* Sub *protocol with honest $\mathcal{V}$ for input $v \in \mathbb{V}$.*

Definition D.4 (Owner-Binding) *A simulation-linkable* BBW *scheme is called owner-binding if for any PPT adversary $\mathcal{A}$ in the experiments $\mathsf{Exp}^{\textit{ob-issue}}_{\mathsf{BBW},\mathcal{A}}(n)$ and $\mathsf{Exp}^{\textit{ob-add-sub}}_{\mathsf{BBW},\mathcal{A}}(n)$ from Figure 12 the advantages of $\mathcal{A}$ defined by*

$$\mathsf{Adv}^{\textit{ob-issue}}_{\mathsf{BBW},\mathcal{A}}(n) := \Pr\Big[\mathsf{Exp}^{\textit{ob-issue}}_{\mathsf{BBW},\mathcal{A}}(n) = 1\Big]$$

$$\mathsf{Adv}^{\textit{ob-add-sub}}_{\mathsf{BBW},\mathcal{A}}(n) := \Pr\Big[\mathsf{Exp}^{\textit{ob-add-sub}}_{\mathsf{BBW},\mathcal{A}}(n) = 1\Big]$$

are negligible in n.

Experiment $\mathsf{Exp}^{\text{ob-issue}}_{\mathsf{BBW},\mathcal{A}}(n)$

$crs \leftarrow \mathsf{Setup}(1^n)$
$(\mathsf{PK}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}) \leftarrow \mathsf{IGen}(crs)$, $(\mathsf{PK}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}) \leftarrow \mathsf{UGen}(crs)$
$b \leftarrow \mathcal{A}^{\mathsf{MalIssue},\mathsf{MalAdd},\mathsf{MalSub}}(crs, \mathsf{PK}_{\mathcal{I}}, \mathsf{PK}_{\mathcal{U}})$
The experiment returns 1 iff $\mathcal{A}$ made a successful call to $\mathsf{MalIssue}(\mathsf{PK}_{\mathcal{U}})$.

Experiment $\mathsf{Exp}^{\text{ob-add-sub}}_{\mathsf{BBW},\mathcal{A}}(n)$

$crs \leftarrow \mathsf{Setup}(1^n)$, $(\mathsf{PK}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}) \leftarrow \mathsf{IGen}(crs)$
$b \leftarrow \mathcal{A}^{\mathsf{MalIssue},\mathsf{MalAdd},\mathsf{MalSub}}(\mathsf{PK}_{\mathcal{I}})$
The experiment returns 1 iff $\mathcal{A}$ made a successful call to MalAdd or MalSub such that $\mathsf{ExtractUID}$ applied to that call outputs a public key $\mathsf{PK}_{\mathcal{U}}$ for which $\mathsf{MalIssue}$ has never been called before.

Figure 12. Owner-binding experiments for Issue and Add/Sub

Definition D.5 (Balance Binding) *A simulation-linkable* BBW *scheme is called balance-binding if for any PPT adversary $\mathcal{A}$ in the experiment $\mathsf{Exp}^{\textit{bb}}_{\mathsf{BBW},\mathcal{A}}(n)$ from Figure 13 the advantage of $\mathcal{A}$ defined by*

$$\mathsf{Adv}^{\textit{bb}}_{\mathsf{BBW},\mathcal{A}}(n) := \Pr\Big[\mathsf{Exp}^{\textit{bb}}_{\mathsf{BBW},\mathcal{A}}(n) = 1\Big] \tag{16}$$

is negligible in n.

Experiment $\mathsf{Exp}^{\mathsf{bb}}_{\mathsf{BBW},\mathcal{A}}$

$crs \leftarrow \mathsf{Setup}(1^n)$
$(\mathsf{PK}_\mathcal{I}, \mathsf{sk}_\mathcal{I}) \leftarrow \mathsf{IGen}(crs)$
$b \leftarrow \mathcal{A}^{\mathsf{MalIssue},\mathsf{MalAdd},\mathsf{MalSub}}(\mathsf{PK}_\mathcal{I})$

The experiment outputs 1 iff either of the following holds:

1. ExtractUID fails to extract a public key for any of the successful calls to MalAdd or MalSub
2. or $\mathcal{A}$ made a successful call to MalSub such that
   - all successful MalIssue/MalAdd/MalSub calls produce unique token version numbers
   - the amount of points $v^* \in \mathbb{V}$ subtracted exceeds the sum of previously collected points w for $\mathsf{PK}_\mathcal{U}$ (extracted from the call to MalSub), i.e.

$$v^* > w = \sum_{v \in V^{\mathsf{Add}}_{\mathsf{PK}_\mathcal{U}}} v - \sum_{v \in V^{\mathsf{Sub}}_{\mathsf{PK}_\mathcal{U}}} v,$$

where $V^{\mathsf{Add}}_{\mathsf{PK}_\mathcal{U}}$ is the list of all accumulation values $v \in \mathbb{V}$ that appeared in previous successful calls to MalAdd for which $\mathsf{PK}_\mathcal{U}$ has been extracted using ExtractUID, and $V^{\mathsf{Sub}}_{\mathsf{PK}_\mathcal{U}}$ for MalSub respectively.

Figure 13. Balance binding experiment.

Definition D.6 (Double-Spending Detection) *A simulation-linkable* BBW *scheme ensures double-spending detection if for any PPT adversary $\mathcal{A}$ in the experiment $\mathsf{Exp}^{dsd}_{\mathsf{BBW},\mathcal{A}}(n)$ from Figure 14 the advantage of $\mathcal{A}$ defined by*

$$\mathsf{Adv}^{dsd}_{\mathsf{BBW},\mathcal{A}}(n) := \Pr\Big[\mathsf{Exp}^{dsd}_{\mathsf{BBW},\mathcal{A}}(n) = 1\Big] \qquad (17)$$

is negligible in n.

Experiment $\mathsf{Exp}^{\mathsf{dsd}}_{\mathsf{BBW},\mathcal{A}}(n)$

$crs \leftarrow \mathsf{Setup}(1^n)$
$(\mathsf{PK}_\mathcal{I}, \mathsf{sk}_\mathcal{I}) \leftarrow \mathsf{IGen}(crs)$
$b \leftarrow \mathcal{A}^{\mathsf{MalIssue},\mathsf{MalAdd},\mathsf{MalSub}}(\mathsf{PK}_\mathcal{I})$

The experiment returns 1 iff $\mathcal{A}$ did two successful MalAdd/MalSub calls resulting in two double-spending tags $dstag_1 = (s, z_1)$ and $dstag_2 = (s, z_2)$ such that at least one of the following conditions is satisfied:

- ExtractUID fails to extract $\mathsf{PK}^{(1)}_\mathcal{U}$ or $\mathsf{PK}^{(2)}_\mathcal{U}$ for the respective calls or
- $\mathsf{PK}^{(1)}_\mathcal{U} \neq \mathsf{PK}^{(2)}_\mathcal{U}$ or
- $\mathsf{IdentDS}(\mathsf{PK}_\mathcal{I}, dstag_1, dstag_2) \neq (\mathsf{PK}^{(1)}_\mathcal{U}, \pi)$ or
- $\mathsf{IdentDS}(\mathsf{PK}_\mathcal{I}, dstag_1, dstag_2) = (\mathsf{PK}^{(1)}_\mathcal{U}, \pi)$ but $\mathsf{VerifyGuilt}(\mathsf{PK}_\mathcal{I}, \mathsf{PK}^{(1)}_\mathcal{U}, \pi) = 0$

Figure 14. Double-spending detection experiment.

We split the proof of Theorem 5.3 in separate proofs for owner-binding, double-spending detection and balance-binding.

Owner-Binding The proof for owner-binding wrt. Issue is a straight-forward reduction to the CDH problem.

Proof (Proof of owner-binding wrt. Add/Sub) *We proceed in a series of games.*

***Game 1** is the real experiment.*

*In **Game 2**, if* ExtractUID *fails for any call to* MalAdd *or* MalSub, *the experiment aborts and returns* 0. *We call this event failure event $\mathcal{F}_1$ (*ExtractUID *failed).*

*In **Game 3**, when $\mathcal{A}$ finished running, we use the extractor $\mathcal{E}^*$ from Theorem 5.1 to extract witnesses for all zero-knowledge proofs. More precisely, for each call to* MalIssue, *we extract a message $m' := (s', w = 0, \mathsf{sk}_\mathcal{U}, u_1)$ and store a record $(\mathsf{PK}_\mathcal{U}, m', m^* := (s' + s'', w = 0, \mathsf{sk}_\mathcal{U}, u_1))$, where $\mathsf{PK}_\mathcal{U}$ is the user public key for which* MalIssue *was called and m^* is the message for which $\mathcal{A}$ obtained a signature, with s'' being the issuers random share of the serial number. For each call to* MalAcc *or* MalVer *we extract a message $m' := (s', w, \mathsf{sk}_\mathcal{U}, u'_1)$ and a message $m := (s, w, \mathsf{sk}_\mathcal{U}, u_1)$ together with a valid signature σ and store a record $(\mathsf{PK}_\mathcal{U}, m', m, \sigma, m^* := (s' + s'', w + v, \mathsf{sk}_\mathcal{U}, u'_1))$, where $\mathsf{PK}_\mathcal{U}$ is the public key extracted by* ExtractUID *and m^* is the message for which $\mathcal{A}$ obtained a signature, with s'' being the issuers random share of the serial number and v the amount of points collected/redeemed. If extraction of any witness fails, the game aborts and returns* 0. *We call this event failure event $\mathcal{F}_2$ (extraction failure).*

*In **Game 4** we check for each extracted message m if there exists a previous call to* MalIssue, MalAcc *or* MalVer *for which our record contains m^* with $m^* = m$, i.e. if every message for which $\mathcal{A}$ proves knowledge of a signature has indeed been signed by the experiment. If this is not the case for any call, the experiment aborts and returns* 0. *We call this event failure event $\mathcal{F}_3$ (forged signature). Let $\mathsf{Adv}^{\mathrm{game-i}}_{\mathsf{BBW},\mathcal{A}}(n) = \Pr\Big[\mathsf{Exp}^{\mathrm{game-i}}_{\mathsf{BBW},\mathcal{A}}(n) = 1\Big]$ denote the advantage of $\mathcal{A}$ in **Game i**. Thus, by definition, $\mathsf{Adv}^{game\text{-}1}_{\mathsf{BBW},\mathcal{A}}(n) = \mathsf{Adv}^{ob\text{-}acc\text{-}ver}_{\mathsf{BBW},\mathcal{A}}(n)$.*
Since each game only differs from the previous one if the respective failure event occurred, we also have $\mathsf{Adv}^{game\text{-}4}_{\mathsf{BBW},\mathcal{A}}(n) = \mathsf{Adv}^{game\text{-}1}_{\mathsf{BBW},\mathcal{A}}(n) - \Pr\Big[\mathsf{Exp}^{\mathrm{game-1}}_{\mathsf{BBW},\mathcal{A}}(n) = 1 \wedge \Big(\bigvee_{i=1}^{3} \mathcal{F}_i\Big)\Big]$.
*Now lets consider the success probability of $\mathcal{A}$ in **Game 4**: As we already ensured that* ExtractUID *always succeeds, the only way for $\mathcal{A}$ to win is to make a call to* MalAcc *or* MalVer *which results in an extracted $\mathsf{PK}_\mathcal{U}$ for which* MalIssue *has not been called previously.*

So consider the first call to MalAcc *or* MalVer *where this happens. We have a record for this call containing a message $m := (s, w, \mathsf{sk}_\mathcal{U}, u_1)$ and a valid signature σ on m. As we ensured that m has been signed in a previous call, we have another call for which the record contains $m^* = m$ and a public key $\hat{\mathsf{PK}}_\mathcal{U}$. But due to the equations shown in the respective zero-knowledge proof (and since soundness for the proof holds when extraction was successful), we know that $\hat{\mathsf{PK}}_\mathcal{U} = \mathsf{PK}_\mathcal{U}$ must hold. Thus, if the call containing m^* was to* MalAcc *or* MalVer, *it is a previous call containing $\mathsf{PK}_\mathcal{U}$, which contradicts the assumption of considering the first such call. If it is a call to* MalIssue, *it contradicts the assumption that* MalIssue *has not been called for $\mathsf{PK}_\mathcal{U}$.*

Thus, it holds that $\mathcal{A}$ can not win this game and we have $\mathsf{Adv}^{game\text{-}4}_{\mathsf{BBW},\mathcal{A}}(n) = 0$. Now, let us discuss the failure events. $\mathcal{F}_1$ only occurs with negligible probability as BBW *is simulation-linkable, so we have $\Pr[\mathcal{F}_1] \leq \mathsf{negl}(n)$. $\mathcal{F}_2$*

also only occurs with negligible probability according to Theorem 5.1, so we have $\Pr[\mathcal{F}_2] \leq \mathsf{negl}(n)$*. When* $\mathcal{F}_3$ *occurs, we extracted a valid signature* σ *on a message* m *that has never been signed. Thus, we can construct an adversary* $\mathcal{B}_{\text{EUF-CMA}}$ *against the* EUF-CMA *security of* S *and it holds that* $\Pr[\mathcal{F}_3] = \mathsf{Adv}^{\text{euf-cma}}_{\mathsf{S},\mathcal{B}_{\text{EUF-CMA}}}(n)$*.*
Putting everything together, we have

$$\begin{aligned}
\mathsf{Adv}^{\textit{ob-acc-ver}}_{\mathsf{BBW},\mathcal{A}}(n) &= \mathsf{Adv}^{\textit{game-1}}_{\mathsf{BBW},\mathcal{A}}(n)\\
&= \Pr\left[\mathsf{Exp}^{\text{game}-1}_{\mathsf{BBW},\mathcal{A}}(n) = 1 \wedge \neg\left(\bigvee_{i=1}^{3} \mathcal{F}_i\right)\right]\\
&+ \Pr\left[\mathsf{Exp}^{\text{game}-1}_{\mathsf{BBW},\mathcal{A}}(n) = 1 \wedge \left(\bigvee_{i=1}^{3} \mathcal{F}_i\right)\right]\\
&\leq \mathsf{Adv}^{\textit{game-4}}_{\mathsf{BBW},\mathcal{A}}(n) + \Pr\left[\left(\bigvee_{i=1}^{3} \mathcal{F}_i\right)\right]\\
&\leq \Pr[\mathcal{F}_1] + \Pr[\mathcal{F}_2] + \Pr[\mathcal{F}_3]\\
&\leq \mathsf{negl}(n) + \mathsf{negl}(n) + \mathsf{Adv}^{\text{euf-cma}}_{\mathsf{S},\mathcal{B}_{\text{EUF-CMA}}}(n)
\end{aligned}$$

which is negligible since S *is* EUF-CMA *secure.* ■

Double-Spend Detection

Proof

*We make use of the same sequence of games **1** to **4** as in the proof of owner-binding. Let* $\mathsf{Adv}^{\text{game}-\text{i}}_{\mathsf{BBW},\mathcal{A}}(n) = \Pr\left[\mathsf{Exp}^{\text{game}-\text{i}}_{\mathsf{BBW},\mathcal{A}}(n) = 1\right]$ *again denote the advantage of* $\mathcal{A}$ *in **Game i**. We thus have that* $\mathsf{Adv}^{\textit{game-1}}_{\mathsf{BBW},\mathcal{A}}(n) = \mathsf{Adv}^{\textit{dsd}}_{\mathsf{BBW},\mathcal{A}}(n)$ *and* $\mathsf{Adv}^{\textit{dsd}}_{\mathsf{BBW},\mathcal{A}}(n) \approx \mathsf{Adv}^{\textit{game-4}}_{\mathsf{BBW},\mathcal{A}}(n) + \mathsf{Adv}^{\text{euf-cma}}_{\mathsf{S},\mathcal{B}_{\text{EUF-CMA}}}(n)$*.*

Now let us again consider the success probability of $\mathcal{A}$ *in **Game 4**:*
Let $\mathsf{PK}^{(1)}_{\mathcal{U}}, m^{(1)} := (s^{(1)}, w^{(1)}, \mathsf{sk}^{(1)}_{\mathcal{U}}, u^{(1)}_1), \sigma^{(1)}$ *be the extracted values from the call resulting in* $dstag_1 = (s, z_1)$ *and* $\mathsf{PK}^{(2)}_{\mathcal{U}}, m^{(2)} := (s^{(2)}, w^{(2)}, \mathsf{sk}^{(2)}_{\mathcal{U}}, u^{(2)}_1), \sigma^{(2)}$ *from the call resulting in* $dstag_2 = (s, z_2)$ *respectively, with* $s^{(1)} = s^{(2)} = s$*. Let also* $z_1 =: (t^{(1)}, u^{(1)}_2)$ *and* $z_2 =: (t^{(2)}, u^{(2)}_2)$*. Note that it holds that* $\mathsf{PK}^{(i)}_{\mathcal{U}} = \mathsf{sk}^{(i)}_{\mathcal{U}} G$*.*

There are 4 ways for $\mathcal{A}$ *to win:*
Case 1 ExtractUID *fails to extract a* $\mathsf{PK}_{\mathcal{U}}$ *for either call*
Case 2 $\mathsf{PK}^{(1)}_{\mathcal{U}} \neq \mathsf{PK}^{(2)}_{\mathcal{U}}$
Case 3 $\mathsf{IdentDS}(\mathsf{PK}_{\mathcal{I}}, dstag_1, dstag_2) \neq (\mathsf{PK}_{\mathcal{U}}, \pi)$
Case 4 $\mathsf{IdentDS}(\mathsf{PK}_{\mathcal{I}}, dstag_1, dstag_2) = (\mathsf{PK}^{(1)}_{\mathcal{U}}, \pi)$ *but* $\mathsf{VerifyGuilt}(\mathsf{PK}_{\mathcal{I}}, \mathsf{PK}^{(1)}_{\mathcal{U}}, \pi) = 0$

Case 1 can never happen, as we already ensured ExtractUID *successfully extracts* $\mathsf{PK}_{\mathcal{U}}$ *for all calls, hence* $\Pr[\textit{Case 1}] = 0$*.*
In Case 2 we have two messages $m^{(1)}$*,* $m^{(2)}$ *that were both signed by the experiment and for which it holds that* $s^{(1)} = s^{(2)}$ *but* $\mathsf{sk}^{(1)}_{\mathcal{U}} \neq \mathsf{sk}^{(2)}_{\mathcal{U}}$ *(since* $\mathsf{PK}^{(1)}_{\mathcal{U}} \neq \mathsf{PK}^{(2)}_{\mathcal{U}}$*). But* $s^{(i)}$ *is chosen uniformly at random from* $\mathbb{Z}_p$ *(observe that for any* $s' \in \mathbb{Z}_p$ *and uniformly randomly chosen* $s'' \leftarrow \mathbb{Z}_p$ *it holds that* $s' + s''$ *is uniformly random over* $\mathbb{Z}_p$*) and thus the probability for* $s^{(1)} = s^{(2)}$ *is at most* $\frac{q^2}{p}$ *where* q *is the amount of successful calls* $\mathcal{A}$ *did to* MalIssue*,* MalAcc *and* MalVer*. Hence, we have* $\Pr[\textit{Case 2}] \leq \frac{q^2}{p}$*.*
For Case 3, recall that we have double spending tags $dstag_1$ *and* $dstag_2$ *with* $t^{(1)} = \mathsf{sk}^{(1)}_{\mathcal{U}} u^{(1)}_2 + u^{(1)}_1$ *and* $t^{(2)} = \mathsf{sk}^{(2)}_{\mathcal{U}} u^{(2)}_2 + u^{(2)}_1$*. From the definition of* IdentDS *it follows that* $\mathsf{IdentDS}(dstag_1, dstag_2) = (\mathsf{PK}^{(1)}_{\mathcal{U}}, \mathsf{sk}^{(1)}_{\mathcal{U}})$ *if the following conditions are satisfied: (1)* $\mathsf{sk}^{(1)}_{\mathcal{U}} = \mathsf{sk}^{(2)}_{\mathcal{U}}$ *(2)* $\mathsf{PK}^{(1)}_{\mathcal{U}} = \mathsf{sk}^{(1)}_{\mathcal{U}} G$ *(3)* $u^{(1)}_2 \neq u^{(2)}_2$ *and (4)* $u^{(1)}_1 = u^{(2)}_1$*.*
Assume $\mathsf{PK}^{(1)}_{\mathcal{U}} = \mathsf{PK}^{(2)}_{\mathcal{U}}$ *(otherwise Case 2 is satisfied). From the definition of* ExtractUID*, it follows that* $\mathsf{PK}^{(1)}_{\mathcal{U}} = \mathsf{sk}^{(1)}_{\mathcal{U}} G, \mathsf{PK}^{(2)}_{\mathcal{U}} = \mathsf{sk}^{(2)}_{\mathcal{U}} G$ *and thus* $\mathsf{sk}^{(1)}_{\mathcal{U}} = \mathsf{sk}^{(2)}_{\mathcal{U}}$*.* $u^{(1)}_2 \neq u^{(2)}_2$ *holds with probability at least* $1 - \frac{q^2}{p}$ *as both are chosen uniformly at random from* $\mathbb{Z}_p$ *by the Accumulator/Verifier. It remains the case that* $u^{(1)}_1 \neq u^{(2)}_1$*. Again, as we already ensured that* $m^{(1)}$ *and* $m^{(2)}$ *were both signed by the experiment during a call to any oracle, this means there were two distinct calls that resulted in the same serial number* s*. Thus, as in Case 2, the probability for this is at most* $\frac{q^2}{p}$*. This leads us to* $\Pr[\textit{Case 3} \,|\, \neg\textit{Case 2}] \leq \frac{q^2}{p} + \frac{q^2}{p}$*.*
Case 4 can never happen due to the definitions of IdentDS *and* VerifyGuilt*: If* IdentDS *outputs* $(\mathsf{PK}_{\mathcal{U}}, \pi)$ *(and not* $\perp$*), it computed* $\mathsf{PK}_{\mathcal{U}}$ *as* πG*, while* VerifyGuilt *checks whether* $\mathsf{PK}_{\mathcal{U}} = \pi G$*. Thus* $\Pr[\textit{Case 4}] = 0$*.*
Hence, $\mathcal{A}$*'s advantage in **Game 4** is* $\mathsf{Adv}^{\textit{game-4}}_{\mathsf{BBW},\mathcal{A}}(n) \leq \frac{3q^2}{p}$ *and we have that*

$$\mathsf{Adv}^{\textit{dsd}}_{\mathsf{BBW},\mathcal{A}}(n) \approx \frac{3q^2}{p} + \mathsf{Adv}^{\text{euf-cma}}_{\mathsf{S},\mathcal{B}_{\text{EUF-CMA}}}(n)$$

which is negligible since S *is* EUF-CMA *secure and* $\frac{q^2}{p}$ *is negligible.* ■

Balance-Binding This proof closely follows the proof of [20, Thm. B.5].

Proof

We show that an adversary cannot make us miscount his balance and that he cannot claim a higher balance than what we counted.

To achieve this, we proceed in a series of games, where the first game is the real experiment and the last game is setup such that it is impossible for the adversary to win. We show that the difference between two such games is either negligible or any adversary able to differentiate between them can be used to build an adversary against one of the building blocks.

Firstly, we now explain the different games.

***Game 1** is the real experiment.*

*In **Game 2**, if* ExtractUID *fails for any call to* MalAcc *or* MalVer*, the experiment aborts and returns* 0*. We call this event failure event* $\mathcal{F}_1$ *(*ExtractUID *failed).*

*In **Game 3**, when* $\mathcal{A}$ *finished running, we use the extractor* $\mathcal{E}^*$ *from Theorem 5.1 to extract witnesses for all zero-knowledge proofs. More precisely, for each call to* MalIssue*, we extract a message* $m' := (s', w = 0, \mathsf{sk}_{\mathcal{U}}, u_1)$ *and store a record*

$(\mathsf{PK}_{\mathcal{U}}, m', m^* := (s_{out} := s' + s'', w = 0, \mathsf{sk}_{\mathcal{U}}, u_1), s'')$, *where* $\mathsf{PK}_{\mathcal{U}}$ *is the user public key for which* MalIssue *was called and* m^* *is the message for which* $\mathcal{A}$ *obtained a signature, with* s'' *being the issuers random share of the serial number. For each call to* MalAcc *or* MalVer *we extract a message* $m' := (s', w, \mathsf{sk}_{\mathcal{U}}, u'_1)$ *and a message* $m := (s_{in} := s, w, \mathsf{sk}_{\mathcal{U}}, u_1)$ *together with a valid signature* σ *and store a record* $(\mathsf{PK}_{\mathcal{U}}, m', m, \sigma_{in}, m^* := (s_{out} := s' + s'', w + v, \mathsf{sk}_{\mathcal{U}}, u'_1), s'')$, *where* $\mathsf{PK}_{\mathcal{U}}$ *is the public key extracted by* ExtractUID *and* m^* *is the message for which* $\mathcal{A}$ *obtained a signature, with* s'' *being the issuers random share of the serial number and* v *the amount of points collected/redeemed. If extraction of any witness fails, the game aborts and returns* 0*. We call this event failure event* $\mathcal{F}_2$ *(extraction failure).*

In the following we view the set of transactions as a directed graph, where each successful transaction is a vertex and there exists an edge from transaction A to transaction B, if the serial number s_{out} *from A is equal to the serial number* s_{in} *of B.*

In ***Game 4*** *we make sure that the indegree of every vertex associated with a call to* MalAcc *or* MalVer *is at least one, i. e. every serial number* s_{in} *that appeared in a call to* MalAcc *or* MalVer *was part of a signed token generated in a previous interaction. Let* $\widehat{rec} = (\widehat{\mathsf{PK}}_{\mathcal{U}}, \hat{m}', \hat{m} := (\hat{s}_{in}, \hat{w}, \widehat{\mathsf{sk}}_{\mathcal{U}}, \hat{u}_1), \hat{\sigma}_{in}, \hat{m}^* := (\hat{s}_{out}, \widehat{w}^*, \widehat{\mathsf{sk}}_{\mathcal{U}}, \hat{u}_1^*), \hat{s}'')$ *denote the record of the transaction considered. Then, the game checks for every successful* MalAcc/MalVer *transaction (containing* $\hat{s}_{in}$*) if there exists a previous transaction whose record contains* s_{out} *with* $s_{out} = \hat{s}_{in}$*. If no such transaction exists, the game aborts and returns 0. We call this event failure event* $\mathcal{F}_3$ *(new serial number).*

In ***Game 5*** *we now make sure that the indegree of every vertex associated with a call to* MalAcc/MalVer *is at most one, i. e. every serial number* s_{in} *that appeared in a call to* MalAcc *or* MalVer *has been signed in exactly one previous transaction. To do so, the game now additionally checks if there is more than one previous transaction containing* s_{out} *such that* $s_{out} = \hat{s}_{in}$*. If at least two such transactions exist, the game aborts and returns 0. We call this event failure event* $\mathcal{F}_4$ *(serial number collision).*

In ***Game 6*** *we make sure that the outdegree of every vertex is at most one, i. e. that every serial number* s_{out} *is used as input in at most one other transaction. To do so, for every call to* MalAcc *or* MalVer *the game checks if there already exists another transaction containing* s_{in} *with* $s_{in} = \hat{s}_{in}$ *and if so aborts and returns 0. We call this event failure event* $\mathcal{F}_5$ *(double spending).*

In ***Game 7*** *we now have for every* MalAcc/MalVer *transaction exactly one previous transaction where* $\hat{s}_{in}$ *was generated and signed. We now make sure that the public key* $\widehat{\mathsf{PK}}_{\mathcal{U}}$ *in each new such transaction is equal to the public key* $\mathsf{PK}_{\mathcal{U}}$ *from the previous transaction. If this is not the case for the first time, the game aborts and return 0. We call this event failure event* $\mathcal{F}_6$ *(miscount).*

In ***Game 8*** *we check for over-claims along the path. By now, our graph consists of unary trees, where the root of each tree represents a call to* MalIssue *and consecutive child nodes represent changes to the balance associated with the respective user (and we made sure that for each such tree there is exactly one user associated with the transaction, namely the one for which* MalIssue *was called). Now, the game first checks that for each* MalVer *transaction, it holds that* $w \geq v$ *(and thus* $\hat{w} \in \mathbb{V}$*). Additionally the game checks each* MalAcc *transaction if for its predecessor transaction record it holds that* $\hat{w} = w + v$ *and for each* MalVer *transaction that* $\hat{w} = w - v$*. If either of these not the case for any transaction, a wrong balance has successfully been claimed and the game aborts and returns 0. We call this event failure event* $\mathcal{F}_7$ *(wrong claim).*

Let $\mathsf{Adv}^{\mathrm{game-i}}_{\mathsf{BBW},\mathcal{A}}(n) = \Pr\left[\mathsf{Exp}^{\mathrm{game-i}}_{\mathsf{BBW},\mathcal{A}}(n) = 1\right]$ *denote the advantage of* $\mathcal{A}$ *in* ***Game i****. Thus, by definition,*

$$\mathsf{Adv}^{\textit{game-1}}_{\mathsf{BBW},\mathcal{A}}(n) = \mathsf{Adv}^{\textit{bb}}_{\mathsf{BBW},\mathcal{A}}(n) \tag{18}$$

Note that in Game ***8****, if none of the failure events occurred, what has been counted as balance for* $\mathsf{PK}_{\mathcal{U}}$ *up to a* MalVer *call coincides with the claimed balance. Hence, the adversary cannot win this game and we have*

$$\mathsf{Adv}^{\textit{game-8}}_{\mathsf{BBW},\mathcal{A}}(n) = 0 \tag{19}$$

Since each game only differs from the previous one if the respective failure event occurred, we also have

$$\begin{aligned}\mathsf{Adv}^{\textit{game-8}}_{\mathsf{BBW},\mathcal{A}}(n) &= \mathsf{Adv}^{\textit{game-1}}_{\mathsf{BBW},\mathcal{A}}(n) - \\ &\Pr\left[\mathsf{Exp}^{\mathrm{game-1}}_{\mathsf{BBW},\mathcal{A}}(n) = 1 \wedge \left(\bigvee_{i=1}^{7} \mathcal{F}_i\right)\right]\end{aligned} \tag{20}$$

Now let us discuss the failure events.

$\mathcal{F}_1$ *only occurs with negligible probability as* BBW *is simulation-linkable, thus*

$$\Pr[\mathcal{F}_1] \leq \mathsf{negl}(n) \tag{21}$$

$\mathcal{F}_2$ *only occurs with negligible probability according to Theorem 5.1, so*

$$\Pr[\mathcal{F}_1] \leq \mathsf{negl}(n). \tag{22}$$

If $\mathcal{F}_3$ *occurs, we have a serial number* $\hat{s}_{in}$ *that didn't occur in previous transactions. As we assume that* P2 *and* P3 *are sound, we have the signature* $\hat{\sigma}_{in}$ *as part of* $\widehat{rec}$*, which is a valid signature for the message* $m = (\hat{s}_{in}, \hat{w}, \widehat{\mathsf{sk}}_{\mathcal{U}}, \hat{u}_1)$ *that has not been signed by the experiment. Hence, we can construct an* EUF-CMA *adversary* $\mathcal{B}_{\text{EUF-CMA}}$ *against* S *with advantage*

$$\mathsf{Adv}^{\text{euf-cma}}_{\mathsf{S},\mathcal{B}_{\text{EUF-CMA}}}(n) \approx \Pr[\mathcal{F}_3] \tag{23}$$

If $\mathcal{F}_4$ *occurs, we have two previous records* rec_1 *and* rec_2 *both containing* $s_{out} = \hat{s}_{in}$*. As* s_{out} *was computed as*

$s' + s''$, where s' was chosen by $\mathcal{A}$ but s'' was chosen uniformly at random from $\mathbb{Z}_p$ by the oracle, s_{out} is uniformly random over $\mathbb{Z}_p$. Thus, the probability for $s_{in1} = s_{in2}$ is at most $\frac{q^2}{p}$, where q is the amount of successful calls $\mathcal{A}$ did to MalIssue, MalAcc *and* MalVer. *So we have*

$$\Pr[\mathcal{F}_4] \leq \frac{q^2}{p} \tag{24}$$

If $\mathcal{F}_5$ occurs, we have a record rec containing $s_{in} = \hat{s}_{in}$. But due to the winning conditions of $\mathcal{A}$, it holds that

$$\Pr\left[\mathsf{Exp}^{\mathrm{game-i}}_{\mathsf{BBW},\mathcal{A}}(n) = 1 \wedge \mathcal{F}_5\right] = 0. \tag{25}$$

If $\mathcal{F}_6$ occurs, we have a previous record rec containing $\mathsf{PK}_{\mathcal{U}} \neq \widehat{\mathsf{PK}}_{\mathcal{U}}$. As we already made sure that each serial number is only part of one signed token, which in this case must be $m = (s_{out}, w + v, \mathsf{sk}_{\mathcal{U}}, u'_1)$ and $\mathsf{PK}_{\mathcal{U}} \neq \widehat{\mathsf{PK}}_{\mathcal{U}}$ implies $\mathsf{sk}_{\mathcal{U}} \neq \widehat{\mathsf{sk}}_{\mathcal{U}}$, the soundness of P2 *and* P3 *means we have another message $\hat{m} = (\hat{s}_{in} = s_{out}, \hat{w}, \widehat{\mathsf{sk}}_{\mathcal{U}}, \hat{u}'_1)$ and valid signature $\hat{\sigma}_{in}$ on $\hat{m}$. As $\hat{m}$ was never signed before, we can construct an adversary $\mathcal{B}_{\text{EUF-CMA}}$ against the* EUF-CMA *security of* S *with advantage*

$$\mathsf{Adv}^{\text{euf-cma}}_{\mathsf{S},\mathcal{B}_{\text{EUF-CMA}}}(n) \approx \Pr[\mathcal{F}_6] \tag{26}$$

If $\mathcal{F}_7$ occurs, there are two possibilities: there exists a MalVer *transaction for which $w < v$ ($\mathcal{F}_{7.1}$) or there exists a* MalAcc *or* MalVer *transaction for which $\hat{w} \neq w + v$ ($\hat{w} \neq w - v$ resp.) holds ($\mathcal{F}_{7.2}$).*

If $\mathcal{F}_{7.1}$ occurs, we can construct an adversary $\mathcal{B}_{wee}$ against witness-extended emulation of RP *with advantage*

$$\mathsf{Adv}^{wee}_{\mathsf{RP},\mathcal{B}_{wee}}(n) \approx \Pr[\mathcal{F}_{7.1}] \tag{27}$$

. *For $\mathcal{F}_{7.2}$ we again have two tokens with the same serial number that differ in another part. So with the same argument as above, we have a message $\hat{m} = (\hat{s}_{in} = s_{out}, \hat{w}, \widehat{\mathsf{sk}}_{\mathcal{U}}, \hat{u}'_1)$ and signature $\hat{\sigma}_{in}$ on $\hat{m}$, but $\hat{m}$ has never been signed (as $\hat{w} \neq w + v$), so we can again construct an adversary $\mathcal{B}_{\text{EUF-CMA}}$ against the* EUF-CMA *security of* S *with advantage*

$$\mathsf{Adv}^{\text{euf-cma}}_{\mathsf{S},\mathcal{B}_{\text{EUF-CMA}}}(n) \approx \Pr[\mathcal{F}_{7.2}] \tag{28}$$

. *Thus, we have*

$$\Pr[\mathcal{F}_7] \leq \mathsf{Adv}^{wee}_{\mathsf{RP},\mathcal{B}_{wee}}(n) + \mathsf{Adv}^{\text{euf-cma}}_{\mathsf{S},\mathcal{B}_{\text{EUF-CMA}}}(n) \tag{29}$$

Putting everything together, we have from (19) to (26) and (29) that

$$\begin{aligned}
\mathsf{Adv}^{bb}_{\mathsf{BBW},\mathcal{A}}(n) &= \mathsf{Adv}^{game\text{-}1}_{\mathsf{BBW},\mathcal{A}}(n) \\
&= \Pr\left[\mathsf{Exp}^{\mathrm{game-1}}_{\mathsf{BBW},\mathcal{A}}(n) = 1 \wedge \neg\left(\bigvee_{i=1}^{7} \mathcal{F}_i\right)\right] \\
&\quad + \Pr\left[\mathsf{Exp}^{\mathrm{game-1}}_{\mathsf{BBW},\mathcal{A}}(n) = 1 \wedge \left(\bigvee_{i=1}^{7} \mathcal{F}_i\right)\right] \\
&= \mathsf{Adv}^{game\text{-}8}_{\mathsf{BBW},\mathcal{A}}(n) \\
&\quad + \Pr\left[\mathsf{Exp}^{\mathrm{game-1}}_{\mathsf{BBW},\mathcal{A}}(n) = 1 \wedge \left(\bigvee_{i=1}^{7} \mathcal{F}_i\right)\right] \\
&\leq \mathsf{negl}(n) + \Pr\left[\left(\bigvee_{i=1}^{7} \mathcal{F}_i\right)\right] \\
&\leq \mathsf{negl}(n) + \mathsf{negl}(n) + \mathsf{negl}(n) \\
&\quad + \mathsf{Adv}^{\text{euf-cma}}_{\mathsf{S},\mathcal{B}_{\text{EUF-CMA}}}(n) + \frac{q^2}{p} \\
&\quad + 0 + \mathsf{Adv}^{\text{euf-cma}}_{\mathsf{S},\mathcal{B}_{\text{EUF-CMA}}}(n) \\
&\quad + \mathsf{Adv}^{wee}_{\mathsf{RP},\mathcal{B}_{wee}}(n) + \mathsf{Adv}^{\text{euf-cma}}_{\mathsf{S},\mathcal{B}_{\text{EUF-CMA}}}(n) \\
&\approx \frac{q^2}{p} + \mathsf{Adv}^{\text{euf-cma}}_{\mathsf{S},\mathcal{B}_{\text{EUF-CMA}}}(n)
\end{aligned}$$

which is negligible as we assumed S *to be* EUF-CMA *secure and $\frac{q^2}{p}$ is negligible.* ■

D.4 User Security/Privacy

Again we state the slightly modified formal definitions for user privacy/security.

Definition D.7 (Privacy-Preserving) *We say that a* BBW *scheme is* privacy-preserving, *if there exist PPT algorithms* SimSetup *and* SimCorrupt *as well as interactive PPT algorithms* SimHonIssue, SimHonAdd *and* SimHonSub *that receive no private user input, such that for all PPT adversaries $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ in the experiments from Figure 2, the advantage $\mathsf{Adv}^{priv}_{\mathsf{BBW},\mathcal{A}}(n)$ of $\mathcal{A}$ defined by*

$$\left|\Pr[\mathsf{Exp}^{priv\text{-}real}_{\mathsf{BBW},\mathcal{A}}(n) = 1] - \Pr[\mathsf{Exp}^{priv\text{-}ideal}_{\mathsf{BBW},\mathcal{A}}(n) = 1]\right| \tag{30}$$

is negligible in n.

Definition D.8 (False-Accusation Protection) *A simulation-linkable* BBW *scheme ensures false-accusation protection if for any PPT adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ in the experiment $\mathsf{Exp}^{facp}_{\mathsf{BBW},\mathcal{A}}(n)$ from Figure 11 the advantage of $\mathcal{A}$ defined by*

$$\mathsf{Adv}^{facp}_{\mathsf{BBW},\mathcal{A}}(n) := \Pr[\mathsf{Exp}^{facp}_{\mathsf{BBW},\mathcal{A}}(n) = 1] \tag{31}$$

is negligible in n.

Experiment $\mathsf{Exp}^{\mathsf{facp}}_{\mathsf{BBW},\mathcal{A}}(n)$

```
$crs \leftarrow \mathsf{Setup}(1^n)$
$(\mathsf{PK}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}) \leftarrow \mathcal{A}_0(crs)$
$(\mathsf{PK}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}) \leftarrow \mathsf{UGen}(crs)$
$\pi \leftarrow \mathcal{A}_1^{\mathsf{RealHonIssue},\mathsf{RealHonAdd},\mathsf{RealHonSub}}(\mathsf{PK}_{\mathcal{I}}, \mathsf{PK}_{\mathcal{U}})$
The experiment returns 1 iff $\mathsf{VerifyGuilt}(\mathsf{PK}_{\mathcal{I}}, \mathsf{PK}_{\mathcal{U}}, \pi) = 1$.
```

Figure 15. False accusation protection experiment

We again split the proof of Theorem 5.4 in separate proofs for privacy-preserving and false-accusation protection.

Privacy-Preserving

Proof
To show that BBW *is privacy-preserving, we define a sequence of games* ***Game 1****, …,* ***Game 5****, where* ***Game 1*** *is the game where all oracles are those from the real world, while* ***Game 5*** *is the game where all oracles are those from the ideal world.*

We denote the experiment with an adversary $\mathcal{A}$ playing the game ***i*** *by $\mathsf{Exp}^{\text{game}-\text{i}}_{\mathsf{BBW},\mathcal{A}}(n)$.*

We write $\mathsf{Setup}_i, \mathsf{HonIssue}_i, \mathsf{HonAdd}_i, \mathsf{HonSub}_i$ and $\mathsf{Corrupt}_i$ to denote the implementations of the oracles in game i (the oracle HonUser *remains unchanged in all games).*

Game 1 *We set $\mathsf{Setup}_1 = \mathsf{Setup}$, $\mathsf{HonIssue}_1 = \mathsf{RealHonIssue}$, $\mathsf{HonAdd}_1 = \mathsf{RealHonAdd}$, $\mathsf{HonSub}_1 = \mathsf{RealHonSub}$ and $\mathsf{Corrupt}_1 = \mathsf{RealCorrupt}$ as in Figures 3 to 6. So in other words, $\mathsf{Exp}^{\text{game}-1}_{\mathsf{BBW},\mathcal{A}}(n)$ is identical to $\mathsf{Exp}^{\textsf{priv-real}}_{\mathsf{BBW},\mathcal{A}}(n)$.*

Game 2 *We modify Setup_2 such that crs_{sig} is not generated by $\mathsf{S.Setup}(gp)$ but instead by $(crs_{\mathsf{sig}}, td_{\mathsf{sig}}) \leftarrow \mathsf{S.SimSetup}(gp)$ (and thus crs_{com} is generated as $(crs_{\mathsf{com}}, td_{\mathsf{com}}) \leftarrow \mathsf{PC.SimSetup}(gp)$), crs_{pok} is not generated by $\mathsf{Z.Setup}(gp)$ but instead by $(crs_{\mathsf{pok}}, td_{\mathsf{pok}}) \leftarrow \mathsf{Z.SimSetup}(gp)$ and crs_{rp} is not generated by $\mathsf{RP.Setup}(gp)$ but instead by $(crs_{\mathsf{rp}}, td_{\mathsf{rp}}) \leftarrow \mathsf{RP.SimSetup}(gp)$.*

Game 3 *We redefine $\mathsf{HonIssue}_3, \mathsf{HonAdd}_3$ and HonSub_3 such that the proofs done with the adversary are simulated (using the trapdoors td_{pok} and td_{rp} to choose the challenge beforehand).*

Game 4 *We modify $\mathsf{HonIssue}_4, \mathsf{HonAdd}_4$ and HonSub_4 so that the commitment C' sent to the adversary is replaced by a commitment to $(0,0,0,0,0)$ and the signature σ_1 is replaced by random values for which the random oracle $\mathcal{H}$ is accordingly programmed so the signature verifies. We also modify $\mathsf{Corrupt}_4$ to equivocate (using td_{com}) the commitment C (which was sent by $\mathcal{A}$ and now is a commitment to $(s'', v, 0, 0, 0)$) to one containing random s, u_1 and the correct $\mathsf{sk}_{\mathcal{U}}$, w and attr with opening information d, and generate a new signature σ by choosing random $\gamma, \omega, \omega', \rho, \rho'_1, \rho'_2, \mu \leftarrow \mathbb{Z}_p$, computing $\zeta = \gamma Z$, $\zeta_1 = \gamma C$, setting $\sigma = (\zeta, \zeta_1, \rho, \omega, \rho'_1, \rho'_2, \omega', \mu, d, \gamma)$ and programming the random oracle $\mathcal{H}$ so that the signature verifies.*

We also need to make sure UVer *still works as expected in $\mathsf{HonIssue}_4, \mathsf{HonAdd}_4$ and HonSub_4. To achieve this we replace the call $\mathsf{UVer}(\mathsf{PK}_{\mathcal{I}}, \mathsf{PK}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, \tau, w^*, attr)$ with $\mathsf{UVer}(\mathsf{PK}_{\mathcal{I}}, 0_{\mathbb{G}}, 0, \tau, v, 0)$.*

Game 5 *We now modify HonAdd_5 and HonSub_5 so that t and s are chosen at random.*

We note that $\mathsf{Exp}^{\text{game}-5}_{\mathsf{BBW},\mathcal{A}}(n)$ is identical to $\mathsf{Exp}^{\textsf{priv-ideal}}_{\mathsf{BBW},\mathcal{A}}(n)$.

We now show that a PPT adversary $\mathcal{A}$ cannot distinguish between games i and $i+1$. By $\mathcal{D}_i$ we denote the probability that $\mathcal{A}$ is able to distinguish ***Game i*** *from* ***Game i + 1****.*

Game 1 to Game 2 *Only the crs is changed and this is perfectly indistinguishable for both crs_{com} and crs_{pok} (see the zero-knowledge property of the proofs and the equivocality property of* PC*) Thus, we have*

$$\Pr[\mathcal{D}_2] = 0 \tag{32}$$

Game 2 to Game 3 *In this hop, the real zero-knowledge proofs are replaced by simulated ones. Note that the proven statements are still valid.*

We introduce 2 new games: in ***Game 2.1****, only* P1 *proofs are replaced by simulated ones. In* ***Game 2.2*** *we also replace the proofs for* P2 *and in* ***Game 2.3*** *additionally proofs for* P3 *are replaced, so* ***Game 2.3*** *=* ***Game 3****.*

Now assume there is an adversary $\mathcal{A}$ that can distinguish between ***Game 2*** *and* ***Game 2.1****. We argue by use of an hybrid argument with games* ***Game 2.0*** *to* ***Game 2.0.****q =* ***Game 2.1****. In* ***Game 2.0.****j, the first j proofs for* P1 *are simulated, while all proofs from $j+1$ to q are real. As $\mathcal{A}$ notices a difference between* ***Game 2.0*** *and* ***Game 2.0.****q, there must be an index $j \in [q]$ such that $\mathcal{A}$ can distinguish between $\mathsf{Exp}^{\text{game}-2.0.\text{j}}_{\mathsf{BBW},\mathcal{A}}(n)$ and $\mathsf{Exp}^{\text{game}-2.0.\text{j}+1}_{\mathsf{BBW},\mathcal{A}}(n)$. We can then construct an adversary $\mathcal{B}_{1-j}$ against the composable perfect zero-knowledge property of* P1*. It then holds that $\Pr[\mathcal{D}_{2.1}] \leq \sum_{i=0}^{q-1} \mathsf{Adv}^{\text{zk}}_{P1,\mathcal{B}_{1-i}}(n)$, but as we assume that* P1 *is perfectly composable zero-knowledge, it holds that $\mathsf{Adv}^{\text{zk}}_{P1,\mathcal{B}_{1-i}}(n) = 0$ for all $i \in [q-1]$ and thus*

$$\Pr[\mathcal{D}_{2.1}] = 0 \tag{33}$$

Next, consider the hop from ***Game 2.1*** *to* ***Game 2.2****. This time the proofs for* P2 *are also replaced by simulated ones. We argue by use of the same hybrid argument as above: in* ***Game 2.1.****j, all proofs for* P1 *are simulated and the first j proofs for* P2 *are simulated, while proofs $j+1$ to q for* P2 *are real. Hence we have that*

$$\Pr[\mathcal{D}_{2.2}] = \mathsf{Adv}^{\text{zk}}_{P2,\mathcal{B}_{1-i}}(n) = 0 \tag{34}$$

The same argument can also be done for the hop from ***Game 2.2*** *to* ***Game 2.3*** *and so to conclude we have*

$$\Pr[\mathcal{D}_3] \leq \Pr[\mathcal{D}_{2.1}] + \Pr[\mathcal{D}_{2.2}] + \Pr[\mathcal{D}_{2.3}] = 0 \tag{35}$$

Game 3 to Game 4 *The difference in* ***Game 4*** *is that all the commitments and signatures are replaced.*

*We first argue about replacing the commitments: Assume there is an adversary $\mathcal{A}$ that can distinguish between $\mathsf{Exp}^{\text{game}-3}_{\mathsf{BBW},\mathcal{A}}(n)$ and $\mathsf{Exp}^{\text{game}-4}_{\mathsf{BBW},\mathcal{A}}(n)$. We argue by use of an hybrid argument with games **Game 3.0** to **Game 3.**q = **Game 4**. In **Game 3.**j, the first j commitments are replaced by simulated commitments while all commitments from $j+1$ to q are real. As $\mathcal{A}$ notices a difference between **Game 3.0** and **Game 3.**q, there must be an index $j \in [q]$ such that $\mathcal{A}$ can distinguish between $\mathsf{Exp}^{\text{game}-3.\text{j}}_{\mathsf{BBW},\mathcal{A}}(n)$ and $\mathsf{Exp}^{\text{game}-3.\text{j}+1}_{\mathsf{BBW},\mathcal{A}}(n)$. We can construct an adversary $\mathcal{C}$ against the equivocality of* PC *that internally runs $\mathcal{A}$ and simulates the oracles. $\mathcal{C}$ has the same advantage in the equiv-exp as $\mathcal{A}$ in distinguishing games 3.j and 3.$j+1$, so*

$$\mathsf{Adv}^{\text{equivocality}}_{\mathsf{PC},\mathcal{C}}(n) = \Pr[\mathcal{D}_{3.j+1}] \tag{36}$$

But as PC *is equivocable, we have that $\mathsf{Adv}^{\text{equivocality}}_{\mathsf{PC},\mathcal{C}}(n) = 0$ and thus $\Pr[\mathcal{D}_{3.j+1}] = 0$ for all $j \in [q-1]$. For the signatures, we have the same argument, but against blindness of* BlindVerify. *Thus, for similarly defined games 3.j and 3.$j+1$ we have*

$$\mathsf{Adv}^{\text{zero}-\text{knowledge}}_{\mathsf{S},\mathcal{C}}(n) = \Pr[\mathcal{D}_{3.j+1}] \tag{37}$$

and thus overall we have

$$\Pr[\mathcal{D}_4] \leq q\mathsf{Adv}^{\text{zero}-\text{knowledge}}_{\mathsf{S},\mathcal{C}}(n) \leq \mathsf{negl}(n) \tag{38}$$

Game 4 to Game 5 *In the last step, the values s and t are chosen at random (if the user has not been corrupted in the previous call). Thus, there are two cases to regard: in Case 1, the adversary has successfully corrupted the user $\mathsf{PK}_\mathcal{U}$ in the previous call and in Case 2 the previous oracle call for $\mathsf{PK}_\mathcal{U}$ was any other oracle than* Corrupt. *In Case 2, this changes nothing for the attacker, as u_1 is chosen uniformly at random in every interaction and thus $t = \mathsf{sk}_\mathcal{U} u_2 + u_1$ is a uniformly random value as any other dependency on u_1 has already been removed in the previous games. The same is true for $s = s' + s''$ where s' is chosen uniformly at random.*

In Case 1, the adversary knows the last value of u_1 as well as s that will be used in the next interaction. As u_2 is chosen from the adversary, he can check if $t = \mathsf{sk}_\mathcal{U} u_2 + u_1$ actually holds (since he also received $\mathsf{sk}_\mathcal{U}$ upon corruption) and whether the correct s is used. For this reason, in the interaction immediately following a corruption, the real user algorithm is used.

Thus, we have

$$\Pr[\mathcal{D}_5] = 0 \tag{39}$$

To conclude, we have that

$$\mathsf{Adv}^{\textit{priv}}_{\mathsf{BBW},\ \mathcal{A}}(n) \leq \mathcal{D}_2 + \mathcal{D}_3 + \mathcal{D}_4 + \mathcal{D}_5 \leq \mathsf{negl}(n) \tag{40}$$

and thus BBW *is privacy-preserving.* ■

False-Accusation Protection

Proof
Assume there is an adversary $\mathcal{A}$ that breaks false-accusation protection. Note that the oracles used in the false-accusation experiment are a subset of those used in the privacy-preserving experiment. Thus we can replace all oracles by the simulation oracles and can distinguish two cases:

Case 1 *$\mathcal{A}$ is still able to output a valid proof of guilt $\pi = \mathsf{sk}_\mathcal{U}$ with non-negligible probability. Since all simulation oracles have besides $\mathsf{PK}_\mathcal{U}$ no input that is related to $\mathsf{sk}_\mathcal{U}$ and are PPT, we can construct an adversary $\mathcal{B}$ against the DLOG experiment that gets $\mathsf{PK}_\mathcal{U}$ as challenge and simulates the false-accusation game for $\mathcal{A}$ and outputs the proof $\mathcal{A}$ returns, which is the discrete logarithm of $\mathsf{PK}_\mathcal{U}$.*

Case 2 *$\mathcal{A}$ is no longer able to output a valid proof of guilt. Then $\mathcal{A}$ can be used to create an adversary $\mathcal{B}$ against the privacy-preserving experiment that distinguishes between the real and the ideal game. As* BBW *is privacy-preserving, it holds that $\mathsf{Adv}^{\textit{priv}}_{\mathsf{BBW},\ \mathcal{B}}(n) \leq \mathsf{negl}(n)$.*

So to conclude, we have that

$$\mathsf{Adv}^{\textit{facp}}_{\mathsf{BBW},\ \mathcal{A}}(n) = \mathsf{Adv}^{\text{dlog}}_{\mathcal{B}}(n) + \mathsf{negl}(n) \tag{41}$$

which is negligible as we assumed the DLOG assumption to hold. ■