BBA+: Improving the Security and Applicability of Privacy-Preserving Point Collection

Gunnar Hartung*
Karlsruhe Institute of Technology
Department of Informatics
Am Fasanengarten 5
76131, Karlsruhe, Germany
gunnar.hartung@kit.edu

Max Hoffmann†
Ruhr-Universität Bochum
Department of Electrical Engineering and Information Technology
Universitätsstraße 150
44801, Bochum, Germany
max.hoffmann@rub.de

Matthias Nagel‡
Karlsruhe Institute of Technology
Department of Informatics
Am Fasanengarten 5
76131, Karlsruhe, Germany
matthias.nagel@kit.edu

Andy Rupp§
Karlsruhe Institute of Technology
Department of Informatics
Am Fasanengarten 5
76131, Karlsruhe, Germany
andy.rupp@kit.edu

ABSTRACT

Black-box accumulation (BBA) was introduced at PETS 2016 as a promising building-block for a variety of user-centric protocols such as loyalty, refund, and incentive systems. Loosely speaking, this building block may be viewed as a cryptographic "piggy bank" that allows a user to collect points (aka incentives, coins, etc.) in an anonymous and unlinkable way. A piggy bank may be "robbed" at some point by a user, letting her spent the collected points, thereby only revealing the total amount inside the piggy bank and a serial number, which is linked to the owner.

In this paper we present BBA+, a definitional framework extending the BBA model in multiple ways: (1) We support offline systems in the sense that there does not need to be a permanent connection to a serial number database to check whether a presented piggy bank has already been robbed. (2) We enforce the collection of "negative points", i.e., points users may not voluntarily collect, as this is for example needed in post- or pre-payment systems. (3) The security property formalized for BBA+ schemes is stronger and more natural than for BBA: Essentially, we demand that the amount claimed to be inside a piggy bank must be exactly the amount legitimately collected with this piggy bank. As piggy bank transactions need to be unlinkable at the same time, defining this property is highly non-trivial. (4) We also define a stronger form of privacy, namely forward and backward privacy.

Besides the framework, we show how to construct a BBA+ system from cryptographic building blocks. A security and privacy proof for the proposed scheme is given within our model. Additionally, we present the promising results of a smartphone-based prototypical implementation. They show that our current implementation may already be useable in practice, allowing to run transactions within a second—while we have not exhausted the potential for optimizations.

CCS CONCEPTS

• **Security and privacy** → *Distributed systems security*; Public key encryption; • **Applied computing** → *Digital cash*; Electronic funds transfer; Secure online transactions;

KEYWORDS

Customer Loyalty Programs, Incentive Systems, Stored-Value Payments, Black-Box Accumulation.

1 INTRODUCTION

In numerous user-centric cyber-physical systems, point collection and redemption mechanisms are one of the core components.

One of the most canonical examples are loyalty systems like the German Payback system [32] or the UK-based Nectar program [4]. Users may collect points at ervery purchase for being loyal customers, and these points can be redeemed at some point in exchange for vouchers, services, or other benefits.

In fact, many other cyber-physical systems try to incentify a certain behavior of users by means of similar mechanisms. For instance, in envisioned mobile sensing scenarios, users need to be encouraged to collect environmental or health data measured with their smart devices and provide this data (enhanced by location-time information) to some operator. In exchange, users will, e.g., receive micropayments they can later use to pay for services provided based on the collected data. In Vehicle-2-Grid scenarios, e-car owners need to be rewarded for the power their e-car batteries provide to the Smart Grid when cars are left at the mall, office, etc.

*The project underlying this report was supported by the German Federal Ministry of Education and Research under Grant No. 01|S15035A. The responsibility for this contents of this publication lies with the author.
†The author is supported by DFG grant PA 587/10-1.
‡This work was supported by the German Federal Ministry of Education and Research within the framework of the project "Sicherheit vernetzter Infrastrukturen (SVI)" in the Competence Center for Applied Security Technology (KASTEL).
§The author is supported by DFG grant RU 1664/3-1 and the Competence Center for Applied Security Technology (KASTEL).

In [23] Jager and Rupp formalized the core functional, security, and privacy requirements of a building block to realize the kind of systems described above. Their building block, called black-box accumulation (BBA), consists of various non-interactive algorithms. When executed correctly by legitimate parties, it allows a user to collect positive points (representing incentives) in an anonymous and unlinkable fashion. Obtaining and redeeming a BBA token is a linkable operation as the unique serial number of the token is revealed in both operations. A permanent connection to a database containing serial numbers of tokens already redeemed is required to prevent double-redemption (aka double-spending) of tokens. Hence, BBA schemes are online systems. Unfortunately, the authors formalized a rather weak form of security, by only demanding that a collusion of malicious users may not be able to redeem more points than the total amount of points issued. In particular, this does not rule out that users may transfer points arbitrarily between tokens (without help) or that an "old" copy of a token is redeemed (i.e., not holding the most current balance).

To summarize, BBA suffers from a number of serious restrictions including fairly weak security guarantees, the need of a permanent database connection, the lack of mechanisms to enforce the collection of "negative" points, and the linkability of token creation and redemption. These shortcomings limit the applicability of BBA as a building block in user-centric systems.

For instance, loyalty system providers do not want their customers to pool or trade their points, which is, however, not excluded by the BBA security definition. Moreover, customers should be allowed to partially redeem collected points. To realize this feature with a BBA scheme, one would need to redeem all points on a token, create a new one, and charge it with the remaining (unspent) points. However, in this way all partial redemptions of a customer are linkable.

Another application requiring features beyond BBA are anonymous reputation systems where a central authority rates the behavior, reliability, or activity of users by issuing reputation points. Similar to loyalty systems, it is undesirable that users are able to pool or trade their reputation points. Additionally, it might be useful in some scenarios to be also able to issue negative reputation points either by subtracting points or having a separate counter for negative ratings.

Yet another application where stronger security, offline capabilities, and negative points are beneficial are pre- or post-payment systems. These systems are employed in many domains like public transportation, toll collection, cashless canteen systems, etc. Typically, in practice, such payment systems are implemented using simplistic RFID-transponder or smartcard-based solutions like the Mifare Classic [29], which essentially offers no security and privacy at all ([15, 18, 19] and more), or the Mifare Desfire [30, 31] also allowing to link all transactions.

1.1 Our Contribution

Definitional framework. We present the BBA+ framework which significantly strengthens the security and broadens the applicability of black-box accumulation compared to [23]: Our framework considers interactive algorithms (protocols) which leads to more intuitive definitions and broadens the class of possible instantiations. Our framework also supports the collection of negative points, and a mechanism to identify users who present an old version of their token (possibly having a higher balance than their most recent one).

We define a strong form of privacy, namely forward and backward privacy: An adversary, including the system operator, must not be able to link transactions of an honest user. This even needs to hold for transactions preceding and succeeding a corruption (except the very next) of the user, during which all of his secrets leak to the adversary. The set of unlinkable transactions not only includes accumulation but also point redemption.

Moreover, we formalize a stronger security property which captures the natural notion that the claimed balance of a token should be exactly the amount legitimately collected with this token. Note that due to the strong privacy property that needs to be satisfied at the same time, defining security is highly-nontrivial. We resolve this issue by demanding that privacy can be removed by a secret trapdoor held by a trusted third party or shared by a couple of semi-trusted parties.

Details on our framework are given in Section 4.

Construction. We propose an instantiation satisfying the properties sketched above. This scheme is a semi-generic construction using public-key encryption, homomorphic trapdoor commitments, digital signatures, and Groth-Sahai non-interactive zero-knowledge proofs over bilinear groups for which the SXDH assumption holds.

To achieve freshness of tokens, we draw from techniques typically used in offline e-cash systems, namely double-spending tags. Here, some double-spending tag, e.g., $t := id \cdot c + r \bmod p$, needs to be revealed when spending an e-coin. This tag contains some user identity information id which is blinded by some secret user randomness r (which has been fixed when the coin was issued) and involves a challenge c freshly chosen by a merchant. No information about id is revealed when the coin is spent once (as r is uniform). However, when the coin is spent a second time, a different challenge c' will be used, while the user randomness r will be the same (because it has been fixed). This enables the bank to extract id using the two double-spending tags t and t'.

Let us briefly sketch our construction which follows but significantly extends the idea of [23]. For the sake of simplicity, each user may only receive a single token bound to his public key. An initial token essentially consists of a (multi-)commitment c and a signature σ on this commitment issued by the system operator. The commitment c contains a user's secret key $\mathsf{sk}_{\mathcal{U}}$, a token version number s, the balance value $w = 0$, and some randomness r that will be used to generate a double-spending tag in the next transaction. Note that s and r are not known to the issuer but c and σ are.

To add (positive or negative) points in an unlinkable fashion, one cannot simply sent over the token. Instead, the user sents a new commitment c' containing the same secret key, the same balance, but a new token version number, and some new randomness. Then he proves in zero-knowledge that c' is indeed new version of his old certified commitment c. Additionally, a double-spending tag (encoding $\mathsf{sk}_{\mathcal{U}}$) for the old token version as well as s is revealed. If the party issuing the points accepts the proof, the homomorphic

property of the commitment scheme is used to add the points to c' which is then signed.

This concludes the simplified description of our construction. More details can be found in Section 5.

Implementation. To verify the suitability of our construction in real-world applications, we built a smartphone implementation and benchmarked the execution times of the BBA+ protocols. Our implementation results (for 254-bit Barreto-Naehrig curves with optimal Ate pairing) show that all protocols can be executed in less than 400 ms on the user side. This leads to the conclusion that our cheme is already usable in practice. Nevertheless, further optimizations are possible at both the algorithmic, as well as the implementation level. Details can be found in Section 6. Note that the authors of BBA [23] only provide rough performance estimates for their scheme.

1.2 Related Work

Besides BBA [23], which we already discussed, only [27] appears to consider a point collection mechanism as a separate, multi-purpose building block. Unfortunately, the security and privacy properties of their protocol – called uCentive – are only informally stated and no proofs are given.

The BBA+ framework shares some aspects with the notion of priced oblivious transfer (POT). POT was introduced by [3] as a tool to protect the privacy of customers buying digital goods. The goal is to allow a buyer to purchase digital goods from a vendor without leaking the "what, when and how much". In the original notion of POT, a user's wallet is not possessed by the user itself. In [3] the vendor manages this information. Consequently, user anonymity cannot be granted and the system is inherently limited to a single vendor. Camenisch et al. [12] extended POTs by anonymity of users and unlinkability of individual transactions which brings it closer to our framework. Nonetheless, the scheme is still limited to a single vendor or a system where all vendors share a joint state in an online fashion, whereas our system is an offline system. Moreover, [12] lacks a full rigorous formal treatment and an implementation.

The techniques we use to instantiate our building block bear some resemblance with P-signatures [7, 22] which have been introduced by [7] as a tool to construct anonymous credentials. A P-signature scheme is a two-party scheme between a user and an issuer. The scheme combines the algorithms of a commitment scheme, a signature scheme and extends them by some additional zero-knowledge protocols that allow the user to prove certain statements about the commitments. More precisely a user can generate commitments to messages. He can ask the issuer to sign the original message inside the commitment using the issuer's secret key without the issuer learning this message. Moreover, the user can generate new commitments and prove the equality of their content or that he knows a valid signature on the message inside a commitment. The scheme in [7] builds on weak Boneh-Boyen signatures [10], Groth-Sahai commitments and Groth-Sahai NIZK proofs [20]. Note that for our building block properties beyond that of a P-signature are needed. For instance, we need to prove additional/different statements about the content of our commitments. Also, users need to be able to obtain new signatures on commitments homomorphically modified by the issuer. Moreover, we build on different signatures and commitments.

2 PRELIMINARIES

We will make use of the common notation to describe cryptographic schemes and define their security properties.

The results of this paper are in the setting of asymmetric bilinear groups. We use the following definition of a bilinear group generator.

Definition 2.1 (prime-order bilinear group generator). A *prime-order bilinear group generator* is a PPT algorithm SetupGrp that on input of a security parameter 1^n outputs a tuple of the form

$$\mathsf{gp} := (G_1, G_2, G_T, e, p, g_1, g_2) \leftarrow \mathsf{SetupGrp}(1^n)$$

where G_1, G_2, G_T are descriptions of cyclic groups of prime order p, $\log p = \Theta(n)$, g_1 is a generator of G_1, g_2 is a generator of G_2, and $e\colon G_1 \times G_2 \to G_T$ is a map (aka pairing) which satisfies the following properties:

- e is efficiently computable
- *Bilinearity:* For all $a \in G_1, b \in G_2, x \in \mathbb{Z}_p$, we have $e(a^x, b) = e(a, b^x) = e(a, b)^x$.
- *Non-Degeneracy:* $e(g_1, g_2)$ generates G_T.

Complexity assumptions. Our construction relies on the SXDH assumption in bilinear groups, which essentially asserts that the DDH assumption holds in both source groups of the bilinear map.

Definition 2.2. We say that the DDH assumption holds with respect to SetupGrp over G_i if the advantage $\mathsf{Adv}^{\mathsf{DDH}}_{\mathsf{SetupGrp},i,\mathcal{A}}(1^n)$ defined by

$$\Pr\left[b = b' \;\middle|\; \begin{array}{c} \mathsf{gp} := (G_1, G_2, G_T, e, p, g_1, g_2) \leftarrow \mathsf{SetupGrp}(1^n) \\ x, y, z \leftarrow \mathbb{Z}_p; h_0 := g_i^{xy}; h_1 := g_i^z; b \leftarrow \{0,1\} \\ b' \leftarrow \mathcal{A}(1^n, \mathsf{gp}, g_i^x, g_i^y, h_b) \end{array}\right]$$

is a negligible function in n for all PPT algorithms $\mathcal{A}$. We say that the SXDH assumption holds with respect to SetupGrp if the above holds for both $i = 1$ and $i = 2$.

We also make use of the Co-CDH assumption which is obviously implied by the SXDH assumption.

Definition 2.3. We say that the Co-CDH assumption holds with respect to SetupGrp if the advantage $\mathsf{Adv}^{\mathsf{CO\text{-}CDH}}_{\mathsf{SetupGrp},\mathcal{A}}(1^n)$ defined by

$$\Pr\left[a = g_2^x \;\middle|\; \begin{array}{c} \mathsf{gp} := (G_1, G_2, G_T, e, p, g_1, g_2) \leftarrow \mathsf{SetupGrp}(1^n) \\ x \leftarrow \mathbb{Z}_p; a \leftarrow \mathcal{A}(1^n, \mathsf{gp}, g_1^x) \end{array}\right]$$

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

3 BUILDING BLOCKS

For our semi-generic construction we draw from F_{gp}-extractable non-interactive zero-knowledge (NIZK) proofs, as well as equivocal homomorphic commitments, digital signatures, and public-key encryption which all need to be compatible with the proof system (e.g., structure-preserving in the case of the Groth-Sahai proof system). In the following, we describe these building blocks in an informal fashion. Formal definitions can be found in Appendix B.

F_{gp}-extractable NIZKs. Let R be a witness relation for some NP language $L = \{x \mid \exists wit \text{ s.t. } (x, wit) \in R\}$. Informally speaking, a zero-knowledge proof scheme is a system that allows a prover P to convince a verifier V that some x given to V is contained in L without V learning anything beyond that fact. In a non-interactive zero-knowledge (NIZK) proof, only one message, the proof π, is sent from P to V for that purpose.

More precisely, a NIZK consists of the four algorithms SetupGrp, SetupPoK, Prove, and Vfy. $\mathsf{SetupGrp}(1^n)$ generates public (group) parameters gp given implicitly to all algorithms. The considered language L_{gp} may depend on gp. SetupPoK(gp) outputs a common reference string CRS. Prove(CRS, x, wit) outputs a proof π on input of $x \in L_{\text{gp}}$ and a corresponding witness wit. Vfy(CRS, x, π) outputs 1 if π is considered a valid proof for $x \in L_{\text{gp}}$, and 0 otherwise. The proof system is called *perfectly complete* if Vfy(CRS, x, π) always accepts proofs generated by Prove(CRS, x, wit). It is called *perfectly sound* if it is impossible to generate a proof π for $x \notin L_{\text{gp}}$ such that $\mathsf{Vfy}(\mathsf{CRS}, x, \pi) = 1$. Moreover, it is called *perfectly F_{gp}-extractable* if there exists some PPT algorithms SetupEPoK and ExtractW such that (1) SetupEPoK outputs some CRS which is perfectly indistinguishable from a real CRS as well as a trapdoor $\mathsf{td}_{\text{epok}}$ and (2) ExtractW is able to exploit this trapdoor to extract $F_{\text{gp}}(wit)$ for an NP-witness wit for $x \in L_{\text{gp}}$ from any a valid proof π. Perfect F_{gp}-extractability implies perfect soundness. Note that if F_{gp} is the identity function, then the system is a real proof of knowledge. However, in our case the domain of F_{gp} consists of tuples of group elements and exponents $e \in \mathbb{Z}_p$, where F_{gp} maps exponents e to g_1^e and acts as the identity function on group elements. This is a property of the Groth-Sahai proof system [17, 20] we have to deal with. Finally, the proof system is called *composable zero-knowledge* if there exist PPT algorithms SetupSPoK and SimProof such that (1) SetupSPoK outputs some CRS which is computationally indistinguishable from a real CRS as well as a trapdoor $\mathsf{td}_{\text{spok}}$ and (2) SimProof can use this trapdoor to generate proofs for $x \in L_{\text{gp}}$ without knowing a witness for x that look like real proofs even if $\mathsf{td}_{\text{spok}}$ is known.

F_{gp}-binding commitments. A commitment scheme allows a user to commit to a message m and publish the result, called commitment c, in a way that m is hidden from others, but also the user cannot claim a different m afterwards when he opens c. In an F_{gp}-binding commitment scheme one commits to a message m but opens the commitment using $F_{\text{gp}}(m)$.

More precisely, a non-interactive commitment scheme consists of the four algorithms SetupGrp, Gen, Com, and Open. $\mathsf{SetupGrp}(1^n)$ generates public (group) parameters gp and Gen(gp) a public common reference string CRS. The parameters gp fix a message space for the commitment scheme. Let F_{gp} be a bijective function on the message space. We call the codomain of F_{gp} the implicit message space. Com takes the CRS and a message m as input and outputs a commitment c as well some decommitment value d (aka opening value). To verify that a commitment can be opened with a message Open is used. It takes CRS, c, d, as well as some implicit message M as input and returns 1 or 0. We call the scheme *correct* if Open always returns 1 on input $(c, d) \leftarrow \mathsf{Com}(\mathsf{CRS}, m)$ and $F_{\text{gp}}(m)$. A commitment scheme is called *hiding* if any PPT adversary $\mathcal{A}$ has negligible advantage to distinguish between the commitments to two messages chosen by $\mathcal{A}$. It is called *F_{gp}-binding* if any PPT adversary has a negligible advantage to find a commitment that can be opened using two different implicit messages $M \neq M'$. Moreover, it is *equivocal* if, roughly speaking, there is a trapdoor for the CRS that allows to efficiently open a commitment with any given implicit message. Finally, the scheme is called *additively homomorphic* if commitments c_1 to m_1 and c_2 to m_2 can efficiently be combined using $\mathsf{CAdd}(c_1, c_2)$, resulting in a commitment c to $m_1 + m_2$.

Digital signatures. A digital signature scheme consists of the four algorithms SetupGrp, Gen, Sgn, and Vfy. $\mathsf{SetupGrp}(1^n)$ generates public (group) parameters gp. The key generation algorithm Gen(gp) outputs a secret key sk and a public key pk. The signing algorithm Sgn outputs a signature σ on input of a message m and sk, and the verification algorithm Vfy decides whether σ is a valid signature on m given pk, m, and σ. A signature scheme is *correct* if Vfy always outputs 1 on input $\sigma \leftarrow \mathsf{Sgn}(\mathsf{sk}, m)$, pk and m. It is called *EUF-CMA secure* if any PPT adversary $\mathcal{A}$ given pk and access to a signature oracle which signs arbitrary messages of his choice, has negligible advantage to compute a signature to a new message.

PKE. A public-key encryption (PKE) scheme consists of the four algorithms SetupGrp, Gen, Enc, and Dec. $\mathsf{SetupGrp}(1^n)$ generates public (group) parameters gp. The key generation algorithm Gen(gp) outputs a secret key sk and a public key pk. The encryption algorithm Enc(pk, m) takes pk and a message m and outputs a ciphertext c. Decryption Dec(sk, c) takes sk and c and outputs a message m or $\perp$. For *correctness*, we want that Dec always outputs m on input $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m)$. A PKE scheme is called *IND-CPA* secure if any PPT adversary $\mathcal{A}$ has negligible advantage to distinguish the ciphertexts of two messages chosen by $\mathcal{A}$.

4 BBA+ DEFINITION

In this section, we introduce BBA+ schemes along with security and privacy definitions appropriate for a variety of applications.

4.1 High-Level System Description

Let us start with an overview of the different parties involved in a BBA+ scheme and an high-level description of the algorithms and protocols they use.

A BBA+ system mainly involves five types of parties: A Trusted Third Party (TTP), an Issuer, an Accumulator, a Verifier, and a User. There might be additional separate parties in certain scenarios like a System Operator, etc.

System setup. To setup the system once, we make use of a Trusted Third Party $\mathcal{T}$ (or a number of mutually distrusting parties doing a multi-party computation). This party computes a common reference string (CRS), which typically consists of a description of the underlying algebraic framework all algorithms and protocols will use as well as certain system-wide public keys. The TTP also computes a trapdoor which can be used to remove the unlinkability of user transactions but which is only needed for definitional purposes. Of course, we need to assume that this trapdoor is not given to anyone (e. g., the Issuer, Accumulator, or Verifier). The TTP could be a (non-governmental) organization trusted by both, Users to protect their privacy and Issuers, Accumulators, and Verifiers to protect system security.

To obtain a working system, the Issuer $\mathcal{I}$ also needs to generate a key pair consisting of a public and a secret key $(\mathsf{pk}_\mathcal{I}, \mathsf{sk}_\mathcal{I})$. The secret key is shared with the Accumulator and Verifier, and can be used to create BBA+ tokens and update their balance. The public key is used to verify the authenticity of such a token.

System operation. In order to participate in the system, a user first needs to generate a key pair. The public key will be used to identify the user in the system and is assumed to be bound to a physical ID such as a passport number, social security number, etc. Of course, for this purpose the public key needs to be unique. We assume that ensuring the uniqueness of user public keys as well as verifying and binding a physical ID to them is done "out-of-band" before calling the BBA+ protocols (in particular the Issue protocol). A simple way to realize the latter could be to make use of external trusted certification authorities.

Issuing tokens. To generate a BBA+ token, a user and the issuer execute the Issue protocol. In this protocol the user proves that he is the owner of the claimed public key $\mathsf{pk}_\mathcal{U}$, for which a token should be generated using $\mathsf{sk}_\mathcal{U}$. As already explained, when this protocol is executed it has been ensured that $\mathsf{pk}_\mathcal{U}$ is unique, bound to a physical ID, and no token has been generated before under $\mathsf{pk}_\mathcal{U}$.[1] This information can be stored in a database, e. g., maintained by the issuer or a separate system operator. The user's protocol output is a BBA+ token with a balance 0.

Collecting points. To add a (postive or negative) value v to the current balance[2] w of a token, the user and the accumulator interact in the scope of the Accum protocol. As these protocol runs should be anonymous and unlinkable the accumulator is only given the secret key it shares with the issuer and the value v. It is not given and may not derive any information about the user it interacts with, provided that this user behaves honestly. The user's output is the updated token with balance $w + v$. The accumulator's output is some double-spending tag, enabling the identification of the user if he uses the old version of the token with balance w in another transaction. To this end, double-spending tags are periodically transmitted to a central database which is regularly checked for two double-spending tags associated with the same token version (expressed by having identical token version numbers). If the DB contains two such records, then algorithm IdentDS can be used to extract the public key of the user this token belongs to as well as a proof (such as his secret key) that the user is guilty. The latter can be verified using algorithm VerifyGuilt. The DB will typically be maintained by the system operator which will coincide in many scenarios with the issuer. Also, IdentDS will be run by this party. VerifyGuilt may be run by anyone, in particular by justice.

Claiming a balance and redeeming points. A user who wants to prove to some verifier that he has a valid token with balance w and possibly, as needed in some applications, redeem v points of w, will interact with the verifier in the scope of the Vfy protocol. Similar to the Accum protocol, also Vfy protocol runs should be anonymous and unlinkable. This is the reason why the verifier does only receive minimal input such as the issuer's secret key and w.[3] The outcome for the user is again an updated token of balance $w + v$ (note that v might be a negative value) which is ready to be used in the next transaction. The verifier's output is a double-spending tag just as before. This data must eventually be transferred to the database already mentioned.

4.2 Formal System Definition

The following definition formalizes our notion of extended black-box accumulation systems that are interactive, offline, and enforce the use of fresh tokens.

Definition 4.1 (BBA+ Scheme). An *extended black-box accumulation (BBA+)* scheme $\mathsf{BBAP} = (\mathsf{Setup}, \mathsf{IGen}, \mathsf{UGen}, \mathsf{Issue}, \mathsf{Accum}, \mathsf{Vfy}, \mathsf{UVer}, \mathsf{IdentDS}, \mathsf{VerifyGuilt})$ with balance and accumulation value space $\mathbb{Z}_p$ (where p may depend on CRS and, in particular, n) consists of the following PPT algorithms and interactive protocols:
$(\mathsf{CRS}, \mathsf{td}) \leftarrow \mathsf{Setup}(1^n)$ The *setup* algorithm takes the security parameter as input and returns a public common reference string CRS and a trapdoor td.[4]
$(\mathsf{pk}_\mathcal{I}, \mathsf{sk}_\mathcal{I}) \leftarrow \mathsf{IGen}(\mathsf{CRS})$ The *issuer's key generation* algorithm takes CRS as input and returns a public and private key pair $(\mathsf{pk}_\mathcal{I}, \mathsf{sk}_\mathcal{I})$, where the $\mathsf{sk}_\mathcal{I}$ will be shared with accumulator and verifier. We assume for convenience that CRS will be part of $\mathsf{pk}_\mathcal{I}$.
$(\mathsf{pk}_\mathcal{U}, \mathsf{sk}_\mathcal{U}) \leftarrow \mathsf{UGen}(\mathsf{CRS})$ The *user's key generation* algorithm takes CRS as input and returns a public-private key pair $(\mathsf{pk}_\mathcal{U}, \mathsf{sk}_\mathcal{U})$ which is used for authentication during token issuance.
$((\tau, b_\mathcal{U}), b_\mathcal{I}) \leftarrow \mathsf{Issue}\langle\mathcal{U}(\mathsf{pk}_\mathcal{I}, \mathsf{pk}_\mathcal{U}, \mathsf{sk}_\mathcal{U}), \mathcal{I}(\mathsf{pk}_\mathcal{I}, \mathsf{sk}_\mathcal{I}, \mathsf{pk}_\mathcal{U})\rangle$ The interactive *token issuing* protocol is executed between a user $\mathcal{U}$, given $\mathsf{pk}_\mathcal{I}$ and his own public and private key $\mathsf{pk}_\mathcal{U}, \mathsf{sk}_\mathcal{U}$ as input, and an issuer $\mathcal{I}$, whose input is $\mathsf{pk}_\mathcal{I}, \mathsf{sk}_\mathcal{I}$ and the public-key $\mathsf{pk}_\mathcal{U}$ of the user $\mathcal{U}$. At the end of the protocol, the user outputs a token τ (with balance 0) and a bit $b_\mathcal{U}$, and the issuer a bit $b_\mathcal{I}$. The bit $b_\mathcal{U}$ (resp. $b_\mathcal{I}$) indicate whether $\mathcal{U}$ (resp. $\mathcal{I}$) accepts the protocol run.
$((\tau^*, b_\mathcal{U}), (\mathsf{dstag}, \mathsf{hid}, b_\mathcal{AC})) \leftarrow \mathsf{Accum}\langle\mathcal{U}(\mathsf{pk}_\mathcal{I}, \mathsf{pk}_\mathcal{U}, \mathsf{sk}_\mathcal{U}, \tau, w, v), \mathcal{AC}(\mathsf{pk}_\mathcal{I}, \mathsf{sk}_\mathcal{I}, v)\rangle$ The interactive *accumulation* protocol is executed between a user $\mathcal{U}$ and an accumulator $\mathcal{AC}$. The user's input is $\mathsf{pk}_\mathcal{I}$, his own public and private key $\mathsf{pk}_\mathcal{U}, \mathsf{sk}_\mathcal{U}$, a token τ with balance w, and the value v. The accumulator's input is $\mathsf{pk}_\mathcal{I}, \mathsf{sk}_\mathcal{I}$, and the value v. At the end of the protocol, the user outputs an updated token τ^* (with balance $w + v$) and a bit $b_\mathcal{U}$. The issuer's output consists of some double spending tag $\mathsf{dstag} = (s, z)$ with token version number s and data z, as well as a hidden user ID hid,[5] as well as a bit $b_\mathcal{AC}$. The bit $b_\mathcal{U}$ (resp. $b_\mathcal{AC}$) indicate whether $\mathcal{U}$ (resp. $\mathcal{AC}$) accepts the protocol run.
$((\tau^*, b_\mathcal{U}), (\mathsf{dstag}, \mathsf{hid}, b_\mathcal{V})) \leftarrow \mathsf{Vfy}\langle\mathcal{U}(\mathsf{pk}_\mathcal{I}, \mathsf{pk}_\mathcal{U}, \mathsf{sk}_\mathcal{U}, \tau, w, v), \mathcal{V}(\mathsf{pk}_\mathcal{I}, \mathsf{sk}_\mathcal{I}, w, v)\rangle$ The interactive *verification and redeeming* protocol is run between a user $\mathcal{U}$ and a verifier $\mathcal{V}$. The inputs and outputs are analogous to those of the Accum protocol, except that

[1] Note that it is possible for a user to have more than one token by allowing him to have more than one public key bound to his name.
[2] Note that the semantics of w is not necessarily fixed to be simply the sum of collected points. For instance, one could also encode two counters into w, one for positive points and one for negative points.
[3] Note that in certain scenarios revealing w may help to link transactions. For such applications, the framework can be extended to only show a bound on the balance or to perform a range proof. However, we deliberately avoid such an proof due to performance reasons. See Appendix G for a discussion.
[4] The trapdoor is needed in the security definition to define the legitimate balance of a token despite token transactions should be unlinkable.
[5] hid is used for definitorial purposes only. In our instantiation, hid will be an encryption of $\mathsf{pk}_\mathcal{U}$.

$\mathcal{V}$ has the current balance w as an additional input, which $\mathcal{U}$ must reveal before the protocol is run.
$b \leftarrow \mathsf{UVer}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, \tau, w)$ The *token verification* algorithm is a non-probabilistic polynomial-time algorithm run by $\mathcal{U}$ which, given $\mathsf{pk}_{\mathcal{I}}$, the user's public and secret key $\mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}$, a token τ, and balance w, outputs a bit b. This bit is 1 if τ is a valid token with balance w owned by the user with public key $\mathsf{pk}_{\mathcal{U}}$.
$(\mathsf{pk}_{\mathcal{U}}, \Pi) \leftarrow \mathsf{IdentDS}(\mathsf{pk}_{\mathcal{I}}, (s_1, z_1), (s_2, z_2))$ The *double-spender detection* algorithm is a non-probabilistic polynomial-time algorithm which is given $\mathsf{pk}_{\mathcal{I}}$ and two double-spending tags (s_1, z_1) and (s_2, z_2). It returns the public key $\mathsf{pk}_{\mathcal{U}}$ of a user and a proof of guilt Π, or it returns an error $\perp$.
$b \leftarrow \mathsf{VerifyGuilt}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \Pi)$ The *guilt verification* algorithm is a non-probabilistic polynomial-time algorithm which is given $\mathsf{pk}_{\mathcal{I}}$, a user public key $\mathsf{pk}_{\mathcal{U}}$ and a proof of guilt Π. It returns 1 if the user with public key $\mathsf{pk}_{\mathcal{U}}$ is considered guilty of double-spending and 0 otherwise.

We defer the straightforward definition of correctness of a *BBA+* scheme to the appendix, see Appendix A.

4.3 System Security

For security we will distinguish between a reduced, simplified model and a more natural, full-fledged model. In the full-fledged model, (cf. Appendix E) the considered adversary can be a collusion of malicious users who additionally may command, eavesdrop on, and adaptively corrupt honest users. In the simplified model, introduced in the following, no interactions with honest users are considered. Fortunately, we can show in a black-box fashion that any scheme which is secure in our reduced model is also secure in the full-fledged model when all protocol messages are additionally encrypted with an IND-CCA secure encryption scheme (cf. Appendix F). Note that privacy will not be affected by extending the protocols with encryption.

With our security definition we will essentially capture three properties:

(1) A token may only be used by its legitimate owner (i. e., the user it was issued to).
(2) For a token one may only claim exactly the amount of points that have legitimately been collected with this token up to this point unless an old version of this token is presented.
(3) Users presenting old tokens can be identified (after the fact).

Formalizing the above notion raises a major problem: It requires to link each transaction with a user and token. However, on the other hand, we will demand that transactions are anonymous and unlinkable. To resolve this issue, we only consider systems where privacy can be abolished given a trapdoor td (which is kept secret by the TTP) to the CRS. We call such schemes *trapdoor-linkable* and formalize them in the following.

When we talk about a successful protocol run in the following, we always mean that this run has been accepted by the issuer, accumulator, or verfier. Let $\mathcal{AC}$'s view of a run of the Accum protocol consist of all its inputs, outputs, and messages sent and received, i. e., $(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}, v, \mathsf{msgs}, s, z, \mathsf{hid}, b_{\mathcal{AC}})$, where $\mathsf{msgs} \in \{0,1\}^*$ is the bitstring of messages sent during the protocol run. Similarly, let $\mathcal{V}$'s view of a run of the Vfy protocol be represented by a tuple $(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}, w, v, \mathsf{msgs}, s, z, \mathsf{hid}, b_{\mathcal{V}})$. For some fixed security parameter $n \in \mathbb{N}$ and $\mathsf{CRS} \leftarrow \mathsf{Setup}(1^n)$ let us consider the set of views of $\mathcal{AC}$, denoted by $\mathcal{V}^{\mathsf{Accum}}_{n,\mathsf{CRS}}$, resulting from any Accum protocol run accepted by $\mathcal{AC}$ with any (possibly malicious) party and any $(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}) \leftarrow \mathsf{IGen}(\mathsf{CRS})$, $v \in \mathbb{Z}_p$ as input to $\mathcal{AC}$. We define $\mathcal{V}^{\mathsf{Vfy}}_{n,\mathsf{CRS}}$ analogously with respect to executions of the Vfy protocol accepted by $\mathcal{V}$.

Definition 4.2 (Trapdoor-Linkability). A BBA+ scheme BBAP is called *trapdoor-linkable* if it satisfies the following conditions:

(1) *Completeness.* Let $n \in \mathbb{N}$, $(\mathsf{CRS}, \mathsf{td}) \leftarrow \mathsf{Setup}(1^n)$, and $\mathsf{view} \in \mathcal{V}^{\mathsf{Accum}}_{n,\mathsf{CRS}}$. Let hid denote the hidden user ID contained in view. Then there exist inputs $\mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, \tau, w$, and random choices for an honest user $\mathcal{U}$ and honest accumulator $\mathcal{AC}$ such that an Accum protocol run between $\mathcal{U}$ and $\mathcal{AC}$ with these inputs and random choices leads to a view $\mathsf{view}' \in \mathcal{V}^{\mathsf{Accum}}_{n,\mathsf{CRS}}$ containing the same hidden user ID hid as view. The same holds for all $\mathsf{view} \in \mathcal{V}^{\mathsf{Vfy}}_{n,\mathsf{CRS}}$ with respect to Vfy.
(2) *Extractability.* There exists a PPT algorithm ExtractUID such that for any $n \in \mathbb{N}$, $(\mathsf{CRS}, \mathsf{td}) \leftarrow \mathsf{Setup}(1^n)$ and $\mathsf{view} = (\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}, v, \mathsf{msgs}, s, z, \mathsf{hid}, 1) \in \mathcal{V}^{\mathsf{Accum}}_{n,\mathsf{CRS}}$ resulting from an Accum protocol run with an honest user on input $\mathsf{pk}_{\mathcal{U}}$, ExtractUID outputs $\mathsf{pk}_{\mathcal{U}}$ on input $(\mathsf{td}, \mathsf{hid})$.[6] The same needs to hold for ExtractUID with respect to views $\mathsf{view} \in \mathcal{V}^{\mathsf{Vfy}}_{n,\mathsf{CRS}}$.

Remark 1. *Note that extractability as defined above implies that any fixed view* view *cannot result from interactions with different users, but is uniquely associated with a single user. Furthermore, by demanding completeness we prevent the use of some odd extraction algorithms that output some special user public key on input of a specifically crafted* hid *that only an adversary is able to generate but not an honest user. Such extraction algorithms may lead to some issues when used in our security definition.*

In the security experiments we are going to formalize, an adversary $\mathcal{A}$ may interact with an honest issuer, accumulator, and verifier an arbitrary number of times and concurrently. Clearly, the adversary playing the role of the user may behave dishonestly and not follow the corresponding protocols. In order to formalize this adversarial setting, we define a couple of oracles the adversary may query.

- $\mathsf{MalIssue}(\mathsf{pk}_{\mathcal{U}})$ 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 Accum protocol with $\mathcal{AC}$ for input $v \in \mathbb{Z}_p$.
- $\mathsf{MalVer}(w, v)$ is used by the adversary to initiate the Vfy protocol with $\mathcal{V}$ for input $w \in \mathbb{Z}_p$ and $v \in \mathbb{Z}_p$.

In the setting described above, we consider several adversarial goals. The first two goals formalized in Theorems 4.3 and 4.4 cover

[6] More generally, we could give ExtractUID the whole view as input. However, this may significantly complicate security proofs as we would need some form of completeness for all building blocks.

Experiment $\mathsf{Exp}^{\text{ob-issue}}_{\mathsf{BBAP},\mathcal{A}}(n)$
$(\mathsf{CRS}, \mathsf{td}) \leftarrow \mathsf{Setup}(1^n)$
$(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}) \leftarrow \mathsf{IGen}(\mathsf{CRS})$
$(\mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}) \leftarrow \mathsf{UGen}(\mathsf{CRS})$
$b \leftarrow \mathcal{A}^{\mathsf{MalIssue},\mathsf{MalAcc},\mathsf{MalVer}}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}})$
The experiment returns 1 iff $\mathcal{A}$ did a successful call to MalIssue on input of the given public-key $\mathsf{pk}_{\mathcal{U}}$.

Figure 1: Owner-binding experiment for Issue**.**

Experiment $\mathsf{Exp}^{\text{ob-acc}}_{\mathsf{BBAP},\mathcal{A}}(n)$
$(\mathsf{CRS}, \mathsf{td}) \leftarrow \mathsf{Setup}(1^n)$
$(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}) \leftarrow \mathsf{IGen}(\mathsf{CRS})$
$b \leftarrow \mathcal{A}^{\mathsf{MalIssue},\mathsf{MalAcc},\mathsf{MalVer}}(\mathsf{pk}_{\mathcal{I}})$
The experiment returns 1 iff $\mathcal{A}$ did a successful call to MalAcc or MalVer such that ExtractUID applied to hid being part of the view of this call outputs a public-key $\mathsf{pk}_{\mathcal{U}}$ for which there has been no successful execution of MalIssue up to this call.

Figure 2: Owner-binding experiment for Accum/Vfy**.**

the owner-binding property, already mentioned, with respect to the different protocols Issue, Accum, and Vfy. Theorem 4.5 formalizes the balance-binding property assuming that no double-spending took place. Consequently, Theorem 4.6 ensures that such double-spendings are indeed hard to accomplish without being identified (and punished).

In Theorem 4.3, we consider the probability that an adversary may succeed in receiving a token in the name of an honest, uncorrupted user (i. e., using the user's public-key). It demands that an adversary may only create tokens in his own name.

Definition 4.3. A trapdoor-linkable BBA+ scheme BBAP is called owner-binding with respect to Issue if for any PPT adversary $\mathcal{A}$ in the experiment $\mathsf{Exp}^{\text{ob-issue}}_{\mathsf{BBAP},\mathcal{A}}(n)$ from Fig. 1 the advantage of $\mathcal{A}$ defined by

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

is negligible in n.

Theorem 4.4 demands that an adversary may not be able to successfully call the accumulation or verification protocol for a forged token, i. e. a token that has not been issued by a legitimate issuer.

Definition 4.4. A trapdoor-linkable BBA+ scheme BBAP is called owner-binding with respect to Accum and Vfy if for any PPT adversary $\mathcal{A}$ in the experiment $\mathsf{Exp}^{\text{ob-acc}}_{\mathsf{BBAP},\mathcal{A}}(n)$ from Fig. 2 the advantage of $\mathcal{A}$ defined by

$$\mathsf{Adv}^{\text{ob-acc}}_{\mathsf{BBAP},\mathcal{A}}(n) := \Pr[\mathsf{Exp}^{\text{ob-acc}}_{\mathsf{BBAP},\mathcal{A}}(n) = 1] \tag{2}$$

is negligible in n.

With Theorem 4.5 we ensure that, unless some token is used twice (which induces the usage of the same token serial number), the claimed balance for a token in the scope of the verification protocol always coincides with sum of points allegedly collected with this token. Note that if this property is violated, then this could mean that the claimed balance is not equal to the "real" balance of the token or that the "real" balance does not coincide with the sum of legitimately collected points associated with this token we have in the records .

Experiment $\mathsf{Exp}^{\text{bb}}_{\mathsf{BBAP},\mathcal{A}}(n)$
$(\mathsf{CRS}, \mathsf{td}) \leftarrow \mathsf{Setup}(1^n)$
$(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}) \leftarrow \mathsf{IGen}(\mathsf{CRS})$
$b \leftarrow \mathcal{A}^{\mathsf{MalIssue},\mathsf{MalAcc},\mathsf{MalVer}}(\mathsf{pk}_{\mathcal{I}})$
The experiment returns 1 iff $\mathcal{A}$ did a successful call to MalVer resulting in a view view = $(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}, w, v, \mathsf{msgs}, s, z, \mathsf{hid}, 1) \in \mathcal{V}^{\mathsf{Vfy}}_{n,\mathsf{CRS}}$ and extracted user public-key $\mathsf{pk}_{\mathcal{U}} \leftarrow \mathsf{ExtractUID}(\mathsf{td}, \mathsf{hid})$ such that the following conditions are satisfied:
- all successful MalIssue/MalAcc calls resulted in different token version numbers and
- the claimed balance $w \in \mathbb{Z}_p$ 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 $v \in \mathbb{Z}_p$ that appeared in previous successfull calls to MalAcc or MalVer for which $\mathsf{pk}_{\mathcal{U}}$ could be extracted using ExtractUID.

Figure 3: Balance binding experiment.

Experiment $\mathsf{Exp}^{\text{dsd}}_{\mathsf{BBAP},\mathcal{A}}(n)$
$(\mathsf{CRS}, \mathsf{td}) \leftarrow \mathsf{Setup}(1^n)$
$(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}) \leftarrow \mathsf{IGen}(\mathsf{CRS})$
$b \leftarrow \mathcal{A}^{\mathsf{MalIssue},\mathsf{MalAcc},\mathsf{MalVer}}(\mathsf{pk}_{\mathcal{I}})$
The experiment returns 1 iff $\mathcal{A}$ did two successful MalAcc/MalVer calls resulting in two views view_1 and view_2 including two double-spending tags $\mathsf{dstag}_1 = (s, z_1)$ and $\mathsf{dstag}_2 = (s, z_2)$ and extracted user public-keys $\mathsf{pk}^{(1)}_{\mathcal{U}}$ and $\mathsf{pk}^{(2)}_{\mathcal{U}}$ (using ExtractUID) such that one of the following conditions is satisfied:
- $\mathsf{pk}^{(1)}_{\mathcal{U}} \neq \mathsf{pk}^{(2)}_{\mathcal{U}}$ or
- $\mathsf{IdentDS}(\mathsf{pk}_{\mathcal{I}}, \mathsf{dstag}_1, \mathsf{dstag}_2) \neq (\mathsf{pk}^{(1)}_{\mathcal{U}}, \Pi)$ or
- $\mathsf{IdentDS}(\mathsf{pk}_{\mathcal{I}}, \mathsf{dstag}_1, \mathsf{dstag}_2) = (\mathsf{pk}^{(1)}_{\mathcal{U}}, \Pi)$ but $\mathsf{VerifyGuilt}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}^{(1)}_{\mathcal{U}}, \Pi) = 0$

Figure 4: Double-spending detection experiment.

Definition 4.5. A trapdoor-linkable BBA+ scheme BBAP is called balance-binding if for any PPT adversary $\mathcal{A}$ in the experiment $\mathsf{Exp}^{\text{bb}}_{\mathsf{BBAP},\mathcal{A}}(n)$ from Fig. 3 the advantage of $\mathcal{A}$ defined by

$$\mathsf{Adv}^{\text{bb}}_{\mathsf{BBAP},\mathcal{A}}(n) := \Pr[\mathsf{Exp}^{\text{bb}}_{\mathsf{BBAP},\mathcal{A}}(n) = 1] \tag{3}$$

is negligible in n.

Theorem 4.6 enforces that two transactions leading to the same token version number have always been initiated by the same user and this user can be identified.

Definition 4.6. A trapdoor-linkable BBA+ scheme BBAP ensures double-spending detection if for any PPT adversary $\mathcal{A}$ in the experiment $\mathsf{Exp}^{\text{dsd}}_{\mathsf{BBAP},\mathcal{A}}(n)$ from Fig. 4 the advantage of $\mathcal{A}$ defined by

$$\mathsf{Adv}^{\text{dsd}}_{\mathsf{BBAP},\mathcal{A}}(n) := \Pr[\mathsf{Exp}^{\text{dsd}}_{\mathsf{BBAP},\mathcal{A}}(n) = 1] \tag{4}$$

is negligible in n.

4.4 User Security and Privacy

This section presents the key security properties for users, protecting them from a dishonest issuer: Firstly, a user should have the privacy guarantee that its individual interactions cannot be exploited for tracking and secondly no operator should be able to forge a proof that a user has allegedly committed a double-spending.

Our privacy definition essentially demands that an adversary, where this could be a collusion of $\mathcal{I}$, $\mathcal{AC}$, and $\mathcal{V}$, may not be able to link the Accum and Vfy transactions of an honest user.

More precisely, it demands that Accum and Vfy do not reveal any information (except for the balance in case of Vfy) that may help in linking transactions. This even needs to hold for transactions preceding and succeeding the corruption (except the very next) of the user thereby leaking all of his secrets to the adversary. Hence, we define a form of *forward and backward* privacy. To this end, our definition follows the real/ideal world paradigm.

In the *real world*, depicted in Fig. 5, first the CRS is generated honestly and the adversary setups a public system key $\mathsf{pk}_{\mathcal{I}}$ of his choice. Then the adversary is allowed to interact with a couple of oracles, that allow the adversary to create a number of honest users and instruct these users to interact with him in the scope of Issue, Accum, or Vfy. Within these interactions the oracles play the role of the honest user, while the adversary plays the issuer, accumulator, or verifier. Whenever an interaction does not successfully terminate from the user's perspective, then this particular user refuses to participate in any future interaction, i. e. the oracles are blocked for the respective $\mathsf{pk}_{\mathcal{U}}$.[7] If the adversary calls an oracle for a blocked user, the oracle simply sends $\perp$-messages. Moreover, for each user no oracle can be called concurrently, i. e. for any arbitrary but fixed $\mathsf{pk}_{\mathcal{U}}$ another oracle can only invoked if no previous oracle call for the same $\mathsf{pk}_{\mathcal{U}}$ is still pending. The oracles the adversary can access are:

- HonUser() creates a new user entity by running $(\mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}) \leftarrow \mathsf{UGen}(\mathsf{CRS})$. The oracle returns $\mathsf{pk}_{\mathcal{U}}$ to the adversary.
- RealHonIssue($\mathsf{pk}_{\mathcal{U}}$) lets the user with public key $\mathsf{pk}_{\mathcal{U}}$ run the Issue protocol with the adversary impersonating the issuer $\mathcal{I}$, provided that $\mathsf{pk}_{\mathcal{U}}$ has not been used before in a call to RealHonIssue which was successful from the user's perspective.
- RealHonAcc($\mathsf{pk}_{\mathcal{U}}, v$) lets the user with public key $\mathsf{pk}_{\mathcal{U}}$ run the Accum protocol with the adversary impersonating the accumulator $\mathcal{AC}$ on common input $v \in \mathbb{Z}_p$, provided that RealHonIssue($\mathsf{pk}_{\mathcal{U}}$) has successfully been called at some point before.
- RealHonVer($\mathsf{pk}_{\mathcal{U}}, w, v$) lets the user with public key $\mathsf{pk}_{\mathcal{U}}$ run the Vfy protocol with the adversary impersonating the verifier $\mathcal{V}$ on common input $v \in \mathbb{Z}_p$ and $w \in \mathbb{Z}_p$, provided that RealHonIssue($\mathsf{pk}_{\mathcal{U}}$) has successfully been called at some point before.
- RealCorrupt($\mathsf{pk}_{\mathcal{U}}$) can be called by the adversary to corrupt the user with public and secret key $(\mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}) \leftarrow$ HonUser(). The oracle outputs the secret key $\mathsf{sk}_{\mathcal{U}}$ of the user as well as the user's most recent token τ to the adversary.

At the end of the game, the adversary outputs a bit.

In the *ideal world*, depicted in Fig. 6, first a CRS along with a simulation trapdoor $\mathsf{td}_{\mathsf{sim}}$ is generated honestly and the adversary setups a public system key $\mathsf{pk}_{\mathcal{I}}$ of his choice, whereby the adversary only receives the CRS as in the real game. Then, the adversary may act exactly like in the real game, by accessing a number of oracles. However, compared to the real world, some oracles are implemented differently in order not to leak information that allows to link transactions. Naturally, the oracles SimHonIssue, SimHonAcc and SimHonVer all receive the public system key $\mathsf{pk}_{\mathcal{I}}$ and the simulation trapdoor $\mathsf{td}_{\mathsf{sim}}$. Although these oracles do not get a full token as input it is unavoidable that in addition to the trapdoor some more information is still provided to the oracles as otherwise simulation becomes impossible: SimHonIssue additionally gets the user's public key $\mathsf{pk}_{\mathcal{U}}$, SimHonAcc gets the accumulation value v and SimHonVer gets the token balance w. The Issue protocol binds $\mathsf{pk}_{\mathcal{U}}$ to a (physical) identity. Hence the issuer expects to see $\mathsf{pk}_{\mathcal{U}}$ as part of the public ZK-statement and $\mathsf{pk}_{\mathcal{U}}$ cannot be omitted. As in our privacy game a single operator plays the role of $\mathcal{I}$, $\mathcal{AC}$ and $\mathcal{V}$ and the single operator dictates with whom it will interact the operator can keep track of all balances per user itself. Consequently, SimHonVer must present the correct balance and thus the ideal game will provide the user oracle with that information.

Also, SimCorrupt needs to come up with plausible secrets like a user's secret key and a token which are "compliant" with the user's previous transactions. After a successful corruption, the oracles run the real protocol in the subsequent call for the affected $\mathsf{pk}_{\mathcal{U}}$ using the "real" secrets for input as returned by SimCorrupt .[8] This means the oracles in the ideal game, distinguish two modes of operation: If the immediately previous call for $\mathsf{pk}_{\mathcal{U}}$ has been a SimCorrupt call, then the oracle just runs the code of a regular honest user $\mathcal{U}$ as in the real game. Otherwise, the oracle runs a user simulation algorithm that *does not* receive any user-related input as described above.

In the ideal game the adversary interacts with the following oracles:

- HonUser() creates a new user entity by running $(\mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}) \leftarrow \mathsf{UGen}(\mathsf{CRS})$. The oracle returns $\mathsf{pk}_{\mathcal{U}}$ to the adversary.
- SimHonIssue($\mathsf{pk}_{\mathcal{U}}$) lets the user with public key $\mathsf{pk}_{\mathcal{U}}$ run the Issue protocol with the adversary impersonating the issuer $\mathcal{I}$, provided that $\mathsf{pk}_{\mathcal{U}}$ has not been used before in a call to SimHonIssue which was successful from the user's perspective. The internal user simulation algorithm $\mathcal{U}_{\mathsf{sim}}$ gets the simulation trapdoor $\mathsf{td}_{\mathsf{sim}}$ and $\mathsf{pk}_{\mathcal{U}}$.
- SimHonAcc($\mathsf{pk}_{\mathcal{U}}, v$) lets the user with public key $\mathsf{pk}_{\mathcal{U}}$ run the Accum protocol with the adversary impersonating $\mathcal{AC}$ on common input $v \in \mathbb{Z}_p$, provided that SimHonIssue($\mathsf{pk}_{\mathcal{U}}$) has successfully been called at some point before. The internal user simulation algorithm $\mathcal{U}_{\mathsf{sim}}$ gets the simulation trapdoor $\mathsf{td}_{\mathsf{sim}}$ and v.

Experiment $\mathsf{Exp}^{\text{priv-real}}_{\text{BBAP},\mathcal{A}}(1^n)$
$(\mathsf{CRS}, \mathsf{td}) \leftarrow \mathsf{Setup}(1^n)$
$(\mathsf{pk}_{\mathcal{I}}, state_0) \leftarrow \mathcal{A}_0(\mathsf{CRS})$
$b \leftarrow \mathcal{A}_1^{\mathsf{HonUser},\mathsf{RealHonIssue},\mathsf{RealHonAcc},\mathsf{RealHonVer},\mathsf{RealCorrupt}}(\mathsf{pk}_{\mathcal{I}}, state_0)$
return b

Figure 5: Real world privacy experiment.

[7] We need to demand that any previous call for $\mathsf{pk}_{\mathcal{U}}$ was successful as otherwise an adversary may simply abort an Accum or Vfy transaction or start two such interactions in parallel and then trigger the user to double-spend its token in a subsequent call which would reveal the user's identity. If the adversary has tried to cheat and has been successfully detected by the user doing so, then this user "leaves" the system.

[8] This is required as the adversary, given all user secrets, may now perform the next interaction of this user on his own and learn a double-spending tag. Running the simulator now needs to result in a second double-spending tag that can be used with the first one to reveal $\mathsf{pk}_{\mathcal{U}}$, which is, however, unknown to the simulator.

Experiment $\mathsf{Exp}^{\text{priv-ideal}}_{\mathsf{BBAP},\mathcal{A}}(1^n)$
$(\mathsf{CRS}, \mathsf{td}_{\mathsf{sim}}) \leftarrow \mathsf{SimSetup}(1^n)$
$(\mathsf{pk}_{\mathcal{I}}, state_0) \leftarrow \mathcal{A}_0(\mathsf{CRS})$
$b \leftarrow \mathcal{A}_1^{\mathsf{HonUser},\mathsf{SimHonIssue},\mathsf{SimHonAcc},\mathsf{SimHonVer},\mathsf{SimCorrupt}}(\mathsf{pk}_{\mathcal{I}}, state_0)$
return b

Figure 6: Ideal world privacy experiment.

Experiment $\mathsf{Exp}^{\text{facp}}_{\mathsf{BBAP},\mathcal{A}}(n)$
$(\mathsf{CRS}, \mathsf{td}) \leftarrow \mathsf{Setup}(1^n)$
$(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}) \leftarrow \mathsf{IGen}(\mathsf{CRS})$
$(\mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}) \leftarrow \mathsf{UGen}(\mathsf{CRS})$
$\Pi \leftarrow \mathcal{A}^{\mathsf{RealHonIssue},\mathsf{RealHonAcc},\mathsf{RealHonVer}}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}})$
The experiment returns 1 iff $\mathsf{VerifyGuilt}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \Pi) = 1$. (The adversary may only call the RealHonIssue, RealHonAcc and RealHonVer oracles on input of the key $\mathsf{pk}_{\mathcal{U}}$ chosen by the experiment.)

Figure 7: False accusation protection experiment.

- $\mathsf{SimHonVer}(\mathsf{pk}_{\mathcal{U}}, w, v)$ lets a user with public key $\mathsf{pk}_{\mathcal{U}}$ run the Vfy protocol with the adversary impersonating the verifier $\mathcal{V}$ on common input $v \in \mathbb{Z}_p$ and $w \in \mathbb{Z}_p$, provided that $\mathsf{SimHonIssue}(\mathsf{pk}_{\mathcal{U}})$ has successfully been called at some point before. The internal user simulation algorithm $\mathcal{U}_{\mathsf{sim}}$ gets the simulation trapdoor $\mathsf{td}_{\mathsf{sim}}$, v and w.
- $\mathsf{SimCorrupt}(\mathsf{pk}_{\mathcal{U}})$ can be called by the adversary to corrupt the user owning the keys $(\mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}})$. The oracle outputs some secret key $\mathsf{sk}'_{\mathcal{U}}$ as well as some token τ'.

At the end of the game, the adversary also outputs a bit.

As already mentioned, for privacy we demand that the real and the ideal world are computationally indistinguishable.

Definition 4.7. We say that a BBA+ scheme BBAP is *privacy-preserving*, if there exist PPT algorithms SimSetup and SimCorrupt as well as interactive PPT algorithms as described in HonIssue, SimHonAcc and SimHonVer, respectively, such that for all PPT adversaries $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ in the experiments from Figs. 5 and 6, the advantage $\mathsf{Adv}^{\text{priv}}_{\mathsf{BBAP},\mathcal{A}}(n)$ of $\mathcal{A}$ defined by

$$\left|\Pr[\mathsf{Exp}^{\text{priv-real}}_{\mathsf{BBAP},\mathcal{A}}(n) = 1] - \Pr[\mathsf{Exp}^{\text{priv-ideal}}_{\mathsf{BBAP},\mathcal{A}}(n) = 1]\right| \tag{5}$$

is negligible in n.

Remark 2. *Our privacy notion is a bit stronger than what is intuitively required. Replacing* RealHonIssue *by* SimHonIssue *could be omitted without loosing any privacy guarantee in a real-world application. The purpose of privacy is to prevent the operators from tracking a user between individual interactions. However, if the user already gets corrupted* before *the first interaction in the scope of* HonAcc *or* HonVer*, then the user has actually never participated in the system as an uncorrupted user but has been under control of the malicious operator right from the beginning. In that case speaking about privacy is pointless. Nonetheless, we introduce a simulated version* SimHonIssue *as this kind of privacy notion directly implies protection against false accusation (see below).*

Finally, Theorem 4.8 demands that honest users cannot be falsely accused of having committed a double-spending by an adversary who generates $\mathsf{pk}_{\mathcal{I}}$ and may coincide with $\mathcal{I}$, $\mathcal{AC}$, or $\mathcal{V}$.

Definition 4.8. A trapdoor-linkable BBA+ scheme BBAP ensures false-accusation protection if for any PPT adversary $\mathcal{A}$ in the experiment $\mathsf{Exp}^{\text{facp}}_{\mathsf{BBAP},\mathcal{A}}(n)$ from Fig. 7 the advantage of $\mathcal{A}$ defined by

$$\mathsf{Adv}^{\text{facp}}_{\mathsf{BBAP},\mathcal{A}}(n) := \Pr[\mathsf{Exp}^{\text{facp}}_{\mathsf{BBAP},\mathcal{A}}(n) = 1] \tag{6}$$

is negligible in n.

5 A BBA+ INSTANTIATION

In this section, we present our "base scheme" BBAP which is secure with respect to the "reduced model" presented in Section 4.3 and privacy-preserving with respect to the model in Section 4.4. As already mentioned, this base protocol can easily be made secure in an extended model where eavesdropping on and corrupting of honest users is allowed, by encrypting all messages transmitted during the protocols Issue, Accum and Vfy. Please refer to Appendix F.1 for details.

5.1 Building Blocks

Let SetupGrp be a bilinear group generator (cf. Definition 2.1) which outputs the description of a bilinear group $\mathsf{gp} := (G_1, G_2, G_T, e, p, g_1, g_2) \leftarrow \mathsf{SetupGrp}(1^n)$ for which the SXDH problem is assumed to be hard. For our construction, we draw from the following building blocks which all make use of SetupGrp as their common group setup algorithm.

NIZKs. For proving correctness of the computations in the scope of the Issue, Accum, and Vfy protocol taking place on the user's side we make use of $F^{(1)}_{\mathsf{gp}}$-, $F^{(2)}_{\mathsf{gp}}$, and $F^{(3)}_{\mathsf{gp}}$-extractable NIZK proof systems, denoted by P1, P2, and P3, respectively. The functions $F^{(1)}_{\mathsf{gp}}$, $F^{(2)}_{\mathsf{gp}}$, and $F^{(3)}_{\mathsf{gp}}$ depend on the considered languages L_1, L_2, and L_3 (defined below), but they have the following in common: They behave as the identity function with respect to group elements and map elements from $\mathbb{Z}_p$ either to G_1 or G_2 (by exponentiation with basis g_1 or g_2) depending on whether these are used as exponents of a G_1 or G_2 element in the language. The proof systems will share a common reference string. More precisely, we demand that there is a shared extraction setup algorithm which generates the CRS and also a single extraction trapdoor for P1, P2, and P3. Let us denote this algorithm by SetupEPoK and its output by $(\mathsf{CRS}_{\mathsf{pok}}, \mathsf{td}_{\mathsf{epok}}) \leftarrow \mathsf{SetupEPoK}(\mathsf{gp})$ in the following. Furthermore, let us denote the prove and verify algorithms of these proof systems by PX.Prove and PX.Vfy, for $1 \le X \le 3$. We will make use of an SXDH-based instantiation of Groth-Sahai proofs [20] for this purpose. Note that GS proofs are not always zero-knowledge (cf. Appendix B.4), but we ensured that they indeed are for the languages we consider.

Homomorphic commitments. In order to form a token and commit to secrets including the user secret key and the token balance, we will make use of an equivocal F'_{gp}-binding homomorphic commitment scheme C for messages from $\mathbb{Z}_p^4$. The commitment space will be G_2. The function F'_{gp} will map $m := (m_1, m_2, m_3, m_4)$ to $M := (g_1^{m_1}, g_1^{m_2}, g_1^{m_3}, g_1^{m_4})$, so G_1^4 is the implicit message space. Moreover, as the user needs to be able to prove that it can open a commitment, the corresponding verification equations must be compatible with our proof systems. We will use a scheme by Abe

Setup(1^n)
$\mathsf{gp} := (p, G_1, G_2, G_T, e, p, g_1, g_2) \leftarrow \mathsf{SetupGrp}(1^n)$
$(\mathsf{CRS}_{\mathsf{pok}}, \mathsf{td}_{\mathsf{epok}}) \leftarrow \mathsf{SetupEPoK}(\mathsf{gp})$
$\mathsf{CRS}_{\mathsf{com}} \leftarrow \mathsf{C.Setup}(\mathsf{gp})$
$(\mathsf{sk}_{\mathcal{T}}, \mathsf{pk}_{\mathcal{T}}) \leftarrow \mathsf{E.Gen}(\mathsf{gp})$
$\mathsf{CRS} := (\mathsf{gp}, \mathsf{CRS}_{\mathsf{com}}, \mathsf{pk}_{\mathcal{T}}, \mathsf{CRS}_{\mathsf{pok}})$
$\mathsf{td} := (\mathsf{td}_{\mathsf{epok}}, \mathsf{sk}_{\mathcal{T}})$
return (CRS, td)

IGen(CRS)
$(\mathsf{pk}_{\mathsf{sig}}, \mathsf{sk}_{\mathsf{sig}}) \leftarrow \mathsf{S.Gen}(\mathsf{CRS})$
return $(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}) := ((\mathsf{CRS}, \mathsf{pk}_{\mathsf{sig}}), \mathsf{sk}_{\mathsf{sig}})$

UGen(CRS)
$y \leftarrow \mathbb{Z}_p$
$(\mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}) := (g_1^y, y)$
return $(\mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}})$

Figure 8: Setup and Key Generation

et al. [2] for this purpose. We denote the CRS generator, commitment, and openening algorithms by C.Setup, C.Com, and C.Open, respectively. Furthermore, as CAdd conincides with multiplication of commitments. We denote this operation by "·".

Signatures. In our protocol, commitments need to be signed by the issuer to form a valid token. Moreover, users need to prove that they know a valid signature without revealing this signature. Hence, we will make use of an EUF-CMA secure signature scheme S for messages over G_2 which is compatible with our proof system. To instantiate this building block, we make use of the structure-preserving signature scheme of Abe et al. [1]. We denote the key generation algorithm, the signing algorithm, and the verification algorithm by S.Gen, S.Sgn, and S.Vfy, respectively.

Encryption. The hidden user ID token will simply be an encryption of a user's public key under a public key contained in the CRS. For this purpose, an IND-CPA secure encrption scheme E for messages in G_1 which is compatible with our proof system suffices. This building block can be intantiated with the ElGamal encryption scheme [16]. We denote the corresponding algorithms by E.Gen, E.Enc, and E.Dec.

5.2 Protocol Description

Figures 8 to 11 summarize the scheme. In the following, we will elaborate on the details of the different protocols and algorithms. Note that an informal but intuitive system description was already given in Section 1.1. We omit repeating the high-level ideas here due to the page limit.

System and user setup. The setup and key generation algorithms are given in Figure 8. The global CRS CRS generated by Setup consists of a CRS $\mathsf{CRS}_{\mathsf{com}}$ for the commitment scheme, a shared CRS $\mathsf{CRS}_{\mathsf{pok}}$ for the three proof systems, as well as a public encryption key $\mathsf{pk}_{\mathcal{T}}$ to generate hidden user id tokens hid. The corresponding trapdoor td contains the extraction trapdoor $\mathsf{td}_{\mathsf{epok}}$ for the proof systems as well as the secret key $\mathsf{sk}_{\mathcal{T}}$ to decrypt hidden id tokens. Thus, ExtractUID can be defined by $\mathsf{ExtractUID}((\mathsf{td}_{\mathsf{epok}}, \mathsf{sk}_{\mathcal{T}}), \mathsf{hid}) := \mathsf{E.Dec}(\mathsf{sk}_{\mathcal{T}}, \mathsf{hid})$.[9] The key pair of the issuer is essentially a signature key pair $(\mathsf{pk}_{\mathsf{sig}}, \mathsf{sk}_{\mathsf{sig}})$. Remember, that the global CRS is included in the public key for convenience, i. e. $(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}) := ((\mathsf{CRS}, \mathsf{pk}_{\mathsf{sig}}), \mathsf{sk}_{\mathsf{sig}})$. The key pair of a user consists of $\mathsf{sk}_{\mathcal{U}} = y$ and $\mathsf{pk}_{\mathcal{U}} = g_1^y$, where $\mathsf{pk}_{\mathcal{U}}$ is used as the user identity and $\mathsf{sk}_{\mathcal{U}}$ to prove this identity in the scope of Issue as well as to prove guilt in VerifyGuilt.

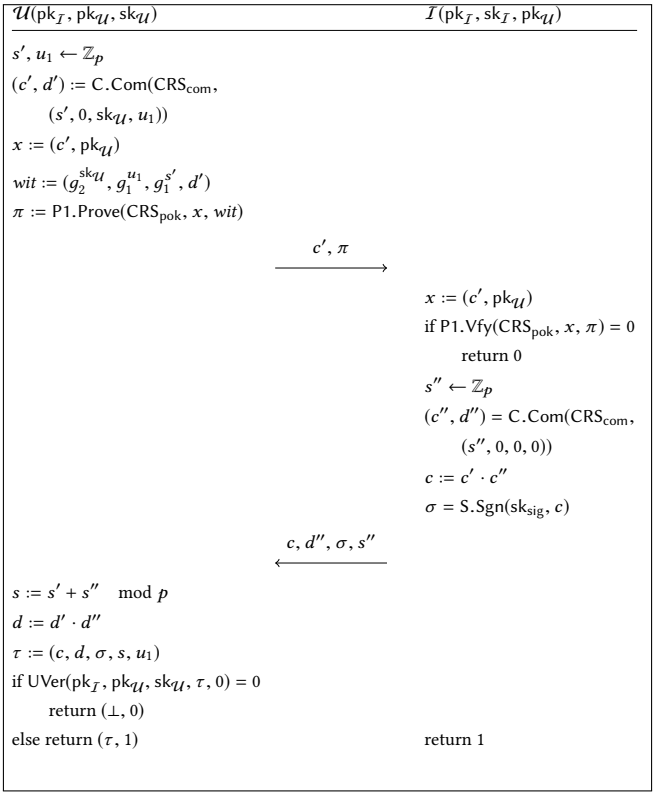

Figure 9: Issue protocol

Issuing tokens. The issue protocol is shown in Fig. 9. Essentially, a valid token consists of a commitment $c \in G_2$ on the token version number $s \in \mathbb{Z}_p$, the balance $w \in \mathbb{Z}_p$, the user secret key $\mathsf{sk}_{\mathcal{U}} \in \mathbb{Z}_p$, and the double-spending tag randomness $u_1 \in \mathbb{Z}_p$, as well as a signature on c under $\mathsf{sk}_{\mathcal{I}}$. The token version number needs to be chosen jointly by $\mathcal{U}$ (choosing an additive share s') and $\mathcal{I}$ (choosing an additive share s'') to ensure unlinkability on the one hand and enable double-spending detection on the other hand. The randomness u_1 is chosen by $\mathcal{U}$ and will be used in the scope of Accum and Vfy to compute double-spending tags. The fact that it is hidden from $\mathcal{I}$ ensures that the double-spending tag will look random if the token is used once. The fact that it is bound to the token ensures that double-spending will reveal the user identity. To generate such a token, $\mathcal{U}$ commits to s', $w = 0$, $\mathsf{sk}_{\mathcal{U}}$, and u_1. It then computes a proof showing that the corresponding commitment c'

[9] Note that, actually, the CRS for the proof systems does not need to be generated by the extraction setup algorithm as $\mathsf{td}_{\mathsf{epok}}$ is not needed by ExtractUID. We decided to do this nevertheless to simplify security proofs: it avoids to first switch to extraction mode in every security proof.

$\underline{\mathsf{UVer}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, \tau, w)}$
parse $(c, d, \sigma, s, u_1) := \tau$
if $\mathsf{pk}_{\mathcal{U}} = g_1^{\mathsf{sk}_{\mathcal{U}}} \wedge \mathsf{C.Open}(\mathsf{CRS}, (g_1^s, g_1^w, \mathsf{pk}_{\mathcal{U}}, g_1^{u_1}), c, d) = 1 \wedge$
$\mathsf{S.Vfy}(\mathsf{pk}_{\mathsf{sig}}, \sigma, c) = 1$
return 1
else return 0

$\underline{\mathsf{IdentDS}(\mathsf{pk}_{\mathcal{I}}, (s_1, z_1), (s_2, z_2))}$
parse $(t, u_2) := z_1, (t', u_2') := z_2$
if $s_1 \neq s_2 \vee u_2 = u_2'$ return $\perp$
else $\mathsf{sk}_{\mathcal{U}} := (t - t') \cdot (u_2 - u_2')^{-1} \bmod p$, $\mathsf{pk}_{\mathcal{U}} := g_1^{\mathsf{sk}_{\mathcal{U}}}$
return $(\mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}})$

$\underline{\mathsf{VerifyGuilt}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \Pi)}$
if $g_1^{\Pi} = \mathsf{pk}_{\mathcal{U}}$ return 1
else return 0

Figure 10: User verification of tokens and double-spending algorithms

has been formed correctly by the owner of $\mathsf{pk}_{\mathcal{U}}$. More precisely, P1 is used to compute a proof π for a statement x from the language $L_{1,\mathsf{pk}_{\mathcal{I}}}$ defined by

$$L_{1,\mathsf{pk}_{\mathcal{I}}} := \left\{ (c', \mathsf{pk}_{\mathcal{U}}) \;\middle|\; \begin{array}{l} \exists\, \mathrm{SKU} \in G_2;\ S', U_1, D' \in G_1 : \\ \mathsf{C.Open}(\mathsf{CRS}_{\mathsf{com}}, (S', 1, \mathsf{pk}_{\mathcal{U}}, U_1), c', D') = 1 \\ e(\mathsf{pk}_{\mathcal{U}}, g_2) = e(g_1, \mathrm{SKU}) \end{array} \right\} \quad (7)$$

Note that the language depends on public parameters like gp, $\mathsf{CRS}_{\mathsf{com}}$, $\mathsf{CRS}_{\mathsf{pok}}$, $\mathsf{pk}_{\mathsf{sig}}$ which are all subsumed in $\mathsf{pk}_{\mathcal{I}}$ and remain fixed after the system has been setup. The first time the language is defined we denote this dependence by a suffix, but later omit this and refer to the language as $L_1 = L_{1,\mathsf{pk}_{\mathcal{I}}}$. Note that the second equation in Eq. (7) actually proves the knowledge of $g_2^{\mathsf{sk}_{\mathcal{U}}}$ (rather than $\mathsf{sk}_{\mathcal{U}}$ itself).[10] However, computing $g_2^{\mathsf{sk}_{\mathcal{U}}}$ without knowing $\mathsf{sk}_{\mathcal{U}}$ (only given $\mathsf{pk}_{\mathcal{U}}$) is assumed to be a hard problem (Co-CDH). Receiving c' and a valid proof π, $\mathcal{I}$ adds a random s'' to s' contained in c' by using the homomorphic property of C. This results in the commitment c which can be opened using the opening value $d = d' \cdot d''$. The commitment c is then signed by $\mathcal{I}$ resulting in signature σ. Please note the asymmetry in the handling of the commitment and its opening value: The user sends its half of the commitment c' but not the opening, while the issuer send the whole commitment c together with his half of the opening d''. The token and s'' are sent over to $\mathcal{U}$ who verifies the correctness by applying UVer. The UVer algorithm is shown in Fig. 10. It verifies that c opens correctly, σ is valid, and $\mathsf{sk}_{\mathcal{U}}$ is the secret key belonging to $\mathsf{pk}_{\mathcal{U}}$.

Collecting points. The Accum protocol is depicted in Figure 11. Given his current token $\tau = (c, d, \sigma, s, u_1)$, a $\mathcal{U}$ receives a random challenge u_2 from $\mathcal{AC}$ to prepare the data component of the double-spending tag $t = \mathsf{sk}_{\mathcal{U}} u_2 + u_1$ for this token. Moreover, $\mathcal{U}$ prepares the generation of a fresh token just like in the Issue protocol. To this

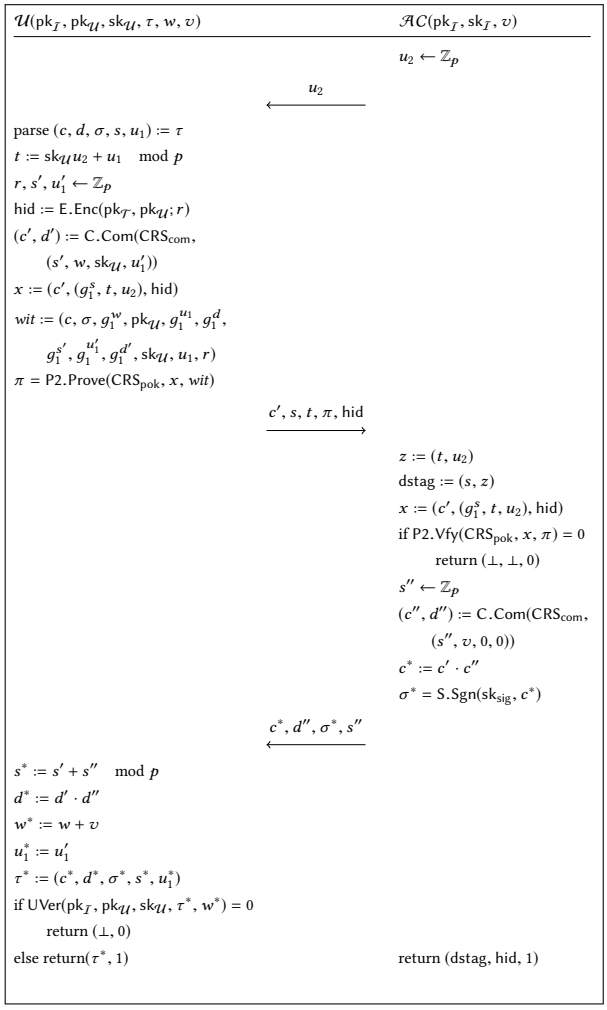

Figure 11: Accumulation protocol

end, it computes a commitment c' containing the same balance w and user secret key $\mathsf{sk}_{\mathcal{U}}$ as c but a fresh share s' for a new token version number and fresh randomness for generating double-spending tags. (Note that in order to ensure unlinkability, one cannot simply send over τ to $\mathcal{AC}$ to accumulate points.) Moreover, hid is generated as a fresh encryption of $\mathsf{pk}_{\mathcal{U}}$. Recall that while hid does not fulfill an obvious function, it is needed for our security definitions. Finally, the user proves that everything has been computed as claimed, i. e., c is a signed commitment; c' is just a "new version" of this commitment containing the same balance and user secret key $\mathsf{sk}_{\mathcal{U}}$; t is part of the double-spending tag "containing" $\mathsf{sk}_{\mathcal{U}}$ and involving the user randomness u_1 fixed in c and the accumulators challenge u_2; and hid contains $\mathsf{pk}_{\mathcal{U}}$ belonging to $\mathsf{sk}_{\mathcal{U}}$. More precisely, P2 is used to compute a proof π for a statement x from the

[10] Note that proving a statement $\exists \mathsf{sk}_{\mathcal{U}} \in \mathbb{Z}_p :\ \mathsf{pk}_{\mathcal{U}} = g_1^{\mathsf{sk}_{\mathcal{U}}}$ instead would not help as we can only extract $g_1^{\mathsf{sk}_{\mathcal{U}}}$ from the proof.

language L_{2,pk_I} defined by

$$\left\{ (c', (S, t, u_2), \mathsf{hid}) \;\middle|\; \begin{array}{l} \exists\, c, \sigma \in G_2; \\ W, \mathsf{pk}_{\mathcal{U}}, U_1, D, S', U_1', D' \in G_1; \\ \mathsf{sk}_{\mathcal{U}}, u_1, r \in \mathbb{Z}_p : \\ \mathsf{E.Enc}(\mathsf{pk}_{\mathcal{T}}, \mathsf{pk}_{\mathcal{U}}; r) = \mathsf{hid} \\ \mathsf{C.Open}(\mathsf{CRS}_{\mathsf{com}}, (S, W, \mathsf{pk}_{\mathcal{U}}, U_1), c, D) = 1 \\ \mathsf{C.Open}(\mathsf{CRS}_{\mathsf{com}}, (S', W, \mathsf{pk}_{\mathcal{U}}, U_1'), c', D') = 1 \\ \mathsf{S.Vfy}(\mathsf{pk}_{\mathsf{sig}}, \sigma, c) = 1 \\ \mathsf{pk}_{\mathcal{U}} = g_1^{\mathsf{sk}_{\mathcal{U}}}, U_1 = g_1^{u_1}, t = \mathsf{sk}_{\mathcal{U}} u_2 + u_1 \end{array} \right\} \quad (8)$$

Upon receiving c', s, z, π, and hid, $\mathcal{AC}$ checks the proof π and updates c' by adding s'' to s' and v to w resulting in commitment c^*. The commitment is signed and c^* together with the accumulator's half of the opening value d'' and the corresponding signature σ^* are sent to the user who verifies them.

Claiming a balance and redeeming points. The Vfy protocol works the same way as the Accum protocol except that the balance w is not treated as a secret anymore. That means the balance (or more precisely $W := g_1^w$) is not a witness but part of the statement x in the language L_{3,pk_I} defined by

$$\left\{ (c', (S, t, u_2), \mathsf{hid}, W) \;\middle|\; \begin{array}{l} \exists\, c, \sigma \in G_2; \\ \mathsf{pk}_{\mathcal{U}}, U_1, D, S', U_1', D' \in G_1; \\ \mathsf{sk}_{\mathcal{U}}, u_1, r \in \mathbb{Z}_p : \\ \mathsf{E.Enc}(\mathsf{pk}_{\mathcal{T}}, \mathsf{pk}_{\mathcal{U}}; r) = \mathsf{hid} \\ \mathsf{C.Open}(\mathsf{CRS}_{\mathsf{com}}, (S, W, \mathsf{pk}_{\mathcal{U}}, U_1), c, D) = 1 \\ \mathsf{C.Open}(\mathsf{CRS}_{\mathsf{com}}, (S', W, \mathsf{pk}_{\mathcal{U}}, U_1'), c', D') = 1 \\ \mathsf{S.Vfy}(\mathsf{pk}_{\mathsf{sig}}, \sigma, c) = 1 \\ \mathsf{pk}_{\mathcal{U}} = g_1^{\mathsf{sk}_{\mathcal{U}}}, U_1 = g_1^{u_1}, t = \mathsf{sk}_{\mathcal{U}} u_2 + u_1 \end{array} \right\} \quad (9)$$

for which $\mathcal{U}$ generates a proof using P3.

Double-spending detection. The IdentDS and VerifyGuilt algorithms are given in Figure 10. To see why IdentDS is working, observe the following: First, to use a specific token τ in a Accum or Vfy protocol run, a (potentially malicious) user is forced to reveal the fixed token version number s. This is because, s is bound to τ by being contained in the signed commitment c and the proof π ensures that the former is indeed the case even if c and σ are not revealed explicitly. Second, c and π also enforce that the fixed user secret key $\mathsf{sk}_{\mathcal{U}}$ and the fixed double-spending tag randomness u_1 as well as a freshly chosen challenge value u_2 are always used in a Accum or Vfy protocol run involving τ to form $z = \mathsf{sk}_{\mathcal{U}} u_2 + u_1$. Hence, double-spending a token will reveal the same token version number $s_1 = s_2$ and involve different challenges $u_2 \neq u_2'$ with overwhelming probability. In this case, we can easily extract $\mathsf{sk}_{\mathcal{U}}$ given $z_1 = \mathsf{sk}_{\mathcal{U}} u_2 + u_1$ and $z_2 = \mathsf{sk}_{\mathcal{U}} u_2' + u_1$. The proof of guilt Π is simply set to be $\mathsf{sk}_{\mathcal{U}}$, which is assumed to be hard to compute given $\mathsf{pk}_{\mathcal{U}}$ only.[11]

Theorem 5.1 (System Security). *Assume the SXDH assumption holds w.r.t.* SetupGrp. *Let us assume that* BBAP *and* E *are correct,* P1, P2, P3 *are perfectly* $F_{\mathsf{gp}}^{(1)}$*-,* $F_{\mathsf{gp}}^{(2)}$*-, and* $F_{\mathsf{gp}}^{(3)}$*-extractable, respectively,* C *is additively homomorphic and* F_{gp}'*-binding, and* S *is EUF-CMA secure. Then* BBAP *is trapdoor-linkable, owner-binding w.r.t.* Issue, *owner-binding w.r.t.* Accum/Vfy, *balance-binding, and ensures double-spending detection.*

Theorem 5.2 (User Security and Privacy). *Assume the SXDH assumption holds w.r.t.* SetupGrp. *Let* E *be an IND-CPA-secure asymmetric encryption scheme,* C *be an additively homomorphic, perfectly hiding and equivocal commitment scheme, and* P1, P2, P3 *be composable zero-knowledge. Then* BBAP *is privacy-preserving and ensures false-accusation protection.*

The theorems follow from our proofs in Appendices C and D. See Appendix B for formal definitions of our building blocks and their properties. Also, see Appendix B for concrete instantiations of E, S, C, and the NIZKs.

6 PERFORMANCE EVALUATION

We evaluate the performance of our BBA+ instantiation by implementing it for a target platform suitable for mobile applications. To this end, network payload and execution time is measured. The results, in terms of execution time, are limited to the user device. We expect the hardware of the issuer, accumulator, or verifier to be much more powerful and therefore to be capable of executing the algorithms in insignificant time. Not included in our estimations are data-transmission times, since they depend on external factors not influenced by BBA+.

As smartphones have become ubiquitous in the developed and developing world, we choose them as our target platform. An additional benefit of this platform is that a developer using our scheme does not have to distribute new hardware and users are already familiar with the functionality of their device.

We evaluate our implementation on a OnePlus 3 smartphone. It features a Snapdragon 820 Quad-Core processor (2 × 2.15 GHz & 2 × 1.6 GHz), 6 GB RAM and runs Android OS v7.1.1 (Nougat). The implementation is done in C++11 using the RELIC toolkit v.0.4.1, an open source crypto-library under the LGPL license [5].

6.1 Bilinear Groups

The digital signature scheme, the commitment scheme and the non-interactive zero-knowledge proof system all build on pairing-friendly elliptic curves. We configured RELIC with curves of 254-bit order, the minimal supported size for pairing-friendly curves that exceed 80 bit security. With this parameter choice the toolkit configures itself to use the Barreto-Naehrig curves Fp254BNb and Fp254n2BNb presented by Aranha et al. for improved efficiency [6, 25].

We select the optimal Ate pairing as RELIC's pairing function since current speed records are achieved using this function [28].

To further optimize the performance of BBA+ one might use a custom implementation of elliptic curves and a compatible bilinear map, which is optimized for this purpose. We emphasize however, that RELIC itself already delivers very promising execution times.

6.2 Prover-Chosen CRS

The prover-chosen CRS as explained in Section X enables some kind of time-memory tradeoff. A fresh prover-chosen CRS has to be generated on each run of a protocol to stay unlinkable. To this end, the transmitted data has to be extended by the prover-chosen CRS,

[11] Of course, we need to show that honest protocol runs will not reveal significant information about $\mathsf{sk}_{\mathcal{U}}$.

Table 1: User execution times for our instantiation

Algorithm	Execution Time [ms]	Data Sent [Bytes]	Data Received [Bytes]
GenSymKey	2.51	98	0
Issue	114.11	672	320
Acc	315.40	4576	320
Vrfy	303.53	4512	320

as well as a NIZK proving that it was correctly generated using the original CRS. This results in 4 additional MSE proofs. However, once the prover-chosen CRS is established, all subsequent Groth-Sahai operations are roughly reduced to 60% of their computational costs.

We implemented the prover-chosen CRS method only for the Accum and Vfy protocols. The Issue protocol would lose performance, as there are too few equations benefiting from the prover-chosen CRS and therefore the setup costs are outweighing the speedup.

6.3 Implementation Results

Table 1 shows the average execution times for the respective protocols on the user device and the amount of data that has to be transmitted from the device to the issuer, accumulator, or verifier and vice versa. To obtain a compact data stream for network transfer while maintaining generality, we serialized each element as a length byte, followed by its internal serialization.

As we implemented our BBA+ instantiation extended by message encryption (c.f. Appendix F.1), Table 1 includes an algorithm GenSymKey, which chooses a random key for the symmetric encryption of all protocol messages with AES-CBC and HMAC-SHA256. This key is encrypted using the TwinDH-based KEM by Cash et al. [13], using the same primitives for its underlying symmetric encryption. As a reference value for an acceptable execution time, we consider one second to be a good upper bound. All sub-protocols of BBA+, Issue, Accum and Vfy, execute in under 350 ms on the user side. If we use for example NFC with its maximum speed of 424 kbit/s, it would take less than 100ms to transmit all data of any of the protocols from/to the communication terminal. This leaves more than 550 ms to transmit data packets over the network to the issuer, accumulator, or verifier, have him compute his part of the protocol and respond to the user. Assuming that the issuer runs a powerful backend, it should not be challenging to execute an entire protocol run in less than a second.

We can therefore conclude that our BBA+ system can be efficiently instantiated and executed.

6.4 Further Optimization

The RELIC toolkit is a multi-purpose library which is not mainly optimized for pairing-based elliptic curve cryptography. By creating a dedicated library focused on a highly optimized implementation of pairing-friendly elliptic curves, the execution time can be reduced. In addition, curves of order 160-bit could be implemented. These would still result in 80 bit of security but would require less computational time. RELIC does not provide such a configuration in the employed version. Regarding the Groth Sahai proof system, there has been some work to optimize efficiency too. Herold et al. proposed a transformation that speeds up verification by a factor of two [21] and Blazy et al. applied batch verification techniques [9]. If these optimizations were integrated into BBA+ as well, the computational complexity of the issuer, accumulator, and verifier could be notably reduced. Also, the user could save computations when generating GS proofs by carefully applying the prover chosen CRS technique from [17].

REFERENCES

[1] Masayuki Abe, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. 2011. Optimal Structure-Preserving Signatures in Asymmetric Bilinear Groups. In *Advances in Cryptology – CRYPTO 2011 (Lecture Notes in Computer Science)*, Phillip Rogaway (Ed.), Vol. 6841. Springer, Heidelberg, Germany, Santa Barbara, CA, USA, 649–666.

[2] Masayuki Abe, Markulf Kohlweiss, Miyako Ohkubo, and Mehdi Tibouchi. 2015. Fully Structure-Preserving Signatures and Shrinking Commitments. In *Advances in Cryptology – EUROCRYPT 2015, Part II (Lecture Notes in Computer Science)*, Elisabeth Oswald and Marc Fischlin (Eds.), Vol. 9057. Springer, Heidelberg, Germany, Sofia, Bulgaria, 35–65.

[3] William Aiello, Yuval Ishai, and Omer Reingold. 2001. Priced Oblivious Transfer: How to Sell Digital Goods. In *Advances in Cryptology – EUROCRYPT 2001 (Lecture Notes in Computer Science)*, Birgit Pfitzmann (Ed.), Vol. 2045. Springer, Heidelberg, Germany, Innsbruck, Austria, 119–135.

[4] Aimia Coalition Loyalty UK Ltd. 2016. The Nectar loyalty program. Online Resource. (2016). https://www.nectar.com/.

[5] D. F. Aranha and C. P. L. Gouvêa. 2016. RELIC is an Efficient Library for Cryptography. Online Resource. (2016). https://github.com/relic-toolkit/relic.

[6] Paulo S. L. M. Barreto and Michael Naehrig. 2006. Pairing-Friendly Elliptic Curves of Prime Order. In *SAC 2005: 12th Annual International Workshop on Selected Areas in Cryptography (Lecture Notes in Computer Science)*, Bart Preneel and Stafford Tavares (Eds.), Vol. 3897. Springer, Heidelberg, Germany, Kingston, Ontario, Canada, 319–331.

[7] Mira Belenkiy, Melissa Chase, Markulf Kohlweiss, and Anna Lysyanskaya. 2008. P-signatures and Noninteractive Anonymous Credentials. In *TCC 2008: 5th Theory of Cryptography Conference (Lecture Notes in Computer Science)*, Ran Canetti (Ed.), Vol. 4948. Springer, Heidelberg, Germany, San Francisco, CA, USA, 356–374.

[8] Mihir Bellare. 2015. New Proofs for NMAC and HMAC: Security Without Collision-Resistance. *Journal of Cryptology* 28, 4 (2015), 844–878.

[9] Olivier Blazy, Georg Fuchsbauer, Malika Izabachène, Amandine Jambert, Hervé Sibert, and Damien Vergnaud. 2010. Batch Groth-Sahai. In *ACNS 10: 8th International Conference on Applied Cryptography and Network Security (Lecture Notes in Computer Science)*, Jianying Zhou and Moti Yung (Eds.), Vol. 6123. Springer, Heidelberg, Germany, Beijing, China, 218–235.

[10] Dan Boneh and Xavier Boyen. 2004. Short Signatures Without Random Oracles. In *Advances in Cryptology – EUROCRYPT 2004 (Lecture Notes in Computer Science)*, Christian Cachin and Jan Camenisch (Eds.), Vol. 3027. Springer, Heidelberg, Germany, Interlaken, Switzerland, 56–73.

[11] Jan Camenisch, Rafik Chaabouni, and abhi shelat. 2008. Efficient Protocols for Set Membership and Range Proofs. In *Advances in Cryptology – ASIACRYPT 2008 (Lecture Notes in Computer Science)*, Josef Pieprzyk (Ed.), Vol. 5350. Springer, Heidelberg, Germany, Melbourne, Australia, 234–252.

[12] Jan Camenisch, Maria Dubovitskaya, and Gregory Neven. 2010. Unlinkable Priced Oblivious Transfer with Rechargeable Wallets. In *FC 2010: 14th International Conference on Financial Cryptography and Data Security (Lecture Notes in Computer Science)*, Radu Sion (Ed.), Vol. 6052. Springer, Heidelberg, Germany, Tenerife, Canary Islands, Spain, 66–81.

[13] David Cash, Eike Kiltz, and Victor Shoup. 2008. The Twin Diffie-Hellman Problem and Applications. In *Advances in Cryptology – EUROCRYPT 2008 (Lecture Notes in Computer Science)*, Nigel P. Smart (Ed.), Vol. 4965. Springer, Heidelberg, Germany, Istanbul, Turkey, 127–145.

[14] Rafik Chaabouni, Helger Lipmaa, and Bingsheng Zhang. 2012. A Non-interactive Range Proof with Constant Communication. In *FC 2012: 16th International Conference on Financial Cryptography and Data Security (Lecture Notes in Computer Science)*, Angelos D. Keromytis (Ed.), Vol. 7397. Springer, Heidelberg, Germany, Kralendijk, Bonaire, 179–199.

[15] Gerhard de Koning Gans, Jaap-Henk Hoepman, and Flavio D. Garcia. 2008. A Practical Attack on the MIFARE Classic. In *Smart Card Research and Advanced Applications: 8th IFIP WG 8.8/11.2 International Conference, Proceedings*, Gilles Grimaud and François-Xavier Standaert (Eds.). Springer, Heidelberg, Germany, London, UK, 267–282.

[16] Taher ElGamal. 1984. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. In *Advances in Cryptology – CRYPTO'84 (Lecture Notes in Computer Science)*, G. R. Blakley and David Chaum (Eds.), Vol. 196. Springer, Heidelberg, Germany, Santa Barbara, CA, USA, 10–18.

[17] Alex Escala and Jens Groth. 2014. Fine-Tuning Groth-Sahai Proofs. In *PKC 2014: 17th International Conference on Theory and Practice of Public Key Cryptography (Lecture Notes in Computer Science)*, Hugo Krawczyk (Ed.), Vol. 8383. Springer, Heidelberg, Germany, Buenos Aires, Argentina, 630–649.

[18] Flavio D. Garcia, Gerhard de Koning Gans, Ruben Muijrers, Peter van Rossum, Roel Verdult, Ronny Wichers Schreur, and Bart Jacobs. 2008. Dismantling MIFARE Classic. In *ESORICS 2008: 13th European Symposium on Research in Computer Security (Lecture Notes in Computer Science)*, Sushil Jajodia and Javier López (Eds.), Vol. 5283. Springer, Heidelberg, Germany, Málaga, Spain, 97–114.

[19] Flavio D. Garcia, Peter van Rossum, Roel Verdult, and Ronny Wichers Schreur. 2009. Wirelessly Pickpocketing a Mifare Classic Card. In *2009 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, Oakland, CA, USA, 3–15.

[20] Jens Groth and Amit Sahai. 2008. Efficient Non-interactive Proof Systems for Bilinear Groups. In *Advances in Cryptology – EUROCRYPT 2008 (Lecture Notes in Computer Science)*, Nigel P. Smart (Ed.), Vol. 4965. Springer, Heidelberg, Germany, Istanbul, Turkey, 415–432.

[21] Gottfried Herold, Julia Hesse, Dennis Hofheinz, Carla Ràfols, and Andy Rupp. 2014. Polynomial Spaces: A New Framework for Composite-to-Prime-Order Transformations. In *Advances in Cryptology – CRYPTO 2014, Part I (Lecture Notes in Computer Science)*, Juan A. Garay and Rosario Gennaro (Eds.), Vol. 8616. Springer, Heidelberg, Germany, Santa Barbara, CA, USA, 261–279.

[22] Malika Izabachène, Benoît Libert, and Damien Vergnaud. 2011. Block-Wise P-Signatures and Non-interactive Anonymous Credentials with Efficient Attributes. In *13th IMA International Conference on Cryptography and Coding (Lecture Notes in Computer Science)*, Liqun Chen (Ed.), Vol. 7089. Springer, Heidelberg, Germany, Oxford, UK, 431–450.

[23] Tibor Jager and Andy Rupp. 2016. Black-Box Accumulation: Collecting Incentives in a Privacy-Preserving Way. *Proceedings on Privacy Enhancing Technologies (PoPETs)* 2016, 3 (2016), 62–82.

[24] Jonathan Katz and Yehuda Lindell. 2007. *Introduction to Modern Cryptography*. Chapman and Hall/CRC Press, London, UK.

[25] Yuto Kawahara, Tetsutaro Kobayashi, Michael Scott, and Akihiro Kato. 2016. *Barreto-Naehrig Curves.* Internet Draft. Internet Engineering Task Force. Work in Progress.

[26] Eike Kiltz, Jiaxin Pan, and Hoeteck Wee. 2015. Structure-Preserving Signatures from Standard Assumptions, Revisited. In *Advances in Cryptology – CRYPTO 2015, Part II (Lecture Notes in Computer Science)*, Rosario Gennaro and Matthew J. B. Robshaw (Eds.), Vol. 9216. Springer, Heidelberg, Germany, Santa Barbara, CA, USA, 275–295.

[27] Milica Milutinovic, Italo Dacosta, Andreas Put, and Bart De Decker. 2015. uCentive: An efficient, anonymous and unlinkable incentives scheme. In *2015 IEEE Trustcom/BigDataSE/ISPA*, Vol. 1. IEEE Computer Society Press, Helsinki, Finland, 588–595.

[28] Dustin Moody, Rene C. Peralta, Ray A. Perlner, Andrew R. Regenscheid, Allen L. Roginsky, and Lidong Chen. 2015. Report on Pairing-based Cryptography. In *Journal of Research of the National Institute of Standards and Technology*, Vol. 120. National Insititute of Standards and Technology, Gaithersburg, MD, USA, 11–27.

[29] NXP Semiconductors Netherlands B.V. 2014. *MIFARE Classic EV1 4K Product Data Sheet Revision 3.1.* NXP Semiconductors Netherlands B.V.

[30] NXP Semiconductors Netherlands B.V. 2016. *MIFARE DESFire EV2 contactless multi-application IC Data Sheet Rev. 2.0.* NXP Semiconductors Netherlands B.V.

[31] David Oswald and Christof Paar. 2011. Breaking Mifare DESFire MF3ICD40: Power Analysis and Templates in the Real World. In *Cryptographic Hardware and Embedded Systems – CHES 2011 (Lecture Notes in Computer Science)*, Bart Preneel and Tsuyoshi Takagi (Eds.), Vol. 6917. Springer, Heidelberg, Germany, Nara, Japan, 207–222.

[32] PAYBACK GmbH. 2016. The Payback loyalty program. Online Resource. (2016). https://www.payback.net/.

APPENDIX

A CORRECTNESS FOR BBA+ SCHEMES

In this section, we formally define *correctness* of BBA+ schemes.

Definition A.1 (BBA+ Correctness). A BBA+ scheme BBAP is called *correct* if all the following properties hold for all $n \in \mathbb{N}$, $(\mathsf{CRS}, \mathsf{td}) \leftarrow \mathsf{Setup}(1^n)$, issuer key-pairs $(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}) \leftarrow \mathsf{IGen}(\mathsf{CRS})$, user key-pairs $(\mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}) \leftarrow \mathsf{UGen}(\mathsf{CRS})$, and parties $\mathcal{U}$, $\mathcal{I}$, $\mathcal{AC}$, and $\mathcal{V}$ honestly following the protocols.

Correctness of issuing. For all outputs of the issue protocol $((\tau, b_{\mathcal{U}}), (b_{\mathcal{I}})) \leftarrow \langle \mathcal{U}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}), \mathcal{I}(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}) \rangle$, it holds that $b_{\mathcal{U}} = b_{\mathcal{I}} = 1$ and $\mathsf{UVer}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, \tau, 0) = 1$.[12]

Correctness of accumulation. For all tokens τ, balance values $w \in \mathbb{Z}_p$ with $\mathsf{UVer}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, \tau, w) = 1$ and all accumulation values $v \in \mathbb{Z}_p$,[13] we have that $((\tau^*, 1), (s, z, \mathsf{hid}, 1)) \leftarrow \mathsf{Accum}\langle \mathcal{U}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, \tau, w, v), \mathcal{AC}(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}, v) \rangle$ as well as $\mathsf{UVer}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, \tau^*, w + v) = 1$.

Correctness of token verification For all tokens τ, balance values $w \in \mathbb{Z}_p$ with $\mathsf{UVer}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, \tau, w) = 1$ and all accumulation values $v \in \mathbb{Z}_p$ with $w + v \in \mathbb{Z}_p$ we have that $((\tau^*, 1), (s, z, \mathsf{hid}, 1)) \leftarrow \mathsf{Vfy}\langle \mathcal{U}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, \tau, w, v), \mathcal{I}(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}, w, v) \rangle$ as well as $\mathsf{UVer}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, \tau^*, w + v) = 1$.

B FORMAL DEFINITIONS AND INSTANTIATIONS OF BUILDING BLOCKS

In the following, we give formal definitions for the building blocks we use in our construction, i.e., encryption, signatures, commitments, and non-interactive zero-knowledge proofs, as well as present possible instantiations of these components.

B.1 Symmetric and Asymmetric Encryption

We define symmetric and asymmetric encryption schemes. The definitions are based on [24], except that Theorem B.1 has been augmented by a SetupGrp algorithm.

Asymmetric Encryption (PKE).

Definition B.1 (Asymmetric Encryption). An *asymmetric encryption scheme* is a tuple of PPT algorithms (SetupGrp, Gen, Enc, Dec) such that:

- SetupGrp takes as input a security parameter 1^n and outputs public parameters gp. We assume that gp is given as implicit input to all algorithms.
- Gen(gp) outputs a pair (pk, sk) of keys, where pk is the (public) encryption key and sk is the (secret) decryption key,
- Enc(pk, m) takes a key pk and a plaintext message $m \in \{0,1\}^*$ and outputs a ciphertext c, and
- Dec(sk, c) takes a key sk and a ciphertext c and outputs a plaintext message m or $\perp$. We assume that Dec is deterministic.

Correctness is required in the usual sense. Security is defined via experiments, given in Figs. 12 and 13. In Fig. 13 $\mathsf{Dec}(\mathsf{sk}, \cdot)$ is an oracle that gets a message m from the adversary and returns $\mathsf{Dec}(\mathsf{sk}, m)$, and $\mathsf{Dec}'(\mathsf{sk}, \cdot)$ is the same, except that it returns $\perp$ on input c^*.

Definition B.2 (CPA Security for Asymmetric Encryption). An asymmetric encryption scheme E is *IND-CPA*-secure if for all PPT adversaries $\mathcal{A}$ in the experiment $\mathsf{Exp}^{\text{ind-cpa-asym}}_{\mathsf{E},\mathcal{A}}(n)$ given in Fig. 12

[12] Note that verifying hat $\mathsf{pk}_{\mathcal{U}}$ is fresh and bound to a valid physical ID is not part of Issue but needs to be done externally.

[13] In practice, one would like to avoid wrap-arounds. However for sufficiently large value space $\mathbb{Z}_p$ (e. g. 64 Bit) and reasonable values v this concern is neglible.

Experiment $\mathsf{Exp}^{\text{ind-cpa-asym}}_{\mathsf{E},\mathcal{A}}(n)$
$\mathsf{gp} \leftarrow \mathsf{SetupGrp}(1^n)$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(\mathsf{gp})$
$(state, m_0, m_1) \leftarrow \mathcal{A}(1^n, \mathsf{pk})$, where $|m_0| = |m_1|$
$b \leftarrow \{0,1\}$, $c^* \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b)$
$b' \leftarrow \mathcal{A}(state, c^*)$
The experiment returns 1 iff $b = b'$.

Figure 12: CPA security experiment for asymmetric encryption.

Experiment $\mathsf{Exp}^{\text{ind-cca-asym}}_{\mathsf{E},\mathcal{A}}(n)$
$\mathsf{gp} \leftarrow \mathsf{SetupGrp}(1^n)$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(\mathsf{gp})$
$(state, m_0, m_1) \leftarrow \mathcal{A}^{\mathsf{Dec}(\mathsf{sk},\cdot)}(1^n, \mathsf{pk})$, where $|m_0| = |m_1|$
$b \leftarrow \{0,1\}$, $c^* \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b)$
$b' \leftarrow \mathcal{A}^{\mathsf{Dec}'(\mathsf{sk},\cdot)}(state, c^*)$
The experiment returns 1 iff $b = b'$.

Figure 13: CCA2 security experiment for asymmetric encryption.

the advantage of $\mathcal{A}$ defined by

$$\mathsf{Adv}^{\text{ind-cpa}}_{\mathsf{E},\mathcal{A}} = \left|\Pr\left[\mathsf{Exp}^{\text{ind-cpa-asym}}_{\mathsf{E},\mathcal{A}}(n) = 1\right] - \frac{1}{2}\right|$$

is negligible in n.

For instantiating the construction of our base protocol, we use the well known ElGamal encryption scheme [16] which is IND-CPA-secure under DDH assumption (implied by the SXDH assumption).

Definition B.3 (CCA2 Security for Asymmetric Encryption). An asymmetric encryption scheme E is *IND-CCA2*-secure if for all PPT adversaries $\mathcal{A}$ in the experiment $\mathsf{Exp}^{\text{ind-cca-asym}}_{\mathsf{E},\mathcal{A}}(n)$ the advantage of $\mathcal{A}$ defined by

$$\mathsf{Adv}^{\text{ind-cca}}_{\mathsf{E},\mathcal{A}} = \left|\Pr\left[\mathsf{Exp}^{\text{ind-cca-asym}}_{\mathsf{E},\mathcal{A}}(n) = 1\right] - \frac{1}{2}\right|$$

is negligible in n.

We use the Twin-DH-based encryption scheme of Cash et al. [13].

Symmetric Encryption (SKE).

Definition B.4 (Symmetric Encryption). A *symmetric encryption scheme* is a tuple of PPT algorithms (Gen, Enc, Dec) such that:

- $\mathsf{Gen}(1^n)$ outputs a (random) key k,
- $\mathsf{Enc}(k, m)$ takes a key k and a plaintext message $m \in \{0,1\}^*$ and outputs a ciphertext c, and
- $\mathsf{Dec}(k, c)$ takes a key k and a ciphertext c and outputs a plaintext message m or $\perp$. We assume that Dec is deterministic.

As for asymmetric encryption, we require correctness in the usual sense. We define a multi-message version of the IND-CCA2 security experiment in Fig. 14. It is a well-known fact that IND-CCA2 security in the multi-message setting is equivalent to standard IND-CCA2 security. (This can be shown via a standard hybrid argument.) In this experiment $\mathsf{Enc}(k,\cdot)$ and $\mathsf{Dec}(k,\cdot)$ denote oracles

Experiment $\mathsf{Exp}^{\text{ind-cca-sym}}_{\mathsf{E},\mathcal{A}}(n)$
$k \leftarrow \mathsf{Gen}(1^n)$
$(state, j, \mathbf{m_0}, \mathbf{m_1}) \leftarrow \mathcal{A}^{\mathsf{Enc}(k,\cdot),\mathsf{Dec}(k,\cdot)}(1^n)$, where $\mathbf{m_0}, \mathbf{m_1}$ are two vectors of $j \in \mathbb{N}$ bitstrings each such that for all $1 \le i \le j$: $|m_{0,i}| = |m_{1,i}|$
$b \leftarrow \{0,1\}$, $\mathbf{c}^* \leftarrow (\mathsf{Enc}(k, m_{b,1}), \ldots, \mathsf{Enc}(k, m_{b,j}))$
$b' \leftarrow \mathcal{A}^{\mathsf{Enc}(k,\cdot),\mathsf{Dec}'(k,\cdot)}(state, \mathbf{c}^*)$
The experiment returns 1 iff $b = b'$.

Figure 14: Multi-message CCA2 security experiment for symmetric encryption.

that return $\mathsf{Enc}(k, m)$ and $\mathsf{Dec}(k, m)$ for an m chosen by the adversary, and $\mathsf{Dec}'(k,\cdot)$ is the same as $\mathsf{Dec}(k,\cdot)$, except that it returns $\perp$ on input of any c_i^* that is contained in $\mathbf{c}^*$.

Definition B.5 (CCA2 Security for Symmetric Encryption). A symmetric encryption scheme E is *IND-CCA2*-secure if for all PPT adversaries $\mathcal{A}$ in the experiment $\mathsf{Exp}^{\text{ind-cca-sym}}_{\mathsf{E},\mathcal{A}}(n)$ the advantage of $\mathcal{A}$ defined by

$$\mathsf{Adv}^{\text{ind-cca}}_{\mathsf{E},\mathcal{A}} = \left|\Pr\left[\mathsf{Exp}^{\text{ind-cca-sym}}_{\mathsf{E},\mathcal{A}}(n) = 1\right] - \frac{1}{2}\right|$$

is negligible.

For example, symmetric encryption can be instantiated with AES in CBC mode together with HMAC based on the SHA-256 hash function. The result will be IND-CCA2-secure if AES is a pseudo-random permutation and the SHA-256 compression function is a PRF when the data input is seen as the key [8].

B.2 Digital Signatures

Definition B.6. A *digital signature scheme* $\mathsf{SIG} := (\mathsf{SetupGrp}, \mathsf{Gen}, \mathsf{Sgn}, \mathsf{Vfy})$ consists of four PPT algorithms.

- SetupGrp takes as input a security parameter 1^n and outputs public parameters gp. We assume that gp is given as implicit input to all algorithms.
- Gen takes gp as input and outputs a key pair (pk, sk). The public key and gp define a message space $\mathcal{M}$.
- Sgn takes as input the secret key sk and a message $m \in \mathcal{M}$, and outputs a signature σ.
- Vfy takes as input the public key pk, a message $m \in \mathcal{M}$, and a purported signature σ, and outputs a bit.

We call SIG correct if for all $n \in \mathbb{N}$, $\mathsf{gp} \leftarrow \mathsf{SetupGrp}(1^n)$, $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(\mathsf{gp})$, $m \in \mathcal{M}$, $\sigma \leftarrow \mathsf{Sgn}(\mathsf{sk}, m)$ we have $1 \leftarrow \mathsf{Vfy}(pk, \sigma, m)$.

We say that SIG is EUF-CMA secure if for all PPT adversaries $\mathcal{A}$ holds that the advantage $\mathsf{Adv}^{\text{euf-cma}}_{\mathcal{A}}(1^n)$ defined by

$$\Pr\left[\mathsf{Vfy}(\mathsf{pk}, \sigma^*, m^*) = 1 \;\middle|\; \begin{array}{c} \mathsf{gp} \leftarrow \mathsf{SetupGrp}(1^n) \\ (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(\mathsf{gp}) \\ (m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sgn}(\mathsf{sk},\cdot)}(1^n, \mathsf{pk}) \\ m^* \notin \{m_1, \ldots, m_q\} \end{array}\right] \tag{10}$$

is negligible in n, where $\mathsf{Sgn}(\mathsf{sk},\cdot)$ is an oracle that, on input m, returns $\mathsf{Sgn}(\mathsf{sk}, m)$, and $\{m_1, \ldots, m_q\}$ denotes the set of messages queried by $\mathcal{A}$ to its oracle.

For our construction, we will use the structure-preserving signature scheme of Abe et at. [1], which is currently the most efficient structure-preserving signature scheme. Its EUF-CMA security proof

is in the generic group model, a restriction we consider reasonable with respect to our goal of constructing a highly efficient BBA+ scheme. An alternative secure in the plain model would be [26].

For the construction in [1], one needs to fix two additional parameters $\mu, \nu \in \mathbb{N}_0$ defining the actual message space. The scheme can be described as follows.

- SetupGrp is a bilinear group generator (cf. Definition 2.1). Let $\mathsf{gp} := (G_1, G_2, G_T, e, p, g_1, g_2, g_T) \leftarrow \mathsf{SetupGrp}(1^n)$ denote the output of $\mathsf{SetupGrp}(1^n)$. Then the message space equals $\mathcal{M} := G_1^\nu \times G_2^\mu$.
- Gen samples vectors $\mathbf{x} \leftarrow \mathbb{Z}_p^\mu$ and $\mathbf{y} \leftarrow \mathbb{Z}_p^\nu$ and integers $\lambda, \lambda' \leftarrow \mathbb{Z}_p$. The public verification key pk is defined as
$$\mathsf{pk} := (g_1^{\mathbf{x}}, g_2^{\mathbf{y}}, g_1^{\lambda}, g_1^{\lambda'}) \in G_1^{\mu+2} \times G_2^{\nu}$$
and the corresponding secret key sk as
$$\mathsf{sk} := (\mathbf{x}, \mathbf{y}, \lambda, \lambda') \in \mathbb{Z}_p^{\mu+\nu+2}$$
- $\mathsf{Sgn}(\mathsf{sk}, (\mathbf{m_1}, \mathbf{m_2}))$ chooses $r \leftarrow \mathbb{Z}_p$, sets $\sigma_1 := g_2^r$, $\sigma_2 := g_2^{\lambda' - r\lambda}(\prod_{i=1}^{\mu} m_{2,i}^{x_i})^{-1}$, $\sigma_3 := g_1^{r^{-1}}(\prod_{i=1}^{\nu} m_{1,i}^{y_i})^{-r^{-1}}$, and returns $\sigma := (\sigma_1, \sigma_2, \sigma_3)$.
- $\mathsf{Vfy}(\mathsf{pk}, (\mathbf{m_1}, \mathbf{m_2}), \sigma)$ returns 1 iff
$$e(g_1^{\lambda}, \sigma_1) e(g_1, \sigma_2) \prod_{i=1}^{\mu} e(g_1^{x_i}, m_{2,i}) = e(g_1^{\lambda'}, g_2)$$
and
$$e(\sigma_3, \sigma_1) \prod_{i=1}^{\nu} e(m_{1,i}, g_2^{y_i}) = e(g_1, g_2)$$

Remark 3. *We instantiate this scheme with $\nu = 0$ and $\mu = 1$, because we only sign a single element (a commitment) from G_2. In this case, we have $\sigma_3 = g_1^{r^{-1}}$.*

B.3 F_{gp}-binding Commitments

Definition B.7. A *commitment scheme* $\mathsf{COM} := (\mathsf{SetupGrp}, \mathsf{Gen}, \mathsf{Com}, \mathsf{Open})$ consists of four algorithms.

- SetupGrp takes as input a security parameter 1^n and outputs public parameters gp. These parameters also define a message space $\mathcal{M}$, an implicit message space $\mathcal{M}'$ and a function $F_{\mathsf{gp}} : \mathcal{M} \to \mathcal{M}'$ mapping a message to its implicit representation. We assume that gp is given as implicit input to all algorithms.
- Gen is a PPT algorithm, which takes gp as input and outputs public parameters $\mathsf{CRS}_{\mathsf{com}}$.
- Com is a PPT algorithm, which takes as input parameters CRS and a message $m \in \mathcal{M}$ and outputs a commitment c to m and some decommitment value d.
- Open is a deterministic polynomial-time algorithm, which takes as input parameters CRS, commitment c, an implicit message $M \in \mathcal{M}'$, and opening d. It returns either 0 or 1.

COM is *correct* if for all $\mathsf{gp} \leftarrow \mathsf{SetupGrp}(1^n)$, $\mathsf{CRS} \leftarrow \mathsf{Gen}(\mathsf{gp})$, $m \in \mathcal{M}$, and $(c, d) \leftarrow \mathsf{Com}(\mathsf{CRS}, m)$ it holds that $1 \leftarrow \mathsf{Open}(\mathsf{CRS}, F_{\mathsf{gp}}(m), c, d)$.

We say that COM is a (computationally) *hiding*, F_{gp}*-binding*, *equivocal* commitment scheme, if it has the following properties:

(1) **Hiding:** For all PPT adversaries $\mathcal{A}$ holds that the advantage $\mathsf{Adv}^{\mathsf{hide}}_{\mathcal{A}}(1^n)$ defined by
$$\left| \Pr\left[b = b' \;\middle|\; \begin{array}{c} \mathsf{gp} \leftarrow \mathsf{SetupGrp}(1^n) \\ \mathsf{CRS} \leftarrow \mathsf{Gen}(\mathsf{gp}) \\ (m_0, m_1, \mathit{state}) \leftarrow \mathcal{A}(1^n, \mathsf{CRS}) \\ b \leftarrow \{0,1\} \\ (c, d) \leftarrow \mathsf{Com}(\mathsf{CRS}, m_b) \\ b' \leftarrow \mathcal{A}(c, \mathit{state}) \end{array} \right] - \frac{1}{2} \right| \tag{11}$$
is negligible in n. The scheme is called statistically hiding if $\mathsf{Adv}^{\mathsf{hide}}_{\mathcal{A}}(1^n)$ is negligible even for an unbounded adversary $\mathcal{A}$.

(2) F_{gp}**-Binding:** For all PPT adversaries $\mathcal{A}$ it holds that the advantage $\mathsf{Adv}^{F_{\mathsf{gp}}\text{-}\mathsf{bind}}_{\mathcal{A}}(1^n)$ defined by
$$\Pr\left[\begin{array}{c} \mathsf{Open}(\mathsf{CRS}, M, c, d) = 1 \\ \wedge \\ \mathsf{Open}(\mathsf{CRS}, M', c, d') = 1 \end{array} \;\middle|\; \begin{array}{c} \mathsf{gp} \leftarrow \mathsf{SetupGrp}(1^n) \\ \mathsf{CRS} \leftarrow \mathsf{Gen}(\mathsf{gp}) \\ (c, M, d, M', d') \\ \leftarrow \mathcal{A}(1^n, \mathsf{CRS}) \\ M \neq M' \end{array} \right] \tag{12}$$
is negligible in n.

(3) **Equivocal:** There exists polynomial-time algorithms SimGen, SimCom and Equiv such that for all PPT adversaries $\mathcal{A}$

(a) we have that the advantage $\mathsf{Adv}^{\mathsf{com\text{-}gen}}_{\mathsf{COM},\mathcal{A}}(n)$ defined by
$$\left| \begin{array}{l} \Pr\left[1 \leftarrow \mathcal{A}(\mathsf{CRS}) \;\middle|\; \begin{array}{c} \mathsf{gp} \leftarrow \mathsf{SetupGrp}(1^n), \\ \mathsf{CRS} \leftarrow \mathsf{Gen}(\mathsf{gp}) \end{array} \right] \\ -\Pr\left[1 \leftarrow \mathcal{A}(\mathsf{CRS}') \;\middle|\; \begin{array}{c} \mathsf{gp} \leftarrow \mathsf{SetupGrp}(1^n), \\ (\mathsf{CRS}', \mathsf{td}_{\mathsf{ecom}}) \leftarrow \mathsf{SimGen}(\mathsf{gp}) \end{array} \right] \end{array} \right| \tag{13}$$
is zero.

(b) we have that the advantage $\mathsf{Adv}^{\equiv}_{\mathsf{COM},\mathcal{A}}(n)$ defined by
$$\left| \begin{array}{l} \Pr\left[1 \leftarrow \mathcal{A}(\mathsf{CRS}', \mathsf{td}_{\mathsf{ecom}}, m, c, d) \;\middle|\; \begin{array}{c} \mathsf{gp} \leftarrow \mathsf{SetupGrp}(1^n), \\ (\mathsf{CRS}', \mathsf{td}_{\mathsf{ecom}}) \leftarrow \mathsf{SimGen}(\mathsf{gp}), \\ m \leftarrow \mathcal{M}, \\ (c, d) \leftarrow \mathsf{Com}(\mathsf{CRS}', m) \end{array} \right] \\ -\Pr\left[1 \leftarrow \mathcal{A}(\mathsf{CRS}', \mathsf{td}_{\mathsf{ecom}}, m, c', d') \;\middle|\; \begin{array}{c} \mathsf{gp} \leftarrow \mathsf{SetupGrp}(1^n), \\ (\mathsf{CRS}', \mathsf{td}_{\mathsf{ecom}}) \leftarrow \mathsf{SimGen}(\mathsf{gp}), \\ (c', r) \leftarrow \mathsf{SimCom}(\mathsf{gp}), \\ m \leftarrow \mathcal{M}, \\ d' \leftarrow \mathsf{Equiv}(\mathsf{CRS}', \mathsf{td}_{\mathsf{ecom}}, m, r) \end{array} \right] \end{array} \right| \tag{14}$$
is neglible in n

Furthermore, assume that the message space of COM is an additive group. Then COM is called *additively homomorphic*, if there exist additional PPT algorithms $c \leftarrow \mathsf{CAdd}(\mathsf{CRS}, c_1, c_2)$ and $d \leftarrow \mathsf{DAdd}(\mathsf{CRS}, d_1, d_2)$ which on input of two commitments and corresponding decommitment values $(c_1, d_1) \leftarrow \mathsf{Com}(\mathsf{CRS}, m_1)$ and $(c_2, d_2) \leftarrow \mathsf{Com}(\mathsf{CRS}, m_2)$, output a commitment c and decommitment d, respectively, such that $\mathsf{Open}(\mathsf{CRS}, c, F_{\mathsf{gp}}(m_1 + m_2), d) = 1$.

We make use of the following shrinking ℓ-message-commitment scheme from [2]:

- SetupGrp is a bilinear group generator (cf. Theorem 2.1). Let $\mathsf{gp} := (G_1, G_2, G_T, e, p, g_1, g_2, g_T) \leftarrow \mathsf{SetupGrp}(1^n)$ denote the output of $\mathsf{SetupGrp}(1^n)$. Then the message space equals $\mathcal{M} := \mathbb{Z}_p^\ell$ and the implicit message space equals

$\mathcal{M}' := G_1^\ell$ with $F'_{\mathsf{gp}} : \mathcal{M} \to \mathcal{M}'$ defined by $F'_{\mathsf{gp}}(m_1, \ldots, m_\ell) := (g_1^{m_1}, \ldots, g_1^{m_\ell})$.

- Gen(gp) samples $x_1, \ldots x_\ell \leftarrow \mathbb{Z}_p^*$ and returns CRS := $(h_1, \ldots, h_\ell) := (g_2^{x_1}, \ldots, g_2^{x_\ell})$.
- SimGen(gp) samples $x_1, \ldots x_\ell \leftarrow \mathbb{Z}_p^*$ and returns CRS := $(h_1, \ldots, h_\ell) := (g_2^{x_1}, \ldots, g_2^{x_\ell})$ together with $\mathsf{td}_{\mathsf{ecom}} := (x_1, \ldots, x_\ell)$.
- Com(CRS, $\mathbf{m}$), where $\mathbf{m} = (m_1, \ldots, m_\ell)$, chooses $r \leftarrow \mathbb{Z}_p$ and returns $(c, d) := (g_2^r \prod_{i=1}^\ell h_i^{m_i}, g_1^r)$.
- SimCom(gp) chooses $r \leftarrow \mathbb{Z}_p$ and returns $(c, r) := (g_2^r, r)$.
- Open(CRS, $\mathbf{M}, c, d$), where $\mathbf{M} = (M_1, \ldots, M_\ell) \in G_1^\ell$ returns 1 if

$$e(g_1, c) = e(d, g_2) \prod_{i=1}^{\ell} e(M_i, h_i)$$

and 0 otherwise.

- Equiv(CRS, $\mathsf{td}_{\mathsf{ecom}}, \mathbf{m}, r$), where $\mathbf{m} = (m_1, \ldots, m_\ell)$ and $\mathsf{td}_{\mathsf{ecom}} = (x_1, \ldots, x_\ell)$, returns $d := g_1^r \prod_{i=1}^\ell g_1^{-x_i m_i}$.

The commitment scheme described above is correct, statistically hiding, equivocal, and F'_{gp}-Binding, for $F'_{\mathsf{gp}}(m_1, \ldots, m_\ell) := (g_1^{m_1}, \ldots, g_1^{m_\ell})$ under the SXDH assumption [2].

B.4 F_{gp}-extractable NIZKs

Definition B.8. Let R be an efficiently verifiable relation containing triples (gp, x, w). We call gp the group setup, x the statement, and w the witness. Given some gp, let L_{gp} be the language containing all statements x such that $(\mathsf{gp}, x, w) \in R$. Let POK := (SetupGrp, SetupPoK, Prove, Vfy) be a tuple of PPT algorithms such that

- SetupGrp takes as input a security parameter 1^n and outputs public parameters gp. We assume that gp is given as implicit input to all algorithms.
- SetupPoK takes as input gp and outputs a (public) common reference string CRS.
- Prove takes as input the common reference string CRS, a statement x, and a witness w with $(\mathsf{gp}, x, w) \in R$ and outputs a proof π.
- Vfy takes as input the common reference string CRS, a statement x, and a proof π and outputs 1 or 0.

POK is called a non-interactive zero-knowledge proof system for R with F_{gp}-extractability, if the following properties are satisfied.

(1) **Perfect completeness:** For all $\mathsf{gp} \leftarrow \mathsf{SetupGrp}(1^n)$, $\mathsf{CRS} \leftarrow \mathsf{SetupPoK}(\mathsf{gp})$, and $(\mathsf{gp}, x, w) \in R$, $\mathsf{Vfy}(\mathsf{CRS}, x, \pi) = 1$ for all proofs $\pi \leftarrow \mathsf{Prove}(\mathsf{CRS}, x, w)$.

(2) **Perfect soundness:** For all (possibly unbounded) adversaries $\mathcal{A}$ we have that

$$\Pr\left[\mathsf{Vfy}(\mathsf{CRS}, x, \pi) = 0 \;\middle|\; \begin{array}{c} \mathsf{gp} \leftarrow \mathsf{SetupGrp}(1^n) \\ \mathsf{CRS} \leftarrow \mathsf{SetupPoK}(\mathsf{gp}) \\ (x, \pi) \leftarrow \mathcal{A}(\mathsf{CRS}) \\ x \notin L_{\mathsf{gp}} \end{array}\right] = 1. \quad (15)$$

(3) **Perfect F_{gp}-extractability:** There exists a polynomial-time extractor (SetupEPoK, ExtractW) such that for all (possibly unbounded) adversaries $\mathcal{A}$

(a) we have that the advantage $\mathsf{Adv}^{\text{pok-ext-setup}}_{\mathsf{POK},\mathcal{A}}(n)$ defined by

$$\left| \begin{array}{l} \Pr\left[1 \leftarrow \mathcal{A}(\mathsf{CRS}) \;\middle|\; \begin{array}{c} \mathsf{gp} \leftarrow \mathsf{SetupGrp}(1^n), \\ \mathsf{CRS} \leftarrow \mathsf{SetupPoK}(\mathsf{gp}) \end{array}\right] \\ -\Pr\left[1 \leftarrow \mathcal{A}(\mathsf{CRS}') \;\middle|\; \begin{array}{c} \mathsf{gp} \leftarrow \mathsf{SetupGrp}(1^n), \\ (\mathsf{CRS}', \mathsf{td}_{\mathsf{epok}}) \leftarrow \mathsf{SetupEPoK}(\mathsf{gp}) \end{array}\right] \end{array} \right| \quad (16)$$

is zero.

(b) we have that the advantage $\mathsf{Adv}^{\text{pok-ext}}_{\mathsf{POK},\mathcal{A}}(n)$ defined by

$$\Pr\left[\begin{array}{c} \exists\, w :\ F_{\mathsf{gp}}(w) = W \\ \wedge\ (\mathsf{gp}, x, w) \in R \end{array} \;\middle|\; \begin{array}{c} \mathsf{gp} \leftarrow \mathsf{SetupGrp}(1^n) \\ (\mathsf{CRS}', \mathsf{td}_{\mathsf{epok}}) \leftarrow \mathsf{SetupEPoK}(\mathsf{gp}) \\ (x, \pi) \leftarrow \mathcal{A}(\mathsf{CRS}') \\ 1 \leftarrow \mathsf{Vfy}(\mathsf{CRS}', x, \pi) \\ W \leftarrow \mathsf{ExtractW}(\mathsf{CRS}', \mathsf{td}_{\mathsf{epok}}, x, \pi) \end{array}\right] \quad (17)$$

is 1.

(4) **Composable Zero-knowledge:** There exists a polynomial-time simulator (SetupSPoK, SimProof) and hint generator GenHint such that for all PPT adversaries $\mathcal{A}$

(a) we have that the advantage $\mathsf{Adv}^{\text{zk-setup}}_{\mathsf{POK},\mathcal{A}}(n)$ defined by

$$\left| \begin{array}{l} \Pr\left[1 \leftarrow \mathcal{A}(\mathsf{CRS}) \;\middle|\; \begin{array}{c} \mathsf{gp} \leftarrow \mathsf{SetupGrp}(1^n), \\ \mathsf{CRS} \leftarrow \mathsf{SetupPoK}(\mathsf{gp}) \end{array}\right] \\ -\Pr\left[1 \leftarrow \mathcal{A}(\mathsf{CRS}') \;\middle|\; \begin{array}{c} \mathsf{gp} \leftarrow \mathsf{SetupGrp}(1^n), \\ \mathit{hint} \leftarrow \mathsf{GenHint}(\mathsf{gp}), \\ (\mathsf{CRS}', \mathsf{td}_{\mathsf{spok}}) \leftarrow \mathsf{SetupSPoK}(\mathsf{gp}, \mathit{hint}), \end{array}\right] \end{array} \right| \quad (18)$$

is negligible in n.

(b) we have that the advantage $\mathsf{Adv}^{\mathsf{zk}}_{\mathsf{POK},\mathcal{A}}(n)$ defined by

$$\left| \begin{array}{l} \Pr[1 \leftarrow \mathcal{A}^{\mathsf{SimProof}'(\mathsf{CRS}', \mathsf{td}_{\mathsf{spok}}, \cdot, \cdot)}(1^n, \mathsf{CRS}', \mathsf{td}_{\mathsf{spok}})] \\ -\Pr[1 \leftarrow \mathcal{A}^{\mathsf{Prove}(\mathsf{CRS}', \cdot, \cdot)}(1^n, \mathsf{CRS}', \mathsf{td}_{\mathsf{spok}})] \end{array} \right| \quad (19)$$

is negligible, where $\mathsf{gp} \leftarrow \mathsf{SetupGrp}(1^n)$, $(\mathsf{CRS}', \mathsf{td}_{\mathsf{spok}}) \leftarrow \mathsf{SetupSPoK}(\mathsf{gp})$, $\mathsf{SimProof}'(\mathsf{CRS}', \mathsf{td}_{\mathsf{spok}}, \cdot, \cdot)$ is an oracle which on input $(x, z) \in R$, returns $\mathsf{SimProof}(\mathsf{CRS}', \mathsf{td}_{\mathsf{spok}}, x)$. Both $\mathsf{SimProof}'$ and Prove return $\perp$ on input $(x, z) \notin R$.

Remark 4. *Note that F_{gp}-extractability actually implies soundness independent of F_{gp}: If there would be a false statement x which verifies, violating soundness, then obviously, there is no witness w for x which violates extractability.*

Let SetupGrp be a bilinear group generator (cf. Theorem 2.1) for which the SXDH assumption (cf. Theorem 2.2) holds and $\mathsf{gp} := (G_1, G_2, G_T, e, p, g_1, g_2, g_T) \leftarrow \mathsf{SetupGrp}(1^n)$ denote the output of SetupGrp. Furthermore, let $X_1, \ldots, X_{m_1} \in G_1$, $x_1, \ldots, x_{m_2} \in \mathbb{Z}_p$, $Y_1, \ldots, Y_{m_3} \in G_2$, and $y_1, \ldots, y_{m_4} \in \mathbb{Z}_p$ denote variables in the following types of equations:

- *Pairing-Product Equation (PPE)*:

$$\prod_{i=1}^{m_3} e(A_i, Y_i) \prod_{i=1}^{m_1} e(X_i, B_i) \prod_{i=1}^{m_1} \prod_{j=1}^{m_3} e(X_i, Y_i)^{\gamma_{i,j}} = t_T$$

for constants $A_i \in G_1$, $B_i \in G_2$, $t_T \in G_T$, $\gamma_{i,j} \in \mathbb{Z}_p$.

- *Multi-Scalar Equation (MSE) over G_1*:

$$\prod_{i=1}^{m_4} A_i^{y_i} \prod_{i=1}^{m_1} X_i^{b_i} \prod_{i=1}^{m_1} \prod_{j=1}^{m_4} X_i^{\gamma_{i,j} y_j} = t_1$$

for constants $A_i, t_1 \in G_1$, $b_i, \gamma_{i,j} \in \mathbb{Z}_p$.

- *Multi-Scalar Equation (MSE) over G_2*:

$$\prod_{i=1}^{m_2} B_i^{x_i} \prod_{i=1}^{m_3} Y_i^{a_i} \prod_{i=1}^{m_2}\prod_{j=1}^{m_3} Y_j^{\gamma_{i,j}x_i} = t_2$$

for constants $B_i, t_2 \in G_2$, $a_i, \gamma_{i,j} \in \mathbb{Z}_p$.

- *Quadratic Equation (QE) over Z_p*:

$$\sum_{i=1}^{m_4} a_i y_i + \sum_{i=1}^{m_2} x_i b_i + \sum_{i=1}^{m_2}\sum_{j=1}^{m_4} \gamma_{i,j} x_i y_j = t$$

for constants $a_i, b_i, \gamma_{i,j}, t \in \mathbb{Z}_p$.

Let L_{gp} be a language containing statements described by the conjunction of n_1 pairing-product equations over gp, n_2 multi-scalar equations over G_1, n_3 multi-scalar equations over G_2, and n_4 quadratic equations over $\mathbb{Z}_p$, where $n_i \in \mathbb{N}_0$ are constants, as well as by witnesses

$$w = (X_1, \ldots, X_{m_1}, x_1, \ldots, x_{m_2}, Y_1, \ldots, Y_{m_3}, y_1, \ldots, y_{m_4}),$$

where $m_i \in \mathbb{N}_0$. Then the Groth-Sahai proof system for L_{gp}, as introduced by [20], is perfectly correct, perfectly sound, and satisfies F_{gp}-extractability [17, 20] for

$$F_{\mathrm{gp}} : G_1^{m_1} \times \mathbb{Z}_p^{m_2} \times G_2^{m_3} \times Z_p^{m_4} \to G_1^{m_1} \times G_1^{m_2} \times G_2^{m_3} \times G_2^{m_4}$$

with

$$F_{\mathrm{gp}}(w) \;:=\; ((X_i)_{i\in[m_1]}, (g_1^{x_i})_{i\in[m_2]}, (Y_i)_{i\in[m_3]}, (g_2^{y_i})_{i\in[m_4]}).$$

It is also known to be composable zero-knowledge [17, 20] as long as for all PPEs in L_{gp} holds that either

- $t_T = 1$ or
- the right-hand side of the PPE can be written as $\prod_{i=1}^{k} e(A_i, B_i)$ for constants $A_i \in G_1$, $B_i \in G_2$, such that for each i either $\mathsf{Dlog}(A_i)$ or $\mathsf{Dlog}(B_i)$ is known prior to the setup of the CRS.

In the latter case, *hint* from Definition B.8 would contain these discrete logarithms which would simply be put (as additional elements) into the Groth-Sahai simulation trapdoor $\mathsf{td}_{\mathsf{spok}}$.

C SYSTEM SECURITY PROOFS

In this section, we state and prove the security properties of our instantiation. We give a full reduction proof for the balance-binding property (Theorem C.5) as this is by far the most involved property. For trapdoor-linkability (Theorem C.1), owner-binding with respect to the issue (Theorem C.2), owner-binding with respect to accumulation and verification (Theorem C.3), and double-spending detection (Theorem C.4), we restrict ourselves to extended proof sketches which, however, can be easily turned into full proofs.

Theorem C.1 (Trapdoor-Linkability). *If* BBAP *and* E *are correct, and* P2 *and* P3 *are perfectly sound then* BBAP *is trapdoor-linkable.*

Proof.

Completeness. We will restrict to show this property for the accumulation protocol as the proof for the verification protocol is very similar. Consider the hidden UID token hid which is part of the output of $\mathcal{AC}$. As π verifies and P2 is sound, we know that hid can be generated using E.Enc with appropriate message and randomness: there exists $m \in G_1$ and $r \in \mathbb{Z}_p$ such that $\mathsf{E.Enc}(\mathsf{pk}_{\mathcal{T}}, m; r) = \mathsf{hid}$. As the message space coincides with the user public key space, m is some valid public key $\mathsf{pk}_{\mathcal{U}}$ for which, by definition of UGen, there is some corresponding secret key $\mathsf{sk}_{\mathcal{U}}$. As we assume that BBAP is correct, an honest user with key pair $(\mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}})$ could have received a valid token τ by executing the Issue protocol. Then, using $\mathsf{pk}_{\mathcal{U}}$, $\mathsf{sk}_{\mathcal{U}}$, τ, $w = 0$, and arbitrary v, this honest user could have executed Accum successfully, which, using randomness r for encrypting $\mathsf{pk}_{\mathcal{U}}$, would have led to hid. Hence, any hidden user hid appearing in a valid accumulation view, can be generated by an honest user.

Extractability. Note that the trapdoor td consist of $\mathsf{td}_{\mathsf{epok}}$ and $\mathsf{sk}_{\mathcal{T}}$. We set $\mathsf{ExtractUID}(\mathsf{td}_{\mathsf{epok}}, \mathsf{hid}) := \mathsf{E.Dec}(\mathsf{sk}_{\mathcal{T}}, \mathsf{hid})$. For an honest user, hid will be the encryption of the user's $\mathsf{pk}_{\mathcal{U}}$ under $\mathsf{pk}_{\mathcal{T}}$. Hence, the output of extraction will always be $\mathsf{pk}_{\mathcal{U}}$. □

Theorem C.2 (Owner-Binding wrt. Issue). *If the CO-CDH assumption holds and* P1 *is perfectly $F_{\mathrm{gp}}^{(1)}$-extractable, then* BBAP *is owner-binding with respect to* Issue.

Proof. As P1 is perfectly $F_{\mathrm{gp}}^{(1)}$-extractable, we can extract $\mathsf{SKU} \in G_2$ from the verifying proof received by Issue. As SKU satisfies the equation

$$e(\mathsf{pk}_{\mathcal{U}}, g_2) = e(g_1, \mathrm{SKU})$$

it needs to be equal to $g_2^{\mathsf{sk}_{\mathcal{U}}}$. Hence, this value is a solution to an instance of the CO-CDH problem. It is straightforward to build a CO-CDH adversary $\mathcal{B}$ from an adversary $\mathcal{A}$ against the owner-binding property wrt. Issue. □

Theorem C.3 (Owner-Binding wrt. Accum and Vfy). *If* P1, P2, P3 *are perfectly $F_{\mathrm{gp}}^{(1)}$-, $F_{\mathrm{gp}}^{(2)}$-, and $F_{\mathrm{gp}}^{(3)}$-extractable, respectively,* C *is F'_{gp}-binding,* S *is EUF-CMA secure, and* E *is correct, then* BBAP *is owner-binding with respect to* Accum *and* Vfy.

Proof. Let us consider the first completed and successful call of $\mathcal{A}$ to either MalAcc or MalVer, where some $\mathsf{pk}_{\mathcal{U}}$ has been extracted, such that, prior to that call, no successful call to MalIssue for $\mathsf{pk}_{\mathcal{U}}$ took place. As P2 and P3 are perfectly $F_{\mathrm{gp}}^{(2)}$-extractable and $F_{\mathrm{gp}}^{(3)}$-extractable, respectively, we can extract the commitment c as well as the a signature σ on c from the verifying proof π that $\mathcal{A}$ sent. Furthermore, we can extract an implicit message $M \in G_1^4$ consisting of S, W, $\widehat{\mathsf{pk}_{\mathcal{U}}}$, and U_1, as well as an opening value D with which c can be opened correctly. Note that due to the perfect binding property of any correct encryption scheme, and the soundness of P2/P3, $\mathsf{pk}_{\mathcal{U}}$ extracted by ExtractUID indeed coincides with $\widehat{\mathsf{pk}_{\mathcal{U}}}$, i.e., $\mathsf{pk}_{\mathcal{U}} = \widehat{\mathsf{pk}_{\mathcal{U}}}$.

Let us first consider the case that c did not occur in a prior, successful call, i.e., has not been sent by $\mathcal{I}$, $\mathcal{AC}$, or $\mathcal{V}$. Then the extracted signature σ, is a signature on a new message, namely c, which has not been signed under $\mathsf{sk}_{\mathcal{I}}$ before. Hence, this violates the EUF-CMA security of S.

Now, let us consider the case that c occured before in a successful call. First, let us assume that c has been sent by the issuer in a successful interaction, where $\mathcal{I}$ received $\widehat{\mathsf{pk}}_{\mathcal{U}}$ as input. Let $\widehat{M}$ and $\widehat{D}$ be the implicit message and opening for c we can extract from the corresponding proof. Due to the soundness of P1, $\widehat{\mathsf{pk}}_{\mathcal{U}}$ must be part of $\widehat{M}$. As we assume that there is no successfull call to MalIssue for $\mathsf{pk}_{\mathcal{U}}$, it holds that $\widehat{\mathsf{pk}}_{\mathcal{U}} \neq \mathsf{pk}_{\mathcal{U}}$. Thus, it follows that $\widehat{M} \neq M$ which violates the F'_{gp}-binding property of C. Second, we can consider the case that c has been sent by an accumulator or verifier (as commitment c^*) before. Using essentially the same argument (and the assumption that this cannot be the first successful call for $\mathsf{pk}_{\mathcal{U}}$, so $\widehat{\mathsf{pk}}_{\mathcal{U}} \neq \mathsf{pk}_{\mathcal{U}}$), we can see that this would also violate the F'_{gp}-binding property. □

Theorem C.4 (Double-Spending Detection). *If* P1, P2, P3 *are perfectly* $F^{(1)}_{\mathsf{gp}}$-, $F^{(2)}_{\mathsf{gp}}$-, *and* $F^{(3)}_{\mathsf{gp}}$*-extractable, respectively,* C *is additively homomorphic and* F'_{gp}*-binding,* S *is EUF-CMA secure, and* E *is correct, then* BBAP *ensures double-spending detection.*

Proof. Let c_1, π_1 and c_2, π_2 denote the commitment and proof, and $s_1 = s_2$ the token version number contained in view_1 and view_2, respectively, that has been sent by the adversary. Furthermore, let $z_1 = (t, u_2)$ and $z_2 = (t', u'_2)$.

Case 1: $\mathsf{pk}^{(1)}_{\mathcal{U}} \neq \mathsf{pk}^{(2)}_{\mathcal{U}}$.
Here, we can distinguish two subcases: either $c_1 = c_2$ or $c_1 \neq c_2$.

(1) $c_1 = c_2$. If $c_1 = c_2$, we know, by using the perfect binding property of E and the soundness of P2 and P3, that this commitment can be opened using $\mathsf{pk}^{(1)}_{\mathcal{U}}$ and $\mathsf{pk}^{(2)}_{\mathcal{U}}$. Hence, by the extractability property of P2 and P3, we can extract two valid, implicit messages $M^{(1)} \neq M^{(2)}$ and two openings $D^{(1)}, D^{(2)}$ for the commitment $c_1 = c_2$. This violates the F'_{gp}-binding property of C.
(2) $c_1 \neq c_2$. Here we distinguish between the case that both commitments occured in a previous transaction and the case that at least one of the commitments is new.
   (a) *c_1 or c_2 is new.* If c_1 or c_2 did not occur in any previous accumulation or verification transaction as message sent from $\mathcal{AC}$ or $\mathcal{V}$, respectively, or any issue transaction, then the EUF-CMA security of S is violated: we can extract the corresponding signature from the corresponding proof and this signature is on a commitment which has never been signed before.
   (b) *c_1 and c_2 occured before.* It remains to consider the case that $c_1 \neq c_2$ did both occur in previous transactions, denoted by T_1 and T_2. However, in this case we have $s_1 = s_2$ only with negligible probability as they have been uniformly chosen from $\mathbb{Z}_p$: Let c'_1, π'_1 denote the commitment and proof *sent* by the adversary in the previous transaction T_1. From π'_1 we can extract an implicit message $M'_1 \in G_1^4$ as well as an opening D'_1 which can be used to open c'_1. Let S' denote the first and W the second component of M'_1. Now, observe that $c_1 = c'_1 \cdot c''_1$, where c''_1 is a commitment to $(s'', v, 0, 0)$ with $s'' \leftarrow \mathbb{Z}_p$ and some $v \in \mathbb{Z}_p$ (where $v = 0$ in the issue protocol). c''_1 can be opened using the trivial opening value 1. Then due to homomorphic property of C and F'_{gp}, c_1 can be opened using D'_1 and message M'_1 with the first and the second message component being replaced by $S' \cdot g_1^{s''}$ and $W \cdot g_1^{v}$. Hence, unless the revealed token version number s_1 is different from $\mathsf{DLog}_{g_1}(S') + s''$ it can be considered a uniformly chosen value from $\mathbb{Z}_p$ as s'' is uniformly chosen. However, in case s_1 differs, we can extract an implicit message $M_1 \neq M'_1$ (where the first message component equals $g_1^{s_1}$) and opening value D_1 from π_1 which can also be used to open c_1. This violates the F'_{gp}-binding property of C. The argument for c_2 is analogous.

Case 2: $\mathsf{IdentDS}(\mathsf{pk}_{\mathcal{I}}, \mathsf{dstag}_1, \mathsf{dstag}_2) \neq (\mathsf{pk}^{(1)}_{\mathcal{U}}, \Pi)$
Let us consider $\mathsf{dstag}_1 = (s, (t, u_2))$ and $\mathsf{dstag}_2 = (s, (t', u'_2))$, where $t = \mathsf{sk}^{(1)}_{\mathcal{U}} u_2 + u_1$ and $t' = \mathsf{sk}^{(2)}_{\mathcal{U}} u'_2 + u'_1$. Note that $\mathsf{IdentDS}(\mathsf{pk}_{\mathcal{I}}, \mathsf{dstag}_1, \mathsf{dstag}_2) = (\mathsf{pk}^{(1)}_{\mathcal{U}}, \mathsf{sk}^{(1)}_{\mathcal{U}})$ if the following conditions are satisfied: $\mathsf{sk}^{(1)}_{\mathcal{U}} = \mathsf{sk}^{(2)}_{\mathcal{U}}$, $\mathsf{pk}^{(1)}_{\mathcal{U}} = g_1^{\mathsf{sk}^{(1)}_{\mathcal{U}}}$, $u_2 \neq u'_2$, and $u_1 = u'_1$. In the following we show that these conditions are satisfied with overwhelming probability and so Case 2 can only occur with negligible probability.

Let us consider the proofs π_1 and π_2. We can assume that $\mathsf{pk}^{(1)}_{\mathcal{U}} = \mathsf{pk}^{(2)}_{\mathcal{U}}$ (otherwise Case 1 is already satisfied). From the soundness of P2 and P3, the perfect binding property of E, and the equations shown in the proofs, it follows that $\mathsf{pk}^{(1)}_{\mathcal{U}} = g_1^{\mathsf{sk}^{(1)}_{\mathcal{U}}}$, $\mathsf{pk}^{(2)}_{\mathcal{U}} = g_1^{\mathsf{sk}^{(2)}_{\mathcal{U}}}$, and $\mathsf{sk}^{(1)}_{\mathcal{U}} = \mathsf{sk}^{(2)}_{\mathcal{U}}$. Moreover, $u_2 \neq u'_2$ with overwhelming probability as those values are chosen uniformly at random by $\mathcal{AC}$ or $\mathcal{V}$. Thus, it remains to exclude the case $u_1 \neq u'_1$. For this, we consider the commitments c_1 and c_2 and distinguish between different subcases:

(1) $c_1 = c_2$. Due to the extractability of P2 and P3, we can extract two implicit messages M_1 and M_2 and corresponding opening values for this commitment from π_1 and π_2. M_1's last component will be $g_1^{u_1}$ and M_2's last component is equal to $g_1^{u'_1}$. Hence, F'_{gp}-binding of C would be violated.
(2) $c_1 \neq c_2$.
   (a) *c_1 or c_2 is new.* If c_1 or c_2 has not occurred in any previous transaction, the EUF-CMA security of S is violated as we can extract the signature on c_i from the corresponding proof π_i.
   (b) *c_1 and c_2 occurred before.* As already argued in the scope of Claim 1, this might only happen with negligible probability since we assume that $s_1 = s_2$.

Case 3: $\mathsf{IdentDS}(\mathsf{pk}_{\mathcal{I}}, \mathsf{dstag}_1, \mathsf{dstag}_2) = (\mathsf{pk}^{(1)}_{\mathcal{U}}, \Pi)$ *but* $\mathsf{VerifyGuilt}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}^{(1)}_{\mathcal{U}}, \Pi) = 0$. It follows immediately from our definition of IdentDS and VerifyGuilt that VerifyGuilt will never output 0 in this case. □

Theorem C.5 (Balance-Binding). *If* P1, P2, P3 *are perfectly* $F^{(1)}_{\mathsf{gp}}$-, $F^{(2)}_{\mathsf{gp}}$-, *and* $F^{(3)}_{\mathsf{gp}}$*-extractable, respectively,* C *is* F'_{gp}*-binding,* S *is EUF-CMA secure, and* E *is correct, then* BBAP *is balance-binding.*

Proof. Roughly speaking, we will prove our theorem by showing that (i) an adversary cannot make us miscount the balance associated with user ID $\mathsf{pk}_{\mathcal{U}}$ and (ii) he cannot overclaim or underclaim this computed balance.

We proceed in a sequence of games. We start with the real experiment $\mathsf{Exp}^{\mathsf{bb}}_{\mathsf{BBAP},\mathcal{A}}(n)$, and then gradually modify this experiment to ensure the properties mentioned above, until we reach an experiment where $\mathcal{A}$ has no chance to win. We show that if any two consecutive games would significantly differ, this results in adversaries against the security of the building blocks.

Let **Game** 1 be the experiment $\mathsf{Exp}^{\mathsf{bb}}_{\mathsf{BBAP},\mathcal{A}}(n)$. Let us additionally assume that the experiment keeps track of the internal details of every transaction which was successful from the issuer's, accumulator's, or verifier's point of view.

More precisely, for every successful MalIssue the experiment stores a record $(\mathsf{pk}_{\mathcal{U}}, \pi, c_{\text{out}}, s'', W = 1, v = 0)$, where $\mathsf{pk}_{\mathcal{U}}$ is the user public key given as input to $\mathcal{I}$, π is the proof sent by $\mathcal{U}$, and $c_{\text{out}} = c$ and s'' are the commitment and the random value (the additive share of the token version number s) sent by $\mathcal{I}$. Moreover, for every successful MalAcc/MalVer the experiment stores a record $(\mathsf{pk}_{\mathcal{U}}, \pi, c_{\text{in}}, c_{\text{out}}, s, s'', W, v)$, where $\mathsf{pk}_{\mathcal{U}}$ is the public key extracted by ExtractUID, v is the value given as input to $\mathcal{AC}/\mathcal{V}$, $c_{\text{out}} = c^*$ and s'' are the commitment and the random value sent by $\mathcal{AC}/\mathcal{V}$, π and s are the proof and the token version number sent by $\mathcal{U}$, and $c_{\text{in}} = c$ is the commitment extracted from π, and W is the balance (as G_1-element) which can be computed from input w to $\mathcal{V}$ and which can be extracted from π in the case of MalAcc calls. As P2 and P3 are perfectly extractable, a verifying proof will always allow this.

Let $\mathsf{Adv}^{\mathsf{Game}_i}_{\mathsf{BBAP},\mathcal{A}}(n) = \Pr[\mathsf{Exp}^{\mathsf{Game}_i}_{\mathsf{BBAP},\mathcal{A}}(n) = 1]$ denote the advantage of $\mathcal{A}$ in Game i. Thus, by definition,

$$\mathsf{Adv}^{\mathsf{Game}_1}_{\mathsf{BBAP},\mathcal{A}}(n) = \mathsf{Adv}^{\mathsf{bb}}_{\mathsf{BBAP},\mathcal{A}}(n) \tag{20}$$

To understand the changes to the original experiment done in the following, it is best to picture the set of all successfull transactions as a directed graph with nodes labelled by the corresponding records as introduced above. There is an edge from node A to node B if A's c_{out} label equals B's c_{in} label.

In **Game** 2, we modify the previous game as follows. The experiment performs a check for every new successful MalAcc/MalVer transaction. Let $\mathsf{rec} := (\widehat{\mathsf{pk}_{\mathcal{U}}}, \hat{\pi}, \hat{c}_{\text{in}}, \hat{c}_{\text{out}}, \hat{s}, \hat{s}'', \hat{W}, \hat{v})$ denote the record of this new transaction. Then the experiment checks all previous MalIssue, MalAcc, and MalVer records whether there exists a record satisfying $c_{\text{out}} = \hat{c}_{\text{in}}$. If this is not the case, the experiment immediately aborts and returns 0. We call this event failure event $\mathfrak{F}_1$ (no predecessor commitment). Note that if this event happens, the signature σ extractable from $\hat{\pi}$ is a valid signature on the new message $\hat{c}_{\text{in}}$, due to the soundness of P2 or P3.

Hence, we can construct an EUF-CMA adversary $\mathcal{B}$ against S with advantage

$$\mathsf{Adv}^{\mathsf{euf\text{-}cma}}_{\mathsf{S},\mathcal{B}}(n) = \Pr[\mathfrak{F}_1] \tag{21}$$

Note that if the failure event does not happen, every commitment c_{in} from a MalAcc/MalVer record has been generated in a previous transaction. Hence, the indegree of every node representing a accumulation/verification transaction is at least one.

In **Game** 3, we again modify the previous game. The experiment now additionally checks whether $\hat{c}_{\text{in}}$ occurs in more than one previous transaction records as c_{out}. As soon as this happens, the experiment immediately aborts and returns 0. We call this event, failure event $\mathfrak{F}_2$ (two predecessor commitments). Let us consider two such previous records rec_1 and rec_2 and let s_1'' and s_2'' be the random numbers sent to $\mathcal{U}$ in these records, respectively. Then we can split up the event into two events $\mathfrak{F}_2^{(s_1''\neq s_2'')}$ and $\mathfrak{F}_2^{(s_1''=s_2'')}$. In the event $\mathfrak{F}_2^{(s_1''\neq s_2'')}$, it additionally holds that $s_1'' \neq s_2''$. Analogously, in the event $\mathfrak{F}_2^{(s_1''=s_2'')}$, it additionally holds that $s_1'' = s_2''$. Note that

$$\Pr[\mathfrak{F}_2^{(s_1''=s_2'')}] = \Pr[\mathfrak{F}_2 \ \wedge \ s_1'' = s_2''] \leq \frac{m^2}{p} \ , \tag{22}$$

where m is a polynomial bound on the number of MalAcc and MalVer queries.

Let us now consider the event $\mathfrak{F}_2^{(s_1''\neq s_2'')}$. As already argued in the proof sketch for Theorem C.4, in this case we can extract two valid openings for $\hat{c}$ such that the corresponding implicit message vectors differ in the token version number component.

Hence, we can construct F'_{gp}-binding adversary C_1 against C with advantage

$$\mathsf{Adv}^{F_{\text{gp}}\text{-bind}}_{\mathsf{C},C_1}(n) = \Pr[\mathfrak{F}_2^{(s_1''\neq s_2'')}] \tag{23}$$

Note that if none of the failure events happen, every commitment c_{in} from a MalAcc/MalVer record has been generated in exactly one previous transaction. Hence, the indegree of every node representing a accumulation/verification transaction is at exactly one.

In **Game** 4, we modify the previous game as follows. The experiment additionally checks whether $\hat{c}$ already occurred as c_{in} in a previous MalAcc/MalVer record. If this is the case, the experiment immediately aborts and returns 0. We call this event, failure event $\mathfrak{F}_3$ (two successor commitments). Let us consider such a previous record rec containing token version number s and proof π. Then we split up the event into the event $\mathfrak{F}_3^{(s\neq\hat{s})}$ where additionally holds that $s \neq \hat{s}$ and the event $\mathfrak{F}_3^{(s=\hat{s})}$ where additionally holds that $s = \hat{s}$. Note that

$$\Pr[\mathsf{Exp}^{\mathsf{Game}_1}_{\mathsf{BBAP},\mathcal{A}}(n) = 1 \wedge \mathfrak{F}_3^{(s=\hat{s})}] = 0 \tag{24}$$

due to the winning conditions of $\mathcal{A}$. If $\mathfrak{F}_3^{(s\neq\hat{s})}$ happens, we can extract two implicit messages $\hat{M} \neq M$ and opening values $\hat{D}$ and D from $\hat{\pi}$ and π, respectively, which can be used to open $\widehat{c_{\text{in}}} = c_{\text{in}}$.

Hence, we can construct F'_{gp}-binding aversary C_2 against C with advantage

$$\mathsf{Adv}^{F_{\text{gp}}\text{-bind}}_{\mathsf{C},C_2}(n) = \Pr[\mathfrak{F}_3^{(s\neq\hat{s})}] \tag{25}$$

Note that if none of the failure events happen, every commitment c_{in} from a MalAcc/MalVer record has been generated by exactly one previous transaction and the corresponding c_{out} is used in at most one subsequent transaction. Hence, the indegree of every node representing a accumulation/verification transaction is at exactly one and its outdegree is at most one.

In **Game** 5, we add yet another check: The experiment additionally checks for each new MalAcc/MalVer record, whether for previous records where $\hat{c}$ occurred as c_{out} (there is exactly one by now) also holds that $\widehat{\mathsf{pk}_{\mathcal{U}}}$ appears as $\mathsf{pk}_{\mathcal{U}}$. In other words, we check if the same user is associated with both the transaction generating

$\hat{c}$ and the transaction making use of $\hat{c}$. If this is not the case for the first time, the experiment immediately aborts and returns 0. We call this event, failure event $\mathfrak{F}_4$ (miscount). Let us consider such a previous record rec containing proof π and user id $\mathsf{pk}_{\mathcal{U}}$. As all proofs are extractable and the hidden id token is perfectly binding, we can now extract two different messages (which at least differ in the third component since $\widehat{\mathsf{pk}}_{\mathcal{U}} \neq \mathsf{pk}_{\mathcal{U}}$) and opening values for $\widehat{c_{\text{in}}} = c_{\text{in}}$ from $\hat{\pi}$ and π.

Hence, we can again construct an F'_{gp}-binding aversary C_3 against C with advantage

$$\mathsf{Adv}^{F_{\text{gp}}\text{-bind}}_{\text{C},C_3}(n) = \Pr[\mathfrak{F}_4] \tag{26}$$

Note that if none of the failure events happen, the following holds for every every commitment c_{in} and c_{out} from a MalAcc/MalVer record: c_{in} has been generated by exactly one previous transaction and c_{out} is used in at most one subsequent transaction. All three transactions are associated with the same user ID. Hence, if we picture the set of all successfull transactions as a directed graph, all paths are labelled with exactly one user ID. Moreover, for each user ID there is at most one such path, since there might be at most one issue transaction for a fixed $\mathsf{pk}_{\mathcal{U}}$. So to compute the legitimate balance for $\mathsf{pk}_{\mathcal{U}}$ one just has to walk along the corresponding path and add up the v values.

In **Game** 6, we check for over- and underclaims along the path: The experiment additionally checks for each new MalAcc/MalVer record, whether for its predecessor record ($c_{\text{out}} = \hat{c}_{\text{in}}$) holds that $\hat{W} = Wg_1^v$. Otherwise, a wrong balance has been successfully claimed in the new transaction. If this the case, the experiment immediately aborts and returns 0. We call this event, failure event $\mathfrak{F}_5$ (wrong claim). If this event happens, we can easily extract two different openings for $\hat{c}_{\text{in}}$ from $\hat{\pi}$ and π which differ in the balance component of the message.

Hence, we can again construct an F'_{gp}-binding aversary C_4 against C with advantage

$$\mathsf{Adv}^{F_{\text{gp}}\text{-bind}}_{\text{C},C_4}(n) = \Pr[\mathfrak{F}_5] \tag{27}$$

Moreover, note that if none of the failure events happened, then 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.

Putting things together. Considering Eq. (20) through Eq. (27) we obtain

$$\begin{aligned}
\mathsf{Adv}^{\text{Game}_1}_{\text{BBAP},\mathcal{A}}(n) &= \Pr\left[\mathsf{Exp}^{\text{Game}_1}_{\text{BBAP},\mathcal{A}}(n) = 1 \wedge \neg\left(\bigvee_{i=1}^{5}\mathfrak{F}_i\right)\right] \\
&+ \Pr\left[\mathsf{Exp}^{\text{Game}_1}_{\text{BBAP},\mathcal{A}}(n) = 1 \wedge \left(\bigvee_{i=1}^{5}\mathfrak{F}_i\right)\right] \\
&= \mathsf{Adv}^{\text{Game}_6}_{\text{BBAP},\mathcal{A}}(n) \\
&+ \Pr\left[\mathsf{Exp}^{\text{Game}_1}_{\text{BBAP},\mathcal{A}}(n) = 1 \wedge \left(\bigvee_{i=1}^{5}\mathfrak{F}_i\right)\right] \\
&\leq \Pr\left[\mathsf{Exp}^{\text{Game}_1}_{\text{BBAP},\mathcal{A}}(n) = 1 \wedge \mathfrak{F}_2\right] \\
&+ \Pr\left[\mathsf{Exp}^{\text{Game}_1}_{\text{BBAP},\mathcal{A}}(n) = 1 \wedge \mathfrak{F}_3\right] \\
&+ \Pr[\mathfrak{F}_1] + \Pr[\mathfrak{F}_4] + \Pr[\mathfrak{F}_5] \\
&= \Pr\left[\mathsf{Exp}^{\text{Game}_1}_{\text{BBAP},\mathcal{A}}(n) = 1 \wedge \mathfrak{F}_2^{(s_1'' \neq s_2'')}\right] \\
&+ \Pr\left[\mathsf{Exp}^{\text{Game}_1}_{\text{BBAP},\mathcal{A}}(n) = 1 \wedge \mathfrak{F}_2^{(s_1'' = s_2'')}\right] \\
&+ \Pr\left[\mathsf{Exp}^{\text{Game}_1}_{\text{BBAP},\mathcal{A}}(n) = 1 \wedge \mathfrak{F}_3^{(s \neq \hat{s})}\right] \\
&+ \Pr\left[\mathsf{Exp}^{\text{Game}_1}_{\text{BBAP},\mathcal{A}}(n) = 1 \wedge \mathfrak{F}_3^{(s = \hat{s})}\right] \\
&+ \Pr[\mathfrak{F}_1] + \Pr[\mathfrak{F}_4] + \Pr[\mathfrak{F}_5] \\
&\leq \Pr[\mathfrak{F}_2^{(s_1'' \neq s_2'')}] + \Pr[\mathfrak{F}_2^{(s_1'' = s_2'')}] + \Pr[\mathfrak{F}_3^{(s \neq \hat{s})}] \\
&+ \Pr\left[\mathsf{Exp}^{\text{Game}_1}_{\text{BBAP},\mathcal{A}}(n) = 1 \wedge \mathfrak{F}_3^{(s = \hat{s})}\right] \\
&+ \Pr[\mathfrak{F}_1] + \Pr[\mathfrak{F}_4] + \Pr[\mathfrak{F}_5] \\
&\leq \mathsf{Adv}^{F_{\text{gp}}\text{-bind}}_{\text{C},C_1}(n) + \frac{m^2}{p} \\
&+ \mathsf{Adv}^{F_{\text{gp}}\text{-bind}}_{\text{C},C_2}(n) + \mathsf{Adv}^{\text{euf-cma}}_{\text{S},\mathcal{B}}(n) \\
&+ \mathsf{Adv}^{F_{\text{gp}}\text{-bind}}_{\text{C},C_3}(n) + \mathsf{Adv}^{F_{\text{gp}}\text{-bind}}_{\text{C},C_4}(n)
\end{aligned} \tag{28}$$

As m is polynomial in n and we assume that S is EUF-CMA secure and C is F'_{gp}-binding, it follows that $\mathcal{A}$'s advantage is negligible. □

D USER SECURITY AND PRIVACY PROOFS

In this section, we show that our construction in Section 5 is privacy-preserving according to Theorem 4.7. Before we state our theorem formally, we define a sequence of games Game 1, …, Game 6. In the proof after Theorem D.1 we show that each pair of consecutive games can only be distinguished with negligible probability.

Game 1 denotes the game where all oracles are running the real protocol and the last Game 6 denotes the game where the oracles are replaced by SimHonIssue, SimHonAcc, SimHonVer and SimCorrupt respectively. We denote the experiment with an adversary $\mathcal{A}$ playing the game i by $\mathsf{Exp}^{\text{Game}_i}_{\text{BBAP},\mathcal{A}}$. As in the real protocols the $\mathcal{U}_{\text{sim}}$ algorithm outputs a tuple $(\cdot, b)$ with $b \in \{0,1\}$ to the oracle to indicate if the protocol execution had been successful from the user's perspective. If an error is reported to the oracle, the oracle replies with $\perp$-messages to the adversary in any future call for the same $\mathsf{pk}_{\mathcal{U}}$ to mimic the real behavior. In the following we give a rough overview about the games. We write Setup_i, $\mathsf{HonIssue}_i$, HonAcc_i, HonVer_i, and $\mathsf{Corrupt}_i$ to denote the implementations of the oracles in the game i. The oracle HonUser remains unchanged in all games.

In **Game 1** we set $\mathsf{Setup}_1 = \mathsf{Setup}$, $\mathsf{HonIssue}_1 = \mathsf{RealHonIssue}$, $\mathsf{HonAcc}_1 = \mathsf{RealHonAcc}$, $\mathsf{HonVer}_1 = \mathsf{RealHonVer}$ and $\mathsf{Corrupt}_1 = \mathsf{RealCorrupt}$ as in Figures 8 to 11. In other words $\mathsf{Exp}^{\text{Game}_1}_{\text{BBAP},\mathcal{A}}$ and $\mathsf{Exp}^{\text{priv-real}}_{\text{BBAP},\mathcal{A}}$ are identical.

In **Game 2** we modify Setup_2 such that CRS_{pok} and CRS_{com} are not created by SetupPoK(gp) and Gen(gp) resp., but by $(\text{CRS}_{\text{pok}}, \text{td}_{\text{spok}}) \leftarrow \text{SetupSPoK}(\text{gp}, \mathit{hint})$ and $(\text{CRS}_{\text{com}}, \text{td}_{\text{ecom}}) \leftarrow \text{SimGen}(\text{gp})$ (cp. Theorems B.7 and B.8). Moreover, we set $\mathit{hint} := \text{td}_{\text{ecom}}$, i. e. the simulator of the NIZK scheme also gets the equivocation trapdoor of the commitment scheme. Otherwise the Groth-Sahai proof system cannot be proven zero-knowledge for the particular type of equations used here. The simulator SimProof needs to know the discrete logarithms of the fixed constants on the right hand-side of the statement in order to successfully simulate a proof. These constants happen to be the group elements used in the commitment scheme and are chosen before the actual statement.

In **Game 3** we redefine HonIssue_3, HonAcc_3 and HonVer_3 such that the oracles (playing the role of the user) send simulated proofs to the adversary. E. g. instead of executing P1.Prove, P2.Prove, P3.Prove (cp. Figs. 9 and 11) the code of the simulator SimProof′.[14]

In **Game 4** the commitments of the user during an interaction within HonIssue_4, HonAcc_4, or HonVer_4 are not created by C.Com but by C.SimCom. If a user is corrupted by calling Corrupt_4, the simulated commitment inside the token is equivoked via C.Equiv before returning the token τ to the adversary such that the commitment contains the correct user key $\text{sk}_{\mathcal{U}}$ and the correct balance w. Moreover, some additional precautions are necessary such that UVer at the end of HonIssue_4, HonAcc_4 or HonVer_4 behaves as expected. Without a corresponding decommitment d' the simulated commitment is information-theoretic empty and thus UVer cannot be called. However, UVer cannot be omitted as otherwise the operator could try to cheat on the user (by adding a wrong value $\tilde{v}$) and check if the user detects such a cheating. This would give the adversary a strategy to distinguish between a real and ideal simulation. For this reason, the original commitment that has been sent to the operator is equivoked as if it had contained the balance $w = 0$ and the private key $\text{sk}_{\mathcal{U}} = 0$ and then the resulting commitment is verified if it contains the new balance $w^* = v$ and the user id $\text{pk}_{\mathcal{U}} = g_1^0 = 1$. As the ZK-statements are nor longer valid, the simulator SimProof is called directly without the wrapper SimProof′.

In the **Game 5** we modify HonAcc_5 and HonVer_5 such that t is randomly chosen and (u_2, t) is not necessarily a point on the line $t = \text{sk}_{\mathcal{U}} u_2 + u_1 \mod p$.

In **Game 6** the encryption of $\text{pk}_{\mathcal{U}}$ for the hid for HonAcc_6 and HonVer_6 is replaced by an encryption of $g_1^0 = 1$. The clear-text $\text{pk}_{\mathcal{U}}$ of HonIssue_6 remains unchanged, because the true $\text{pk}_{\mathcal{U}}$ is part of the statement and must not be changed or otherwise the issuer would notice a difference. We note that $\text{Exp}^{\text{Game}_6}_{\text{BBAP},\mathcal{A}}$ and $\text{Exp}^{\text{priv-ideal}}_{\text{BBAP},\mathcal{A}}$ are identical. See Figs. 15 to 18 for the details of all modifications.

Remark 5 (Global State for Oracles). *Please note that all the oracles* HonUser, HonIssue, HonAcc, HonVer *and* Corrupt *use a joint, global state. This global state stores the* $\text{sk}_{\mathcal{U}}$*, the current balance* w *and the last message sent from the adversary to the user for each user* $\text{pk}_{\mathcal{U}}$*. This is necessary as the adversary is the only operator in the privacy game and thus knows what balance* w *and what signature* σ *is expected upon corruption of a particular user. This global state*

[14] N.b.: To be precise, there are three different simulators SimProof′: one that simulates proof for each of P1.Prove, P2.Prove and P3.Prove

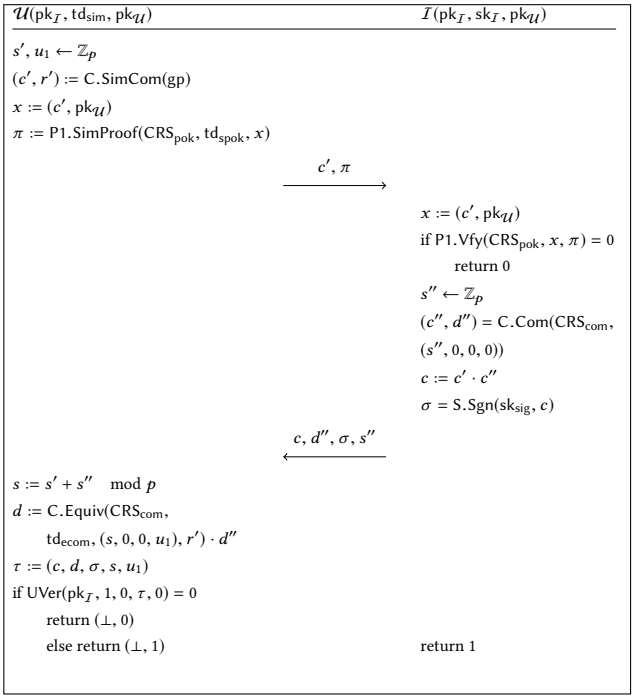

Figure 15: Issue protocol of Game 6

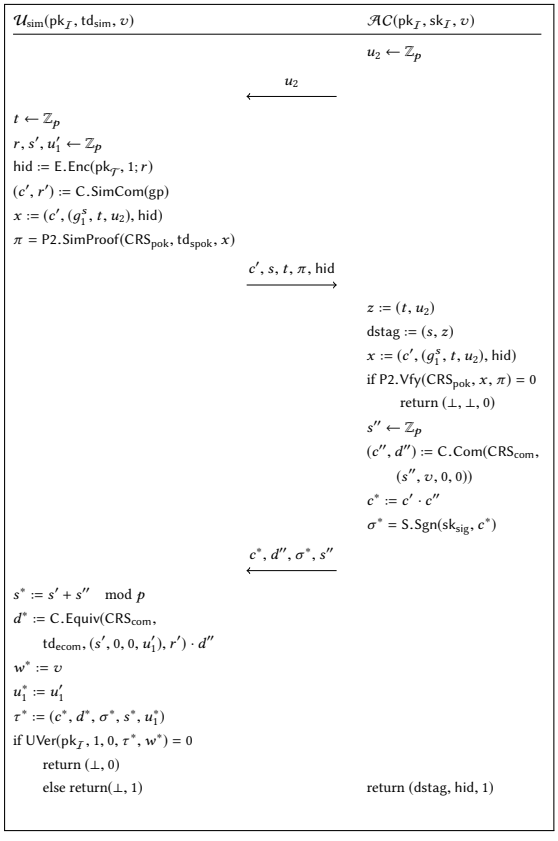

Figure 16: Accumulation protocol of Game 6

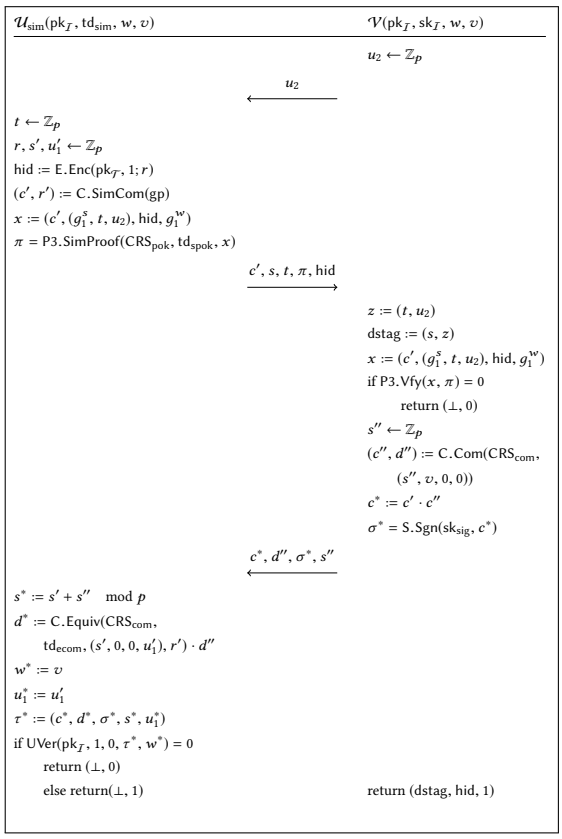

Figure 17: Verification protocol of Game 6

Corrupt($\text{pk}_\mathcal{U}$)

Get the latest commitment c^* and signature σ^* that were sent by the adversary during an HonAcc or HonVer to the oracle for this $\text{pk}_\mathcal{U}$. Let r' be the equivocation information that was created by C.SimCom during that interaction. Let w we the recent balance of the user that the adversary expects due to all interactions in HonAcc or HonVer.

$s^*, u_1^* \leftarrow \mathbb{Z}_p$

$d^* := \text{C.Equiv}(\text{CRS}_{\text{com}}, \text{td}_{\text{ecom}}, (s', w, \text{sk}_\mathcal{U}, u_1'), r')$

$\tau^* := (c^*, d^*, \sigma^*, s^*, u_1^*)$

return $(\text{pk}_\mathcal{U}, \text{sk}_\mathcal{U}, \tau^*, w)$

Figure 18: Corruption algorithm of Game 6

is never read (only updated) by HonIssue, HonAcc, *or* HonVer *and thus privacy is guaranteed as long as the user is uncorrupted.*

Theorem D.1. *If* P1, P2, P3 *are composable zero-knowledge,* C *is equivocable and* E *is IND-CPA secure, then* BBAP *is privacy-preserving (cp. Theorem 4.7).*

Proof. *From Game 1 to Game 2:* This game hop only changes how the the CRS is created during the setup phase. However, this is indistinguishable for both CRS_{pok} and CRS_{com} (see the composable zero-knowledge property of Theorem B.8 and the equivocality property of Theorem B.7, resp., condition (a) each)

From Game 2 to Game 3: This game hop replaces the real proofs by simulated proofs. Note that we prove valid statements, hence SimProof returns an actual proof π and not $\perp$. We proof the indistinguishability of this both games by a nested game-hopping argument and three reductions to the composable zero-knowledge property of the NIZK. First all proofs of the proof system P1 are replaced by simulated proofs. Assume there exists an adversary $\mathcal{A}'$ that notices a difference. Then we can construct an adversary $\mathcal{A}$ that has a non-negligible advantage $\text{Adv}^{\text{zk}}_{\text{POK},\mathcal{A}}(n)$. Internally $\mathcal{A}$ runs $\mathcal{A}'$ and simulates all user oracles (HonIssue, HonAcc, HonVer, ...) for $\mathcal{A}'$. All calls to P1.Prove are forwarded by $\mathcal{A}$ to its own oracle in the challenge game which is either P1.Prove or P1.SimProof. $\mathcal{A}$ outputs whatever $\mathcal{A}'$ outputs. This completes the description of the first sub-experiment. In the second step all proofs of the proof system P2 are replaced by simulated proofs. Assume that there exists an adversary $\mathcal{B}'$ that notices a difference. Again, we construct an adversary $\mathcal{B}$ that externally plays the ZK-game of the proof system P2 and internally runs $\mathcal{B}'$. However, this time $\mathcal{B}$ needs to give simulated proofs for P1, if it internally plays the role of the user within the scope of HonIssue. Note, that the adversary $\mathcal{B}$ also gets the simulation trapdoor td_{spok} in the ZK-game, hence $\mathcal{B}$ can run P1.SimProof itself. In the third sub-experiment all proofs of P3 are replaced by simulated ones using the same proof argument again.

From Game 3 to Game 4: Assume there is an adversary $\mathcal{A}'$ that notices a difference between $\text{Exp}^{\text{Game}_3}_{\text{BBAP},\mathcal{A}}$ and $\text{Exp}^{\text{Game}_4}_{\text{BBAP},\mathcal{A}}$. Again we argue by a sequence of nested games $\text{Game}_{3,0}$ to $\text{Game}_{3,m} = \text{Game}_4$. In game $\text{Game}_{3,j}$ the first j commitments are replaced by simulated commitments while all commitments from $j+1$ to m are real. As the commitment consists of a single message, this ordering is well-defined. As $\mathcal{A}'$ notices a difference between $\text{Game}_{3,0}$ to $\text{Game}_{3,m}$ there must be an index $j \in [m]$ such that $\mathcal{A}'$ can distinguish $\text{Game}_{3,j}$ and $\text{Game}_{3,j-1}$. We use this to construct an adversary $\mathcal{A}$ against the equivocality of the commitment scheme (cp. Theorem B.7, Item 3b). Internally, $\mathcal{A}$ runs $\mathcal{A}'$ and simulates the user oracles for $\mathcal{A}'$. $\mathcal{A}$ can equivoce the first $j-1$ commitments internally, because $\mathcal{A}$ also get the equivocation trapdoor as its input. $\mathcal{A}$ forwards the j'th commitment to the challenger and plays all remaining commitments real. Then $\mathcal{A}$ has the same advantage $\text{Adv}^{\equiv}_{\text{COM},\mathcal{A}}(n)$ as $\mathcal{A}'$ in distinguishing between $\text{Game}_{3,j}$ and $\text{Game}_{3,j-1}$.

From Game 4 to Game 5: We have to distinguish two cases: (a) this interaction follows a corruption for the same user $\text{pk}_\mathcal{U}$, or (b) the preceding interaction for the same user was any other oracle call but Corrupt. In case (b) this game does not change anything from the adversaries perspective. As u_1 is uniformly chosen in every interaction, the user uses a new linear equation every time, hence $t = \text{sk}_\mathcal{U} u_2 + u_1$ is uniform as well and we have already removed any other dependence on u_1 in the previous game hops. However, in case (a) the adversary knows the latest value u_1 that will be used by the user in the next interaction. As u_2 is chosen from the adversary anyway, the adversary is able to check if $(u_2, t) := (u_2, \text{pk}_\mathcal{U} u_2 + u_1)$

mod p is actually a point on the correct line. For this reason the real algorithm of the user is executed in the interaction that follows a corruption.

From Game 5 to Game 6: First note, that hid never gets decrypted and the secret key is unknown to the adversary. Assume that there is an efficient adversary $\mathcal{A}'$ that can distinguish between $\mathsf{Exp}^{\mathsf{Game}_5}_{\mathsf{BBAP},\mathcal{A}}$ and $\mathsf{Exp}^{\mathsf{Game}_6}_{\mathsf{BBAP},\mathcal{A}}$. Again we construct a sequence of nested games $\mathsf{Game}_{5,0}$ to $\mathsf{Game}_{5,m} = \mathsf{Game}_6$, whereby $\mathsf{Game}_{5,k}$ denotes the game in that the first k sessions encrypt the constant g_1 for hid and the remaining sessions encrypt the real $\mathsf{pk}_{\mathcal{U}}$. If $\mathcal{A}'$ can distinguish $\mathsf{Game}_{5,k}$ from $\mathsf{Game}_{5,k+1}$, then we can construct an efficient adversary $\mathcal{A}$ against the IND-CPA security of the encryption scheme. Internally, $\mathcal{A}$ runs $\mathcal{A}'$ plays the honest user oracles for $\mathcal{A}'$ and either asks the challenger to encrypt either the real $\mathsf{pk}_{\mathcal{U}}$ or g_1 during the k'th session. $\mathcal{A}$ outputs whatever $\mathcal{A}'$ outputs. □

Finally, we prove that our BBA+ scheme also protects the user against a false accusation of double-spending according to Theorem 4.8. The proof is almost generic in the sense that the statement is nearly immediately implied by the privacy property of the BBA+ scheme.

Theorem D.2. *If* BBAP *is privacy-preserving and the calculation of discrete logarithms in G_1 is a computationally hard problem, then* BBAP *is secure against false accusation of double-spending.*

Proof. Assume there is an efficient adversary $\mathcal{A}$ that breaks the false-accusation protection (cp. Theorem 4.8). Observe that the adversary $\mathcal{A}$ in the experiment $\mathsf{Exp}^{\mathsf{facp}}_{\mathsf{BBAP},\mathcal{A}}(n)$ (see. Fig. 7) has access to the oracles RealHonIssue, RealHonAcc, RealHonVer for exactly one user identity $\mathsf{pk}_{\mathcal{U}}$ that was chosen by the experiment in advance. This is a strict subset of the oracles an adversary in the privacy game is allowed to use. We replace all these oracles by simulated oracles SimHonIssue, SimHonAcc, SimHonVer and distinguish two cases:

(1) Either $\mathcal{A}$ still outputs a valid proof Π of guilt with non-negligible probability. Essentially, this means the adversary outputs the discrete logarithm $\mathsf{sk}_{\mathcal{U}}$ of $\mathsf{pk}_{\mathcal{U}}$. All simulation oracles are PPT and are – except for $\mathsf{pk}_{\mathcal{U}}$ – only equipped with input that is information-theoretic independent of $\mathsf{sk}_{\mathcal{U}}$. Hence, we could construct an adversary $\mathcal{A}'$ that gets $\mathsf{pk}_{\mathcal{U}}$ as input, incorporates $\mathcal{A}$ together with all oracles and outputs the discrete logarithm of $\mathsf{pk}_{\mathcal{U}}$. This contradicts the CDH assumption.
(2) Or $\mathcal{A}$ is not able to output a valid proof Π of guilt anymore, i. e. we have $\mathsf{VerifyGuilt}(\mathsf{CRS}, \mathsf{pk}_{\mathcal{U}}, \Pi) = 0$ with overwhelming probability. In this case we can use $\mathcal{A}$ to construct an efficient adversary $\mathcal{A}'$ that breaks the privacy of our BBA+ scheme. First $\mathcal{A}'$ generates a user honestly for $\mathcal{A}$ by using its own HonUser oracle in the privacy game. Then $\mathcal{A}'$ executes $\mathcal{A}$ and forwards all oracle calls to its own oracles. After $\mathcal{A}$ has terminated $\mathcal{A}'$ returns the result of $\mathsf{VerifyGuilt}(\mathsf{CRS}, \mathsf{pk}_{\mathcal{U}}, \Pi)$ as its own output to the experiment. Hence, $\mathcal{A}'$ uses the validity of the proof of guilt Π to distinguish a real or ideal execution. □

E FULL-FLEDGED SECURITY MODEL

In this section, we outline an extended security model for BBA+ schemes, where malicious users may additionally corrupt honest users and passively eavesdrop on protocol executions between honest users and the issuers, accumulators and verifiers. In particular, we give alternative, more general definitions for the following security properties:

- owner-binding with respect to Issue,
- owner-binding with respect to Accum and Vfy,
- balance-binding,
- double-spending detection, and
- false accusation protection.

The security experiments we give in this section resemble the ones from Section 4.3.

We claim that if a scheme fulfills our simplified security definitions given in Section 4.3, then the scheme (or at least an encrypted version of that scheme) is secure according to the following definitions. Note though, that we have not conducted formal proofs for this. Proof *sketches* are given in Appendix F. We first explain the key differences for the first four security properties (protecting the issuer from dishonest users), before we move to false accusation protection and privacy (protecting the users from the issuer).

E.1 System Security

Our alternative definitions for the first four properties given above are more general in that most of them provide the adversary with additional oracles. Using these oracles, the adversary may instruct a number of honest users to engage in protocol executions and obtain the transcripts of these executions. With respect to these honest users the adversary is passive: It may not tamper with the communication between the honest users and the issuer. It may also not force a user to behave maliciously or run protocols concurrently but only sequentially just as an honest user would always do.[15] The adversary may at any time (after a completed protocol run) corrupt one of the honest users and receive all of his secrets (like the BBA+ token, the user secret key, etc.) in this way. From this point on, the adversary can impersonate this user and the user may not be instructed anymore to engage in any protocols. In fact, the adversary could also play the role of the honest user if he wishes to do so.

Apart from being able to eavesdrop on honest users, the adversary may additionally participate in protocol executions with honest issuers, accumulators and verifiers as a malicious user, as in Section 4.3. In these protocol executions, the adversary may arbitrarily deviate from the protocol.

We now specify the additional oracles that the adversary may access:

- HonUser and Corrupt, as defined in Section 4.4.
- HonIssue($\mathsf{pk}_{\mathcal{U}}$) lets the honest user with public key $\mathsf{pk}_{\mathcal{U}}$ generated by HonUser() run the Issue protocol with an honest issuer provided that $\mathsf{pk}_{\mathcal{U}}$ has not been used before

[15] We leave creating a BBA+ scheme that has provable security against active attackers as an open problem. We believe that our base scheme can be converted to achieve active security by having messages transmitted over a secure channel, i.e. a channel that does not just ensure confidentiality (as our encryption does) but also integrity, message ordering, replay protection and the like. Proving this is future work.

Experiment $\mathsf{Exp}^{\text{ob-issue-full}}_{\mathsf{BBAP},\mathcal{A}}(n)$
$(\mathsf{CRS}, \mathsf{td}) \leftarrow \mathsf{Setup}(1^n)$
$(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}) \leftarrow \mathsf{IGen}(\mathsf{CRS})$
$b \leftarrow \mathcal{A}^{\mathsf{HonUser},\mathsf{HonIssue},\mathsf{HonAcc},\mathsf{HonVer},\mathsf{Corrupt},\mathsf{MalIssue},\mathsf{MalAcc},\mathsf{MalVer}}(\mathsf{CRS}, \mathsf{pk}_{\mathcal{I}})$
The experiment returns 1 iff $\mathcal{A}$ did a successful call to MalIssue on input of a public-key $\mathsf{pk}_{\mathcal{U}}$ generated by HonUser for which Corrupt has not been executed up to this MalIssue-call.

Figure 19: Alternative owner-binding experiment for BBAP **with respect to** Issue**.**

in a successful call to HonIssue or MalIssue, and there is no pending MalIssue call for $\mathsf{pk}_{\mathcal{U}}$. The latter ensures we only have a single token issued per user public key.

- $\mathsf{HonAcc}(\mathsf{pk}_{\mathcal{U}}, v)$ lets the honest user with public key $\mathsf{pk}_{\mathcal{U}}$ generated by HonUser() run the Accum protocol with the honest issuer $\mathcal{I}$ on common input $v \in \mathbb{Z}_p$, provided that $\mathsf{HonIssue}(\mathsf{pk}_{\mathcal{U}})$ has successfully been called and $\mathsf{Corrupt}(\mathsf{pk}_{\mathcal{U}})$ has not been called before. In this protocol run, the user will use the the most recent BBAP token τ and balance value w (received in the previous call for $\mathsf{pk}_{\mathcal{U}}$, which might have been a HonIssue, HonAcc or HonVer call).
- $\mathsf{HonVer}(\mathsf{pk}_{\mathcal{U}}, v)$ lets the honest user with public key $\mathsf{pk}_{\mathcal{U}}$ generated by HonUser() run the Vfy protocol with the honest issuer $\mathcal{I}$ on common input $v \in \mathbb{Z}_p$, and $w \in \mathbb{Z}_p$, where w is the current balance of the user with public key $\mathsf{pk}_{\mathcal{U}}$. This oracle may only be called if $\mathsf{HonIssue}(\mathsf{pk}_{\mathcal{U}})$ has sucessfully been called and $\mathsf{Corrupt}(\mathsf{pk}_{\mathcal{U}})$ has not been called before. In this protocol run, the user will use the the most recent BBAP token τ and balance value w.

The oracles HonIssue, HonAcc and HonVer return the transcript of all exchanged protocol messages to the adversary.

In addition to these oracles, the adversary $\mathcal{A}$ may call the MalIssue, MalAcc, and MalVer oracles as defined in Section 4. However, $\mathcal{A}$ may not call MalIssue on input of a public key that has been used with HonIssue before.

each user no oracle can be called concurrently, i. e. for any arbitrary but fixed $\mathsf{pk}_{\mathcal{U}}$ another oracle can only invoked if no previous oracle call for the same $\mathsf{pk}_{\mathcal{U}}$ is still pending.

We are now ready to state the modified security experiments and definitions. As stated before, they differ in the oracles available to the attacker, and the winning conditions have been adjusted accordingly. The security experiment for the following definition is given in Fig. 19. (The definition itself is completely analogous to Theorem 4.3.) In this experiment, the adversary needs not register a specific key $\mathsf{pk}_{\mathcal{U}}$ via the Issue protocol, but may choose among the keys generated by the HonUser oracle.

Definition E.1. A trapdoor-linkable BBA+ scheme BBAP is called owner-binding with respect to Issue *under eavesdropping on honest users* if for any PPT adversary $\mathcal{A}$ in the experiment $\mathsf{Exp}^{\text{ob-issue-full}}_{\mathsf{BBAP},\mathcal{A}}(n)$ the advantage of $\mathcal{A}$ defined by

$$\mathsf{Adv}^{\text{ob-issue-full}}_{\mathsf{BBAP},\mathcal{A}}(n) := \Pr[\mathsf{Exp}^{\text{ob-issue-full}}_{\mathsf{BBAP},\mathcal{A}}(n) = 1] \tag{29}$$

is negligible in n.

Theorem E.2 demands that an adversary may not be able to successfully call the accumulation or verification protocols for a forged token or a token not owned by him but an honest user. Note that a token is owned by the adversary if he has created this token by calling MalIssue or if the token belongs to a corrupted, previously honest user.

Experiment $\mathsf{Exp}^{\text{ob-acc-full}}_{\mathsf{BBAP},\mathcal{A}}(n)$
$(\mathsf{CRS}, \mathsf{td}) \leftarrow \mathsf{Setup}(1^n)$
$(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}) \leftarrow \mathsf{IGen}(\mathsf{CRS})$
$b \leftarrow \mathcal{A}^{\mathsf{HonUser},\mathsf{HonIssue},\mathsf{HonAcc},\mathsf{HonVer},\mathsf{Corrupt},\mathsf{MalIssue},\mathsf{MalAcc},\mathsf{MalVer}}(\mathsf{CRS}, \mathsf{pk}_{\mathcal{I}})$
The experiment returns 1 iff $\mathcal{A}$ did a successful call to MalAcc or MalVer such that ExtractUID applied to the hid being part of the view of this call outputs a public-key $\mathsf{pk}_{\mathcal{U}}$ for which
- there has been no successful execution of MalIssue or HonIssue or
- there has been a successful execution of HonIssue but no call to Corrupt

up to this MalAcc/MalVer-call.

Figure 20: Alternative owner-binding experiment for BBAP **with respect to** Accum**.**

Experiment $\mathsf{Exp}^{\text{bb-full}}_{\mathsf{BBAP},\mathcal{A}}(n)$
$(\mathsf{CRS}, \mathsf{td}) \leftarrow \mathsf{Setup}(1^n)$
$(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}) \leftarrow \mathsf{IGen}(\mathsf{CRS})$
$b \leftarrow \mathcal{A}^{\mathsf{HonUser},\mathsf{HonIssue},\mathsf{HonAcc},\mathsf{HonVer},\mathsf{Corrupt},\mathsf{MalIssue},\mathsf{MalAcc},\mathsf{MalVer}}(\mathsf{CRS}, \mathsf{pk}_{\mathcal{I}})$
The experiment returns 1 iff $\mathcal{A}$ did a successful call to MalVer resulting in a view $\mathsf{view} = (\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}, w, v, \mathsf{msgs}, s, z, 1) \in \mathcal{V}^{\mathsf{Vfy}}_{n,\mathsf{CRS}}$ and extracted user public-key $\mathsf{pk}_{\mathcal{U}} \leftarrow \mathsf{ExtractUID}^{\mathsf{Vfy}}(\mathsf{td}, \mathsf{view})$ such that the following conditions are satisfied:
- all successful MalIssue and MalAcc calls resulted in different token version numbers and
- the claimed balance $w \in \mathbb{Z}_p$ does not equal the sum of previously collected accumulation values v for $\mathsf{pk}_{\mathcal{U}}$, i.e.,

$$w \neq \sum_{v \in V_{\mathsf{pk}_{\mathcal{U}}}} v \in \mathbb{Z}_p,$$

where $V_{\mathsf{pk}_{\mathcal{U}}}$ is the list of all accumulation values $v \in \mathbb{Z}_p$ that appeared in previous calls to MalAcc, HonAcc or MalVer, HonVer for which $\mathsf{pk}_{\mathcal{U}}$ could be extracted using ExtractUID.

Figure 21: Balance binding experiment for BBAP**.**

Definition E.2. A trapdoor-linkable BBA+ scheme BBAP is called owner-binding with respect to Accum and Vfy *under eavesdropping on honest users* if for any PPT adversary $\mathcal{A}$ in the experiment $\mathsf{Exp}^{\text{ob-acc-full}}_{\mathsf{BBAP},\mathcal{A}}(n)$ from Fig. 20 the advantage of $\mathcal{A}$ defined by

$$\mathsf{Adv}^{\text{ob-acc-full}}_{\mathsf{BBAP},\mathcal{A}}(n) := \Pr[\mathsf{Exp}^{\text{ob-acc-full}}_{\mathsf{BBAP},\mathcal{A}}(n) = 1] \tag{30}$$

is negligible in n.

The following definition for the balance-binding property is analogous to the definition from Section 4.3, except that it has been adapted to account for the additional oracles.

Definition E.3. A trapdoor-linkable BBA+ scheme BBAP is called balance-binding *under eavesdropping on honest users* if for any PPT adversary $\mathcal{A}$ in the experiment $\mathsf{Exp}^{\text{bb-full}}_{\mathsf{BBAP},\mathcal{A}}(n)$ from Fig. 21 the advantage of $\mathcal{A}$ defined by

$$\mathsf{Adv}^{\text{bb-full}}_{\mathsf{BBAP},\mathcal{A}}(n) := \Pr[\mathsf{Exp}^{\text{bb-full}}_{\mathsf{BBAP},\mathcal{A}}(n) = 1] \tag{31}$$

is negligible in n.

Theorem E.4 enforces that two transactions leading to the same token version number have always been initiated by the same user and that this user can be identified.

Definition E.4. A trapdoor-linkable BBA+ scheme BBAP ensures double-spending detection *under eavesdropping on honest users* if

Experiment $\mathsf{Exp}^{\text{dsd-full}}_{\mathsf{BBAP},\mathcal{A}}(n)$
$(\mathsf{CRS}, \mathsf{td}) \leftarrow \mathsf{Setup}(1^n)$
$(\mathsf{pk}_I, \mathsf{sk}_I) \leftarrow \mathsf{IGen}(\mathsf{CRS})$
$b \leftarrow \mathcal{A}^{\mathsf{HonUser},\mathsf{HonIssue},\mathsf{HonAcc},\mathsf{HonVer},\mathsf{Corrupt},\mathsf{MalIssue},\mathsf{MalAcc},\mathsf{MalVer}}(\mathsf{CRS}, \mathsf{pk}_I)$
The experiment returns 1 iff $\mathcal{A}$ did two successful MalAcc/MalVer calls resulting in two views view_1 and view_2 including two double-spending tags $\mathsf{dstag}_1 = (s, z_1)$ and $\mathsf{dstag}_2 = (s, z_2)$ and extracted user public-keys $\mathsf{pk}^{(1)}_{\mathcal{U}}$ and $\mathsf{pk}^{(2)}_{\mathcal{U}}$ (using ExtractUID) such that one of the following conditions is satisfied:
- $\mathsf{pk}^{(1)}_{\mathcal{U}} \neq \mathsf{pk}^{(2)}_{\mathcal{U}}$ or
- $\mathsf{IdentDS}(\mathsf{dstag}_1, \mathsf{dstag}_2) \neq (\mathsf{pk}^{(1)}_{\mathcal{U}}, \Pi)$ or $\mathsf{VerifyGuilt}(\mathsf{pk}_I, \mathsf{pk}^{(1)}_{\mathcal{U}}, \Pi) = 0$

Figure 22: Double-spending detection experiment for BBAP**.**

Experiment $\mathsf{Exp}^{\text{facp-full}}_{\mathsf{BBAP},\mathcal{A}}(n)$
$(\mathsf{CRS}, \mathsf{td}) \leftarrow \mathsf{Setup}(1^n)$
$(\mathsf{pk}_I, \mathit{state}) \leftarrow \mathcal{A}(\mathsf{CRS})$
$(\mathsf{pk}_{\mathcal{U}}, \Pi) \leftarrow$
$\mathcal{A}^{\mathsf{HonUser},\mathsf{HonIssue},\mathsf{HonAcc},\mathsf{HonVer},\mathsf{Corrupt}}(\mathit{state})$
The experiment returns 1 iff $\mathsf{pk}_{\mathcal{U}}$ is a public-key generated by HonUser for which Corrupt has not been called and $\mathsf{VerifyGuilt}(\mathsf{pk}_I, \mathsf{pk}_{\mathcal{U}}, \Pi) = 1$.

Figure 23: False accusation protection experiment for BBAP**.**

for any PPT adversary $\mathcal{A}$ in the experiment $\mathsf{Exp}^{\text{dsd-full}}_{\mathsf{BBAP},\mathcal{A}}(n)$ from Fig. 22 the advantage of $\mathcal{A}$ defined by

$$\mathsf{Adv}^{\text{dsd-full}}_{\mathsf{BBAP},\mathcal{A}}(n) := \Pr[\mathsf{Exp}^{\text{dsd-full}}_{\mathsf{BBAP},\mathcal{A}}(n) = 1] \tag{32}$$

is negligible in n.

E.2 User Security and Privacy

We now move to the security definitions for protecting users from malicious issuers.

We give an alternative definition for the false accusation protection property, where the adversary may additionally corrupt honest users. Furthermore, the adversary is not restricted to produce a convincing double-spending proof against a specific public key $\mathsf{pk}^*_{\mathcal{U}}$, but $\mathcal{A}$ may pick any $\mathsf{pk}_{\mathcal{U}}$ output by the HonUser oracle.

We don't provide the adversary with the MalIssue, MalAcc or MalVer oracles, since these oracles would model interactions in which both parties are corrupted, and the attacker could simulate such interactions for itself.

Definition E.5. A trapdoor-linkable BBA+ scheme BBAP ensures false-accusation protection *under eavesdropping on honest users* if for any PPT adversary $\mathcal{A}$ in the experiment $\mathsf{Exp}^{\text{facp-full}}_{\mathsf{BBAP},\mathcal{A}}(n)$ from Fig. 23 the advantage of $\mathcal{A}$ defined by

$$\mathsf{Adv}^{\text{facp-full}}_{\mathsf{BBAP},\mathcal{A}}(n) := \Pr[\mathsf{Exp}^{\text{facp-full}}_{\mathsf{BBAP},\mathcal{A}}(n) = 1] \tag{33}$$

is negligible in n.

We do not give an alternative definition for privacy.

F RELATION BETWEEN OUR SECURITY MODELS

In this section, we argue that a BBA+ scheme that is secure according to our simplified security definitions (given in Section 4) is secure in the more complex security model given in Appendix E, if all messages sent during the HonIssue, HonAcc and HonVer protocols are encrypted as described in Appendix F.1. Proof sketches are given in Sections F.2–F.6.

F.1 Extending security by encryption

Any BBA+ base protocol can be transformed to an encrypted scheme as follows:

- During issuer key generation, additionally generate a group description $\mathsf{gpSetupGrp}(1^n)$ and a key-pair $(\mathsf{pk}_{\mathsf{Enc}}, \mathsf{sk}_{\mathsf{Enc}})$ of a secure PKE scheme and include $\mathsf{pk}_{\mathsf{Enc}}$ and $\mathsf{sk}_{\mathsf{Enc}}$ in the issuer's public and secret key, respectively.
- In the protocols Issue, Accum and Vfy, before the first message is sent, let the user choose a random key k for an IND-CCA2-secure symmetric encryption scheme, encrypt it under $\mathsf{pk}_{\mathsf{Enc}}$, and send the resulting ciphertext to the issuer, accumulator or verifier, respectively, who will decrypt it to obtain k. Then during the protocol execution, have all outgoing messages encrypted under k. When receiving a message, decrypt the message using k. If the decryption algorithm outputs $\perp$, abort the protocol. When the base protocol finishes, output whatever the base protocol did output.

It should be intuitively clear that this transformation does not reduce the security of a BBA+ scheme, however, this is not-so-obvious for the *trapdoor linkability* property, without which security is not even defined. We state the following propositions with regard to the encrypted scheme:

Proposition F.1 (Correctness, Informal). *If the encryption schemes used by the transformation given above are correct and the base BBA+ scheme is correct, then the encrypted BBA+ scheme is correct.*

Proposition F.2 (Completeness, Informal). *If the base BBA+ scheme is complete, then the encrypted scheme is complete.*

Proof sketch: Let view be a view of one of the protocols Accum of the encrypted scheme that ends with the issuer accepting the protocol and outputting the hidden user id hid. Then, firstly, all decryptions must have succeeded (because the issuer had aborted otherwise) and, secondly, the base protocol must have accepted with the decrypted messages. Then the decrypted messages of the encrypted protocol are a view view′ that lets $\mathcal{AC}$ (of the base protocol) output hid. Encrypting all messages in view′ with an arbitrary honestly generated symmetric key $k \leftarrow \mathsf{Gen}(1^n)$ and prefixing an encryption of k under the issuer's public encryption key $\mathsf{pk}_{\mathsf{Enc}}$ yields a view of the encrypted scheme that lets $\mathcal{AC}$ (of the encrypted protocol) output hid. The argument for views of the Vfy protocol is analogous.

Proposition F.3 (Extractability, Informal). *If the base scheme is extractable, then the encrypted scheme is extractable.*

Proof sketch: The ExtractUID algorithm of the base scheme can be used without modification.

F.2 Owner-Binding with respect to Issue

We want to show that a BBA+ scheme BBAP secure according to Theorem 4.3 is also secure according to Theorem E.1. To see

Experiment $\mathsf{Exp}^{\text{ob-acc-mod}}_{\mathsf{BBAP},\mathcal{A}}(n)$

$(\mathsf{CRS}, \mathsf{td}) \leftarrow \mathsf{Setup}(1^n)$

$(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}) \leftarrow \mathsf{IGen}(\mathsf{CRS})$

run $\mathcal{A}^{\mathsf{HonUser},\mathsf{HonIssue},\mathsf{HonAcc},\mathsf{HonVer},\mathsf{MalIssue},\mathsf{MalAcc},\mathsf{MalVer}}(\mathsf{CRS}, \mathsf{pk}_{\mathcal{I}})$

The experiment returns 1 iff $\mathcal{A}$ did a successful call to MalAcc or MalVer such that ExtractUID applied hid that is part of the resulting outputs a public-key $\mathsf{pk}^*_{\mathcal{U}}$ for which there has been no successful execution of MalIssue or HonIssue up to this MalAcc (or MalVer) call.

Figure 24: Intermediary owner-binding experiment for BBAP **with respect to** Accum**.**

this we must show that an attacker $\mathcal{A}$ in the security experiment $\mathsf{Exp}^{\text{ob-issue-full}}_{\mathsf{BBAP},\mathcal{A}}$ from Fig. 19 can be transformed into an attacker $\mathcal{B}$ in the simplified model (see Fig. 1). This is achieved as follows.

Let CRS, $\mathsf{pk}_{\mathcal{I}}$, $\mathsf{pk}_{\mathcal{U}}$ be $\mathcal{B}$'s input. $\mathcal{B}$ guesses the index of the HonUser-query returning the $\mathsf{pk}_{\mathcal{U}}$ for which $\mathcal{A}$ will eventually successfully run MalIssue. $\mathcal{B}$ then internally simulates $\mathcal{A}$ with input CRS, $\mathsf{pk}_{\mathcal{I}}$. When answering $\mathcal{A}$'s HonUser queries, return the challenge public key $\mathsf{pk}_{\mathcal{U}}$ on that query, and generate all other user keys honestly. If the guess was correct, HonIssue, HonAcc and HonVer queries for all other users can be simulated by running honest executions of the respective protocol via the MalIssue, MalAcc and MalVer oracles. User corruptions (for all users except the user with the key $\mathsf{pk}_{\mathcal{U}}$) can be dealt with by revealing the current token as well as the respective $\mathsf{sk}_{\mathcal{U}}$ to $\mathcal{A}$. Furthermore, if $\mathcal{B}$ has guessed the HonUser query correctly, then HonIssue, HonAcc, HonVer and Corrupt queries for $\mathsf{pk}_{\mathcal{U}}$ are not allowed to $\mathcal{A}$.

Thus, $\mathcal{B}$ wins the game $\mathsf{Exp}^{\text{ob-issue}}_{\mathsf{BBAP},\mathcal{A}}$ from Fig. 1 if $\mathcal{A}$ wins the game $\mathsf{Exp}^{\text{ob-issue-full}}_{\mathsf{BBAP},\mathcal{A}}$ and $\mathcal{B}$ has guessed the right HonUser call.

F.3 Owner-Binding with respect to Accum and Vfy

We argue that if a "base" BBA+ scheme BBAP is secure according to Theorem 4.4, then the encrypted version of that protocol (see Appendix F.1) is secure according to Theorem E.2.

We introduce two additional "intermediate" security experiments, see Fig. 24. In these security experiments $\mathcal{A}$ does not have access to the Corrupt oracle. Furthermore, in the second intermediate experiment, the HonIssue, HonAcc and HonVer oracles do not return the encrypted messages sent during an honest protocol run, but return encryptions of uniformly random strings of the respective lengths. (We denote these modified oracles by HonIssue′, HonAcc′, HonVer′, respectively.)

Our high-level proof strategy is as follows: In a first step, we will show that an attacker with non-negligible probability of winning $\mathsf{Exp}^{\text{ob-acc-full}}_{\mathsf{BBAP},\mathcal{A}}$ against the encrypted scheme can be transformed into two attackers $\mathcal{B}$, C, at least one of which will win the first game $\mathsf{Exp}^{\text{ob-acc-mod}}_{\mathsf{BBAP},\mathcal{A}}$ given in Fig. 24 with non-negligible probability. The second step is to show that the security with regard to the first intermediate game implies security wrt. the second intermediate game by a hybrid argument using the security of the encryption schemes. The third step is to show that an attacker that wins the second intermediate game against the *encrypted* scheme can be used to break the security of the *base* scheme wrt. Theorem 4.4.

We start with the first step. Observe that in experiment $\mathsf{Exp}^{\text{ob-acc-full}}_{\mathsf{BBAP},\mathcal{A}}$, the attacker $\mathcal{A}$ wins if he makes a successful (as seen from $\mathcal{AC}$'s or $\mathcal{V}$'s perspective) call to MalAcc or MalVer on a public key $\mathsf{pk}_{\mathcal{U}}$ which fulfills one of two conditions:

(1) $\mathcal{A}$ has made neither a MalIssue- nor an HonIssue-query for $\mathsf{pk}_{\mathcal{U}}$
(2) $\mathcal{A}$ has made an HonIssue-query for $\mathsf{pk}_{\mathcal{U}}$, but no Corrupt-query.

Let E_1 be the event that $\mathcal{A}$ wins the game with condition 1 fulfilled, and E_2 be the event that $\mathcal{A}$ wins with condition 2. Clearly, $\Pr[E_1] + \Pr[E_2] = \Pr[\mathsf{Exp}^{\text{ob-acc-full}}_{\mathsf{BBAP},\mathcal{A}} = 1] =: \varepsilon$, and thus $\Pr[E_1] \geq \varepsilon/2$ or $\Pr[E_2] \geq \varepsilon/2$.

We now construct an adversary $\mathcal{B}$ that will break $\mathsf{Exp}^{\text{ob-acc-mod}}_{\mathsf{BBAP},\mathcal{A}}$ if $\Pr[E_1]$ is non-negligible. $\mathcal{B}$ internally simulates the $\mathsf{Exp}^{\text{ob-acc-full}}_{\mathsf{BBAP},\mathcal{A}}$ game. The HonUser, HonIssue, HonAcc, and HonVer oracles are implemented by generating key-pairs with UGen, and honestly running the protocols via $\mathcal{B}$'s MalIssue, MalAcc and MalVer oracles, respectively. $\mathcal{A}$'s MalIssue, MalAcc, and MalVer oracles are forwarded to the $\mathsf{Exp}^{\text{ob-acc-mod}}_{\mathsf{BBAP},\mathcal{B}}$ experiment by $\mathcal{B}$. The Corrupt oracle is implemented by revealing the data honestly generated while simulating the HonUser, HonIssue, HonAcc and HonVer oracles. When $\mathcal{A}$ terminates, winning the $\mathsf{Exp}^{\text{ob-acc-full}}_{\mathsf{BBAP},\mathcal{A}}$ game with event E_1, then $\mathcal{B}$ will also win the $\mathsf{Exp}^{\text{ob-acc-mod}}_{\mathsf{BBAP},\mathcal{B}}$ game, since $\mathcal{A}$'s MalAcc/MalVer call will have been forwarded by $\mathcal{B}$ to the $\mathsf{Exp}^{\text{ob-acc-mod}}_{\mathsf{BBAP},\mathcal{B}}$ game. The key $\mathsf{pk}^*_{\mathcal{U}}$ extracted in this game will not have been used by $\mathcal{B}$ in a call to its MalIssue oracle, since $\mathcal{B}$ only uses its MalIssue oracle for keys that $\mathcal{A}$ submits to its HonIssue or MalIssue oracles, and (by the definition of event E_1) $\mathcal{A}$ has never used the public key $\mathsf{pk}^*_{\mathcal{U}}$ with these oracles.

We construct a similar attacker C for the case where $\Pr[E_2]$ is non-negligible. C guesses a random index i for a HonUser query by $\mathcal{A}$. When $\mathcal{A}$ makes its i-th call to the HonUser oracle, then C calls its own HonUser oracle, stores the resulting public key $\mathsf{pk}_{\mathcal{U}}$ and returns it to $\mathcal{A}$. On all other queries to the HonUser oracle, C proceeds as $\mathcal{B}$ did: it generates the respective keys itself. For the stored key $\mathsf{pk}_{\mathcal{U}}$, the HonIssue, HonAcc, HonVer oracles are simulated using C's respective oracle. For all other user public keys, C honestly runs the respective protocol over its MalIssue, MalAcc and MalVer oracle. If $\mathcal{A}$ calls Corrupt with the key $\mathsf{pk}_{\mathcal{U}}$, C aborts. When Corrupt is called for another user public key, C reveals the most recent token as well as the secret key to $\mathcal{A}$. C forwards MalIssue, MalAcc and MalVer calls by $\mathcal{A}$ to its own respective oracle.

Note that if $\Pr[E_2] \geq \varepsilon/2$, then $\mathcal{A}$ must make a MalAcc call where ExtractUID yields a public key $\mathsf{pk}^*_{\mathcal{U}}$ returned by the HonUser oracle, which has never been used with the Corrupt oracle. If $\mathsf{pk}_{\mathcal{U}} = \mathsf{pk}^*_{\mathcal{U}}$ (which happens with non-negligible probability), then C does not abort, and wins the $\mathsf{Exp}^{\text{ob-acc-mod}}_{\mathsf{BBAP},C}$ game. This concludes the first step of our proof strategy.

We now continue with the second step: We need to show that an adversary $\mathcal{A}$ (which may be one of the attackers described above, but might be completely independent of these as well) against an encrypted scheme in the first $\mathsf{Exp}^{\text{ob-acc-mod}}_{\mathsf{BBAP},\mathcal{A}}$ game can be used to build an attacker $\mathcal{B}$ against the same scheme in the second $\mathsf{Exp}^{\text{ob-acc-mod}}_{\mathsf{BBAP},\mathcal{B}}$

game, where the HonIssue, HonAcc and HonVer oracles return encryptions of random strings instead of encrypted protocol messages.

We show that an adversary $\mathcal{A}$'s success probability can differ at most negligibly between these games by a hybrid argument, reducing to the security of the two encryption schemes: symmetric encryption scheme used for sending base protocol messages and the asymmetric encryption scheme used for transmitting the symmetric key.

Assume that an attacker $\mathcal{A}$'s success probabilities in the two $\mathsf{Exp}^{\text{ob-acc-mod}}_{\mathsf{BBAP},\mathcal{A}}$ games differ by a non-negligible function. Let q be an upper bound on the total number of HonIssue, HonAcc and HonVer calls by $\mathcal{A}$. We define $2q+1$ hybrid games as follows: In the hybrids $0 \le i \le q$, calls to the HonIssue, HonAcc and HonVer return transcripts of honest protocol executions, with the exception that during the first i calls, the (asymmetric) encryption of the symmetric key is replaced by an encryption of *another* independently selected random symmetric key. The symmetric ciphertexts are kept without modifications. (Thus, while the base protocol messages are encrypted under some key k_0, the initial message will contain an encryption of a key k_1, which will differ from k_0 with high probability.) Thus, in hybrid q, while the base protocol messages are encrypted under up to q random symmetric keys, all encryptions of the symmetric keys under the issuer's public key have been replaced by encryptions of independent random keys. This changes in the following hybrids: In the hybrids $q \le q+i \le 2q$, additionally, the ciphertexts of the base protocol messages of the first i calls to HonIssue, HonAcc and HonVer are replaced by encryptions of random strings. Thus, in hybrid $2q$, all ciphertexts returned by the HonIssue, HonAcc and HonVer oracles have been replaced by encryptions of random data.

Clearly, the 0-th hybrid is equal to the first $\mathsf{Exp}^{\text{ob-acc-mod}}_{\mathsf{BBAP},\mathcal{A}}$ game, whereas hybrid $2q$ is equal to the second $\mathsf{Exp}^{\text{ob-acc-mod}}_{\mathsf{BBAP},\mathcal{A}}$ game.

Now, if there is a non-negligible difference between the two $\mathsf{Exp}^{\text{ob-acc-mod}}_{\mathsf{BBAP},\mathcal{A}}$ games, there must be an i such that there is a non-negligible difference between hybrid i and hybrid $i-1$.

We show that this non-negligible difference can be exploited to break the security of (one of) the underlying encryption schemes. More precisely, we construct two attackers $\mathcal{D}_1$, $\mathcal{D}_2$ against the IND-CCA2 security of the asymmetric and symmetric encryption schemes, respectively. We then show that at least one of these two attacker must have a non-negligible advantage in the IND-CCA2 game if the difference of hybrids i and $i-1$ is non-negligible.

Assume that there is a non-negligible difference between hybrids i and $i-1$ for $i \le q$. We construct a distinguisher $\mathcal{D}_1$ in the IND-CCA2 game against the public key encryption scheme employed. $\mathcal{D}_1$ receives a public key of an encryption scheme $\mathsf{pk}_{\mathsf{Enc}}$ as input, and generates a CRS, and a BBAP key pair $(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}})$ as in experiment $\mathsf{Exp}^{\text{ob-acc-mod}}_{\mathsf{BBAP},\mathcal{A}}$. It includes its challenge public key $\mathsf{pk}_{\mathsf{Enc}}$ in the issuer's public key of the encrypted BBA+ protocol and passes the resulting key to $\mathcal{A}$, along with the CRS. It then internally simulates $\mathcal{A}$.

The HonUser oracle is implemented by $\mathcal{D}_1$ choosing a random key pair $(\mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}) \leftarrow \mathsf{UGen}(\mathsf{CRS})$ and returning $\mathsf{pk}_{\mathcal{U}}$. For the first $i-1$ calls by $\mathcal{A}$ to one of the HonIssue, HonAcc or HonVer oracles, $\mathcal{D}_1$ simulates an honest execution of the respective (base) protocol (recall that $\mathcal{D}_1$ knows $\mathsf{sk}_{\mathcal{U}}$, $\mathsf{sk}_{\mathcal{I}}$ and the most recent token), encrypts all messages under a randomly selected symmetric key k_0, encrypts another randomly selected symmetric key k_1 under $\mathsf{pk}_{\mathsf{Enc}}$, and sends all ciphertexts to $\mathcal{A}$. For the i-th call, $\mathcal{D}_1$ runs the respective protocol as well, encrypts all base protocol messages under a randomly chosen key k_0, and outputs the chosen symmetric key k_0 as m_0, and another randomly selected symmetric key k_1 as m_1 to the IND-CCA2 game. It then returns its challenge ciphertext c^* along with the encryptions of all base protocol messages to $\mathcal{A}$. For all later calls, $\mathcal{D}_1$ simulates the encrypted protocol without modifications.

Whenever $\mathcal{A}$ makes a call to MalIssue, MalAcc or MalVer, $\mathcal{D}_1$ uses its decryption oracle to decrypt the initial message sent by $\mathcal{A}$, thereby obtaining the symmetric key k used for the base protocol messages. (If the initial message matches $\mathcal{D}_1$'s challenge ciphertext c^*, $\mathcal{D}_1$ continues as if the decryption oracle had returned $k := k_0$.) It then uses that key to decrypt the messages sent via the oracles, uses the (self-generated) $\mathsf{sk}_{\mathcal{I}}$ to simulate the issuer side of the base protocol, and encrypts its answers with k and sends them back to $\mathcal{A}$.

$\mathcal{D}_1$ outputs whatever the respective hybrid game would output. By construction, $\mathcal{D}_1$ simulates either hybrid $i-1$ or hybrid i for $\mathcal{A}$. Since $\mathcal{A}$'s success probabilities in winning these two hybrids differ non-negligibly (by assumption), $\mathcal{D}_1$'s success probability in winning the IND-CCA2 game is non-negligible.

Now assume that $\mathcal{A}$'s success probabilities in two hybrids $q+i$ and $q+i-1$ differ (for some $i \in \{1,\dots,q\}$) by a non-negligible amount. We construct an attacker $\mathcal{D}_2$ in the multi-message IND-CCA2 game (where the attacker may output two *vectors* of messages) against the symmetric encryption scheme employed. $\mathcal{D}_2$ generates $(\mathsf{CRS},\mathsf{td}) \leftarrow \mathsf{Setup}(1^n)$, $(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}) \leftarrow \mathsf{IGen}(\mathsf{CRS})$, $\mathsf{gp} \leftarrow \mathsf{SetupGrp}(1^n)$ and $(\mathsf{pk}_{\mathsf{Enc}}, \mathsf{sk}_{\mathsf{Enc}}) \leftarrow \mathsf{Gen}(\mathsf{gp})$. Furthermore, it chooses a "substitute" symmetric key k_s. It then internally simulates $\mathcal{A}$ with input $(\mathsf{CRS}, (\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathsf{Enc}}))$.

$\mathcal{A}$'s HonUser oracle is implemented as before: by randomly selecting a key-pair $(\mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}) \leftarrow \mathsf{UGen}(\mathsf{CRS})$. The HonIssue, HonAcc and HonVer oracles are implemented as follows. Upon the first i invocations, $\mathcal{D}_2$ simulates the execution of the respective base protocol, selects two random symmetric keys k_0, k_1, encrypts k_0 under $\mathsf{pk}_{\mathsf{Enc}}$ and encrypts random strings of the same lengths as the protocol messages under k_1, and returns the ciphertexts to $\mathcal{A}$. Upon the i-th invocation, $\mathcal{D}_2$ simulates the respective protocol of the base scheme, and stores all resulting messages. It then outputs the vector of these messages as one part of its challenge, and a vector of random strings of the same lengths as its second part of the challenge. When it receives the vector of challenge ciphertexts c^*, it returns these encrypted messages, along with an encryption of the "substitute" key k_s under $\mathsf{pk}_{\mathsf{Enc}}$. For all later invocations, $\mathcal{D}_2$ simulates the base protocol, encrypts the messages under a random symmetric key k_0, encrypts another randomly selected symmetric key k_1 under $\mathsf{pk}_{\mathsf{Enc}}$, and returns the ciphertexts to $\mathcal{A}$.

When $\mathcal{A}$ makes a query to the MalIssue, MalAcc or MalVer oracles, $\mathcal{D}_2$ decrypts the symmetric key sent by $\mathcal{A}$ using $\mathsf{sk}_{\mathsf{Enc}}$. If it does not match the substitute key k_s, then $\mathcal{D}_2$ computes the next protocol message sent by $\mathcal{I}$, $\mathcal{AC}$ or $\mathcal{V}$, respectively, using $\mathsf{sk}_{\mathcal{I}}$. If the symmetric key matches k_s, however, $\mathcal{D}_2$ checks if the respective

message sent by $\mathcal{A}$ is one of its challenge ciphertexts contained in c^*. If it is, $\mathcal{D}_2$ continues as if it had decrypted the ciphertext to the message of the base protocol that $\mathcal{D}_2$ stored while simulating the i-th invocation of an HonIssue, HonAcc or HonVer oracle. If the ciphertext does not match any of $\mathcal{D}_2$'s challenge ciphertexts, $\mathcal{D}_2$ decrypts it using its decryption oracle. Once $\mathcal{D}_2$ has "decrypted" the ciphertext in one way or another, it simulates the respective protocol of the base scheme using $\mathsf{sk}_{\mathcal{I}}$. The resulting answer is encrypted as follows: If the symmetric key used matches k_s, $\mathcal{D}_2$ uses its encryption oracle. If the symmetric key differs from k_s, then $\mathcal{D}_2$ uses the regular symmetric encryption algorithm with the key sent by $\mathcal{A}$.

By construction, $\mathcal{D}_2$ simulates either hybrid $q+i-1$ or hybrid $q+i$ for $\mathcal{A}$. Since $\mathcal{A}$ success probabilities in winning these two hybrids differ non-negligibly (by assumption), $\mathcal{D}_2$'s success probability in winning the multi-message IND-CCA2 game is non-negligible.

This concludes the second step of our proof: Using the hybrid argument, we have shown that $\mathcal{A}$'s success probability in the hybrids 0 and $2q$ (which equal the two variants of the $\mathsf{Exp}^{\text{ob-acc-mod}}_{\text{BBAP},\mathcal{A}}$ game), differ on non-negligibly if both encryption schemes are IND-CCA2 secure.

Finally, we move to the third step of our proof sketch. We construct an adversary $\mathcal{B}$ in the $\mathsf{Exp}^{\text{ob-acc}}_{\text{BBAP},\mathcal{B}}$ game against the *base* scheme that simulates $\mathcal{A}$ in the second intermediate experiment with the *encrypted* scheme. $\mathcal{B}$ internally generates an encryption key pair $(\mathsf{pk}_{\mathsf{Enc}}, \mathsf{sk}_{\mathsf{Enc}})$, augments its input $\mathsf{pk}_{\mathcal{I}}$ with $\mathsf{pk}_{\mathsf{Enc}}$ and hands the resulting $\mathsf{pk}'_{\mathcal{I}}$ to $\mathcal{A}$. When $\mathcal{A}$ queries the HonUser oracle, $\mathcal{B}$ generates a random key pair $(\mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}})$ and returns $\mathsf{pk}_{\mathcal{U}}$. Upon HonIssue, HonAcc, or HonVer queries, $\mathcal{B}$ returns transcripts that contain encryptions of random symmetric keys k under $\mathsf{pk}_{\mathsf{Enc}}$ and encryptions of random strings under independently selected symmetric keys k'. $\mathcal{A}$'s MalIssue, MalAcc and MalVer queries are decrypted with $\mathsf{sk}_{\mathsf{Enc}}$, and the decrypted messages are then passed on to the $\mathsf{Exp}^{\text{ob-acc}}_{\text{BBAP},\mathcal{B}}$ game by $\mathcal{B}$. The answers of $\mathcal{B}$'s oracles are encrypted by $\mathcal{B}$ and then returned to $\mathcal{A}$. With this setup, $\mathcal{B}$ perfectly simulates the second intermediate game for $\mathcal{A}$. Furthermore, $\mathcal{B}$ wins the $\mathsf{Exp}^{\text{ob-acc}}_{\text{BBAP},\mathcal{B}}$ game if $\mathcal{A}$ wins the second intermediate game.

This concludes the proof for the third step of our proof sketch, and thus our proof sketch as a whole.

F.4 Balance Binding

Let $\mathcal{A}$ be an adversary in the $\mathsf{Exp}^{\text{bb-full}}_{\text{BBAP},\mathcal{A}}$ experiment. We construct an adversary $\mathcal{B}$, winning the $\mathsf{Exp}^{\text{bb}}_{\text{BBAP},\mathcal{B}}$ experiment with the same success probability as $\mathcal{A}$: $\mathcal{B}$ internally simulates the $\mathsf{Exp}^{\text{bb-full}}_{\text{BBAP},\mathcal{A}}$ experiment for $\mathcal{A}$. When $\mathcal{A}$ makes a call to one of the MalIssue, MalAcc or MalVer oracles, $\mathcal{B}$ just passes the messages on without modification. When $\mathcal{A}$ uses the HonUser oracle, $\mathcal{B}$ generates a key pair with $(\mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}) \leftarrow \mathsf{UGen}$, stores $\mathsf{sk}_{\mathcal{U}}$ for later use and returns $\mathsf{pk}_{\mathcal{U}}$. On a HonIssue, HonAcc or HonVer query by $\mathcal{A}$, $\mathcal{B}$ executes the respective protocol via its own MalIssue, MalAcc or MalVer oracles honestly, stores the results (i.e., the updated token) and then returns the transcript of all exchanged messages to $\mathcal{A}$. If $\mathcal{A}$ makes a query to the Corrupt oracle, $\mathcal{B}$ returns the respective secret key along with the most recent token for that user.

If $\mathcal{A}$ wins the game, then (by definition) $\mathcal{A}$ must have made a MalVer call which returned a serial number s^* which was never returned before, and in which $\mathcal{A}$ claimed a wrong balance. Let S be the set of serial numbers s returned by $\mathcal{B}$'s MalAcc and MalVer oracles, S_{Hon} be the set of serial numbers s returned by $\mathcal{B}$ to $\mathcal{A}$ on calls to the HonAcc or HonVer oracles, and s_{Mal} be the set of all s returned by $\mathcal{B}$ on $\mathcal{A}$'s calls to the MalAcc and MalVer oracles. Then (by construction of $\mathcal{B}$) we have $S = S_{\text{Hon}} \cup S_{\text{Mal}}$. Thus, if s^* was not returned to $\mathcal{A}$ before (which means $s^* \notin S_{\text{Hon}} \cup S_{\text{Mal}}$), then, clearly, $s \notin S$, so s^* was never returned to $\mathcal{B}$ as well. Furthermore, since the ExtractUID algorithms are guaranteed to return the input key $\mathsf{pk}_{\mathcal{U}}$ on honest protocol executions, $\mathcal{B}$'s simulation of the HonAcc and HonVer oracles does not falsify the set $V_{\mathsf{pk}_{\mathcal{U}}}$ from Fig. 3. Thus, $\mathcal{B}$ has the same success probability in winning the $\mathsf{Exp}^{\text{bb}}_{\text{BBAP},\mathcal{B}}$ game as $\mathcal{A}$ in the $\mathsf{Exp}^{\text{bb-full}}_{\text{BBAP},\mathcal{A}}$ game.

F.5 Double Spending Detection

Assuming an adversary $\mathcal{A}$ with non-negligible success probability in the $\mathsf{Exp}^{\text{dsd-full}}_{\text{BBAP},\mathcal{A}}$ game, we construct an adversary $\mathcal{B}$ in the $\mathsf{Exp}^{\text{dsd}}_{\text{BBAP},\mathcal{B}}$ game that has at least the same success probability. $\mathcal{B}$ internally simulates the $\mathsf{Exp}^{\text{dsd}}_{\text{BBAP},\mathcal{A}}$ game for $\mathcal{A}$. $\mathcal{A}$'s oracle queries are answered exactly as in Appendix F.4. Now, assume that $\mathcal{A}$ has done two MalAcc/MalVer queries that fulfill the conditions stated in Fig. 22. Since the MalAcc/MalVer queries of $\mathcal{B}$ are a superset of $\mathcal{A}$'s MalAcc/MalVer queries, and the conditions are identical in both experiments, $\mathcal{B}$ must also win the $\mathsf{Exp}^{\text{dsd}}_{\text{BBAP},\mathcal{B}}$ game.

F.6 Framing Protection

An attacker $\mathcal{A}$ in the $\mathsf{Exp}^{\text{facp}}_{\text{BBAP},\mathcal{A}}$ can be transformed into an attacker $\mathcal{B}$ game, in a way that $\mathcal{B}$'s success probability equals $\mathcal{A}$'s: $\mathcal{B}$ receives a CRS and a user public key $\mathsf{pk}^*_{\mathcal{U}}$ as input, and forwards CRS to $\mathcal{A}$, which will output an issuer public key $\mathsf{pk}_{\mathcal{I}}$ that $\mathcal{B}$ stores for later.

$\mathcal{B}$ guesses a random index i for an HonUser query by $\mathcal{A}$. On $\mathcal{A}$'s i-th HonUser query $\mathcal{B}$ returns $\mathsf{pk}^*_{\mathcal{U}}$. On all other HonUser queries $\mathcal{B}$ calls its own HonUser oracle. $\mathcal{A}$'s HonIssue, HonAcc, and HonVer calls are passed through to $\mathcal{B}$'s oracles. $\mathcal{A}$'s Corrupt queries are passed through as well, except if $\mathcal{A}$ calls Corrupt on input of $\mathsf{pk}^*_{\mathcal{U}}$: In this case, $\mathcal{B}$ will abort.

If $\mathcal{B}$ has correctly guessed the index i of $\mathcal{A}$'s HonUser query returning the user public key that $\mathcal{A}$ outputs in the end, and $\mathcal{A}$ wins the $\mathsf{Exp}^{\text{facp-full}}_{\text{BBAP},\mathcal{A}}$ game then $\mathcal{A}$ must not have made a Corrupt call on input of $\mathsf{pk}^*_{\mathcal{U}}$, and thus $\mathcal{B}$ will not abort. Furthermore, in this case, $\mathcal{B}$ can output the previously stored $\mathsf{pk}_{\mathcal{I}}$ and $\mathcal{A}$'s proof Π to win the $\mathsf{Exp}^{\text{facp}}_{\text{BBAP},\mathcal{B}}$ game.

G RANGE PROOFS

There are a variety of applications where it might be desirable not to publish the current balance $w \in \mathbb{Z}_p$ during the verification and redemption phase. To overcome this issue our protocol could be extended by a range proof system such as [14] or [11]. Although there has been great progress to increase the efficiency of those proof systems, we deliberately did not include such a proof in our basic scheme, as we are convinced that those range proofs are still

too inefficient for practical deployment on low-end hardware like mobile devices. Nonetheless, we implemented such a range proof and will give some concrete numbers in this appendix.

G.1 High Level Overview

We roughly explain the idea of the range proof in [11]. Firstly, we recap the trivial approach to prove that a balance w is at least the redeemed value v, i. e. $w \in \mathcal{S} := \{v, \ldots, N_{\max}\}$ whereby $N_{\max}$ denotes the biggest integer that can be represented. The verifier generates a signature on every element of $\mathcal{S}$ and the prover proves in ZK that it knows a signature on its balance w. Obviously, this approach is prohibitive if $|\mathcal{S}| \propto \left|\mathbb{Z}_p\right|$ for $p \in \Theta(\text{poly}(1^n))$ as this yields an exponentially large set $\mathcal{S}$ and therefore number of signatures.

In [11] Camenisch et al. exploit a q-nary representation of the secret w with at most $\eta_{\max}$ digits to overcome this problem. Hereby, $q, \eta_{\max} \in \mathbb{N}$ are design parameters that are chosen during system setup such that $q^{\eta_{\max}} \leq \left|\mathbb{Z}_p\right|$. For a fixed q the maximal admissible number of digits $\eta_{\max}$ that provides a representation for most secrets $s \in \mathbb{Z}_p$ is $\eta_{\max} \leq \lfloor \log_q p \rfloor \in O(n)$. Now, the verifier must only generate q signatures on each element in $\{0, \ldots, q-1\}$, the prover generates an q-nary representation of the secret $s = \sum_{j=0}^{\eta-1<\eta_{\max}-1} s_j q^j$ and then proves for each digit s_j ($j \in \{0, \ldots, \eta - 1\}$) that the digit is contained in the set $\{0, \ldots, q-1\}$, i. e. that it knows a signature for it.

Please note that this range proof is only applicable to a $\mathbb{Z}_p$-subset whose size is a power of q. E. g. depending on the tangible choice of q and $\eta_{\max}$ there are $\left|\mathbb{Z}_p\right| - q^{\eta_{\max}}$ elements of the prime-order group that elude a q-nary representation. In practical terms this means that only a subset of $\mathbb{Z}_p$ can be used and "illegal" balances have to be avoided by the protocol. (See discussion in the next subsection.) Although cleartext balances are restricted to a much smaller domain, this does not weaken security as randomness and therefore ciphertexts/commitments are still varying over the whole range.

Moreover, the basic range proof only allows to show that a secret s can be represented with $\eta \leq \eta_{\max}$ digits, i. e. that $s \in \{0, \ldots, q^\eta - 1\}$ holds. But we need to prove a statement $w \in \{v, \ldots, N_{\max}\}$ about a secret w and usually neither interval limit is located at a q-power. The idea is to suitable shift w and proof a q-nary representation of the shifted value. In order to overcome the issue with non-aligned interval limits the prover conducts two range proofs and shows that the shifted value lies in two different intervals whose limits are aligned and whose intersection is the claimed range. For details on the actual calculation see the section after the next.

G.2 Design Choices and Notation

Efficient range proofs heavily depend on the representation of the elements with individual digits and then proofing statements about the digits in zero-knowledge. This representation leaves space for some design decisions. The design parameters q and $\eta_{\max}$ are a trade-off between the number of signatures and the size of the NIZK statement. Please note, that the signatures can be pre-computed and re-used for all NIZKs. Hence, a greater q and a smaller $\eta_{\max}$ is usually beneficial.

In the BBA+ scheme, the accumulation value v and the balance w are elements of the prime-order group $\mathbb{Z}_p$. We fix the representation

$$\mathbb{Z}_p = \{0, \ldots, p-1\} \subset \mathbb{N}, \tag{34}$$

i. e. we interpret elements of $\mathbb{Z}_p$ as *positive* numbers with the usual $\leq$-order inherited from $\mathbb{N}$. This also implies that we keep the Accum and Vfy algorithms separated. We accumulate points by addition of positive numbers and redeem points by *subtraction* of positive numbers. We do not perform a range check for accumulation, but only check if the redeemed points v are smaller than the current balance w, e. g. if the user does not overdraw its account. If an accumulation results in an overflow and thus in a smaller balance, we regard that as the user's problem. We consider a plus of points as beneficial to the user and the user who knows its previous balance and the accumulation value should have obviate this problem before engaging in the Accum protocol.

We choose

$$2 \leq q \leq p - 1 \tag{35}$$

as the base of the q-nary representation. The maximum number of digits to represent values in $\mathbb{Z}_p$ is set to

$$\eta_{\max} \leq \lfloor \log_q p \rfloor. \tag{36}$$

Hence, the biggest integer that can be represented equals

$$N_{\max} := q^{\eta_{\max}} - 1. \tag{37}$$

Note that this means that the elements $\{N_{\max} + 1, \ldots, p-1\} \subset \mathbb{Z}_p$ cannot be represented by the positional number system and thus are "illegal" elements. We again argue that it is the user's responsibility to ensure that an execution of Accum does not result into an illegal balance. Similar to the issue with the wrap-around the user is harmed, because it will possess a broke token afterwards and cannot successfully prove statements about its balance during Vfy. The user should avoid this situation of its own accord.

Moreover, for the ease of later notation we denote the first $\eta_{\max}$ q-powers of g_1 by

$$Q_j := g_1^{(q^j)} \qquad \text{for } j = 0, \ldots, \eta_{\max} - 1. \tag{38}$$

These constants are an F_{gp}-mapping of all relevant magnitudes of the positional digit system. Please note that whenever any constant Q_j appears in a formula the party can compute $g_1^{(q^j)}$ by itself. However, in the security proof the simulator of the NIKZ needs to know the discrete log, i. e. the constants are part of its hint *hint*. In practice, it might be even beneficial to pre-compute Q_j and include them into the CRS such that they can be looked up quickly when needed.

The system constants q, $\eta_{\max}$, $N_{\max}$ and (optionally) Q_0, …, $Q_{\eta_{\max}-1}$ are included in the global CRS.

Alternative design choice. Alternatively, we could choose to use the representation $\mathbb{Z}_p = \left\{-\frac{p-1}{2}, \ldots, 0, \ldots, \frac{p-1}{2}\right\}$ and allow *negative* numbers. Then the Accum and Vfy protocols could be unified using a two-sided range proof and regarding redemption as the accumulation of negative points. In this case the maximum number of digits is $\eta_{\max} \leq \left\lfloor \log_q \left(\frac{p-1}{2}\right) \right\rfloor$ plus an additional sign bit.

G.3 Concrete Range Proof

As already stated we want to proof $w \in \{v, \ldots, N_{\max}\}$ and need to shift the values such that the interval limits fit into the proof scheme. In preparation let $\eta \in [\eta_{\max}]$ be defined as

$$\eta := \lfloor \log_q(N_{\max} - v) \rfloor + 1 \quad \Leftrightarrow \quad q^{\eta-1} \leq N_{\max} - v < q^{\eta} \tag{39}$$

and

$$N := q^{\eta} - 1, \tag{40}$$

i. e. $N + 1$ is the smallest q-power greater than $N_{\max} - v$. It follows

$$\begin{aligned}
& w \in \{v, \ldots, N_{\max}\} \\
\Leftrightarrow\ & N_{\max} - w \in \{0, \ldots, N_{\max} - v\} \\
\Leftrightarrow\ & N_{\max} - w \in \{0, \ldots, N\} \cap \{N_{\max} - v - N, \ldots, N_{\max} - v\} \\
\Leftrightarrow\ & \begin{cases} N_{\max} - w \in \{0, \ldots, N\} \wedge \\ N + v - w \in \{0, \ldots, N\} \end{cases} \\
\Leftrightarrow\ & \begin{cases} \exists\, w'_0, \ldots, w'_{\eta-1} \in \{0, \ldots, q-1\} : N_{\max} - w = \sum_{j=0}^{\eta-1} w'_j q^j \\ \exists\, w''_0, \ldots, w''_{\eta-1} \in \{0, \ldots, q-1\} : N + v - w = \sum_{j=0}^{\eta-1} w''_j q^j \end{cases}
\end{aligned} \tag{41}$$

In the BBA+ scheme the user proves that it has created a randomized version of a signed commitment to its balance and that both commitments open to the same value. Openings of commitments are elements from the implicit message space $\mathcal{M}'$. For this reason w is not directly part of the witness but an F_{gp}-mapping $W = g_1^w$ and equation (41) translates into a statement about group elements. For an F_{gp}-mapped balance $W \in G_1$ the user must prove

$$\exists\, w'_0, \ldots, w'_{\eta-1} \in \mathbb{Z}_p : W \prod_{j=0}^{\eta-1} Q_j^{w'_j} = g_1^{N_{\max}} \tag{42}$$

$$\exists\, w''_0, \ldots, w''_{\eta-1} \in \mathbb{Z}_p : W \prod_{j=0}^{\eta-1} Q_j^{w''_j} = g_1^{N+v} \tag{43}$$

using $Q_j \in G_1$ as defined by equation (38). These are MSEs and therefore fit into our Groth-Sahai proof system.

Note, that in contrast to (41) the equations (42), (43) make a statement about exponents w'_j, w''_j in $\mathbb{Z}_p$ and not in $\{0, \ldots, q-1\}$. Hence, the user must additionally prove that w'_j, w''_j are valid digits, i. e. actually elements in $\{0, \ldots, q\}$. The user shows for each digit that it knows a suitable signature σ_i that has been issued before. For each digit $i \in \{0, \ldots, q-1\}$ let

$$\sigma_i := \text{Sgn}(\text{sk}_{\text{sig}}, g_2^i) \tag{44}$$

be a corresponding signature. Then the user must prove

$$\forall\, j \in \{0, \ldots, \eta-1\} : \quad \text{Vfy}(\text{pk}_{\text{sig}}, g_2^{w'_j}, \sigma_{w'_j}) = 1 \quad \wedge \quad \text{Vfy}(\text{pk}_{\text{sig}}, g_2^{w''_j}, \sigma_{w''_j}) = 1. \tag{45}$$

These expand into two PPEs each. In summary, including a range proof expands the NIZK by 2 MSEs for correctness of representation and 4η PPEs for correctness of the digits.

IGen(CRS)
$(\text{pk}_{\text{sig}}, \text{sk}_{\text{sig}}) \leftarrow \text{S.Gen(CRS)}$
for $j = 0$ to $q-1$ do $\sigma_j \leftarrow \text{S.Sgn}(\text{sk}_{\text{sig}}, g_2^j)$
return $(\text{pk}_{\mathcal{I}}, \text{sk}_{\mathcal{I}}) := ((\text{CRS}, \text{pk}_{\text{sig}}, \sigma_0, \ldots, \sigma_{q-1}), \text{sk}_{\text{sig}})$

Figure 25: Augmented Issuer Key Generation

G.4 System Setup

The key generation for the issuer must be augmented such that it additionally generates a signature σ_i for each digit $i \in \{0, \ldots, q-1\}$ as in eq. (44). We include these signatures into its public key $\text{pk}_{\mathcal{I}}$. See Fig. 25 for the augmented Issue algorithm.

G.5 Verify and Redeem

In the simple verify-and-redeem protocol Vfy the current balance w of the token was made public and part of the statement. For convenience, we recap the language $L_{3,\text{pk}_{\mathcal{I}}}$ of the simple verify-and-redeem protocol.

$$L_{3,\text{pk}_{\mathcal{I}}} = \left\{ (c', (S, t, u_2), \text{hid}, W) \;\middle|\; \begin{array}{l} \exists\, c, \sigma \in G_2; \\ \text{pk}_{\mathcal{U}}, U_1, D, S', U'_1, D' \in G_1; \\ \text{sk}_{\mathcal{U}}, u_1, r \in \mathbb{Z}_p : \\ \text{E.Enc}(\text{pk}_{\mathcal{T}}, \text{pk}_{\mathcal{U}}; r) = \text{hid} \\ \text{C.Open}(\text{CRS}_{\text{com}}, (S, W, \text{pk}_{\mathcal{U}}, U_1), c, D) = 1 \\ \text{C.Open}(\text{CRS}_{\text{com}}, (S', W, \text{pk}_{\mathcal{U}}, U'_1), c', D') = 1 \\ \text{S.Vfy}(\text{pk}_{\text{sig}}, \sigma, c) = 1 \\ \text{pk}_{\mathcal{U}} = g_1^{\text{sk}_{\mathcal{U}}}, U_1 = g_1^{u_1}, t = \text{sk}_{\mathcal{U}} u_2 + u_1 \end{array} \right\} \tag{46}$$

In the augmented verify-and-redeem protocol the redeemed value v — that is known to both parties anyway — becomes part of the statement and the user proves in zero-knowledge that $w \in \{v, \ldots, N_{\max}\}$ holds. Due to a technical subtlety, we do not have *one* language but $\eta_{\max}$ languages with a varying number of constraints. Before the zero-knowledge proof the user $\mathcal{U}$ and verifier $\mathcal{V}$ have to calculate an $\eta \in [\eta_{\max}]$ according to eq. (39) and then

Table 2: Parameter Settings for Range Proofs

Parameter	Value
q	16
$\eta_{\max}$	4
$N_{\max}$	65535

select the "correct" language $L'_{3,\mathsf{pk}_{\mathcal{I}},\eta}$.

$$L'_{3,\mathsf{pk}_{\mathcal{I}},\eta} = \left\{ (c', (S, t, u_2), \mathsf{hid}, v) \;\middle|\; \begin{array}{l} \exists\, c, \sigma \in G_2; \\ W, \mathsf{pk}_{\mathcal{U}}, U_1, D, S', U'_1, D' \in G_1; \\ \mathsf{sk}_{\mathcal{U}}, u_1, r \in \mathbb{Z}_p; \\ w'_0, \ldots, w'_{\eta-1}, w''_0, \ldots, w''_{\eta-1} \in \{0, \ldots, q-1\} : \\ \mathsf{E.Enc}(\mathsf{pk}_{\mathcal{T}}, \mathsf{pk}_{\mathcal{U}}; r) = \mathsf{hid} \\ \mathsf{C.Open}(\mathsf{CRS}_{\mathsf{com}}, (S, W, \mathsf{pk}_{\mathcal{U}}, U_1), c, D) = 1 \\ \mathsf{C.Open}(\mathsf{CRS}_{\mathsf{com}}, (S', W, \mathsf{pk}_{\mathcal{U}}, U'_1), c', D') = 1 \\ \mathsf{S.Vfy}(\mathsf{pk}_{\mathsf{sig}}, \sigma, c) = 1 \\ \mathsf{pk}_{\mathcal{U}} = g_1^{\mathsf{sk}_{\mathcal{U}}}, U_1 = g_1^{u_1}, t = \mathsf{sk}_{\mathcal{U}} u_2 + u_1 \\ W \prod_{j=0}^{\eta-1} Q_j^{w'_j} = g_1^{N_{\max}} \\ W \prod_{j=0}^{\eta-1} Q_j^{w''_j} = g_1^{N+v} \\ \mathsf{S.Vfy}(\mathsf{pk}_{\mathsf{sig}}, \sigma_{w'_0}, g_2^{w'_0}) = 1 \\ \quad \vdots \\ \mathsf{S.Vfy}(\mathsf{pk}_{\mathsf{sig}}, \sigma_{w'_{\eta-1}}, g_2^{w'_{\eta-1}}) = 1 \\ \mathsf{S.Vfy}(\mathsf{pk}_{\mathsf{sig}}, \sigma_{w''_0}, g_2^{w''_0}) = 1 \\ \quad \vdots \\ \mathsf{S.Vfy}(\mathsf{pk}_{\mathsf{sig}}, \sigma_{w''_{\eta-1}}, g_2^{w''_{\eta-1}}) = 1 \end{array} \right\} \tag{47}$$

We shortly highlight the differences: In line 2 of Eq. (47) the F_{gp}-mapping W of the current balance becomes part of the witness (this is a reminiscence of the Accum-protocol). In line 4 2η digits w'_j and w''_j ($j \in \{0, \ldots, \eta-1\}$) are included into the witness. Line 10 and 11 actually prove that an equation of the balance w and the redemption value v have a certain digit representation. The last 2η lines prove that the digits w'_j and w''_j are indeed *valid* digits, i. e. that they are in the range $\{0, \ldots, q-1\}$ by showing that the user knows a valid signature for each of them.

The augmented Vfy protocol is depicted in Fig. 26.

G.6 Performance Evaluation

For our implementation we decided for the parameters as in Table 2. Using a hexadecimal number system with at most four digits results into 16 additional signatures during the setup phase and 2 additional MSEs plus at most 16 PPEs for the NIZK in the Vfy protocol.

H APPLICATIONS AND FUTURE WORK

In this work, we have introduced Black-Box Accumulators as a general-purpose building block for higher-level applications. In this section, we give some example applications, and explain how BBA schemes can be used in the respective scenario. In some applications, it would be beneficial to use a slightly modified version of BBAs than we have introduced them here. We explain the modifications below.

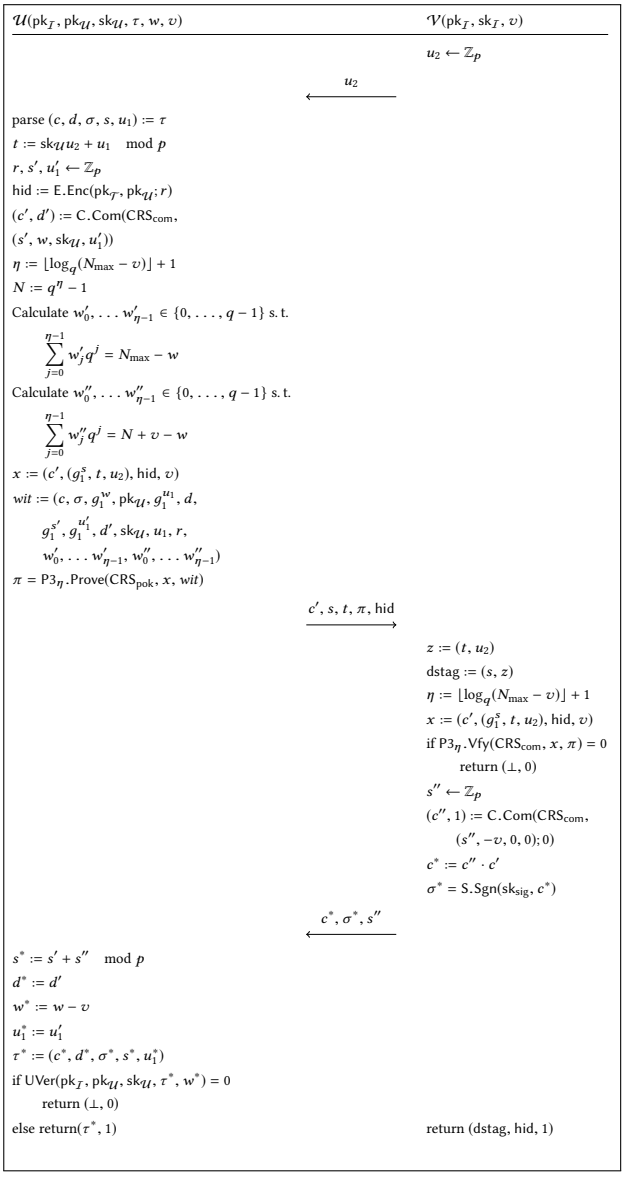

Figure 26: Augmented Verification and Redemption Protocol

H.1 Applications

H.1.1 Customer Loyalty Systems. As a basic application we outline how Black-Box Accumulators can be used to create a customer loyalty system where individual transactions are unlinkable, thus retaining the customer's privacy. When the operator starts a loyalty program, he asks a trusted third party to generate a CRS by running Setup. The trusted third party then publishes the CRS and securely erases the trapdoor. The operator may then generate a key pair $(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}) \leftarrow \mathsf{IGen}(\mathsf{CRS})$ and publishes $\mathsf{pk}_{\mathcal{I}}$.

When a customer registers with the loyalty program, he generates a user key pair $(\mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}) \leftarrow \mathsf{UGen}(\mathsf{CRS})$, and executes the Issue protocol with the operator to obtain a token. The operator

verifies the name and address of the user and stores this information together with $\mathsf{pk}_{\mathcal{U}}$. The owner-binding property wrt. Issue guarantees that it is hard to register a public key $\mathsf{pk}_{\mathcal{U}}$ without knowing $\mathsf{sk}_{\mathcal{U}}$, i. e., only a user who knows $\mathsf{sk}_{\mathcal{U}}$ can register $\mathsf{pk}_{\mathcal{U}}$.

When the customer purchases a product or service, he executes the $\mathcal{AC}$ protocol with the operator, where v is the number of points that the customer receives. When the customer wants to redeem some points (say, $v' \in \mathbb{N}$), he unveils his current balance w to the operator, who checks that $w \geq v'$. The parties then execute the Vfy protocol with $v = -v'$. The balance-binding property prevents a user from unveiling an incorrect balance.

In order to detect double-spending of collected points, the operator regularly scans his database for double-spending tags with identical serial numbers. If there are some, he runs IdentDS in order to obtain the public key $\mathsf{pk}_{\mathcal{U}}$ of the user who committed double-spending and a proof of guilt Π. He then looks up the name and address of the user with key $\mathsf{pk}_{\mathcal{U}}$ and can contact him about this issue. In case the user denies the double-spending, the operator can convince anyone of the double-spending with the proof Π.

Here, the onwer-binding property wrt. Accum and Vfy makes sure that only tokens can be used that are bound to key $\mathsf{pk}_{\mathcal{U}}$ that has been registered before. Furthermore, the double-spending detection property guarantees that a) the public key $\mathsf{pk}_{\mathcal{U}}$ of the user is uniquely determined, b) the IdentDS algorithm returns the correct key $\mathsf{pk}_{\mathcal{U}}$, and c) the proof Π will in fact convince any third party.

Moreover, the false-accusation protection offered by the BBA+ scheme makes sure that an operator can not produce convincing proof of guilt Π for a user that did not commit double-spending.

Finally, while the user is not perfectly anonymous (because he registered with his name and address), his privacy is nonetheless well-protected, due to the privacy property of the BBA+ scheme: Since the operator's views of transactions are computationally indistinguishable from views that are independent of the real user public keys, these transactions are computationally unlinkable. Thus, while the operator knows all transactions and the corresponding goods and services traded in each, he can not tell which transactions belong to the same user. Thus, it is intractable to create personalized profiles of users which may reveal sensitive personal information.

H.1.2 Vehicle to Grid Power Transfer. As the world slowly moves toward more ecologically friendly, renewable and natural sources of energy, the problem of storing large amounts of energy is emerging. For example, the supply of solar and wind power depend on external circumstances, and thus energy harvested from these sources must be stored in order to have energy when the supply is low, e. g. at night.

One approach for storing this energy is to use the batteries of electric cars while they are parked. This is known as vehicle-to-grid power transfer, and involves micro-trading. In this setting, Black-Box Accumulators can be used to realize the transfer of money. Here, a provider of electricity takes the role of the issuer, accumulator and verifier, respectively. The car owner (or his car, acting autonomously on behalf of him) is a user. When the supply of energy is low and the car has sufficient energy in its battery, the car will autonomously sell energy from its battery to the energy provider, and run the Accum protocol to collect points or credit in return. When the supply of energy is high and the car's battery is low, the car may buy power from the provider, and pay by redeeming previously collected points using Vfy. When the points collected by the car are drained, the user may buy additional points (using cash, bank transfer or another commonplace payment method) to charge is token. Likewise, when the car has accumulated a large number of points, the car owner may then redeem the points collected by his car, in order to get paid for the electricity his car has provided.

H.1.3 Stored-Value Card Systems. A common application of so-called stored-value card systems is public transportion, where users present a smartcard upon entering or leaving the transportation system (or both), and the smartcard-based system replaces a traditional ticket-based system. In these systems, each smartcard is linked to a virtual wallet that contains a certain amount of money. Money is withdrawn from the wallet for using the transportation system, and the wallet can be refilled at automated vending machines in exchange for a classical payment in cash or via credit card.

Using Black-Box Accumulators analogously to the ways described above, one may create a stored-value card system where the wallet is actually stored on the card being used, while still guaranteeing that (a) the card's stored value can not be manipulated, (b) double-spenders can be identified, and (c) nothing is leaked beyond the card's current value.

In the scenario of public transportation described above, it is likely that users will charge their token once and then repeatedly pay small amounts using the Vfy protocol, each time revealing their current balance. One might therefore raise the argument that there is a certain risk that users may be tracked by their balance. However, we believe that this will be hard to conduct in practice, since most users will have balances within a certain range. Furthermore, as stored-value card systems are usually deployed in metropolitan areas, we expect the number of persons having the same balance to be quite large. Thus, since the balance is the only information leaked to the Verifier, tracking users will be very hard.

H.1.4 Anonymous Reputation Systems. Anonymous reputation systems are systems where users can collect points for positive reputation, e. g. in public support forums or in a participatory sensing context, and users with higher reputation can access or use designated services reserved for well-reputed users. Applying Black-Box Accumulators to this problem is analogous to the aforementioned applications.

H.2 Active Adversaries

In the security model described in Section 4.3, we consider adversaries that may arbitrarily deviate from the protocol in order to achieve their goals. In Appendix E, we give an extended security model, where the adversary may additionally eavesdrop on protocol executions of honest users in several security experiments. We prove that if a BBA+ scheme satisfies the simpler security model from Section 4.3, then an *encrypted* version of that scheme satisfies the extended definition. Thus, these two security models are mostly equivalent.

However, even in our extended security model, the adversary is passive with respect to honest users, i. e. he may not tamper with protocol executions of honest users. While we believe this is

a realistic model for several of the applications mentioned above, where the user and the issuer are often in direct contact with each other, it may be insufficient when communication is done indirectly.

We therefore consider it interesting to extend our model to adversaries that are not restricted to passively eavesdrop on honest users, but that may additional arbitrarily tamper with their communication as a man-in-the-middle. However, we leave extending our model to such "active" adversaries as future work.