This is the preprint version of an article with the same title that will appear at ACM CCS 2017 on Oct. 30–Nov. 3, 2017, Dallas, TX, USA. DOI: 10.1145/3133956.3134071

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

Gunnar Hartung[*]
Karlsruhe Institute of Technology
Department of Informatics
Karlsruhe, Germany
gunnar.hartung@kit.edu

Max Hoffmann[†]
Ruhr-Universität Bochum
Horst Görtz Institute for IT-Security
Bochum, Germany
max.hoffmann@rub.de

Matthias Nagel[‡]
Karlsruhe Institute of Technology
Department of Informatics
Karlsruhe, Germany
matthias.nagel@kit.edu

Andy Rupp[§]
Karlsruhe Institute of Technology
Department of Informatics
Karlsruhe, Germany
andy.rupp@kit.edu

ABSTRACT

Black-box accumulation (BBA) has recently been introduced as a 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 spend the collected points, thereby only revealing the total amount inside the piggy bank and its unique serial number.

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" which users may not voluntarily collect, as this is, for example, needed in pre-payment or reputation 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 and present the promising results of a smartphone-based prototypical implementation. They show that our current instantiation 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, Reputation Systems, Black-Box Accumulation.

1 INTRODUCTION

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

Well-known examples include loyalty programs like the German Payback system [33] or the UK-based Nectar program [4]. Users may collect points at every purchase for being loyal customers (and for revealing some information about their purchases). Later, collected points may be redeemed 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 [15], users should 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 receive micropayments they can use to pay for services based on the collected data. In Vehicle-to-Grid scenarios [27], e-car owners are rewarded for the power their e-car batteries provide to the Smart Grid when cars are parked at the mall, office, etc.

In [25] Jager and Rupp formalized the core functional, security, and privacy requirements of a building block to realize the kind of systems described above. Their new building block, called black-box accumulation (BBA), consists of a set of non-interactive algorithms to generate, manipulate, and show properties about a BBA token (aka piggy bank). When correctly executed by legitimate parties, it allows a user to collect *positive* points (representing incentives) in an anonymous and unlinkable fashion. More precisely, in the BBA framework we have four parties: user, issuer, accumulator, and

[*] 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.

[‡] The author is 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).

verifier. In the beginning, a user receives a BBA token generated by the issuer (using his secret key) which is bound to a unique serial number known to both parties. All points are collected using this single, constant-size accumulation token. To this end, a user blinds and unblinds the token before and after every transaction with an accumulator (who uses the same secret key as the issuer). When redeeming the token, the sum of all collected points as well as the serial number is revealed to the verifier.

Obviously, 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. Moreover, 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 to them. In particular, this does not rule out that users may transfer points arbitrarily between their BBA tokens (without help). Also, an "old" copy of a BBA token (i.e., one not holding the most current balance) can be used for the accumulation or redemption of points. In this way, a user could easily get rid of "negative" points.

To summarize, the original BBA framework suffers from a number of serious restrictions including (1) fairly weak security guarantees, (2) the need of a permanent database connection, (3) the lack of mechanisms to enforce the collection of "negative" points, and (4) 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.

Other applications 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 able to issue negative reputation points either by subtracting points or having a separate counter for negative ratings.

Yet another set of applications which benefit from stronger security, offline capabilities, and negative points, are pre- or post-payment systems. These systems are employed in many domains like public transportation, toll collection, cashless canteen systems, etc. In practice, such payment systems are typically implemented using simple RFID-transponder or smartcard-based solutions like the Mifare Classic [30], which essentially offers no security and privacy at all ([16, 19, 20] and more), or the Mifare Desfire [31, 32] also allowing to link all transactions.

1.1 Our Contribution

Definitional framework. We present the BBA+ framework which addresses the restrictions of BBA discussed above, thereby significantly strengthening the security and broadining the applicability of black-box accumulation.

Our framework offers the following additional features: We consider interactive algorithms (protocols) which leads to more intuitive definitions and broadens the class of possible instantiations. The 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).

Our security model is game-based. We formalize a stronger security property which captures the natural notion that the claimed balance for each BBA+ token must be exactly the amount legitimately collected with this token up to this point. Note that due to the strong privacy property that needs to be satisfied in parallel, 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. Our security model consists of a reduced model and a full-fledged model, where the full-fledged model additionally considers eavesdropping on honest users. In the full version of this paper [22], we define the full-fledged model in detail. Moreover, we show that any system secure in the reduced model is also secure in the full-fledged model provided that we additionally encrypt all protocol messages using an IND-CCA secure scheme.

Our privacy model is simulation-based. 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 (except the very next) a corruption 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. In this context, note that the lifetime of a BBA+ token is essentially unrestricted, and, in contrast to a BBA token, does not end with the first redemption of points.

Details on our framework are given in Section 3.

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 (GS) 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., $t := id \cdot u_2 + u_1 \bmod p$, needs to be revealed when spending an e-coin. This tag contains user identity information id which is blinded by some secret user randomness u_1 (which has been fixed when the coin was issued) and involves a challenge u_2 freshly chosen by a merchant. No information about id is revealed when the coin is spent once (as u_1 is uniformly drawn). However, when the coin is spent a second time, a different challenge u_2' is used in the revealed tag $t' := id \cdot u_2' + u_1 \bmod p$, while the user randomness u_1 and id are the same (which needs to be enforced by the protocol). This enables the bank to extract id using the two double-spending tags t and t' and challenge values u_2 and u_2'.

Let us briefly sketch our construction which follows but significantly extends the idea of [25]. For the sake of simplicity, each user may only receive a single token bound to his ID. An initial BBA+ token jointly generated by the issuer and the user essentially consists of a (multi-)commitment c and a signature σ on this commitment under the issuer's secret key. The commitment c contains a user's secret key $\mathrm{sk}_{\mathcal{U}}$, a token version number s, the balance value $w = 0$, and some randomness u_1 that will be used to generate a double-spending tag in the next transaction. Note that $\mathrm{sk}_{\mathcal{U}}$, s, and u_1 are not known to the issuer but $\mathrm{pk}_{\mathcal{U}}$, c, and σ are.

To add (positive or negative) points in an unlinkable fashion, one cannot simply send over the token (c, σ) to the accumulator. Instead, the user sends a new commitment c' containing the same secret key $\mathrm{sk}_{\mathcal{U}}$, the same balance w, but a new token version number s',[1] and some new randomness u'_1. Then he proves in zero-knowledge that c' is indeed a new, correctly modified version of his old certified commitment c. Additionally, the proof ensures that a double-spending tag (encoding $\mathrm{sk}_{\mathcal{U}}$) along with the version number s for the old token (c, σ) is revealed. The version number is used to index the double-spending tags in the database. If the accumulator accepts the proof, the homomorphic property of the commitment scheme is used to add v points to c' which is then signed. The new token (c^*, σ^*) is sent to the user. Verifying the balance of a token and redeeming points works analogously, except that the balance w is revealed to the verifier. Note that the computational complexity of all operations as well as the token size is independent of the number of points to be transferred or stored. This concludes the simplified description of our construction. More details can be found in Section 4.

While our construction is fairly intuitive and draws from techniques also commonly used in e-cash or P-signatures, there are technical differences to these concepts as explained in the related work section. The main challenge was to carefully combine these techniques into a protocol that it is both provably secure and efficient at the same time. For instance, proving that a token can only be used with its legitimate balance is highly non-trivial. Other technical challenges arise from building on the Groth-Sahai proof system. GS proofs are efficient and secure in the CRS model but require particular care, as they are no proper proofs-of-knowledge for witness components over $\mathbb{Z}_p$ and not always zero-knowledge. For example, to prove statements about shrinking multi-commitments to $\mathbb{Z}_p$-messages, which we use to obtain compact tokens and proofs, the employed commitment scheme needs to satisfy a non-standard binding property (F-binding).

Implementation. In order to assess the suitability of our construction for real-world applications, we implemented our BBA+ instantiation and measured execution times on a smartphone. Our implementation results show that all protocols can be executed in less than 400 ms on the user side (for 254-bit Barreto-Naehrig curves with optimal Ate pairing). This leads to the conclusion that our instantiation is already usable in practice and can be efficiently executed with current, well-established hardware. Details on the implementation and measurements, as well as ideas for further optimizations can be found in Section 6.

1.2 Related Work

At first sight, the problem of privacy-preserving point collection might appear easily solvable using (offline) e-cash: user and accumulator may execute the e-coin withdrawal protocol to collect several points. All collected coins may later be redeemed using the spend protocol (multiple times). However, besides not being very efficient, because coins typically cannot be aggregated, this also violates user privacy as in traditional offline e-cash, e.g. [12], withdrawing e-coins is identifying. This is because the identity of a user needs to be encoded into an e-coin during withdrawal to enable double-spending detection. In our system, we initially encode this identity into the user's token (aka wallet or piggy bank) which is used to collect points and not into a point itself.

Even transferable e-cash, e.g. [6], does not achieve our goals. In such a scheme, the ownership of a coin can be transferred anonymously and unlinkably between users multiple times without the help of the bank. Applied to our scenario, an accumulator could thus withdraw e-coins, possibly from the issuer acting as bank, and transfer them anonymously to a user. However, an impossibility result by Canard and Gouget [13] implies that an adversary impersonating issuer, accumulators, and verifiers would be able to link a user's collection and redeeming transactions. Moreover, transferable e-cash allows users to transfer e-coins arbitrarily among each other, a property which is undesirable in our scenario as users would be able to pool their points.

Besides BBA [25], which we already discussed, only [28] appears to consider a point collection mechanism as a multi-purpose building block on its own. However, the proposed protocol—called uCentive—targets a simpler scenario than we do: incentives are not accumulatable on the user's side but stored and redeemed individually, negative points are not supported, and double-spending detection is done online rather than offline. BBA+ and uCentive also differ regarding the use of cryptographic building blocks: uCentive makes use of anonymous credentials and partially blind signatures. Moreover, the security and privacy properties of their system are only informally stated and no proofs are given.

BBA+ shares some aspects with the notion of priced oblivious transfer (POT). POT was introduced by Aiello et al. [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. [11] 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, [11] lacks a rigorous formal treatment and an implementation.

The techniques we use to instantiate BBA+ bear some resemblance with P-signatures [8, 24] which have been introduced by [8] 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

[1] Actually, the new version number is jointly chosen by user and issuer. s' is only the user's share.

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 prove that he knows a valid signature on the message inside a commitment or generate two new commitments and prove the equality of their content. The scheme in [8] builds on weak Boneh-Boyen signatures [9], Groth-Sahai commitments and Groth-Sahai NIZK proofs [21].

To instantiate a BBA+ scheme, properties beyond that of a P-signature scheme are needed. To see this, let us try to build a BBA+ scheme from P-signatures. We would like to embed our piggy bank as a message, encoding the balance, a serial number, a user ID, and maybe some additional information. This message could then be blindly signed by the issuer using the P-signature scheme. To collect some points, the user would generate a fresh commitment to the message and send it to the accumulator along with a proof showing that he has a signature on the message inside the commitment. Note that this step can be done repeatedly using the same message, as P-signatures do not include a mechanism (double-spending detection) to prevent a user from showing the same message twice. In fact, such a mechanism is not required for standard anonymous credentials. However, in our setting this is necessary as a user could present an old piggy bank to collect or redeem points and thus get rid of negative points. Apart from this, there are other shortcomings. After a user proved that she has a valid piggy bank, it needs to be updated by the accumulator to add new points. To realize this with the P-signature scheme, the user would like to update the balance in the old message accordingly, commit to this updated message and prove that the two messages (for which the accumulator obtained commitments) are closely "related" (i.e., only differ wrt. balance and serial number). However, a P-signature scheme only offers a protocol for showing that two commitments contain the *same* message.

From a technical perspective, in our BBA+ instantiation, commitments rather than messages inside commitments as in [8, 24] are signed. Thus, in combination with Groth-Sahai NIZK proofs we make use of F-binding commitments (cf. Section 2.2) rather than F-unforgeable signatures as in [8].

2 PRELIMINARIES

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

2.1 Bilinear Groups and Assumptions

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 $a \in G_1, b \in G_2$, $x, y \in \mathbb{Z}_p$, we have $e(a^x, b^y) = e(a,b)^{xy}$.
- *Non-Degeneracy:* $e(g_1, g_2)$ generates G_T.

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

$$\left| \Pr\left[b = b' \;\middle|\; \begin{array}{c} \mathsf{gp} := (G_1, G_2, G_T, e, p, g_1, g_2) \leftarrow \mathsf{SetupGrp}(1^n) \\ x, y, z \leftarrow \mathbb{Z}_p; h_0 := g_i^{xy}; h_1 := g_i^z; b \leftarrow \{0,1\} \\ b' \leftarrow \mathcal{A}(1^n, \mathsf{gp}, g_i^x, g_i^y, h_b) \end{array} \right] - \frac{1}{2} \right|$$

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 $\mathcal{A}$.

2.2 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. In the following, we describe these building blocks in an informal fashion appropriate to understand the construction. Formal definitions can be found in the full version [22].

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 (group-based) NIZK proof system consists of the algorithms SetupGrp, SetupPoK, Prove, and Vfy. $\mathsf{SetupGrp}(1^n)$ generates public parameters gp given implicitly to all algorithms. The considered language L_{gp} may depend on gp. SetupPoK(gp) outputs a common reference string CRS. $\mathsf{Prove}(\mathsf{CRS}, x, wit)$ outputs a proof π on input of $x \in L_{\mathsf{gp}}$ and a valid witness wit with $(x, wit) \in R$, else Prove outputs $\perp$. $\mathsf{Vfy}(\mathsf{CRS}, x, \pi)$ outputs 1 if π is considered a valid proof for $x \in L_{\mathsf{gp}}$, and 0 otherwise. The proof system is called *perfectly complete* if $\mathsf{Vfy}(\mathsf{CRS}, x, \pi)$ always accepts proofs generated by $\mathsf{Prove}(\mathsf{CRS}, x, wit)$. It is called *perfectly sound* if it is impossible to generate a proof π for $x \notin L_{\mathrm{gp}}$ 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 $\mathrm{td}_{\mathrm{epok}}$ and (2) ExtractW is able to exploit this trapdoor to extract $F_{\mathrm{gp}}(wit)$ for an NP-witness *wit* for $x \in L_{\mathrm{gp}}$ from any 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 $g_1^e \in G_1$ or $g_2^e \in G_2$ (depending on the context e is used in) and acts as the identity function on group elements. This is a property of the Groth-Sahai proof system [18, 21] 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 $\mathrm{td}_{\mathrm{spok}}$ and (2) SimProof can use this trapdoor to generate proofs for x (not necessarily in L_{gp}) without knowing a witness that look real even if $\mathrm{td}_{\mathrm{spok}}$ 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 $F_{\mathrm{gp}}(m)$.

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) creates 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. To verify that a commitment can be opened to a message Open is used. It takes CRS, c, d, as well as some implicit message SM 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 $F_{\mathrm{gp}}(m)$. 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 to any given implicit message. Finally, the scheme is called *additively homomorphic* if commitments c_1 to m_1 and c_2 to m_2 with decommitment values d_1 and d_2, respectively, can efficiently be combined using $\mathsf{CAdd}(c_1, c_2)$, resulting in a commitment c to $m_1 + m_2$ with decommitment value $d \leftarrow \mathsf{DAdd}(d_1, d_2)$.

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. 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, 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 $\mathsf{Enc}(\mathsf{pk}, m)$ takes pk and a message m and outputs a ciphertext c. The decryption algorithm $\mathsf{Dec}(\mathsf{sk}, c)$ 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 $\mathcal{A}$.

3 BBA+ DEFINITION

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

3.1 High-Level System Description

Let us start with an overview of the different parties involved in a BBA+ scheme and a 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. Issuers, accumulators and verifiers are subsumed under the term operators. In particular, they need to trust each other as the share the same secret key (see below).

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 used by all algorithms and protocols 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 is 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 uses $\mathsf{sk}_{\mathcal{U}}$ to prove that he is the owner of the claimed public key $\mathsf{pk}_{\mathcal{U}}$ for which a token should be generated. 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 $\mathsf{pk}_{\mathcal{U}}$.[2] 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 balance 0.

Collecting points. To add a (positive or negative) value v to the current balance[3] 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 number. If the DB contains two such records, then the 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 the algorithm VerifyGuilt. The DB is typically maintained by a system operator who coincides with the issuer in many scenarios. Also, IdentDS is 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 who possibly wants to redeem v points, interacts 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.[4] 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.

3.2 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 3.1 (BBA+ Scheme). An *extended black-box accumulation (BBA+)* scheme BBAP = (Setup, IGen, UGen, Issue, Accum, Vfy, UVer, IdentDS, 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.[5]

$(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}) \leftarrow \mathsf{IGen}(\mathsf{CRS})$. 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 $\mathsf{sk}_{\mathcal{I}}$ is shared with accumulator and verifier. We assume for convenience that CRS is 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) along with a bit $b_{\mathcal{U}}$. The issuer's output is 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, a hidden user ID hid,[6] 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 $\mathcal{V}$ receives the current token balance w as an additional input.

$b \leftarrow \mathsf{UVer}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, \tau, w)$. The *token verification* algorithm is a deterministic 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 a balance w owned by the user with public key $\mathsf{pk}_{\mathcal{U}}$.

$(\mathsf{pk}_{\mathcal{U}}, \Pi) \leftarrow \mathsf{IdentDS}(\mathsf{pk}_{\mathcal{I}}, \mathsf{dstag}_1, \mathsf{dstag}_2)$. The *double-spender detection* algorithm is a deterministic polynomial-time algorithm which is given $\mathsf{pk}_{\mathcal{I}}$ and two double-spending tags $\mathsf{dstag}_1 = (s_1, z_1)$ and $\mathsf{dstag}_2 = (s_2, z_2)$. It returns the public key $\mathsf{pk}_{\mathcal{U}}$ of a user and a proof of guilt Π, or it returns an error $\perp$.

[2] 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.

[3] 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.

[4] In certain scenarios revealing w may significantly help to link transactions. For such applications, the framework can be extended to only show a bound on the balance (cf. Section 8.1). Due to efficiency reasons, we omit this feature in our basic system.

[5] The trapdoor is needed in the security definition to define the legitimate balance of a token although token transactions are unlinkable.

[6] hid is used for definitorial purposes only. In our instantiation, hid is an encryption of $\mathsf{pk}_{\mathcal{U}}$.

$b \leftarrow \mathsf{VerifyGuilt}(\mathsf{pk}_\mathcal{I}, \mathsf{pk}_\mathcal{U}, \Pi)$. The *guilt verification* algorithm is a deterministic 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.

Correctness for BBA+ schemes is fairly straightforward:

Definition 3.2 (BBA+ Correctness). A BBA+ scheme BBAP is called *correct* if all of 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 $((\tau, b_\mathcal{U}), (b_\mathcal{I})) \leftarrow \mathsf{Issue}\langle \mathcal{U}(\mathsf{pk}_\mathcal{I}, \mathsf{pk}_\mathcal{U}, \mathsf{sk}_\mathcal{U}), \mathcal{I}(\mathsf{pk}_\mathcal{I}, \mathsf{sk}_\mathcal{I}, \mathsf{pk}_\mathcal{U})\rangle$, it holds that $b_\mathcal{U} = b_\mathcal{I} = 1$ and $\mathsf{UVer}(\mathsf{pk}_\mathcal{I}, \mathsf{pk}_\mathcal{U}, \mathsf{sk}_\mathcal{U}, \tau, 0) = 1$.

Correctness of accumulation. For all tokens τ, balances $w \in \mathbb{Z}_p$ with $\mathsf{UVer}(\mathsf{pk}_\mathcal{I}, \mathsf{pk}_\mathcal{U}, \mathsf{sk}_\mathcal{U}, \tau, w) = 1$ and all values $v \in \mathbb{Z}_p$, we have that $((\tau^*, 1), (s, z, \mathsf{hid}, 1)) \leftarrow \mathsf{Accum}\langle \mathcal{U}(\mathsf{pk}_\mathcal{I}, \mathsf{pk}_\mathcal{U}, \mathsf{sk}_\mathcal{U}, \tau, w, v), \mathcal{AC}(\mathsf{pk}_\mathcal{I}, \mathsf{sk}_\mathcal{I}, v)\rangle$ and $\mathsf{UVer}(\mathsf{pk}_\mathcal{I}, \mathsf{pk}_\mathcal{U}, \mathsf{sk}_\mathcal{U}, \tau^*, w + v) = 1$.[7]

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

3.3 Definition of System Security

For security we distinguish between a reduced, simplified model and a more natural, full-fledged model that is given in the full version of the paper [22]. In the full-fledged model, the adversary can be a collusion of malicious users who additionally may command, eavesdrop on, and adaptively corrupt honest users. In the reduced model, introduced in the following, no interactions with honest users are considered. Fortunately, we can show in a black-box fashion that any scheme secure in the reduced model is also secure in the full-fledged model if all protocol messages are additionally encrypted with an IND-CCA secure encryption scheme. Note that privacy is not affected by extending the protocols with encryption.

With our security definition, we essentially capture three properties:

1. A token may only be created in the name of and used by its legitimate owner (owner-binding).
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 the token is presented (balance-binding).
3. Users presenting old tokens can be identified after the fact.

Formalizing the notion above raises a major problem: It requires to link each transaction with a user and token. However, on the other hand, we 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 verifier. 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., $(\mathsf{pk}_\mathcal{I}, \mathsf{sk}_\mathcal{I}, v, \mathsf{msgs}, \mathsf{dstag}, \mathsf{hid}, b_\mathcal{AC})$, where $\mathsf{msgs} \in \{0,1\}^*$ is the bit string 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 $(\mathsf{pk}_\mathcal{I}, \mathsf{sk}_\mathcal{I}, w, v, \mathsf{msgs}, \mathsf{dstag}, \mathsf{hid}, b_\mathcal{V})$. 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 3.3 (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 $\mathsf{view} = (\mathsf{pk}_\mathcal{I}, \mathsf{sk}_\mathcal{I}, v, \mathsf{msgs}, \mathsf{dstaghid}, 1) \in \mathcal{V}^{\mathsf{Accum}}_{n,\mathsf{CRS}}$ resulting from an Accum protocol run with an honest user on input $\mathsf{pk}_\mathcal{U}$, ExtractUID outputs $\mathsf{pk}_\mathcal{U}$ on input $(\mathsf{td}, \mathsf{hid})$. 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 concurrently interact with an honest issuer, accumulator, and verifier an arbitrary number of times. 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 Definitions 3.4 and 3.5 cover

[7] To simplify definitions, subtraction by v is not handled as a separate operation but by adding $v' := p - v \bmod p$. In an implementation, one may prefer having a subtraction operation though.

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

$(\mathsf{CRS}, \mathsf{td}) \leftarrow \mathsf{Setup}(1^n)$
$(\mathsf{pk}_\mathcal{I}, \mathsf{sk}_\mathcal{I}) \leftarrow \mathsf{IGen}(\mathsf{CRS})$
$(\mathsf{pk}_\mathcal{U}, \mathsf{sk}_\mathcal{U}) \leftarrow \mathsf{UGen}(\mathsf{CRS})$
$b \leftarrow \mathcal{A}^{\mathsf{MalIssue}, \mathsf{MalAcc}, \mathsf{MalVer}}(\mathsf{pk}_\mathcal{I}, \mathsf{pk}_\mathcal{U})$
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 $\mathsf{Exp}^{\mathsf{ob\text{-}acc\text{-}ver}}_{\mathsf{BBAP},\mathcal{A}}(n)$

$(\mathsf{CRS}, \mathsf{td}) \leftarrow \mathsf{Setup}(1^n)$
$(\mathsf{pk}_\mathcal{I}, \mathsf{sk}_\mathcal{I}) \leftarrow \mathsf{IGen}(\mathsf{CRS})$
$b \leftarrow \mathcal{A}^{\mathsf{MalIssue}, \mathsf{MalAcc}, \mathsf{MalVer}}(\mathsf{pk}_\mathcal{I})$
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 with respect to the different protocols Issue, Accum, and Vfy. Definition 3.6 formalizes the balance-binding property assuming that no double-spending took place. Consequently, Definition 3.7 ensures that such double-spendings are indeed hard to accomplish without being identified.

In Definition 3.4, 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 3.4. A trapdoor-linkable BBA+ scheme BBAP is called owner-binding with respect to Issue if for any PPT adversary $\mathcal{A}$ in the experiment $\mathsf{Exp}^{\mathsf{ob\text{-}issue}}_{\mathsf{BBAP},\mathcal{A}}(n)$ from Fig. 1 the advantage of $\mathcal{A}$ defined by

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

is negligible in n.

Definition 3.5 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 3.5. 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 $\mathsf{Exp}^{\mathsf{ob\text{-}acc\text{-}ver}}_{\mathsf{BBAP},\mathcal{A}}(n)$ from Fig. 2 the advantage of $\mathcal{A}$ defined by

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

is negligible in n.

With Definition 3.6 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 (1) the claimed balance is not equal to the "real" balance of the presented token or that (2) the "real" balance does not coincide with the sum of legitimately collected points associated with this token in the records.

Experiment $\mathsf{Exp}^{\mathsf{bb}}_{\mathsf{BBAP},\mathcal{A}}(n)$

$(\mathsf{CRS}, \mathsf{td}) \leftarrow \mathsf{Setup}(1^n)$
$(\mathsf{pk}_\mathcal{I}, \mathsf{sk}_\mathcal{I}) \leftarrow \mathsf{IGen}(\mathsf{CRS})$
$b \leftarrow \mathcal{A}^{\mathsf{MalIssue}, \mathsf{MalAcc}, \mathsf{MalVer}}(\mathsf{pk}_\mathcal{I})$
The experiment returns 1 iff $\mathcal{A}$ did a successful call to MalVer resulting in a view $\mathsf{view} = (\mathsf{pk}_\mathcal{I}, \mathsf{sk}_\mathcal{I}, w, v, \mathsf{msgs}, \mathsf{dstag}, \mathsf{hid}, 1) \in \mathcal{V}^{\mathsf{Vfy}}_{n,\mathsf{CRS}}$ 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 produced unique token version numbers
- 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 $\mathsf{Exp}^{\mathsf{dsd}}_{\mathsf{BBAP},\mathcal{A}}(n)$

$(\mathsf{CRS}, \mathsf{td}) \leftarrow \mathsf{Setup}(1^n)$
$(\mathsf{pk}_\mathcal{I}, \mathsf{sk}_\mathcal{I}) \leftarrow \mathsf{IGen}(\mathsf{CRS})$
$b \leftarrow \mathcal{A}^{\mathsf{MalIssue}, \mathsf{MalAcc}, \mathsf{MalVer}}(\mathsf{pk}_\mathcal{I})$
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 at least one of the following conditions is satisfied:

- $\mathsf{pk}^{(1)}_\mathcal{U} \neq \mathsf{pk}^{(2)}_\mathcal{U}$ or
- $\mathsf{IdentDS}(\mathsf{pk}_\mathcal{I}, \mathsf{dstag}_1, \mathsf{dstag}_2) \neq (\mathsf{pk}^{(1)}_\mathcal{U}, \Pi)$ 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 3.6. A trapdoor-linkable BBA+ scheme BBAP is called balance-binding if for any PPT adversary $\mathcal{A}$ in the experiment $\mathsf{Exp}^{\mathsf{bb}}_{\mathsf{BBAP},\mathcal{A}}(n)$ from Fig. 3 the advantage of $\mathcal{A}$ defined by

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

is negligible in n.

Definition 3.7 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 3.7. A trapdoor-linkable BBA+ scheme BBAP ensures double-spending detection if for any PPT adversary $\mathcal{A}$ in the experiment $\mathsf{Exp}^{\mathsf{dsd}}_{\mathsf{BBAP},\mathcal{A}}(n)$ from Fig. 4 the advantage of $\mathcal{A}$ defined by

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

is negligible in n.

3.4 Definition of User Security and Privacy

This section presents the key security properties for users, protecting them from dishonest operators: 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, which 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, the definition demands that Accum and Vfy do not reveal any information (except for the balance in case of Vfy) that may help

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

Figure 5: Real world privacy experiment.

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

Figure 6: Ideal world privacy experiment.

in linking transactions. This even needs to hold for transactions preceding and succeeding (except the very next) the corruption of the user. 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 chooses 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 $\mathsf{pk}_{\mathcal{U}}$.[8] 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 be 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.
- $\mathsf{RealHonIssue}(\mathsf{pk}_{\mathcal{U}})$ 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.
- $\mathsf{RealHonAcc}(\mathsf{pk}_{\mathcal{U}}, v)$ lets the user with public key $\mathsf{pk}_{\mathcal{U}}$ run the Accum protocol with the adversary impersonating the accumulator $\mathcal{AC}$ on input $v \in \mathbb{Z}_p$, provided that the oracle $\mathsf{RealHonIssue}(\mathsf{pk}_{\mathcal{U}})$ has successfully been called at some point before.
- $\mathsf{RealHonVer}(\mathsf{pk}_{\mathcal{U}}, v)$ lets the user with public key $\mathsf{pk}_{\mathcal{U}}$ run the Vfy protocol with the adversary impersonating the verifier $\mathcal{V}$ on input $v \in \mathbb{Z}_p$ provided that $\mathsf{RealHonIssue}(\mathsf{pk}_{\mathcal{U}})$ has successfully been called at some point before.
- $\mathsf{RealCorrupt}(\mathsf{pk}_{\mathcal{U}})$ can be called by the adversary to corrupt the user with public and secret key $(\mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}) \leftarrow \mathsf{HonUser}()$. The oracle outputs the secret key $\mathsf{sk}_{\mathcal{U}}$ of the user as well as the user's most recent token τ along with the balance w.

In the *ideal world*, depicted in Fig. 6, first a CRS along with a simulation trapdoor $\mathsf{td}_{\mathsf{sim}}$ is generated honestly. The adversary only receives the CRS as input like 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.

The basic idea is that all messages that are sent from a user to the adversary in the scope of the protocols Vfy and $\mathcal{AC}$ are simulated, i.e., are generated without any user-related data and thus are information-theoretic independent. Nonetheless, if the adversary corrupts a specific user by means of the SimCorrupt oracle, the adversary expects to see a correct secret key $\mathsf{sk}_{\mathcal{U}}$, a plausible token τ, and a correct balance w. As the adversary commands all users, he can keep track of all balances and the simulation must do the same. Hence, the oracles are implemented in a very specific way.

The adversary interacts with one global user simulator $\mathcal{U}_{\mathsf{sim}}$ that keeps track of all interactions and the oracles are interfaces of this simulator. Internally, the simulator stores and updates for each $\mathsf{pk}_{\mathcal{U}}$ the corresponding $\mathsf{sk}_{\mathcal{U}}$, the current balance w and the latest state object. This state object enables the simulator to come up with a correct token upon corruption of a specific user. When the adversary invokes an oracle (aka interface) the user simulator $\mathcal{U}_{\mathsf{sim}}$ internally invokes a corresponding simulation algorithm that *does not* receive any user-related input and returns an updated state object to the simulator. Messages between the internal simulation algorithm and the adversary are forwarded by the simulator.

If the adversary calls SimHonAcc or SimHonVer for a user with ID $\mathsf{pk}_{\mathcal{U}}$ that has been successfully corrupted in the previous call for this $\mathsf{pk}_{\mathcal{U}}$, then the user simulator does not run the simulation algorithm but executes the real protocol using the information that has been returned by SimCorrupt.[9]

In the ideal game, the user simulator $\mathcal{U}_{\mathsf{sim}}$ behaves as follows upon an adversarial invocation of the oracles (aka interfaces):

- HonUser() creates a new user entity by running $(\mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}) \leftarrow \mathsf{UGen}(\mathsf{CRS})$. The simulator returns $\mathsf{pk}_{\mathcal{U}}$ to the adversary and stores $\mathsf{sk}_{\mathcal{U}}$ in the internal database.
- $\mathsf{SimHonIssue}(\mathsf{pk}_{\mathcal{U}})$ simulates a user with public key $\mathsf{pk}_{\mathcal{U}}$ who runs 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 simulation algorithm gets $\mathsf{pk}_{\mathcal{U}}$ but not $\mathsf{sk}_{\mathcal{U}}$.
- $\mathsf{SimHonAcc}(\mathsf{pk}_{\mathcal{U}}, v)$ simulates a user with public key $\mathsf{pk}_{\mathcal{U}}$ that runs the Accum protocol with the adversary impersonating $\mathcal{AC}$ on common input $v \in \mathbb{Z}_p$, provided that $\mathsf{SimHonIssue}(\mathsf{pk}_{\mathcal{U}})$ has successfully been called at some

[8] 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.

[9] 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 $\mathsf{pk}_{\mathcal{U}}$.

```
Experiment $\mathsf{Exp}^{\mathsf{facp}}_{\mathsf{BBAP},\mathcal{A}}(n)$
  $(\mathsf{CRS}, \mathsf{td}) \leftarrow \mathsf{Setup}(1^n)$
  $(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}) \leftarrow \mathsf{IGen}(\mathsf{CRS})$
  $(\mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}) \leftarrow \mathsf{UGen}(\mathsf{CRS})$
  $\Pi \leftarrow \mathcal{A}^{\mathsf{RealHonIssue},\mathsf{RealHonAcc},\mathsf{RealHonVer}}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}})$
  The experiment returns 1 iff $\mathsf{VerifyGuilt}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \Pi) = 1$.
```

Figure 7: False accusation protection experiment.

point before. The internal simulation algorithm only gets v but not τ, w, $\mathsf{pk}_{\mathcal{U}}$ nor $\mathsf{sk}_{\mathcal{U}}$.

- $\mathsf{SimHonVer}(\mathsf{pk}_{\mathcal{U}}, v)$ simulates a user with public key $\mathsf{pk}_{\mathcal{U}}$ that runs the Vfy protocol with the adversary impersonating the verifier $\mathcal{V}$ on common input $v \in \mathbb{Z}_p$ provided that $\mathsf{SimHonIssue}(\mathsf{pk}_{\mathcal{U}})$ has successfully been called at some point before. The user simulator looks up the recent balance w associated with $\mathsf{pk}_{\mathcal{U}}$ in the database and calls the simulation algorithm. The internal simulation algorithm gets v and w but not τ, $\mathsf{pk}_{\mathcal{U}}$ nor $\mathsf{sk}_{\mathcal{U}}$.
- $\mathsf{SimCorrupt}(\mathsf{pk}_{\mathcal{U}})$ can be called by the adversary to corrupt the user with public key $\mathsf{pk}_{\mathcal{U}}$. The simulator looks up $\mathsf{sk}_{\mathcal{U}}$ associated with $\mathsf{pk}_{\mathcal{U}}$, the current balance w, and the state object from the database. It then calls the internal simulation algorithm on this input. The output of the algorithm is a tuple $(\mathsf{sk}_{\mathcal{U}}, w, \tau)$ and is returned to the adversary.

Please note that the internal simulation algorithms of SimHonIssue, SimHonAcc, SimHonVer don't get a token, $\mathsf{sk}_{\mathcal{U}}$ nor the state object as input. However, upon corruption the internal algorithm of $\mathsf{SimCorrupt}(\mathsf{pk}_{\mathcal{U}})$ must come up with a valid token as part of its output $(\mathsf{sk}_{\mathcal{U}}, w, \tau)$. Essentially, this means that this algorithm must create a plausible token out of the latest state object. But this object has been created by algorithms that have never received any user-specific input.

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

Definition 3.8. 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 SimHonIssue, 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 $\mathsf{Adv}^{\mathsf{priv}}_{\mathsf{BBAP},\mathcal{A}}(n)$ of $\mathcal{A}$ defined by

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

is negligible in n.

Finally, Definition 3.9 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 $\mathcal{I}$, $\mathcal{AC}$, and $\mathcal{V}$.

Definition 3.9. A trapdoor-linkable BBA+ scheme BBAP ensures false-accusation protection if for any PPT adversary $\mathcal{A}$ in the experiment $\mathsf{Exp}^{\mathsf{facp}}_{\mathsf{BBAP},\mathcal{A}}(n)$ from Fig. 7 the advantage of $\mathcal{A}$ defined by

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

is negligible in n.

4 BBA+ INSTANTIATION

In this section, we present our basic scheme BBAP which is secure with respect to the "reduced model" presented in Section 3.3 and privacy-preserving with respect to the model in Section 3.4. As already mentioned, this basic protocol can easily be made secure in a full-fledged model in which eavesdropping on and corrupting of honest users is allowed, by encrypting all messages transmitted during the protocols Issue, Accum and Vfy. Please refer to the full version of the paper [22] for details.

4.1 Building Blocks

Let SetupGrp be a bilinear group generator (cf. Definition 2.1) which outputs the description of a bilinear group $gp := (G_1, G_2, G_T, e, p, g_1, g_2) \leftarrow \mathsf{SetupGrp}(1^n)$ for which the SXDH problem is assumed to be hard. For our construction, we draw from the following building blocks (cf. Section 2.2 for informal definitions) which all make use of SetupGrp as their common group setup algorithm.

NIZKs. For proving that a user behaves honestly in the scope of the Issue, Accum, and Vfy protocol, 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 $L^{(1)}_{\mathsf{pk}_{\mathcal{I}}}$, $L^{(2)}_{\mathsf{pk}_{\mathcal{I}}}$, and $L^{(3)}_{\mathsf{pk}_{\mathcal{I}}}$ (defined later), 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 of the 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 share a common reference string. More precisely, we demand that there is a shared setup algorithm SetupPoK which generates the CRS and also a shared setup algorithm SetupEPoK that additionally generates a single extraction trapdoor for P1, P2, and P3. In the following, let us denote their output by $\mathsf{CRS}_{\mathsf{pok}} \leftarrow \mathsf{SetupPoK}(\mathsf{gp})$ and $(\mathsf{CRS}_{\mathsf{pok}}, \mathsf{td}_{\mathsf{epok}}) \leftarrow \mathsf{SetupEPoK}(\mathsf{gp})$ respectively. Furthermore, let us denote the prove and verify algorithms of these proof systems by $\mathsf{P}X.\mathsf{Prove}$ and $\mathsf{P}X.\mathsf{Vfy}$, for $1 \leq X \leq 3$. We make use of an SXDH-based instantiation of Groth-Sahai proofs [21] for this purpose. Note that GS proofs are not always zero-knowledge, but we ensured that they indeed are for the languages we consider (cf. [22]).

Homomorphic commitments. In order to form a token and commit to secrets including the user secret key and the token balance, we make use of an equivocal F'_{gp}-binding homomorphic commitment scheme C for messages from $\mathbb{Z}^4_p$. The commitment space is G_2 and decommitment values are elements of G_1. The function F'_{gp} maps $m := (m_1, m_2, m_3, m_4)$ 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 she can open a commitment, the corresponding verification equations must be compatible with our proof systems. We use a scheme by Abe et al. [2] for this purpose. We denote the CRS generator, commitment, and opening algorithms by C.Setup, C.Com, and C.Open, respectively. Furthermore, as CAdd and DAdd coincide with the multiplication of commitments and decommitment values, respectively, we denote both operations by "·".

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

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

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

Figure 8: Setup and Key Generation

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 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. hid will be a simple encryption of a user's public key under a public key contained in the CRS. For this purpose, an IND-CPA secure encryption scheme E for messages in G_1 which is compatible with our proof system suffices. This building block can be instantiated with the ElGamal encryption scheme [17]. We denote the corresponding algorithms by E.Gen, E.Enc, and E.Dec.

4.2 Protocol Description

Figures 8 to 11 summarize the scheme. In the following, we elaborate on the details of the different protocols and algorithms.

System and user setup. The setup and key generation algorithms are given in Fig. 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 the hidden user ID hid. The corresponding trapdoor td equals the secret key $\mathsf{sk}_{\mathcal{T}}$ to decrypt hidden IDs. Thus, we have $\mathsf{ExtractUID}(\mathsf{td}, \mathsf{hid}) := \mathsf{E.Dec}(\mathsf{sk}_{\mathcal{T}}, \mathsf{hid})$. 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. $\mathsf{pk}_{\mathcal{I}} := (\mathsf{CRS}, \mathsf{pk}_{\mathsf{sig}})$. 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.

```
$\mathcal{U}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}})$                                  $\mathcal{I}(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}})$
$s', u_1 \leftarrow \mathbb{Z}_p$
$(c', d') := \mathsf{C.Com}(\mathsf{CRS}_{\mathsf{com}},$
      $(s', 0, \mathsf{sk}_{\mathcal{U}}, u_1))$
$x := (c', \mathsf{pk}_{\mathcal{U}})$
$wit := (g_2^{\mathsf{sk}_{\mathcal{U}}}, g_1^{u_1}, g_1^{s'}, d')$
$\pi := \mathsf{P1.Prove}(\mathsf{CRS}_{\mathsf{pok}}, x, wit)$
                              $\xrightarrow{c', \pi}$
                                                  $x := (c', \mathsf{pk}_{\mathcal{U}})$
                                                  if $\mathsf{P1.Vfy}(\mathsf{CRS}_{\mathsf{pok}}, x, \pi) = 0$
                                                        return 0
                                                  $s'' \leftarrow \mathbb{Z}_p$
                                                  $(c'', d'') = \mathsf{C.Com}(\mathsf{CRS}_{\mathsf{com}},$
                                                        $(s'', 0, 0, 0))$
                                                  $c := c' \cdot c''$
                                                  $\sigma = \mathsf{S.Sgn}(\mathsf{sk}_{\mathsf{sig}}, c)$
                              $\xleftarrow{c, d'', \sigma, s''}$
$s := s' + s'' \mod p$
$d := d' \cdot d''$
$\tau := (c, d, \sigma, s, u_1)$
if $\mathsf{UVer}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, \tau, 0) = 0$
      return $(\perp, 0)$
else return $(\tau, 1)$                           return 1
```

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 randomness $u_1 \in \mathbb{Z}_p$ of the double-spending tag, 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 is 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 looks random if the token is used once. The fact that it is bound to the token ensures that double-spending reveals 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' 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

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

The language depends on public parameters such as 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. Note that the second equation in Eq. (7)

```
$\mathsf{UVer}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, \tau, w)$
  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
```

```
$\mathsf{IdentDS}(\mathsf{pk}_{\mathcal{I}}, (s_1, z_1), (s_2, z_2))$
  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 $(\mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}})$
```

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

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

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 decommitment value $d = d' \cdot d''$. The commitment c is then signed by $\mathcal{I}$ resulting in signature σ. The values token c, σ, s'', and d'' are sent over to $\mathcal{U}$ who verifies the correctness of the new token τ 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 $\mathsf{pk}_{\mathcal{U}}$.

```
$\mathcal{U}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, \tau, w, v)$                                   $\mathcal{AC}(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}, v)$
                                                                  $u_2 \leftarrow \mathbb{Z}_p$
                                    $\xleftarrow{u_2}$
parse $(c, d, \sigma, s, u_1) := \tau$
$t := \mathsf{sk}_{\mathcal{U}} u_2 + u_1 \mod p$
$r, s', u_1' \leftarrow \mathbb{Z}_p$
$\mathsf{hid} := \mathsf{E.Enc}(\mathsf{pk}_{\mathcal{T}}, \mathsf{pk}_{\mathcal{U}}; r)$
$(c', d') := \mathsf{C.Com}(\mathsf{CRS}_{\mathsf{com}},$
      $(s', w, \mathsf{sk}_{\mathcal{U}}, u_1'))$
$x := (c', (g_1^s, t, u_2), \mathsf{hid})$
$wit := (c, \sigma, g_1^w, \mathsf{pk}_{\mathcal{U}}, g_1^{u_1}, g_1^d,$
      $g_1^{s'}, g_1^{u_1'}, g_1^{d'}, \mathsf{sk}_{\mathcal{U}}, u_1, r)$
$\pi = \mathsf{P2.Prove}(\mathsf{CRS}_{\mathsf{pok}}, x, wit)$
                                    $\xrightarrow{c', s, t, \pi, \mathsf{hid}}$
                                                                  $z := (t, u_2)$
                                                                  $\mathsf{dstag} := (s, z)$
                                                                  $x := (c', (g_1^s, t, u_2), \mathsf{hid})$
                                                                  if $\mathsf{P2.Vfy}(\mathsf{CRS}_{\mathsf{pok}}, x, \pi) = 0$
                                                                        return $(\perp, \perp, 0)$
                                                                  $s'' \leftarrow \mathbb{Z}_p$
                                                                  $(c'', d'') := \mathsf{C.Com}(\mathsf{CRS}_{\mathsf{com}},$
                                                                        $(s'', v, 0, 0))$
                                                                  $c^* := c' \cdot c''$
                                                                  $\sigma^* = \mathsf{S.Sgn}(\mathsf{sk}_{\mathsf{sig}}, c^*)$
                                    $\xleftarrow{c^*, d'', \sigma^*, s''}$
$s^* := s' + s'' \mod p$
$d^* := d' \cdot d''$
$w^* := w + v$
$u_1^* := u_1'$
$\tau^* := (c^*, d^*, \sigma^*, s^*, u_1^*)$
if $\mathsf{UVer}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, \tau^*, w^*) = 0$
      return $(\perp, 0)$
else return$(\tau^*, 1)$                                          return $(\mathsf{dstag}, \mathsf{hid}, 1)$
```

Figure 11: Accumulation protocol

Collecting points. The Accum protocol is depicted in Fig. 11. Given his current token $\tau = (c, d, \sigma, s, u_1)$, $\mathcal{U}$ receives a random challenge u_2 from $\mathcal{AC}$ to prepare the t-part of the double-spending tag as $t = \mathsf{sk}_{\mathcal{U}} u_2 + u_1$ for this token. Moreover, $\mathcal{U}$ prepares the generation of a fresh token just like in the Issue protocol. To this 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. 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: c is a signed commitment; c' is just a "new version" of this commitment containing the same balance w and user secret key $\mathsf{sk}_{\mathcal{U}}$; t contains $\mathsf{sk}_{\mathcal{U}}$ and the user randomness u_1 (fixed in c) as well as the accumulator's 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 language $L^{(2)}_{\mathsf{pk}_{\mathcal{I}}}$ defined by

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

Upon receiving c', s, t, 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^* along with the accumulator's half of the decommitment 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

[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.

more precisely $W := g_1^w$) is not a witness but part of the statement x in the language $L^{(3)}_{\mathsf{pk}_{\mathcal{I}}}$ defined by

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

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

Double-spending detection. The IdentDS and VerifyGuilt algorithms are given in Fig. 10. To see why IdentDS is working, observe the following: Firstly, to use a specific token τ in an Accum or Vfy protocol run, a (potentially malicious) user is forced to reveal the fixed token version number s, because s is bound to τ by being contained in the signed commitment c. The proof π asserts that this is indeed the case even if c and σ are not revealed explicitly. Secondly, for a specific token τ the commitment c and the proof π also enforce that the fixed user secret key $\mathsf{sk}_{\mathcal{U}}$ and the fixed randomness u_1 of the double-spending tag as well as a freshly chosen challenge value u_2 are always used in a Accum or Vfy protocol run to calculate $t = \mathsf{sk}_{\mathcal{U}} u_2 + u_1$. Hence, double-spending a token reveals 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 $t = \mathsf{sk}_{\mathcal{U}} u_2 + u_1$ and $t' = \mathsf{sk}_{\mathcal{U}} u_2' + u_1$. 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]

5 SECURITY AND PRIVACY THEOREMS

The following theorems state that the instantiation described in the previous section fulfills the security and privacy definitions given in Sections 3.3 and 3.4, provided that all the building blocks are secure. For a formal definition of our building blocks and their properties as well as complete proofs of Theorems 5.1 to 5.7, we refer the reader to the full version of our paper [22]. In the following we only formally state the theorems and point out the main proof ideas.

5.1 System Security

Theorem 5.1 (Trapdoor-Linkability). *If* BBAP *and* E *are correct, and* P2 *and* P3 *are perfectly sound, then* BBAP *is trapdoor-linkable.*

Recall that trapdoor-linkability demands completeness and extractability. The completeness property is satisfied since the soundness of P2 (P3) ensures that any hid which is the output of $\mathcal{AC}$ ($\mathcal{V}$) is a proper encryption of some valid public key $\mathsf{pk}_{\mathcal{U}}$. As BBAP is correct, an honest user holding $\mathsf{pk}_{\mathcal{U}}$ could have created a token and afterwards used this token which resulted in the hidden user ID hid (assuming appropriate encryption randomness). Extractability is satisfied, provided that E is correct, since for an honest user, hid will always be the encryption of his public key $\mathsf{pk}_{\mathcal{U}}$.

Theorem 5.2 (Owner-Binding wrt. Issue). *If the Co-CDH assumption holds and* P1 *is perfectly* $F^{(1)}_{\mathrm{gp}}$*-extractable, then* BBAP *is owner-binding with respect to* Issue*.*

The extractability property of P1 asserts that $g_2^{\mathsf{sk}_{\mathcal{U}}}$ can be extracted from the proof given during the protocol. This value is a solution to the Co-CDH instance $g_1, g_2, g_1^{\mathsf{sk}_{\mathcal{U}}} = \mathsf{pk}_{\mathcal{U}}$.

Theorem 5.3 (Owner-Binding wrt. Accum and Vfy). *If* P1, P2, P3 *are perfectly* $F^{(1)}_{\mathrm{gp}}$*-,* $F^{(2)}_{\mathrm{gp}}$*-, and* $F^{(3)}_{\mathrm{gp}}$*-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*.*

Consider the first call to MalAcc or MalVer that fulfills the winning condition of the experiment. The soundness of the NIZKs, the F'_{gp}-binding of C and the correctness of E ensure that the public key $\mathsf{pk}_{\mathcal{U}}$ extracted from the corresponding NIZK proof and the key $\mathsf{pk}'_{\mathcal{U}}$ extracted by ExtractUID from the corresponding hid token are well-defined and identical. Now, distinguish two cases with respect to the commitment c (which fixes $\mathsf{pk}_{\mathcal{U}}$) extracted from the proof: a) either c is a fresh commitment or b) it is a replayed commitment from a previous protocol invocation. Case a) is an immediate violation of the EUF-CMA security of the signature scheme, as only signed commitments can lead to acceptable protocol executions. Case b) implies that the adversary was able to equivocate an old commitment from some $\widehat{\mathsf{pk}}_{\mathcal{U}}$ to $\mathsf{pk}_{\mathcal{U}}$. This contradicts the F'_{gp}-binding property of the commitment (after exploitation of the soundness of the NIZKs).

Theorem 5.4 (Double-Spending Detection). *If* P1, P2, P3 *are perfectly* $F^{(1)}_{\mathrm{gp}}$*-,* $F^{(2)}_{\mathrm{gp}}$*-, and* $F^{(3)}_{\mathrm{gp}}$*-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.*

The proof considers each of the three possible winning conditions separately. The last winning condition can be immediately ruled out by our protocol definition.

If the first winning condition holds, there are two double-spending tags with the same token version number s but different user public keys $\mathsf{pk}^{(1)}_{\mathcal{U}}$ and $\mathsf{pk}^{(2)}_{\mathcal{U}}$. Consider the commitments c_1 and c_2 that can be extracted from the proofs sent by the adversary in each of the corresponding protocol executions. If $c_1 = c_2$ holds, this violates the F'_{gp}-binding property of C as the adversary is able to equivoke the same commitment to two different user IDs. If $c_1 \neq c_2$ holds, two sub-cases needs to be considered. If at least one of the commitments is a fresh commitment, this immediately contradicts the EUF-CMA security of S by the same argument as in Theorem 5.3. If both commitments are replayed commitments of former protocol invocations, they are already associated to some version numbers s_1 and s_2, respectively. As version numbers are uniformly drawn from $\mathbb{Z}_p$ in each protocol invocation, $s_i = s$ only holds with negligible probability. (Here, the Blum-like coin tossing requires the homomorphism of C.) Hence, the adversary is able to equivoke the commitments to the winning version number s which again contradicts the F'_{gp}-binding property.

If the second winning condition holds, ExtractUID extracts $\mathsf{pk}^{(1)}_{\mathcal{U}} = \mathsf{pk}^{(2)}_{\mathcal{U}}$ from the protocol invocations but IdentDS returns a different value (or $\perp$) given $\mathrm{dstag}_1 = (s, (t, u_2))$ and $\mathrm{dstag}_2 = (s, (t', u_2'))$.

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

From the extractability of the NIZKs and the correctness of E, it follows that t and t' have been computed using the same $\mathsf{sk}_{\mathcal{U}}$. Hence, considering $\mathsf{IdentDS}$, either $u_2 = u_2'$ or $u_1 \neq u_1'$. The first case only occurs with negligible probability. The second case implies that that either both corresponding commitments (fixing u_1 and u_1') are equal or not. Thus, again either the F_{gp}'-binding property of C is violated or the EUF-CMA security of S.

Theorem 5.5 (Balance-Binding). *If* $\mathsf{P1}$, $\mathsf{P2}$, $\mathsf{P3}$ *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 balance-binding.*

The proof proceeds in a sequence of game hops from the real game to a modified game where the adversary always loses by definition of the game. We show that if any two consecutive games would significantly differ, this results in adversaries against the security of the building blocks. The high-level idea is to augment each game by additional sanity checks between the adversary's individual oracle calls to $\mathsf{MalIssue}$, MalAcc or MalVer. These checks transform the original game into one which asserts that (1) an adversary cannot make the experiment miscount the balance associated with user ID $\mathsf{pk}_{\mathcal{U}}$ and (2) an adversary cannot overclaim or underclaim this computed balance. The technical challenge is to properly formalize these sanity checks. To this end, a directed graph is introduced, where a path links the individual protocol executions that belong to the same user. The checks assert that the adversary cannot deviate from the correct balance along each path.

5.2 User Security and Privacy

Theorem 5.6. *If* $\mathsf{P1}$, $\mathsf{P2}$, $\mathsf{P3}$ *are composable zero-knowledge,* C *is equivocable and* E *is IND-CPA secure, then* BBAP *is privacy-preserving.*

This proof proceeds in a sequence of game hops that gradually transform the real game into the ideal game in which no privacy infringement can occur as all message are independent of user-specific data. The game hops are as follows: a) replace the honest CRS by a CRS that allows simulation of NIZK proofs and equivocation of commitments, b) replace all proofs by simulated proofs, c) replace commitments by fake commitments, d) choose a random value instead of an honest double-spending tag, e) replace the hid by an encryption of g_1^0. The complete proof has to deal with two technical subtleties: a) The subroutine UVer needs to be re-defined properly. If the commitments are replaced by fake commitments the unmodified version always fails and thus an honest party playing the user aborts. However, the check cannot just be skipped entirely as otherwise the adversary could just send garbage, check if the user aborts and thus easily distinguish between real and ideal. b) The experiment needs to keep track of all interactions such that it can come up with a correct balance upon corruption. To this end, the oracles need some sort of shared state.

Theorem 5.7. *If* BBAP *is privacy-preserving and the calculation of discrete logarithms in* G_1 *is a computationally hard problem, then* BBAP *ensures false accusation protection.*

The oracles which the adversary is allowed to use are a subset of the oracles of the privacy game. Assume there is an efficient adversary that wins the false-accusation game with non-negligible probability. Replace all oracles by their ideal counterparts using the same techniques as in the privacy proof. We distinguish two cases: a) The adversary still outputs $\mathsf{sk}_{\mathcal{U}}$ with non-negligible probability. This contradicts the DL assumption in G_1. b) The adversary does not output the correct $\mathsf{sk}_{\mathcal{U}}$. This implies the existence of an adversary who can distinguish between the real and the ideal privacy game.

Table 1: User execution times for our instantiation

Algorithm	Execution Time [ms]	Data Sent [Bytes]	Data Received [Bytes]
Issue	115.27	672	320
Acc	385.61	3728	320
Vrfy	375.73	3664	320

6 PERFORMANCE EVALUATION

We evaluate the performance of our BBA+ instantiation by measuring execution times of the BBA+ protocols using a practical implementation. To this end, network payload and execution time on the user's device is measured. We selected the smartphone as a target platform suitable for mobile applications, since it has become a familiar companion in everyday life to the majority of potential users. An additional benefit of this platform is that a developer using our scheme does not have to distribute any new hardware and users are already acquainted with their device.

The issuer's, accumulator's or verifier's performance is not measured, as we expect their hardware to be much more powerful. Not included in our measurements are data-transmission times, since they depend on external factors not influenced by BBA+. However, we provide estimations based on the prevalent transmission technology NFC.

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++14 using the RELIC toolkit v.0.4.1, an open source library with support for pairing-friendly elliptic curves 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 $\mathsf{Fp254BNb}$ and $\mathsf{Fp254n2BNb}$ presented by Aranha et al. [7, 26].

We select the optimal Ate pairing as RELIC's pairing function since current speed records are achieved using this function [29].

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

6.2 Implementation Results

Table 1 shows the average execution times for the respective BBA+ 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. The size of a data packet could further be reduced using compression algorithms. Note that these are average execution times measured on a smartphone during regular use. The operating system's scheduler interferes with the computations and thus single protocol execution times may vary.

As a reference value for an acceptable execution time, we consider one second to be a reasonable upper bound. All protocols of BBA+ execute in less than 400ms on the user's side. If we use, for example, NFC with its maximum transmission speed of 424 kbit/s, it would take less than 80ms to transmit all data for any of the protocols from/to the communication terminal. This leaves more than 500ms to transmit data packets over the network to the protocol partner, have him compute his part of the protocol and respond to the user. Assuming that the issuer, accumulator, and verifier run a powerful back-end, it should not be challenging to execute an entire protocol run in less than a second.

6.3 Further Optimizations

The RELIC toolkit is a multi-purpose library which is not mainly optimized for pairing-based elliptic curve cryptography. Hence, we might be able speed up our implementation by creating a dedicated library focused on a highly optimized implementation of a pairing-friendly elliptic curve.

Regarding the Groth-Sahai proof system, we expect that a user could significantly reduce his computations for generating a GS proof by carefully applying the prover chosen CRS technique from [18]. Moreover, there are ways to optimize the efficiency of the verifier of a GS proof considerably. For instance, Herold et al. [23] recently proposed a batch verification technique dedicated to Groth-Sahai proofs. Integrating these optimizations into our BBA+ instantion, could notably reduce the computational complexity of the issuer, accumulator, and verifier.

7 APPLYING BBA+

In the following, we sketch important aspects when applying BBA+ in some selected scenarios. From a high-level perspective, applying BBA+ to these scenarios seems mostly straightforward. Nonetheless, there are some technical subtleties that needs to taken into account, including:

- To guarantee security and privacy, the CRS needs to be set up by a party which is trusted by both the operator (issuer, accumulator, verifier) and the user.
- As the parties $\mathcal{I}$, $\mathcal{AC}$, and $\mathcal{V}$ all share the same secret key, they need to trust each other (in particular, not to collude with a malicious user).
- The values $|v|$ being collected in a specific scenario need to be upper bounded in a way such that no balance "wraparounds" (in $\mathbb{Z}_p$) occur during the lifetime of a token. Otherwise, a user could wrongfully lose points. He could also gain points if negative balances are allowed (e.g., when considering a balance space $\mathbb{Z}_p$ represented by $\{-\frac{p-1}{2}, \dots, 0, \dots, \frac{p-1}{2}\}$).
- It needs to be ensured that the application meets the restrictions imposed by the BBA+ security model. In particular, user registration needs to be done out-of-band: The application needs to verify and store the (physical) identity of a user and make sure his chosen public-key $\mathsf{pk}_{\mathcal{U}}$ is unique. Furthermore, it must be ensured that that only a single token is issued per $\mathsf{pk}_{\mathcal{U}}$. If for the same (natural) user more than one token is required, then he may register multiple $\mathsf{pk}_{\mathcal{U}}$—one for each token. Management of (natural) users and association of each $\mathsf{pk}_{\mathcal{U}}$ with a physical identity is out of the scope of the BBA+ scheme.
- Users need to be forced to actually run the Accum or Vfy protocol if the value v to be added is negative. How this can be accomplished is application-specific. Remember, if an application ensures this, the user is bound to continue with the updated token in the next interaction and cannot re-use an older version with a higher balance due to double-spending detection.

7.1 Customer Loyalty Systems

As a basic application, we outline how BBA+ can be used to create a privacy-preserving loyalty system for customer retention. When the operator initiates a loyalty program, he asks a trusted third party (e.g., the Federal Commissioner for Data Protection and Freedom of Information) to generate a CRS by running Setup. The trusted third party then publishes the CRS and securely stores the trapdoor.

When the CRS has been created, the operator may generate a key pair $(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}) \leftarrow \mathsf{IGen}(\mathsf{CRS})$ and publish $\mathsf{pk}_{\mathcal{I}}$. Before a customer is allowed to participate in the loyalty program, he must register himself. The operator verifies the (physical) identity of the user, ensures that $\mathsf{pk}_{\mathcal{U}}$ is unique and stores all information in some kind of CRM system. Then the user and the operator execute the Issue protocol to create a token for the user.

When the customer purchases some products, he executes the $\mathcal{AC}$ protocol with the operator (e.g., represented by the cash register), where v is the number of points that the customer receives.

If the customer wants to redeem some points (say, $v' \in \mathbb{N}$) to obtain a voucher, he unveils his current balance w to the operator (represented by some token remdemption machine), who checks that $w \geq v'$. The parties then execute the Vfy protocol with $v := -v' = p - v' \bmod p$. Note that there is no need to force the customer to execute Vfy with a negative v, as he has an incentive to do so (obtaining a voucher).

Using a balance space $\mathbb{Z}_p = \{0, \dots, p-1\}$, balance wraparounds are not an issue in this scenario: First, we avoid wraparounds that could be caused by subtracting values by only executing Vfy if $w \geq v$. Second, adding up positive points will not result in a wraparound neither. To achieve a reasonable level of security, p is in the order of 2^{250} (cf. Section 6.1) while a typical loyalty program grants one point per expensed dollar. Hence, a user needs to spent about \$ 10^{75} before a wraparound occurs.

In order to detect double-spending of collected points, the operator regularly scans his database for double-spending tags with identical token version numbers. If there are some, he runs $\mathsf{IdentDS}$ 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.

7.2 Micropayments in Vehicle-to-Grid

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 be able to revert to this energy when the natural supply is low, e. g. at night. One approach for storing and resupplying 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, BBA+ can be used to realize the transfer of money. Here, an electricity provider may take the role of the issuer, accumulator, and verifier, respectively. The car owner (or his car, acting autonomously) is a user. Depending on the supply of energy in the net and the charging level of the car's battery, the car autonomously sells or buys energy in exchange for points.

This raises the question how the user can be forced to actually run the Vfy protocol in order to lose points when charging his battery. The simplest solution is to let the user prepay before any charging takes place. A drawback of this approach is that a user would lose money if he wants to leave before the charging process is completed. For a more comfortable solution, this approach can be augmented by a subsequent refund step (using Accum) if the prepaid amount is not used up when leaving. This allows the user to overbuy and gain more flexibility. Finally, our specific implementation enables a third solution preventing prepayments. The Vfy protocol can be interleaved with the actual charging of energy. Before the charging starts, the Vfy protocol is executed until the message from the user to the operator has been sent. This ensures that the operator has already learned a double-spending tag for the current token. Please note, that the final value v that must be paid (which is unknown by now) is not yet required by the protocol. This input can be postponed. After the charging terminates, v is calculated and the Vfy protocol is resumed. If a user decides to "run away" without paying and does not finish the Vfy protocol run, he will be detected during the next interaction. This is because he has not received a fresh version of the token, but must reuse the old version a second time.

When the points collected by the car are exhausted, the user may buy additional points (using cash, bank transfer or another commonplace payment method) to recharge his token. Likewise, when the car has accumulated a large number of points, the car owner may redeem the points collected by his car, in order to get paid for the electricity his car has provided.

8 FUTURE EXTENSIONS

In this section we sketch how BBA+ could be extended by range proofs and also point out future work regarding active adversaries.

8.1 Range Proofs

There are a variety of applications where it might be desirable not to reveal the current balance w during the verification and redemption protocol. To overcome this issue, the Vfy protocol could be extended by a range proof system such as [10] or [14]. Although there has been great progress to increase the efficiency of those proof systems, we deliberately did not include one in our basic scheme, as even recent range proofs are still computationally expensive. So they may considerably slow down the execution on low-end hardware like mobile devices. Nonetheless, as for certain scenarios this privacy/efficiency tradeoff might be worthwhile, we sketch how range proofs could be integrated into our instantiation.

8.1.1 High Level Overview. In the following, we explain the idea of the range proof in [10]. Firstly, let us recap the trivial approach to prove that a balance w is at least the redeemed value v, i. e. $w \in \mathcal{S} := \{v, \dots, N_{\max}\}$ with $N_{\max}$ being an upper bound on the balance space. Here, the verifier would generate a signature on every element of $\mathcal{S}$ and the prover would prove in ZK that it knows a signature on its balance w. Obviously, this approach is prohibitive if $\mathcal{S}$ grows proportional to the underlying group $\mathbb{Z}_p$ as this yields an exponentially large set for $\log p \in \Theta(n)$.

In [10] Camenisch et al. exploit a q-ary representation of the secret with at most $\eta_{\max}$ digits to overcome this problem. Here, $q, \eta_{\max} \in \mathbb{N}$ are design parameters that are chosen during system setup such that $N_{\max} := q^{\eta_{\max}} - 1 < p$. For a fixed q, the maximal admissible number of digits $\eta_{\max}$ to represent a value is bounded by $\eta_{\max} \leq \lfloor \log_q p \rfloor \in \mathcal{O}(n)$. Assume the prover wants to prove that a secret $x \in \mathbb{Z}_p$ is contained in the range $\{0, \dots, q^{\eta} - 1\}$ for some $\eta \leq \eta_{\max}$. The verifier needs to generate only one signature on each element in $\{0, \dots, q - 1\}$, the prover generates a q-ary representation of the secret $x = \sum_{j=0}^{\eta-1} x_j q^j$ and then proves for each digit x_j ($j \in \{0, \dots, \eta - 1\}$) that the digit is contained in the set $\{0, \dots, q - 1\}$, 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. Depending on the tangible choice of q and $\eta_{\max}$ there are $p - N_{\max} + 1$ elements of the underlying group that cannot be represented. 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.

Moreover, the basic range proof only allows to show that a secret x can be represented with $\eta \leq \eta_{\max}$ digits, i. e. that $x \in \{0, \dots, q^{\eta} - 1\}$ holds. But we need to prove a statement $w \in \{v, \dots, N_{\max}\}$ about a secret w and usually neither interval limit is located at a q-power. In order to overcome this issue the prover conducts two range proofs about a suitable shifted value 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 Section 8.1.3.

8.1.2 Design Choices and Notation. The efficiency of range proofs heavily depends on the representation of the elements with individual digits. This 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.

We fix the representation $\mathbb{Z}_p = \{0, \dots, p - 1\} \subset \mathbb{N}$, i. e. we interpret elements of $\mathbb{Z}_p$ as *positive* numbers with the usual $\leq$-order inherited from $\mathbb{N}$. This means we keep the protocols Accum and Vfy separated and only extend the latter by a range proof. Moreover, for the ease of notation we denote the first $\eta_{\max}$ q-powers of g_1 by $Q_j := g_1^{(q^j)}$ for $j = 0, \dots, \eta_{\max} - 1$. These public constants are an $F_{\mathrm{gp}}^{(3)}$-mapping of all relevant magnitudes of the positional digit system. The system constants q, $\eta_{\max}$, $N_{\max}$ are included in the CRS.

8.1.3 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 $\eta := \lfloor \log_q(N_{\max} - v) \rfloor + 1$ and $N := q^{\eta} - 1$, i. e. $N + 1$ is the smallest q-power greater than $N_{\max} - v$. It follows

$$\begin{aligned}
&w \in \{v, \ldots, N_{\max}\} \\
\Leftrightarrow\ & N_{\max} - w \in \{0, \ldots, N_{\max} - v\} \\
\Leftrightarrow\ & N_{\max} - w \in \{0, \ldots, N\} \cap \{N_{\max} - v - N, \ldots, N_{\max} - v\} \\
\Leftrightarrow\ & \begin{cases} N_{\max} - w \in \{0, \ldots, N\} \ \wedge \\ N + v - w \in \{0, \ldots, N\} \end{cases} \\
\Leftrightarrow\ & \begin{cases} \exists\, w'_0, \ldots, w'_{\eta-1} \in \{0, \ldots, q-1\} : N_{\max} - w = \sum_{j=0}^{\eta-1} w'_j q^j \\ \\ \exists\, w''_0, \ldots, w''_{\eta-1} \in \{0, \ldots, q-1\} : N + v - w = \sum_{j=0}^{\eta-1} w''_j q^j \end{cases}
\end{aligned} \tag{10}$$

In our instantiation of the BBA+ scheme, openings of commitments are elements from the implicit message space G_1. For this reason, w does not directly become part of the witness but an $F_{\mathrm{gp}}^{(3)}$-mapping $W = g_1^{w}$ and Eq. (10) translates into a statement about group elements. For an $F_{\mathrm{gp}}^{(3)}$-mapped balance $W \in G_1$ the user proves

$$\exists\, w'_0, \ldots, w'_{\eta-1} \in \mathbb{Z}_p : W \prod_{j=0}^{\eta-1} Q_j^{w'_j} = g_1^{N_{\max}} \tag{11}$$

$$\exists\, w''_0, \ldots, w''_{\eta-1} \in \mathbb{Z}_p : W \prod_{j=0}^{\eta-1} Q_j^{w''_j} = g_1^{N+v}. \tag{12}$$

These are multi-scalar multiplication equations (MSEs) and therefore fit into our Groth-Sahai proof system.[12]

Note, that in contrast to Eq. (10) the Eqs. (11) and (12) do not assert that w'_j, $w''_j \in \{0, \ldots, q-1\}$. Hence, the user must additionally prove that w'_j, w''_j are indeed valid digits. For each digit $i \in \{0, \ldots, q-1\}$ let $\sigma_i := \mathsf{S.Sgn}(\mathsf{sk}_{\mathrm{sig}}, g_2^{i})$ be a corresponding signature using signature scheme S. Then the user proves

$$\begin{aligned}
&\forall\, j \in \{0, \ldots, \eta - 1\} : \\
&\mathsf{S.Vfy}(\mathsf{pk}_{\mathrm{sig}}, g_2^{w'_j}, \sigma_{w'_j}) = 1 \quad \wedge \quad \mathsf{S.Vfy}(\mathsf{pk}_{\mathrm{sig}}, g_2^{w''_j}, \sigma_{w''_j}) = 1.
\end{aligned} \tag{13}$$

in order to show that it knows a valid signature for each digit. These expand into two power-product equations (PPEs) each, using the signature scheme from [1]. In summary, including a range proof into the BBA+ scheme increases the NIZK of the Vfy protocol by 2 MSEs for correctness of representation and 4η PPEs for correctness of the digits.

This concludes our description. We leave it as future work to implement such a range proof and evaluate the absolute runtime penalty for concrete choices of q and $\eta_{\max}$. Moreover, the security model and proofs must be adopted to the new setting in order to formally reestablish the security guarantees.

8.2 Fully Active Adversaries

In the security model described in Section 3.3, we consider adversaries that may arbitrarily deviate from the protocol. In the full version of the paper [22], we give a full-fledged security model, where the adversary may additionally eavesdrop on protocol executions of honest users in several security experiments.

However, even in our full-fledged model, the adversary does not control the network and therefore cannot tamper with the messages of honest parties but is restricted to passively eavesdrop. In other words, man-in-the-middle attacks are excluded by the model. While we believe this is a realistic model for several of the applications mentioned before, where the user and the operator (issuer, accumulator, and verifier) are in direct contact with each other, it may be insufficient if communication is done over a wide-area network.

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

ACKNOWLEDGMENTS

We would like to thank Jessica Koch and Valerie Fetzer for fruitful discussions on black-box accumulation and their comments on earlier version of this paper. Moreover, we also would like to thank the anonymous ACM CCS reviewers for their constructive comments.

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] Foteini Baldimtsi, Melissa Chase, Georg Fuchsbauer, and Markulf Kohlweiss. 2015. Anonymous Transferable E-Cash. In *Public-Key Cryptography - PKC 2015 - 18th IACR International Conference on Practice and Theory in Public-Key Cryptography, Gaithersburg, MD, USA, March 30 - April 1, 2015, Proceedings (Lecture Notes in Computer Science)*, Jonathan Katz (Ed.), Vol. 9020. Springer, 101–124. https://doi.org/10.1007/978-3-662-46447-2_5

[7] 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.

[8] 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.

[12] Groth-Sahai distinguish between different types of equations that are supported by their proof system.

[9] 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.

[10] 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.

[11] 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.

[12] Jan Camenisch, Susan Hohenberger, and Anna Lysyanskaya. 2005. Compact E-Cash. In *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings (Lecture Notes in Computer Science)*, Ronald Cramer (Ed.), Vol. 3494. Springer, 302–321. https://doi.org/10.1007/11426639_18

[13] Sébastien Canard and Aline Gouget. 2008. Anonymity in Transferable E-cash. In *Applied Cryptography and Network Security, 6th International Conference, ACNS 2008, New York, NY, USA, June 3-6, 2008. Proceedings (Lecture Notes in Computer Science)*, Steven M. Bellovin, Rosario Gennaro, Angelos D. Keromytis, and Moti Yung (Eds.), Vol. 5037. 207–223. https://doi.org/10.1007/978-3-540-68914-0_13

[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] Delphine Christin, Andreas Reinhardt, Salil S. Kanhere, and Matthias Hollick. 2011. A survey on privacy in mobile participatory sensing applications. *Journal of Systems and Software* 84, 11 (2011), 1928–1946.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] Gunnar Hartung, Max Hoffmann, Matthias Nagel, and Andy Rupp. 2017. BBA+: Improving the Security and Applicability of Privacy-Preserving Point Collection (full paper). (2017). http://homepage.rub.de/andy.rupp/papers/bbap_full_paper.pdf.

[23] Gottfried Herold, Max Hoffmann, Michael Klooß, Carla Ràfols, and Andy Rupp. 2017. New Techniques for Structural Batch Verification in Bilinear Groups with Applications to Groth-Sahai Proofs. Cryptology ePrint Archive, Report 2017/802. (2017). http://eprint.iacr.org/2017/802.

[24] 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.

[25] 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.

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

[27] Willett Kempton and Jasna Tomic. 2005. Vehicle-to-grid power fundamentals: Calculating capacity and net revenue. *Elsevier Journal of Power Sources* 144, 1 (2005), 268–279.

[28] 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.

[29] 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.

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

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

[32] 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.

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