Device-Bound Anonymous Credentials With(out) Trusted Hardware

Karla Friedrichs[1], Franklin Harding[2] *, Anja Lehmann[1], and Anna Lysyanskaya[2]

[1] Hasso Plattner Institute, University of Potsdam, Germany
{karla.friedrichs,anja.lehmann}@hpi.de
[2] Brown University, USA {franklin_harding,anna_lysyanskaya}@brown.edu

Abstract. Anonymous Credentials enable privacy-preserving authentication. To ensure non-transferability of credentials among corrupt users, they can additionally be *device-bound*. Therein, a credential is tied to a key protected by a *secure element* (SE), usually a hardware component, and any presentation of the credential requires a fresh contribution of the SE. Despite being a fundamental concern of user credentials, device binding for Anonymous Credentials is relatively unstudied. Existing constructions either require multiple calls to the SE, or need the SE to keep state – violating the design principles of resource-limited SEs. Further, constructions that are compatible with the most mature credential scheme BBS rely on the honesty of the SE for privacy, which is hard to vet given that SEs are black-box components. In this work, we thoroughly study *Device-Bound Anonymous Credentials* (DBACs). We model DBACs to ensure not only unforgeability and non-transferability of credentials, but also user privacy, even when the SE is subverted or fully corrupted. We also define *blind* DBACs, in which the SE learns nothing about the credential presentations it helped compute. This targets the design of a remote, cloud-based SE which is a deployment model considered for the EU Digital Identity wallet. Finally, we present three simple and round-optimal constructions for device binding of BBS credentials, prove their security in the AGM+ROM, and privacy unconditionally. The SE remains extremely lightweight, computing only a single BLS or Schnorr signature. A blind variant of the BLS-based construction yields the first protocol to enable privacy-preserving device binding for Anonymous Credentials when used with a remote SE.

1 Introduction

Anonymous Credentials (ACs) are the privacy-preserving version of attribute-based certificates. The user gets a credential from a trusted issuer, asserting a set of user attributes, which can be shown to a verifier in an unlinkable manner that reveals nothing besides the relevant predicate over the attributes. ACs are well-studied; the first efficient schemes were proposed more than 20 years ago [?,?,?].

* Paper approved as a Master's report for this author.

Due to several digital identity projects around the globe and standardization efforts in the IETF and W3C [?,?], ACs have recently received renewed attention. For instance, the so-called European Digital Identity (EUDI) wallet envisions strong and privacy-preserving user authentication for EU citizens through a credential-based smartphone application. The corresponding regulation [?] mandates that the technical solution must satisfy unlinkability, untraceability and selective disclosure – making ACs the perfect choice [?].

Besides privacy, an important concern of the regulation is that credentials be tied to individuals in a way that prevents their cloning or transfer. As a result, the technical bodies tasked with realizing the EUDI vision have emphasized the need for binding credentials to uncloneable keys via *device binding* (e.g., [?]).

Device Binding. In a *device-bound* credential system, the user device, such as a laptop or a smartphone, is assumed to contain a special secure element (SE). This SE guarantees that "jail-breaking", or copying cryptographic keys, is hard; most modern devices today come with such hardware. To issue a credential, the issuer signs the device-protected public key along with the list of user attributes. In secure elements currently on the market, this key is the public key of a standard signature scheme; each presentation of such a device-bound credential also requires a fresh proof of knowledge of the device secret key. The motivation behind involving the secure element in the user's credential presentation is twofold: first, accidental (or malicious) exposure of a user Alice's credentials prevents an attacker from impersonating her. Second, the use of the SE makes it impossible for a malicious Alice to let her friend Bob copy and make use of her credentials.

Lack of Appropriate Device Binding for AC. Device binding is a natural approach to non-transferability of credentials, and the first notable exploration in this vein was the study of direct anonymous attestation (DAA) [?]. In DAA, the user consists of an untrusted host machine and a trusted-platform module (TPM) responsible for non-transferability. The TPM2.0 specification implements an elegant design: it provides hardware interfaces to a two-round Schnorr signing protocol, from which device binding for several credential schemes, such as BBS [?,?], CL [?,?], PS [?] or U-Prove [?] can be bootstrapped [?].

The DAA line of work has two drawbacks: first, the only sound security models [?,?] are in the Universal Composability (UC) model [?]. They are highly complex and cover the entire DAA application, such that neither the formal model nor the device binding aspect can easily be re-used for credential systems. Second, DAA needs two TPM calls per credential, and requires the TPM to keep state in between. This is incompatible with the design principle that interactions with SEs should be isolated and monolithic. Additionally, the employed TPM interfaces are *not* available on SEs within user phones.

For realizing device binding for AC with a single call to the SE, Hesse et al. [?] investigate device binding for offline presentation, where the user's phone is used to authenticate towards a physical terminal. This work has similar drawbacks to DAA: the security model is in the UC framework (making it too tailored to the application) and the protocol now requires the SE and host device to share a

credential-specific state, which is unrealistic when the SE is expected to be used for multiple user credentials. The work of Hanzlik and Slamanig [?] defines ACs with a core-helper-structure, but their proposed protocol requires a new class of structure-preserving primitives, and is not compatible with BBS credentials.

Recently, Orange Innovation [?] proposes new ideas for device binding for BBS credentials, where the secure hardware only computes standard Schnorr or ECDSA signatures (although over a pairing-friendly curve), but the paper lacks a formal model and the provided security proofs appear to have some gaps. Other recent ideas for device binding include converting non-anonymous credentials that use device binding into anonymous variants [?,?]. Those works are tailored towards secure elements based on ECDSA, and are using more-or-less generic zero-knowledge proof systems for proving possession of credentials, rather than relying on more efficient anonymous credential techniques. Moreover, they do not consider the additional privacy risks associated with using a potentially untrustworthy SE. In particular, these schemes would allow an adversary that controls both the SE and the verifier to deanonymize the user.

In conclusion, for SE architectures with a simple query-response structure, no provably-secure version of device binding for established anonymous credential schemes such as the BBS family [?,?,?,?] exists.

In Hardware We Trust? All aforementioned works put *full* trust into the SE, with just one exception [?]. By design, SEs are shielded, essentially black-box components that cannot be inspected and analyzed publicly. While upholding unforgeability assurances in this way is in the interest of the manufacturer, the effect on privacy might be opposite. Manufacturers or nation-state adversaries could tamper with the design in a way that leaves unforgeability intact but introduces subliminal channels in order to track the users' actions through the computations executed by their devices. Similarly, if the credential is visible for the SE, the SE could perform a denial of service attack if the user tries to show a credential that the SE does not approve. Thus, relying on the trustworthiness of a black-box hardware component for the privacy of billions of users is certainly not desirable. Users who do not fully trust the SE manufacturer should reject device binding schemes which do not provide sufficient privacy guarantees.

In the context of DAA, guaranteeing privacy in the presence of subverted TPMs has been studied by Camenisch et al. [?] (again, in the UC model). Their protocol makes a non-standard approach for designing the main anonymous credential scheme, and is not compatible with BBS signatures or alike.

Remote Device Binding. While we should not trust hardware components for privacy, relying on them for the non-transferability of credentials is reasonable – with one major caveat: the average smartphone does not have the necessary type of SE, yet! The EUDI regulation [?] demands SEs with the highest level of security certification ("Level-of-Assurance high") which are only included in certain flagship phones. To address this "underequipped-ness", the regulation explicitly allows for a *remote* HSM (Hardware Security Module) [?, Section 4.5.2]:

the remote server acts in place of a local SE; ensuring that it interacts with the correct user via state-of-the-art (non-anonymous) multi-factor authentication.

This approach significantly aggravates the privacy risk. Recall that device binding enforces that the device contributes to *each* presentation of the credential. To tightly bind a device binding proof to a specific presentation, some session nonce and identifier of the targeted service may be included in the information attested by the device. Revealing this presentation context to a central service would void ACs of all unlinkability and unobservability guarantees – so clearly, protocols developed for a local SE cannot simply be copied to a cloud version. Despite these aggravated privacy concerns, which we share fully, we do see value in searching for a better solution for the remote HSM setting: not only do several European member states, e.g., Germany, Netherlands, and Sweden, plan to adopt this option [?,?,?]. But also in the spirit of digital sovereignty there might be value in replacing specialized SE hardware, supplied by few actors world-wide, with a more autonomous infrastructure.

Scheme	1-Round	Stateless	Privacy	BBS	Blind
DAA [?]	✗	✗	Honest SE	✓	✗
DAA+ [?]	✓	✓	Subverted SE	✗	✗
Hanzlik and Slamanig [?]	✓	✓	Honest SE	✗	✗
Hesse et al. [?]	✓	✗	Honest SE	✓	✗
BBS-Schnorr / BBS# [?]	✓	✓	Honest SE	✓	✗
BBS-BLS (this work)	✓	✓	Subverted SE	✓	✗
BBS-BlindBLS (this work)	✓	✓	Malicious SE	✓	✓

Table 1. Comparing AC schemes with some notion of device binding. The "1-Round" column refers to the interaction between the user and SE required to compute a showing. "Stateless" refers to the SE. "BBS" highlights compatibility with the currently developed BBS standard, the most likely candidate for real-world AC deployment.

1.1 Our Contributions

We rigorously define security for device-bound anonymous credentials, capturing different levels of trust in the SE. We then propose instantiations that significantly improve the state of the art for each trust level. All constructions rely on BBS signatures [?,?,?] for the main attribute credentials, and require the SE to only compute a standard Schnorr or BLS [?] signature. This enables smooth integration based on standardized [?,?] cryptographic schemes that the cryptography community stands behind [?]. Table 1 summarizes our results in comparison with prior work. In more detail, our contributions are as follows:

Security Model. We formalize *device-bound anonymous credentials* (DBAC), where credentials are bound to a device public key dpk, and presentations of these

credentials require the assistance of a hardware component that is in possession of the corresponding secret key *dsk*. Recall that in an anonymous credential presentation, a user with attributes $\boldsymbol{a}$ and a predicate ϕ convinces a verifier that $\phi(\boldsymbol{a}) = \mathsf{True}$ without revealing anything else about the attributes; this presentation is tied to a particular context *ctx* that may include information such as the verifier's identity or a session nonce. To cater to different deployment settings, we model two variants of DBAC: *context-aware* and *context-blind* schemes. In the *context-aware* version, the SE learns the presentation context *ctx* with every request, which is suited for settings where the SE is a hardware component run locally by the user. The *context-blind* (BDBAC) scheme targets the remote HSM setting, and does not reveal the context when querying for the SE's contribution to a presentation. We propose a unified security framework that captures the desired *unforgeability* and *unlinkability* requirements of (B)DBAC for different trust assumptions put into the hardware component.

For privacy, we present a hierarchy of four **unlinkability** notions; each of them ensures that a presentation computed by an honest user, from its real credential and device public key *dpk*, is indistinguishable from one computed by a simulator without access to either information:

Unlink-0: The SE and the user's host device are fully honest, and the SE has no communication channels to the outside world except through the device.

Unlink-1: The SE is fully trusted, but not perfectly shielded. An attacker gets temporary access to the SE's interfaces, and this must not impact the privacy of presentations made by the honest user.

Unlink-2: The same as before but the SE algorithms can be subverted, i.e. replaced with arbitrary stateless code that is run within the SE.

Unlink-3: The hardware component is actively under the adversary's control. This is most reasonable for the *remote HSM* setting, where the device binding part is entirely run by an external, and possibly fully malicious entity.

We stress that when employing a remote HSM for a DBAC scheme, the only meaningful notion of privacy is Unlink-3 for the *context-blind* variant. Additionally, any BDBAC scheme must satisfy the privacy property of **obliviousness** (SE-Obliv), meaning that the SE does not learn anything from an interaction with a user. This may also be relevant in the local SE and context-aware setting to prevent denial-of-service attacks targeted to users with selected attributes. Obliviousness is not only needed for privacy, but also strengthens unforgeability and robustness when the SE is corrupt: it ensures that no information about the user's credential or predicate is leaked to the SE.

Unforgeability expresses the necessity of the hardware's contribution to each presentation, in addition to the conventional guarantees, ensuring the unforgeability of attributes and freshness of presentations. Here, the device binding guarantees rather depend on whether we are in the context-blind version or not:

Unf-DBAC: For the context-aware version, our notion guarantees classic unforgeability for the device binding part. That is, for every credential bound to a device key *dpk*, it is infeasible for the adversary to come up with a presentation for a context ctx^* that was never queried to the SE.

UNF-BDBAC: When the device no longer learns the *ctx* it approves, our notion guarantees a mix of classic and one-more unforgeability. For a corrupt user whose *dpk* is protected through an honest hardware component, the best we can hope for is a one-more bound. For the combination of honest user and honest device, however, there is a stronger guarantee: as we know which contexts the honest user intended to create presentations for, any deviation is considered a forgery.

Round-Optimal Device Binding for BBS. We propose three constructions for device binding of BBS-based credentials, following the idea recently proposed by Orange Innovation [**?**] as BBS#. The SE public key $dpk = dsk \cdot H_0$ is a component of the attribute commitment formed within a BBS credential, which the issuer signs (so here H_0 is a generator used in the BBS credential). The user now has a credential on the vector $(dsk, \boldsymbol{a})$ ($\boldsymbol{a}$ being the vector of her attributes) but does not know *dsk* explicitly. Previous schemes let the hardware component contribute the *dsk* part to the overall BBS proof; in contrast, BBS# relies on *key-homomorphic* signatures. The user re-randomizes *dpk* into $\overline{dpk} \leftarrow dpk + r_{\text{key}} H_0$. Roughly, the user proves knowledge of a BBS signature on a group element C and that they know r_{key} such that $C + r_{\text{key}} H_0 = \overline{dpk}$. For device binding, the SE computes a signature on *ctx* under *dsk* that the user then adapts to $\overline{dpk}$ and reveals to the verifier. This is *round-optimal*, as only a single query to the SE is required. We show that both standard Schnorr and BLS signatures [**?**] can be used for the device signature, which leads us to the following variants:

BBS-Schnorr: Our first construction is exactly the BBS# proposal with weak Fiat-Shamir Schnorr signatures (over $\mathbb{G}_1$ of a pairing-friendly curve) for *dsk*. We prove the construction satisfies UNF-DBAC and UNLINK-1. Overall, BBS-Schnorr is the first provably-secure round-optimal and stateless device binding protocol for BBS. However, we show explicit subversion attacks.

BBS-BLS: Our second construction replaces Schnorr with BLS which eliminates subversion attacks thanks to the uniqueness of BLS signatures. It achieves UNLINK-2 and remains round-optimal and unforgeable. The main technical difficulty is showing that the adapted BLS signature serves as a straight-line extractable proof of knowledge of the underlying secret key in the AGM *even when composed with the other elements of our scheme such as BBS.*

BBS-BlindBLS: To get context blindness and satisfy our strongest privacy notation, we use the blind version of BLS signatures for the device signatures. This provably satisfies UNLINK-3, SE-OBLIV, and UNF-BDBAC.

Overall, all schemes are round-optimal and achieve unforgeability under standard assumptions in the ROM+AGM, and statistical privacy. The fact that the SE only computes a well-established and standard signature is a major advantage for deployability: hardware-based SEs are not expected to serve application-specific purposes but rather provide simple and multi-purpose interfaces.

Extendability & Compatibility of the BLS-Based Schemes. A notable feature of our two BLS-based variants is that they are fully compatible. That is, using BLS

for the device signature, the same main credential scheme can support the local SE and remote HSM setting at the same time, verification is identical. This addresses an important real-world constraint: given that it is unlikely that all users will (or want to) own phones with the necessary high-assurance hardware SEs, a remote HSM deployment is not just needed for the initial phase, but rather will prevail. Finally, BLS signatures can be distributed and thresholdized [**?**], while fully hiding the distributed structure from the verifier. This allows extensions, e.g., enabling users to rely on multiple devices, combining a local and remote key, or letting the remote HSM internally use key distribution for added security.

2 Preliminaries

The security parameter is λ. We initialize a key-value map data structure with $M \leftarrow ()$, set a key k to be a value v with $M[k] \leftarrow v$, and retrieve the value at a key k with $M[k]$. We use $\leftarrow\!\!\$\ S$ to denote sampling of an element uniformly at random from a set S. If Alg is a probabilistic algorithm, then $\mathsf{Alg}(\mathsf{parameters}; \rho)$ denotes running Alg with parameters $\mathsf{parameters}$ and random coins ρ. If Alg is a probabilistic algorithm and the coins are not given explicitly in a call, then we assume they are sampled uniformly at random.

Pairings. In this paper we restrict ourselves to the type-3 [**?**] pairing setting, described as follows. Syntactically, we require a bilinear group generator algorithm $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, G_1, G_2, \mathsf{e}) \leftarrow \mathsf{BGGen}(1^\lambda)$ where p is the shared order of groups $\mathbb{G}_1, \mathbb{G}_2$, and $\mathbb{G}_T$, G_1 and G_2 are the generators of the source groups $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively, and $\mathsf{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is an efficiently computable function satisfying $\mathsf{e}(aG_1, bG_2) = ab \cdot \mathsf{e}(G_1, G_2)$ and $\mathsf{e}(G_1, G_2) \neq 1_{\mathbb{G}_T}$.

Cryptographic Assumptions and Idealized Models. All presented device binding protocols rely on the Random Oracle Model (ROM) [**?**] and the Algebraic Group Model (AGM) [**?**] for their security analysis. In the AGM all adversaries are presumed to be algebraic, formally defined in Appendix A.

Signatures and Blind Signatures. A signature scheme Sig defined with respect to a message space $\mathcal{M}$ and a size parameter ℓ is a set of algorithms $(\mathsf{Setup}, \mathsf{KGen}, \mathsf{Sign}, \mathsf{Verify})$ such that $\mathsf{Sig.Setup}(1^\lambda, \ell)$ outputs the parameters par, and $\ell + 1$ is the number of messages; $\mathsf{Sig.KGen}(par)$ outputs a key pair (sk, pk); $\mathsf{Sig.Sign}(sk, \boldsymbol{m})$ outputs a signature sig on a list of messages $\boldsymbol{m} \in \mathcal{M}^{\ell+1}$; and $\mathsf{Sig.Verify}(pk, sig, \boldsymbol{m})$ outputs 1 if the signature is valid. Sig must also satisfy the standard correctness and unforgeability (EUF-CMA) properties, which we give in Appendix A.

A (round-optimal) blind signature scheme BSig is a set of algorithms $(\mathsf{Setup}, \mathsf{KGen}, \mathsf{Blind}, \mathsf{BSign}, \mathsf{Unblind}, \mathsf{Verify})$ defined with respect to message space $\mathcal{M}$ and size parameter ℓ, where $\mathsf{Setup}, \mathsf{KGen}, \mathsf{Verify}$ are as for ordinary signatures. $\mathsf{BSig.Blind}(pk, \boldsymbol{m})$ outputs a one-time unblinding key bst and a blinded message $\overline{m}$; $\mathsf{BSig.BSign}(sk, \overline{m})$ outputs a blinded signature $\overline{sig}$; and $\mathsf{BSig.Unblind}(bst, \overline{sig})$

outputs an unblinded signature sig. We require the standard notions of correctness and (weak) one-more unforgeability, defined formally in Appendix A. Weak one-more unforgeability requires that an adversary interacting n times with a signer cannot output more than n valid signatures on distinct messages.

Boneh-Boyen-Shacham (BBS) Signatures. The Boneh-Boyen-Shacham (BBS) signature scheme [**?**] allows for extremely efficient zero-knowledge proofs of knowledge of a signature with perfect zero-knowledge, which makes it very useful for practical ACs with everlasting privacy [**?**].

$\mathsf{BBS.Setup}(1^\lambda, \ell)$**:** output $\mathsf{BGGen}(1^\lambda)$ and generators $(H_0, \ldots, H_\ell) \leftarrow\!\!\$\ \mathbb{G}_1$
$\mathsf{BBS.KGen}(par)$**:** sample $sk \leftarrow\!\!\$\ \mathbb{Z}_p$, set $pk \leftarrow sk \cdot G_2$ and output (sk, pk)
$\mathsf{BBS.Sign}(sk, \boldsymbol{m})$**:** sample $e \leftarrow\!\!\$\ \mathbb{Z}_p$, compute $A \leftarrow \frac{1}{sk+e}(G_1 + \langle \boldsymbol{m}, (H_i) \rangle)$, where $\langle \boldsymbol{m}, (H_i) \rangle$ is the dot product of $\boldsymbol{m}$ and generators $(H_i)_{i \in [\ell]}$, output (A, e)
$\mathsf{BBS.Verify}(pk, sig, \boldsymbol{m})$**:** parse $(A, e) \leftarrow sig$ and output 1 if $\mathsf{e}(A, pk + eG_2) = \mathsf{e}(G_1 + \langle \boldsymbol{m}, (H_i) \rangle, G_2) \ \wedge \ A \neq 0_{\mathbb{G}_1}$, otherwise 0

3 Device-Bound Anonymous Credentials (DBAC)

In a credential system, there are three types of participants: the users, the issuer(s) and the verifiers. The issuer issues credentials to the users; a user Alice's credential σ attests that Alice has certain identity attributes $\boldsymbol{a}$. Alice can then use σ in a specific context ctx to compute a credential presentation τ that convinces a verifier that her identity attributes $\boldsymbol{a}$ satisfy a predicate ϕ.

In a *device-bound* credential system there are additionally the secure elements (SE). An SE can be either a local hardware component run within the user's main device, or a cloud service. Either way, it generates a *device key* pair (dsk, dpk) to which Alice's credential is bound; the SE is the only component of Alice's system that's in possession of dsk, while dpk is known to Alice herself and (possibly) the issuer. Any presentation of the device-bound credential must involve the device holding dsk, explicitly approving the presentation.

Outsourcing the secure element to the cloud introduces a privacy challenge: considering that the context typically includes a unique session nonce or information that binds the authentication to the targeted service, revealing the context to an external SE entirely voids the privacy properties of the anonymous credential system. Therefore, we introduce two variants of DBACs: *context-aware* and *context-blind* (BDBAC). We define the syntax of both variants in this section, followed by the unforgeability (Section 4) and privacy properties (Section 5).

3.1 Algorithms

Setup and Keys. With $(isk, ipk) \leftarrow \mathsf{IssKGen}(par)$ and $(dsk, dpk) \leftarrow \mathsf{DevKGen}(par)$ we generate issuer and device key pairs for shared parameters par.

Issuance. A credential σ is generated through algorithm Issue on input the issuer secret key isk, the device public key dpk and the user's attributes $\boldsymbol{a}$. The credential is then said to be device-bound to dpk. We consider device- and attribute-*hiding* during issuance out of scope for this paper because it can likely be accomplished using standard techniques, such as in [**?**]. Upon receiving the credential σ, the user runs VfCred to verify it with respect to ipk, dpk, and $\boldsymbol{a}$.

Presentation. To create a presentation τ for a predicate ϕ satisfied by attributes $\boldsymbol{a}$ though a device-bound credential σ for dpk, the user must interact with the SE through the showing protocol. $\mathsf{ShowUser}$ is the user's host device's side of the protocol, and ShowSE is the SE's side. Both take as input the context ctx, which typically encodes a session nonce and the targeted service the user wants to authenticate to. The user also takes $ipk, dpk, \sigma, \boldsymbol{a}$, and ϕ as input, whereas the device only gets dsk along with ctx.

Definition 1 (DBAC). *A (context-aware) Device-Bound Anonymous Credential scheme (*DBAC*), defined with respect to a predicate family $\Phi = \{\Phi_{par}\}_{par}$ and message space $\mathcal{M} = \{\mathcal{M}_{par}\}_{par}$ is a set of algorithms* ($\mathsf{Setup}, \mathsf{IssKGen}, \mathsf{DevKGen}, \mathsf{Issue}, \mathsf{ShowUser}, \mathsf{ShowSE}, \mathsf{Verify}, \mathsf{VfCred}$) *with the following syntax:*

$\mathsf{Setup}(1^\lambda, \ell) \to par$**:** *outputs the public parameters for the scheme where ℓ is the number of attributes for a credential description.*

$\mathsf{IssKGen}(par) \to (isk, ipk)$**:** *generates a key pair for the issuer.*

$\mathsf{DevKGen}(par) \to (dsk, dpk)$**:** *generates a key pair for the secure element.*

$\mathsf{Issue}(isk, dpk, \boldsymbol{a}) \to \sigma$**:** *run by the issuer to generate a credential bound to device key dpk and for attributes $\boldsymbol{a} \in \mathcal{M}$.*

$\langle \mathsf{ShowUser}(ipk, dpk, \sigma, \boldsymbol{a}, ctx, \phi) \rightleftharpoons \mathsf{ShowSE}(dsk, ipk, ctx) \rangle \to (\tau, \perp)$**:** *an interactive protocol between the user and the device, where at the end the user obtains a presentation of the given predicate and context. All schemes in our work are round-optimal and have three-move showing, so we restrict the* DBAC *syntax to this simpler setting with the following algorithms:*

- $\mathsf{ShowUser}_1(ipk, dpk, \sigma, \boldsymbol{a}, ctx, \phi) \to (ust, umsg)$
- $\mathsf{ShowSE}_1(dsk, ipk, umsg, ctx) \to smsg$
- $\mathsf{ShowUser}_2(ust, smsg) \to \tau$

$\mathsf{Verify}(ipk, \tau, ctx, \phi) \to 0/1$**:** *run by the verifier to check that a presentation is valid w.r.t. a specific context and predicate.*

$\mathsf{VfCred}(ipk, \sigma, dpk, \boldsymbol{a}) \to 0/1$**:** *run by the user to verify that a credential is valid w.r.t. a device public key and set of attributes.*

Context-Aware vs. Context-Blind. In the definition above, the device learns the presentation context ctx when providing its contribution, where the context typically encodes the verifier's identity and session the presentation should be bound to. Revealing the context to the device allows for a stronger binding of the device's contribution, or even enables the device to condition and refuse presentations based on the context; this can be a bug or a feature. Thus, already based on its syntax, a DBAC scheme is *context-aware*.

Context-awareness is not always desirable. A context-aware SE may stop its user from obtaining a presentation for a context it disapproves of. Also, for a cloud-based SE, context awareness clearly is at odds with privacy. Thus, we also need to consider, both syntactically and in terms of security properties, schemes that do not take *ctx* as input. This is what we call a *(context)-blind* Device-Bound Anonymous Credential (BDBAC). This version is syntactically identical to a DBAC scheme except that the device does not take *ctx* as an argument:

Definition 2 (BDBAC). *A context-blind Device-Bound Anonymous Credential scheme (BDBAC) is defined exactly as DBAC except for ShowSE for which the input is now defined as* $\mathsf{ShowSE}(dsk, ipk)$ *(and* $\mathsf{ShowSE}_1(dsk, ipk, umsg)$ *in the concrete three-move variant).*

Comparison to DAA Syntax. Direct Anonymous Attestation (DAA) [**?**] provides a richer interface for the trusted platform module (TPM) than DBAC does for a secure element (SE). Specifically, a TPM can compute a pseudonym from the device secret key and a given scope (or "base" in DAA terminology). This pseudonym can then be used in an outside protocol, for example in order to preserve a linkable state across several interactions for the same verifier. We stress here that DBACs do not have this feature, and no part of the presentation τ can be considered a *device-based pseudonym*. However, it is possible to express pseudonyms through our predicate notation, where the pseudonym gets seeded through an attribute that is contained in the user's credential.

Correctness. Presentations that are generated in an honest interaction between SE with device key (dsk, dpk) and a user, from an honestly generated credential bound to dpk and for a predicate ϕ satisfied by the attributes, will verify correctly. For space reasons, the formal definition is given in Appendix B.

4 Unforgeability

Our unforgeability definition for (B)DBAC covers multiple aspects: attribute unforgeability, presentation freshness, and the device binding non-transferability. The full game is shown in Figure 1, in a context-aware and a context-blind version. It turns out the latter makes it harder to tell apart trivial from relevant forgeries, as blind SEs no longer see what presentations they are participating in. We point out the differences between the two versions in the following.

High-Level Structure. The issuer is honest in the game, verifiers are corrupt, and users and SEs may be corrupt or honest. With a corrupt user, the adversary gets access to their credentials, and with a corrupt SE the adversary learns the device secret key *dsk*. Oracles model the adversarial control over users and devices: it can create users and SEs, choosing their corruption state, generate credentials, and request presentations from honest users or SEs. Eventually, the adversary outputs a forgery $(\tau_i^*, ctx_i^*, \phi_i^*)_{i=1}^k$ (where $k = 1$ in the non-blind game).

Both games require that the device public key dpk^* be extractable from the adversary's output.[3] More precisely, we require an efficient extractor, consisting of two algorithms, Ext.Setup for setting up the parameters, and Ext.EKey for tying each presentation to a specific device public key. Therein, the parameters *par* output by Ext.Setup must be indistinguishable from those output by (B)DBAC.Setup. The adversary wins by outputting presentations τ_i^* that are valid for ctx_i^*, ϕ_i^* (for all k outputs) and $(dpk^*, ctx_1^*, \phi_1^*)$ is a non-trivial forgery (or k is larger than a trivial bound in BDBAC). What constitutes a non-trivial forgery depends on the device key to which the forged presentation(s) correspond(s) – dpk^* would remain hidden in the real-world operation of the system, but is extracted in the game. The set of trivial forgeries is described in the sequel.

Oracles and Records. $\mathcal{O}_{\mathsf{NewSE}}$ lets the adversary create an honest SE, which it can later corrupt using $\mathcal{O}_{\mathsf{CorruptSE}}$; or the SE can be made corrupt from the get-go. At this point $\mathcal{A}$ also sets the user to *honest* or *corrupt*, which is recorded by storing *dpk*'s associated with honest users to `HUsers`. This set `HUsers` remains crucial throughout the game, as oracle behavior and win cases are conditioned on the user's corruption state.

Using $\mathcal{O}_{\mathsf{Issue}}$, the adversary can obtain a credential on attributes of its choice for corrupt users ($dpk \notin$ `HUsers`) and for a previously registered device key dpk. On the other hand, $\mathcal{O}_{\mathsf{hIssue}}$ creates (but does not output) a credential for an honest user with an honest SE. The adversary can later request presentations of such honest users' credentials through $\mathcal{O}_{\mathsf{hShow}}$. The description of $\mathcal{O}_{\mathsf{hShow}}$ for DBAC differs purely syntactically from that of BDBAC, taking into account that the context *ctx* is one of the inputs to ShowSE in DBAC but not in BDBAC.

Finally, the oracle $\mathcal{O}_{\mathsf{ShowSE}}$ models the case where a corrupt user owns a credential bound to an honest SE's dpk. So, using this oracle the adversary can create presentations of its device-bound credentials.

Winning Conditions. The adversary wins by computing a "non-trivial" presentation. Recall that of this presentation we know the extracted dpk^*, a context ctx^*, and a predicate ϕ^*. In our game, we collect all trivial presentations from the maintained records, and check none captures the adversary's output. We classify "trivial" presentations into three categories, depending on the corruption setting of SE and user. These corruption settings are checked by looking for matching dpk in `CSEs`, and matching credentials in `CCreds`, respectively.

T_1 **(Corrupt User, Corrupt SE):** If dpk^* is corrupt and associated to a corrupt user, then all presentations satisfiable from the adversarially owned credentials bound to dpk^* are trivial. Thus, here the adversary only wins if it can create a presentation for a predicate ϕ^* that is not satisfied by any of the attributes issued to dpk^* (reflected in `CCreds`).

[3] One could formulate unforgeability in a morally weaker way: instead of requiring that every credential showing be tied to a specific device, we may require that the adversary be unable to show a credential that it could not have computed using any of its devices and queries to honest participants.

Fig. 1. Unforgeability games for context-aware and context-blind DBAC schemes. The extractor EKey also gets access to the entire game state before seeing $\mathcal{A}$'s output (e.g. isk, $\mathtt{SEKeys}$, and all of the signatures output by Issue), and can program random oracles. Additionally, $\mathcal{O}_* = (\mathcal{O}_{\mathsf{NewSE}}, \mathcal{O}_{\mathsf{Issue}}, \mathcal{O}_{\mathsf{ShowSE}}, \mathcal{O}_{\mathsf{hIssue}}, \mathcal{O}_{\mathsf{hShow}}, \mathcal{O}_{\mathsf{CorruptSE}})$.

T_2 **(Corrupt User, Honest SE):** If the user is corrupt, but uses a dpk^* of an honest SE, we additionally get the device binding guarantees. The winning condition in this case varies somewhat in the two games. In the non-blind DBAC game, the adversary wins with a presentation that is not satisfied by any of the corrupt users' credentials, or for a context ctx^* that was never queried to the corresponding SE (reflected in `Granted`). The second condition is not enforceable in the blind BDBAC game, and we revert to a one-more condition. That is, the adversary wins if it outputs more valid presentations than it made queries to the SE for dpk^* (reflected in `GrantedCount`).

T_3 **(Honest User, Honest SE):** This winning condition is the strictest, and the adversary wins if it creates a presentation for a context and predicate that it never queried to the honest user/SE (reflected in `HShown`).

Record	Stored Tuples	Explanation
`SEKeys`	$\{(dsk, dpk)\}$	All device key pairs generated by the game.
`CSEs`	$\{(dpk)\}$	Corrupt SEs.
`HUsers`	$\{(dpk)\}$	Device keys belonging to honest users.
`CCreds`	$\{(dpk, \boldsymbol{a})\}$	Attributes issued to corrupt users (through $\mathcal{O}_{\mathsf{Issue}}$).
`HCreds`	$\{(dpk, \boldsymbol{a}, \sigma)\}$	Issued credentials for honest users (through $\mathcal{O}_{\mathsf{hIssue}}$).
`HShown`	$\{(dpk, ctx, \phi)\}$	Queried presentations (context and predicate) for honest user and honest SE (through $\mathcal{O}_{\mathsf{hShow}}$).
`Granted`	$\{(dpk, ctx)\}$	Queried presentations (only context) for honest SE of corrupt users (only in non-blind DBAC version).
`GrantedCount`	$\{(dpk, k)\}$	Number of presentations k the honest SE of dpk has contributed to (only in blind BDBAC version).

Table 2. Records kept in our unforgeability games.

Our game builds the entire set of trivial combinations for these three types when determining the winning condition. This is done for the sake of readability, but will result in an infinite set when generated naively. Clearly, the sets have finite representations for which membership can efficiently be checked.

Definition 3 (Unforgeability). *A* (B)DBAC *scheme Π is unforgeable if there exists an extractor* $\mathsf{Ext} = (\mathsf{Setup}, \mathsf{EKey})$ *such that for all PPT adversaries $\mathcal{A}$ and $\ell \in \mathbb{N}$ the following conditions hold: (1) The distributions of par from Π.*Setup *and* Ext.Setup *are indistinguishable. (2) The following advantage is negligible:*

$$\mathsf{Adv}^{\mathit{UNF}}_{\Pi,\ell,\mathsf{Ext}}(\mathcal{A}, \lambda) := \Pr[\mathsf{Unf}^{\mathcal{A}}_{\Pi,\ell,\mathsf{Ext}}(\lambda) = 1]$$

Honest User, Corrupt SE Setting? In our winning conditions we do not treat the case where an honest user is relying on a corrupt SE. In the hardware SE setting this seems implausible, but in the remote HSM setting it is more realistic: an adversary could break into the central HSM managing all users' keys without corrupting a specific user's phone. Instead of accounting for this case in our unforgeability definition, we introduce an additional and standalone property, *SE obliviousness*, in Section 5.4. SE obliviousness guarantees that a malicious SE cannot learn anything from a presentation request beyond the context (in

DBAC), or even learns nothing at all (in BDBAC). Clearly, if the corrupt SE learns nothing from the interaction with the user, it cannot get any additional advantage over an adversary not controlling the SE. The unforgeability guarantees for such an honest user/corrupt SE combination are then equivalent to the honest user/honest SE case (Type-3 forgeries) in our game. Thus, if a (B)DBAC scheme is expected to provide unforgeability in the scenario where the SE can be subverted or corrupted, but the host remains honest, the scheme must satisfy both unforgeability and SE obliviousness.

Remark on Secure Key Generation. Our model assumes (semi-) honest key generation for the SE. A more general version of this security game would let the adversary also inject fully maliciously generated *dpk*'s into the game. Satisfying this is straightforward on the protocol level by appending an appropriate NIZK to *dpk*'s. As our definition is already complex, we opted for the simpler version.

5 Privacy

The standard privacy guarantee of anonymous credentials is that a presentation τ does not reveal anything about the user other than the fact that $\phi(\boldsymbol{a}) = \mathsf{True}$, and is unlinkable to any of the user's other credential presentations. This considers the issuer and verifier to be corrupt and collude with each other. For a device-bound AC system, we have another possibly malicious actor: the secure element. In the setting where the SE is provided as an external and central cloud service catering to millions of users, privacy should clearly not rely on the honesty of the SE, but rather hold even if that entity is fully malicious. In the hardware SE setting, full corruption seems less likely, but at the same time the shielded nature makes subversion attacks harder to detect.

Therefore, we formulate the privacy goal for AC – *unlinkability* of presentations – in a hierarchy of four levels, giving the adversary increasing control over the user's SE: in the most basic notion (Unlink-0), the SE is honest and the adversary can interact with the SE only through the honest user. In Unlink-1, the SE is still honest, but the host device is not exempt from temporary corruption, i.e., the adversary can engage directly with the honest SE's interface. Unlink-2 captures the setting where the SE's behavior can be statically subverted, e.g., by malicious actors in the hardware supply chain. Finally, full corruption, where the adversary has a live view and control of the SE, is defined through Unlink-3.

Beyond unlinkability of the presentations, we want to limit a malicious SE's knowledge of presentations a user requests. On the one hand, facing full SE corruption, unlinkability is completely undermined if the SE observes the *ctx* or another unique detail of the presentation it signs. On the other hand, a subverted or malicious SE could be programmed to deny service based on the user's attributes or shown predicates. Therefore, we model *SE obliviousness* (SE-Obliv), requiring that the SE does not learn the user's private input to the presentation request, i.e., her credential, attributes and predicate. For context-blind BDBAC this also includes the context, and obliviousness becomes an essential property that a BDBAC scheme *must* satisfy in addition to Unlink-3.

In our privacy model we disregard issuance, since it is not the focus of this work, but we provide a brief discussion of possible pitfalls in Appendix E.

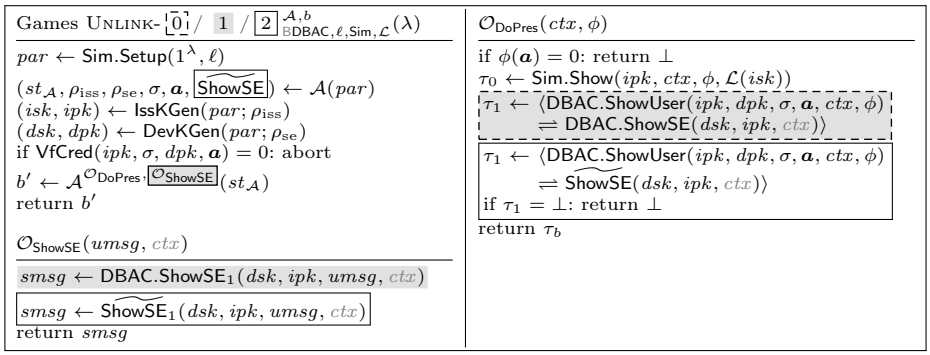

Fig. 2. Games for UNLINK-0 (plain and solid box code), UNLINK-1 (plain and highlighted code), and UNLINK-2 (plain and dashed box code). For BDBAC, the *ctx* is not sent to the SE, indicated by gray text color.

5.1 UNLINK-0, UNLINK-1: Privacy With Honest SEs

The first three security games are similar enough to provide them together in Figure 2. We present all properties in the single-user, single-credential setting, but show the multi-user, multi-credential notions to be equivalent in Appendix C. The definitions use a simulation-based approach: they require the existence of a simulator consisting of Sim.Setup and Sim.Show, where parameters generated by Sim.Setup are indistinguishable from those generated by (B)DBAC.Setup. The adversary plays a game parameterized by a bit b. Challenges are available through the $\mathcal{O}_{\mathsf{DoPres}}$ oracle, which returns a presentation τ_b to an adversarially provided credential, predicate and context. If $b = 0$, the presentation is merely simulated via Sim.Show that takes the public information ϕ and *ctx* as input, but not the user's credential. Thus, by definition, the presentation τ_0 cannot reveal any information beyond what is revealed through the predicate. If $b = 1$, then the challenger executes the DBAC system the same way an honest user would do for the underlying σ and $\boldsymbol{a}$. We require that the adversary should not be able to distinguish the "ideal world" with $b = 0$ from the "real world" with $b = 1$.

Our first two unlinkability variants consider *honest* SEs. The "baseline" anonymity game UNLINK-0 is identical to anonymity if the SE was just a part of the showing algorithm. Here, the SE is fully under the user's control at all times. We model semi-honest issuer and device keys, assuming that all parties verify the public keys they receive from each other. UNLINK-0 is realized by the game described above without any modifications.

UNLINK-0 assumes that the honest user can perfectly shield the SE, and does not allow the adversary to directly interact with it. Thus, a scheme where

the SE reveals a trapdoor-like object to the host that allows to de-anonymize presentations would satisfy UNLINK-0 – although even a temporary corruption of the user's phone could retroactively break their privacy. Therefore, we strengthen the game in UNLINK-1, where the host device may fall into the adversary's hands. We model this by granting $\mathcal{A}$ access to an additional $\mathcal{O}_{\mathsf{ShowSE}}$ oracle, so it can directly interact with the SE, and see the respective outputs.

Additional Leakage $\mathcal{L}$. Simulating knowledge of a credential with only *ipk* can be difficult, especially in the type-3 pairing setting that our constructions are in. Here the *ipk* is in one group, and parts of the credential are in the other, without an efficient isomorphism. Tessaro and Zhu [**?**] observe that a simulation may only be feasible with respect to a leakage function $\mathcal{L}$ that provides (minimal) additional knowledge about the issuer key to Sim.

5.2 UNLINK-2: Privacy with Subverted SEs

Next, we allow $\mathcal{A}$ to corrupt SEs in a limited form. UNLINK-2 allows *algorithm substitution attacks*: at the beginning of the game, the adversary may re-program the SE. Importantly, $\mathcal{A}$ does not have ongoing control or access. This models the adversary's ability to tamper with an SE's integrated circuit during production. We capture this scenario in a (non-adaptive, non-continuous) *algorithm substitution attack* ([**?**], earlier discussed as *kleptography* [**?**,**?**]). A similar approach to modeling SE corruption was taken by [**?**].

The adversary specifies a subverted $\widetilde{\mathsf{ShowSE}}$ during the game's initialization, and throughout the game $\widetilde{\mathsf{ShowSE}}$ will be run in place of the honest $\mathsf{DBAC.ShowSE}$. Implicitly, if presentations are still simulatable, this also captures that whatever the SE learns during signing cannot be exfiltrated to an outside adversary via some subliminal channel unnoticed by the user.

A slight subtlety is that our definition must account for the fact that the adversary can trivially distinguish the output of $\widetilde{\mathsf{ShowSE}}$ from that of the simulator if $\widetilde{\mathsf{ShowSE}}$ outputs garbage that makes the host algorithm fail. Therefore, we follow related work on subversion resilience (e.g., [**?**]) by having the game abort if $\tau_1 = \perp$, i.e., the honest user failed to compute a presentation. This seems reasonable; in a secure scheme the user notices malformed SE outputs and does not output any presentation, maintaining unlinkability.

We emphasize that the corruption of SE in UNLINK-2 is relatively weak, in particular, the SE is *stateless*, and this model mainly serves as a stepping stone to the next, where we treat fully malicious SEs.

5.3 UNLINK-3: Privacy with Full SE Corruption

Finally, we model unlinkability with a *fully corrupt* SE: we remove the isolation imposed in UNLINK-2 and give the adversary full control over the SE. This model is particularly relevant when the SE is run through a cloud HSM.

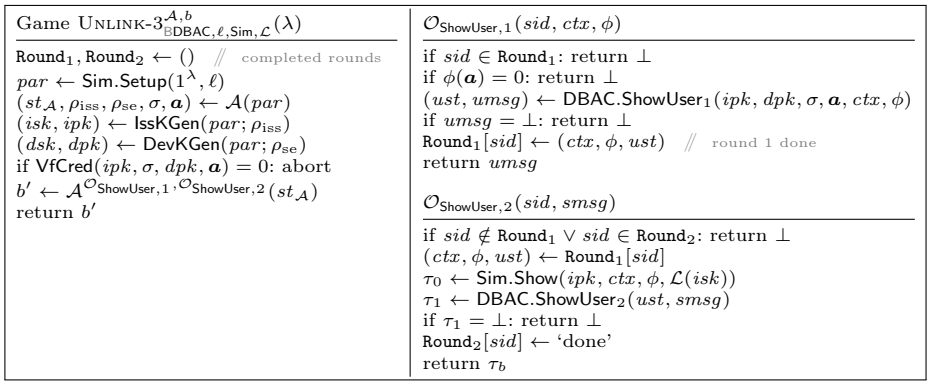

Fig. 3. UNLINK-3 game (fully corrupt SE) for DBAC and BDBAC (no differences).

This changes the game significantly. Instead of giving the adversary access to $\mathcal{O}_{\mathsf{DoPres}}$ which internally ran both the user and SE part, we now provide two oracles for the two user algorithms (as the SE part in between is fully controlled by $\mathcal{A}$ now). In $\mathcal{O}_{\mathsf{ShowUser},1}$, $\mathcal{A}$ directs an honest user to initiate the computation of a credential presentation, causing the honest user to send a message to $\mathcal{A}$. The adversary can respond in place of the SE with an arbitrary *smsg* to $\mathcal{O}_{\mathsf{ShowUser},2}$. Upon *smsg*, the latter oracle responds with a challenge, either by completing the real interaction according to DBAC ($b = 1$), or by simulating a completely independent presentation ($b = 0$). This captures that τ is not linkable to *dpk*, non-disclosed attributes or the credential even if the adversary actively participates in its creation on behalf of the SE.

5.4 SE Obliviousness

If the SE is fully corrupt, UNLINK-3 alone is not a sufficient privacy guarantee. So far, we enforced that presentations are not linkable to a particular user – but a malicious SE possibly still learns a lot about what the user is up to, depending on how *umsg* is formed, such as the contexts they use, predicates, or even the full credential. The implications are threefold: (1) Unlinkability can break completely if the adversary simply de-anonymizes a user by the (nonce-like) context that appears in a presentation; (2) Without obliviousness an SE can potentially learn information about, e.g., the user's credential, and refuse service to them based on the contents; and (3) If the SE learns the full credential, it is in the position to create presentation of an honest user entirely on its own.

For these reasons, we introduce *SE obliviousness* (SE-OBLIV). This property ensures that the intermediate messages *umsg* reveal nothing about the user's presentation intentions, except that it needs *dpk* (which the SE already knows). In Figure 4, we present the SE-OBLIV game. It is somewhat similar in spirit to UNLINK-3, in that it grants the adversary access to the two steps of the honest user presentation through oracles $\mathcal{O}_{\mathsf{ShowUser},1}$ and $\mathcal{O}_{\mathsf{ShowUser},2}$. But here, it

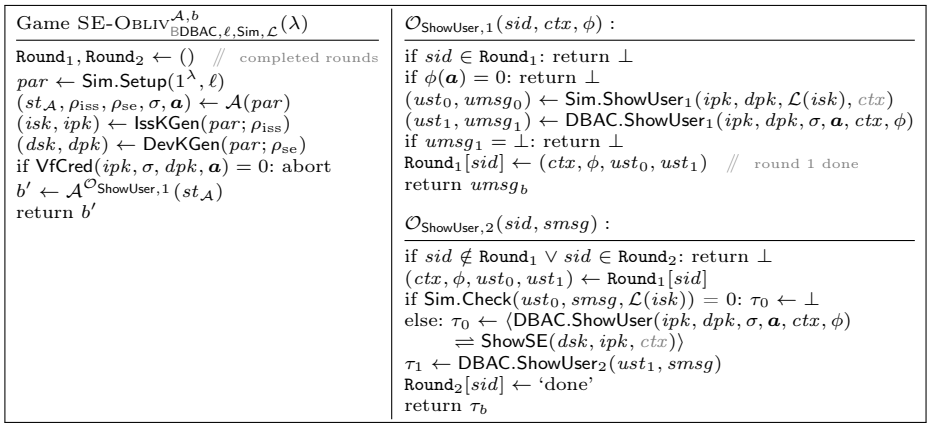

Fig. 4. SE obliviousness game. The simulator's $\mathsf{ShowUser}_1$ algorithm receives the context *ctx* for DBAC, the context is omitted for BDBAC, indicated by gray text color.

creates the challenge already in the first oracle: $\mathcal{O}_{\mathsf{ShowUser},1}$ either responds with a normally created *umsg* ($b = 1$) or a simulated one ($b = 0$). The simulator creating $umsg_0$ does not get the user's credential, attributes, or requested predicate (and for BDBAC neither the context *ctx*). This models that the user's message to the SE hides all that information.

We want obliviousness to hold even if the SE colludes with an outside adversary and therefore $\mathcal{A}$ learns the finished presentations. This is what $\mathcal{O}_{\mathsf{ShowUser},2}$ models, where $\mathcal{A}$ can send responses *smsg* and receives the presentation τ_b. Thereby, the oracle always outputs a "real" presentation: in the real world ($b = 1$), this presentation is built using *smsg*, and in the ideal world ($b = 0$), we compute a fresh showing from scratch (using the original context and predicate).

A subtlety arises here: $\mathcal{O}_{\mathsf{ShowUser},2}$ entirely ignores $\mathcal{A}'s$ input message *smsg* when computing τ_b in the ideal world. Thus, it will succeed in producing a valid presentation even when *smsg* is invalid. We cannot have the $\mathsf{ShowUser}_2$ algorithm perform a check, either, since we returned a simulated message $umsg_0$ in the first oracle, on which the adversary has now computed on. Hence, to prevent the security game from outputting τ_b when *smsg* is invalid, we require the simulator to provide an algorithm Sim.Check that verifies *smsg* for $b = 0$.

Privacy Limitation Against a Fully Malicious (Remote) SE. We stress two points about user privacy in the setting where the SE is controlled by the adversary in real time, such as, potentially, in the remote HSM setting. First, a combination of using a context-blind BDBAC, satisfying UNLINK-3 and SE-OBLIV properties is required for any meaningful notion of privacy. Secondly, even with these properties, *the overall achievable privacy is lower than in the other corruption settings*! The SE still observes metadata on the presentations it creates, namely the number and timing of showings. In particular, the timing side-channel is impossible to remove when the SE is required for every presentation and the

presentation shall be (blindly) bound to a fresh session nonce provided by the verifier. Thus, while we believe that the remote SE is a useful alternative that makes device binding available to users that do not own a flagship phone, there is some inherent loss of privacy.

5.5 Formal Definitions and Relations

All privacy games are stated for the context-aware DBAC and context-blind BDBAC version, where the BDBAC version only drops the *ctx* input from all interactions with the honest or corrupt SE. For UNLINK-3 both notions are equivalent. Thus, slightly abusing notation we now formalize what it means for a (B)DBAC scheme to satisfy the privacy notions corresponding to each game:

Definition 4 (UNLINK-{0,1,2,3} and SE-OBLIV). *A* (B)DBAC *scheme* Π *satisfies* X *property for* $\mathsf{X} \in \{$*UNLINK-0,UNLINK-1,UNLINK-2,UNLINK-3, SE-OBLIV*$\}$ *w.r.t. a leakage function* $\mathcal{L}$ *if there exists a simulator consisting of* $(\mathsf{Setup}, \mathsf{Show})$ *for UNLINK and* $(\mathsf{Setup}, \mathsf{ShowUser}_1, \mathsf{Check})$ *for SE-OBLIV, such that for all PPT adversaries* $\mathcal{A}$ *and* $\ell \in \mathbb{N}$ *the following two conditions hold: (1) the parameters generated by* Sim.Setup *are indistinguishable from those generated by* Π.Setup*. (2) The following advantage is negligible:*

$$\mathsf{Adv}^{\mathsf{X}}_{\Pi,\ell,\mathsf{Sim},\mathcal{L}}(\mathcal{A},\lambda) := |\Pr[\mathsf{X}^{\mathcal{A},b=0}_{\Pi,\ell,\mathsf{Sim},\mathcal{L}}(\lambda)=1] - \Pr[\mathsf{X}^{\mathcal{A},b=1}_{\Pi,\ell,\mathsf{Sim},\mathcal{L}}(\lambda)=1]|$$

The four privacy properties UNLINK-{0,1,2,3} are progressively strictly stronger. This is straightforward to prove: we strictly increased the adversarial control over the SE in every step. UNLINK-1 is only subtly stronger than UNLINK-0, but helps to ensure intermediate SE messages *smsg* do not leak anything that would de-anonymize presentations. Our BBS-Schnorr construction presented in the next section separates UNLINK-1 and UNLINK-2: it relies on a Schnorr signature for the device contribution, which a subverted SE can exploit to encode tracking information into the randomness used therein. Finally, UNLINK-3 allows the adversary to actively participate in computing the challenge τ_b – BBS-BLS is an example where this ability breaks unlinkability, as the SE observes a somewhat unique identifier of the final presentation. We provide the lemmas and proofs for these relations in Appendix D.

Note that SE-OBLIV is orthogonal to UNLINK and intuitively, so a DBAC scheme that satisfies SE-OBLIV in addition to UNLINK-2 or UNLINK-3 becomes strictly more secure.

6 Our (B)DBAC Constructions

We now present three constructions that provide the desired device binding while decreasingly relying on the trustworthiness of the SE for their privacy guarantees. All constructions use BBS for the core attribute credential and are highly efficient and round-optimal: essentially, they only require a standard Schnorr or (blind) BLS signature from the device. Our construction from *blind* BLS signatures

yields the first device-bound AC scheme that is context-blind and can be used for the remote HSM setting currently designed for the EUDI wallet in Europe.

All our constructions build on the "secure splitting" idea introduced by Orange Innovation for their BBS# proposal [**?**]. We start by describing this idea and the common blueprint used in all constructions before presenting the concrete variants and their security analysis in detail.

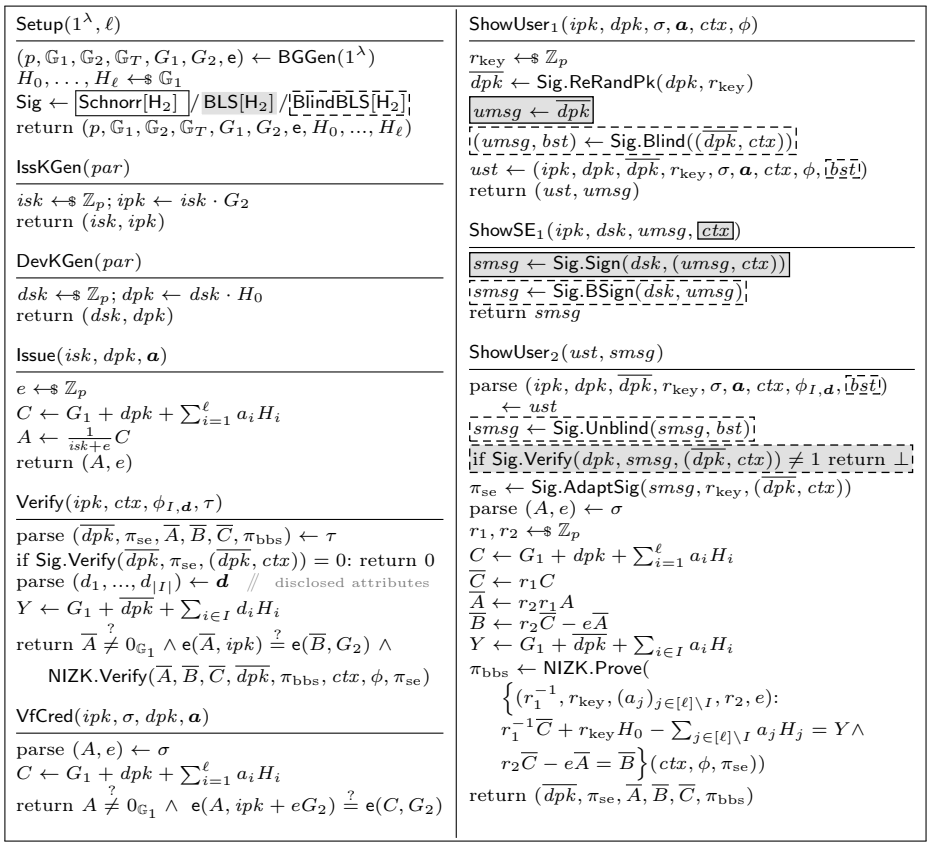

Fig. 5. Common skeleton of the constructions. BBS-Schnorr is boxed, BBS-BLS is highlighted, and BBS-BlindBLS is dash-boxed. The instantiations for the SE signature part Sig for each variant are given in the respective subsection.

6.1 Construction Blueprint

The starting point is to combine BBS credentials with a signature scheme for the device contribution that has public keys that fitting the BBS structure, i.e., are group elements in $\mathbb{G}_1$ of a pairing-friendly curve. This allows to embed the

device *public* key $dpk = dsk \cdot H_0$ (H_0 is a dedicated generator used in the BBS credential) as an implicit attribute, which the issuer "blindly" signs along with the user's attributes. The user obtains a credential on $(dsk, \boldsymbol{a})$ (where $\boldsymbol{a}$ denotes the vector of her attributes) but does not know dsk explicitly.

The intriguing idea put forward by Desmoulins et al. [**?**] is to rely on a *key-homomorphic* signature scheme [**?**,**?**] for the device part, and decouple the device signature contribution and the BBS proof when creating device-bound presentations. A *key-homomorphic* or *adaptable* signature scheme enables a party holding a signature sig under public key pk for message m to compute a signature sig' which is valid under a related pk' on message m. At a syntactic level, the Desmoulins et al. idea is compatible with any key-homomorphic signature scheme where key pairs are of the form $(dsk, dsk \cdot H_0)$ and there is an adaptation algorithm $\mathsf{AdaptSig}$ such that for $r_{\text{key}} \in \mathbb{Z}_p$ and any signature sig under pk on m we have that $sig' \leftarrow \mathsf{AdaptSig}(sig, r_{\text{key}}, m)$ is a valid signature under the public key $pk + r_{\text{key}} H_0$, still for message m.

In the blueprint construction, this mechanism is used during a showing as follows. The user first samples $r_{\text{key}} \leftarrow\!\!\$\ \mathbb{Z}_p$ and sets $\overline{dpk} \leftarrow dpk + r_{\text{key}} H_0$. Then they ask the SE for a signature sig on message $(\overline{dpk}, ctx)$. The user computes $sig' \leftarrow \mathsf{AdaptSig}(sig, r_{\text{key}}, (\overline{dpk}, ctx))$. Finally, they prove knowledge of a BBS credential adapted for $\overline{dpk}$, and reveal both $\overline{dpk}$ and the adapted signature sig' in the clear alongside the BBS proof of knowledge.

Desmoulins et al. [**?**] presented this idea specifically for weak Fiat-Shamir Schnorr and pre-hashed ECDSA as the key-homomorphic device signature, and did not give an in-depth security analysis. In particular, their work simply assumes that the device signature serves as a straight-line extractable proof of knowledge of the secret key without any formal justification that this holds when composing the signature with other components of the scheme (BBS). In our analysis we provide a formal proof that this holds for both weak Fiat-Shamir Schnorr and BLS signatures, where the latter is non-trivial. Our use of the key-homomorphic signature syntax is for presentation purposes only, and we emphasize that we do not have a generic theorem for arbitrary key-homomorphic signature schemes, due to the aforementioned extraction issues.

We show both the blueprint and the construction-specific algorithms in Figure 5. For $\ell \in \mathbb{N}$, all of our constructions are defined for the message space $\mathcal{M}_{par} := \mathbb{Z}_p^\ell$ and predicate family $\Phi_{par} := \{\phi_{I,\boldsymbol{d}}\}$ where $\phi_{I,\boldsymbol{d}}(a_1, \ldots, a_\ell) = 1$ if $a_i = \boldsymbol{d}_i$ for all $i \in I$, and otherwise 0, i.e., the "selective disclosure" predicate family. Where these details are not important, we shorten $\phi_{I,\boldsymbol{d}}$ to ϕ. All of our constructions make use of a bilinear group generator BGGen.

Credential Issuance. Credentials in our construction are standard BBS credentials with $\ell + 1$ attributes. Device keys are of the form $(dsk, dsk \cdot H_0)$. To keep this key secret, issuance is modified slightly to include the group element dpk directly instead of specifying dsk as an ordinary attribute. In practice, the issuer will need to ensure the user is indeed in possession of the dsk, but we

omit this step in our model. The issuer samples a nonce $e \in \mathbb{Z}_p$ and computes $\sigma \leftarrow \left(\frac{1}{isk+e}(G_1 + dpk + \sum_{i=1}^{\ell} a_i H_i), e\right)$.

Presentation. The "secure splitting" technique splits the presentation of the BBS credential into two parts, computed jointly between SE and user. The SE knows dsk, whereas the user holds the credential and knows the attributes. The goal is to allow the user to compute a presentation tied to some ctx, but require them to interact with the SE who approves the ctx. For simplicity, consider the setting where $\ell = 2$ and the first attribute is disclosed. As a warm-up, think of the user proving they possess a BBS signature on group element C and that they know a_2 such that $C - a_2 H_2 = G_1 + dpk + a_1 H_1$, while the SE provides a signature on ctx under dpk. At the moment, this does not yet provide unlinkability, because dpk stays the same. To get around this, we can take advantage of the signature key-homomorphism. The user samples a blinding factor r_{key} and re-randomizes dpk as $\overline{dpk} \leftarrow dpk + r_{\text{key}} H_0$. The SE computes a signature sig on message $(\overline{dpk}, ctx)$, which the user transforms into a signature sig' under $\overline{dpk}$ using $\mathsf{AdaptSig}$. Then (still simplified) the user proves knowledge of a BBS signature on C and that they know r_{key}, a_2 such that $C + r_{\text{key}} H_0 - a_2 H_2 = G_1 + \overline{dpk} + a_1 H_1$, and they reveal the transformed SE signature. Figure 5 shows the values $ctx, \phi, \pi_{\text{se}}$ that the NIZK must be bound to for our security model.[4] Verification follows naturally by verifying π_{se} is valid under the re-randomized $\overline{dpk}$ and verifying the NIZK.

$\overline{dpk}$ Is Not a Pseudonym. Recall from Section 3 that DBAC does not include pseudonyms by design. We stress that $\overline{dpk}$, although its use in the construction might resemble that of a pseudonym at first glance, does not behave as such. A $\overline{dpk}$ is in no way linked to a specific underlying dpk, in particular, two colluding SEs can easily target the same $\overline{dpk}$ in a presentation.

6.2 BBS-Schnorr Construction (Unlink-1)

The first construction BBS-Schnorr instantiates our blueprint from Figure 5 with (weak Fiat-Shamir) Schnorr signatures as the SE's key-homomorphic signature scheme, following the BBS# proposal [?]. The protocol has not been formally analyzed in [?], and we prove that it satisfies the desired unforgeability, but only weak privacy (Unlink-1), requiring a fully trusted SE.

Schnorr Signatures. The Schnorr signature scheme is a signature of knowledge of the secret key in the formal sense: an extractor can extract the secret key from an algorithm that produces Schnorr signatures, via rewinding in the ROM [?]. In fact, Schnorr can be generalized to a signature of knowledge of a witness for *any linear discrete-logarithm relationship*, and extraction is straight-line in the AGM+ROM [?]. Schnorr is pairing-free, but for convenience, we directly make use of the bilinear DBAC parameters, in particular, the generator H_0 for

[4] In real-world deployments, standard practice is to hash all public values to ensure the strongest possible notion of non-malleability, e.g. *par* and *ipk*.

the public key. This way, the signature keys already fit their use in the BBS credential. We use the weak Fiat-Shamir transform version of Schnorr and parameterize it in terms of a hash function $\mathsf{H} : \{0,1\} \to \mathbb{Z}_p$. For simplicity, we specify the scheme for signing a single message, as used in the construction.

$\mathsf{Schnorr.KGen}(par)$**:** sample $sk \leftarrow\!\!\$\ \mathbb{Z}_p$ and set $pk \leftarrow sk \cdot H_0$, output (sk, pk)
$\mathsf{Schnorr.Sign}(sk, m)$**:** sample $\omega \leftarrow\!\!\$\ \mathbb{Z}_p$, set $r \leftarrow \omega H_0$, compute $c = \mathsf{H}(r, m)$ and $s \leftarrow \omega + c \cdot sk$, output (c, s)
$\mathsf{Schnorr.Verify}(pk, sig, m)$**:** parse $(c, s) \leftarrow sig$ and output 0 if this fails; then output 1 if $c = \mathsf{H}(sH_0 - c \cdot pk, m)$, else 0
$\mathsf{Schnorr.ReRandPk}(pk, r_{\mathbf{key}})$**:** return $pk + r_{\text{key}} \cdot H_0$
$\mathsf{Schnorr.AdaptSig}(sig, r_{\mathbf{key}}, m)$**:** parse $(c, s) \leftarrow sig$ and output $(c, s + c \cdot r_{\text{key}})$

Now we are ready to define the construction formally:

Definition 5 (BBS-Schnorr). *The (context-aware)* DBAC *scheme BBS-Schnorr is defined w.r.t. the predicate family "selective disclosure"* $\Phi \coloneqq \{\phi_{I,\boldsymbol{d}}\}$ *and message space* $\mathcal{M} \coloneqq \mathbb{Z}_p^\ell$ *as a set of algorithms* $(\mathsf{Setup}, \mathsf{IssKGen}, \mathsf{DevKGen}, \mathsf{Issue}, \mathsf{ShowUser}, \mathsf{ShowSE}, \mathsf{Verify})$ *given in Figure 5, using signature algorithms* $\mathsf{Sig} = (\mathsf{Sign}, \mathsf{Verify}, \mathsf{ReRandPk}, \mathsf{AdaptSig})$ *defined according to* Schnorr *instantiated with a hash function* $\mathsf{H}_2 : \{0,1\}^* \to \mathbb{Z}_p$*. The NIZK uses* $\mathsf{H}_1 : \{0,1\}^* \to \mathbb{Z}_p$*.*

Unforgeability. We formally describe the extractor and prove that the BBS-Schnorr construction satisfies unforgeability in Appendix F. Here we give a brief overview of the result and proof.

Theorem 1 (BBS-Schnorr UNF). *If DL is hard in* $\mathbb{G}_1$ *for* BGGen*, and* BBS *and* Schnorr *satisfy EUF-CMA, then BBS-Schnorr is* UNF *in the ROM+AGM.*

Proof (sketch). The extractor works by using standard AGM techniques to straight-line extract the witness for the NIZK proving knowledge of the BBS credential. In particular, this allows for extracting r_{key}, and the extractor simply outputs $\overline{dpk} - r_{\text{key}} H_0$. The honest user showing oracle is simulated by programming H_1 outputs. Using the fact that *dsk* is known for every SE, we describe how the extractor can be modified to extract dsk_1^* such that $dsk_1^* H_0 = dpk_1^* = \overline{dpk} - r_{\text{key}} H_0$. With this in place, we first rule out three classes of forgeries: *attribute*, *invalid key*, and *malleability* forgeries, showing that any forgery of this type can be used to break the EUF-CMA security of BBS. Our next goal is to rule out *context* forgeries, but this first requires a change to the extractor. We need to rule out the possibility of $\mathcal{A}$ giving an algebraic representation in terms of some uncorrupted dpk_i, and we do so by using a guessing argument and showing that such an adversary breaks DL. Finally we can rule out context forgeries, which we do by describing a reduction which plays the EUF-CMA game for Schnorr by guessing the i^* such that the extracted key is dpk_i, and simulating any queries to $\mathcal{O}_{\mathsf{ShowSE}}$ using the signing oracle. By the prior argument about the extractor, we know that $\mathcal{A}$ cannot break the extractor by abusing the fact that the reduction does not know the discrete logarithm of dpk_i.

Privacy. BBS-Schnorr achieves only the lower UNLINK-0 and UNLINK-1 properties, which both assume honest SEs. Intuitively, if both the SE and user part of a presentation are executed honestly, then $\overline{dpk}$ is information-theoretically blinded, and the NIZK over the credential hides non-disclosed attributes and the true dpk. We start by proving UNLINK-1 (UNLINK-0 follows by implication), then outline how unlinkability breaks in the presence of corrupt SEs.

Theorem 2 (BBS-Schnorr UNLINK-1). *If* $\mathsf{BBS}[\mathsf{H}_1]$ *is zero-knowledge and* H_1 *is a RO, then for leakage* $\mathcal{L} : isk \to (U \leftarrow\!\$\ \mathbb{G}_1, isk \cdot U)$, *BBS-Schnorr is UNLINK-1.*

Proof (sketch). BBS-Schnorr presentations can be created by a simulator oblivious to dsk, dpk, σ and $\boldsymbol{a}$, as follows. Since the key-homomorphism hides perfectly what key originally signed the SE signature, the simulator might as well use an independent key pair $(dsk', dpk') \leftarrow \mathsf{DevKGen}(par)$ to sign. We make use of the BBS zero-knowledge simulator $\mathsf{Sim}_{\mathsf{BBS}}$ to create π_{bbs}, but first need to generate fitting public values $\overline{A}, \overline{B}$, and $\overline{C}$. The latter can be totally random, but we need to ensure $\overline{B} = isk \cdot \overline{A}$. This is a small technical challenge, as the simulator only has $ipk \in \mathbb{G}_2$ at hands, but $\overline{A}$ and $\overline{B}$ are in $\mathbb{G}_1$. Therefore, as noted by [**?**], we will need to supply a leakage $(U \leftarrow\!\$\ \mathbb{G}_1, isk \cdot U)$ to the simulator, essentially an additional issuer public key in $\mathbb{G}_1$. Sim re-randomizes this leakage to obtain $\overline{A} \leftarrow \delta U$ and $\overline{B} \leftarrow \delta iskU$, and has $\mathsf{Sim}_{\mathsf{BBS}}$ create the NIZK.

We show UNLINK-1 by transitioning from real to simulated presentations in three main steps: we start by signing with dsk', but ensure to re-randomize key and signature into the original $\overline{dpk}$ (setting $r_{\text{key}}{}' \leftarrow dsk + r_{\text{key}} - dsk'$), such that π_{bbs} does not break since we still know the mapping from the credential's dpk to $\overline{dpk}$. Next, we let $\mathsf{Sim}_{\mathsf{BBS}}$ create π_{bbs}. Finally, we are able to freely choose the re-randomization of dsk', which gets rid of the last dependence to dsk. The full proof can be found in Appendix G.1.

No privacy against compromised SEs. BBS-Schnorr does not guarantee privacy if an adversary compromises SEs; even against algorithm substitution attacks. In short, standard techniques for subverting signatures can be used to embed additional information in randomness of the SE-created Schnorr signature that is part of a presentation. Thus the device's identity could be leaked to a colluding verifier, breaking UNLINK-2. We describe this attack in more detail in Appendix G.1. Additionally, this construction does not achieve SE-OBLIV since the blind $\overline{dpk}$ observable by the SE allows to re-identify presentations. Note how this leakage was not exploitable in UNLINK-2 since $\mathcal{A}$ has no full view of the SE, and the SE is also stateless, so it cannot record what it is sees and tell an outside adversary later.

6.3 BBS-BLS Construction (UNLINK-2)

The Schnorr-based construction is not subversion-resilient, since the signatures include signer-chosen randomness which can be abused as a subliminal channel from the SE to the outside. To eliminate this degree of freedom for a corrupted

SE, our second construction replaces Schnorr with a deterministic signature scheme, namely BLS.

This yields the first subversion-resilient DBAC construction that is compatible with BBS credentials. Apart from satisfying stronger privacy than BBS-Schnorr, this version also enables for flexible support of distributed device binding, as BLS signatures can easily be thresholdized or aggregated for multi-signatures.

BLS Signatures. The Boneh-Lynn-Shacham (BLS) [?] signature scheme is defined with respect to BGGen and a hash function $\mathsf{H} : \{0,1\} \to \mathbb{G}_2$. There are several instantiations of BLS signatures (see also the IETF standard [?]), here for compatibility with the BBS credential we use a type-3 pairing version with keys in $\mathbb{G}_1$ and signatures in $\mathbb{G}_2$. This version is proven secure under the co-CDH assumption in the ROM [?]. We specify the algorithms below, again re-using the DBAC parameters. BLS.ReRandPk is identical to Schnorr.

$\mathsf{BLS.KGen}(par)$**:** sample $sk \leftarrow\!\$\ \mathbb{Z}_p$ and set $pk \leftarrow sk \cdot H_0$, output (sk, pk)
$\mathsf{BLS.Sign}(sk, m)$**:** output $sig \leftarrow sk \cdot \mathsf{H}(m)$
$\mathsf{BLS.Verify}(pk, sig, m)$**:** output $\mathsf{e}(H_0, sig) = \mathsf{e}(pk, \mathsf{H}(m))$.
$\mathsf{BLS.AdaptSig}(sig, r_{\mathbf{key}}, m)$**:** output $sig + r_{\mathrm{key}} \cdot \mathsf{H}(m)$

Thus, BBS-BlindBLS is our blueprint using BLS signatures as Sig:

Definition 6 (BBS-BLS). *The (context-aware)* DBAC *scheme BBS-BLS is defined w.r.t. the predicate family "selective disclosure"* $\Phi := \{\phi_{I,\boldsymbol{d}}\}$ *and message space* $\mathcal{M} := \mathbb{Z}_p^\ell$ *as a set of algorithms* $(\mathsf{Setup}, \mathsf{IssKGen}, \mathsf{DevKGen}, \mathsf{Issue}, \mathsf{ShowUser}, \mathsf{ShowSE}, \mathsf{Verify})$ *given in Figure 5, using signature algorithms* $\mathsf{Sig} = (\mathsf{Sign}, \mathsf{Verify}, \mathsf{ReRandPk}, \mathsf{AdaptSig})$ *defined according to* BLS *instantiated with a hash function* $\mathsf{H}_2 : \{0,1\}^* \to \mathbb{Z}_p$*. The NIZK uses* $\mathsf{H}_1 : \{0,1\}^* \to \mathbb{Z}_p$*.*

Unforgeability. The full proof can be found in Appendix F.2 and mainly follows the same structure of Theorem 1. Below we sketch the main technical differences.

Theorem 3 (BBS-BLS UNF). *If DL is hard in* $\mathbb{G}_1$ *and* $\mathbb{G}_2$ *for* BGGen *and* BBS *and* BLS *satisfy EUF-CMA, then BBS-BLS satisfies* UNF *in the ROM+AGM.*

Proof (sketch). A minor difference is that to rule out context forgeries we obviously reduce to the EUF-CMA security of BLS instead of Schnorr. The bigger issue is showing that we can straight-line extract the secret key from the adapted BLS signature, and for this we introduce a new assumption which we prove hard in the AGM if DL is hard in both $\mathbb{G}_1$ and $\mathbb{G}_2$. This lemma can be seen as an analog of the result on the tight EUF-CMA security of BLS in the AGM by Fuchsbauer et al. [?], except that here we are in the type-3 pairing setting, and additional difficulties are encountered due to the composition of BLS with the rest of the scheme. Roughly speaking, the lemma says that given generators $H_0, H_1, \ldots, H_\ell$, after committing to coefficients $(a_i)_{i=0}^\ell$ such that for $X = \hat{a}G_1 + \sum_{i=0}^\ell a_i H_i$ and then being given a group element $Y \in \mathbb{G}_2$, it should be hard to find σ such that

$\mathsf{e}(H_0, \sigma) = \mathsf{e}(X, Y)$ and $a \neq 0$ for some $a \in \{\hat{a}, (a_i)_{i=1}^{\ell}\}$. This analysis may be of independent interest, as it effectively shows that the type-3 version of BLS, with the *pk* hashed alongside the message, is a straight-line extractable proof of knowledge of the secret key *in a way that is fairly robust to composition*, in the AGM+ROM.

Privacy. The deterministic signature scheme BLS and the additional user-side check we introduced makes our second construction achieve Unlink-2 – privacy in the presence of subverted SEs. Intuitively, limiting the SE to a fixed artifact that is accepted and forwarded by the user to a potential outside adversary prevents embedding even a single bit of information into this artifact.

Theorem 4 (BBS-BLS Unlink-2). *If* $\mathsf{BBS}[\mathsf{H}_1]$ *is zero-knowledge and* H_1 *a RO, then for leakage* $\mathcal{L} : isk \to (U \leftarrow\!\$\ \mathbb{G}_1, isk \cdot U)$, *BBS-BLS is Unlink-2.*

Proof (sketch). We follow the same simulation strategy as for BBS-Schnorr, but preprend an additional game hop that excludes that $\widetilde{\mathsf{ShowSE}}$ outputs anything else than an honest SE signature (or else, $\mathcal{O}_{\mathsf{DoPres}}$ would abort and no simulation is necessary). This is easy to argue, since BLS is deterministic and $\mathsf{ShowUser}_2$ rejects anything but the unique correct BLS signature. The full proof can be found in Appendix G.2.

BBS-BLS Does Not Achieve Unlink-3 or SE-Obliv. The construction reveals to the SE the message $(\overline{dpk}, ctx)$ that it is signing, and this allows a malicious SE to recognize the presentations it is later signing. Formally, this allows a simple attack against both Unlink-3 and SE-Obliv, and both in the context-aware and blind versions: first, call $\mathcal{O}_{\mathsf{ShowUser},1}$ and take note of the $\overline{dpk}$ contained in the returned *umsg*. Respond with an honestly computed *smsg* to $\mathcal{O}_{\mathsf{ShowUser},2}$. If the $\overline{dpk}$ in the resulting presentation matches the earlier key, we are in the real world, if there is a mismatch, the presentation was created by a simulator disconnected from the first round of interaction.

6.4 BBS-BlindBLS Construction (BDBAC, Unlink-3, and SE-Obliv)

To achieve our strongest privacy notion in the context-blind setting and SE obliviousness, we replace BLS with *blind BLS*, yielding our BBS-BlindBLS scheme. This is the first construction for device-bound credentials that can safely be used in the remote HSM setting envisioned by the EUDI. Generally, it enables a natural deployment path to use device-bound BBS credentials until local SEs are equipped with the necessary APIs. Further, note that this construction is compatible with BBS-BLS as both have identical issuance and verification. That is, the two schemes can be run in parallel and support users with local SEs and users that rely on the cloud solution with the same main credential scheme.

Blind BLS. Boldyreva [**?**] gave a blind signature version of BLS and showed it to be OM-Unf under a one-more-CDH assumption in gap groups. We give the blind algorithms (with *multiplicative* blinding) in the following; all other algorithms including the key homomorphism remain unchanged from plain BLS. The statistical blinding of BlindBLS hides from the SE what it is signing, thus overcoming the unlinkability limitation we encountered in the prior construction.

$\mathsf{BlindBLS.Blind}(m)$**:** sample $bst \leftarrow\!\!\$\ \mathbb{Z}_p$, compute $\overline{m} \leftarrow bst \cdot \mathsf{H}(m)$, output $(\overline{m}, bst)$
$\mathsf{BlindBLS.BSign}(sk, \overline{m})$**:** compute $\overline{sig} \leftarrow sk \cdot \overline{m}$ and output $\overline{sig}$
$\mathsf{BlindBLS.Unblind}(\overline{sig}, bst)$**:** compute $sig \leftarrow (bst)^{-1} \cdot \overline{sig}$ and return sig

Again, the construction is then simply defined as the blueprint combined with the chosen signature scheme:

Definition 7 (BBS-BlindBLS). *The* BDBAC *scheme BBS-BlindBLS is defined w.r.t. the predicate family "selective disclosure"* $\Phi := \{\phi_{I,\boldsymbol{d}}\}$ *and message space* $\mathcal{M} := \mathbb{Z}_p^\ell$ *as the set of algorithms* (Setup, IssKGen, DevKGen, Issue, ShowUser, ShowSE, Verify) *given in Figure 5, using signature algorithms* Sig = (Blind, BSign, Unblind, Verify, ReRandPk, AdaptSig) *defined according to* BLS *instantiated with a hash function* $\mathsf{H}_2 : \{0,1\}^* \to \mathbb{Z}_p$*. The NIZK uses* $\mathsf{H}_1 : \{0,1\}^* \to \mathbb{Z}_p$*.*

Unforgeability. The full proof, given in Appendix F.2, is quite similar to the proof of Theorem 3. The main difference compared to the context-aware setting is an additional hybrid where we rule out the possibility of a one-more forgery by reduction to the (weak) one-more unforgeability of blind BLS.

Theorem 5 (BBS-BlindBLS Unf). *If DL is hard in* $\mathbb{G}_1$ *and* $\mathbb{G}_2$ *for* BGGen*,* BBS *satisfies EUF-CMA security, and* BlindBLS *satisfies (weak) one-more unforgeability, then BBS-BlindBLS satisfies* BDBAC *Unf in the ROM+AGM.*

Privacy. BBS-BlindBLS achieves the strongest privacy guarantee Unlink-3 and context-blind SE-Obliv: not only does it limit the SE to a unique acceptable output, hindering any form of subliminal channel, but it also hides the signed message from the SE via *blind* signatures. We first show SE-Obliv relying on the statistical blindness of BlindBLS. Once we have established that the intermediate *umsg* the SE sees reveals nothing about the presentation the SE helps to create, we employ the same strategy and simulator as for BBS-BLS to show the entire presentation is simulatable and Unlink-3 is indeed achieved.

Theorem 6 (BBS-BlindBLS SE-Obliv). *Let* H_2 *be a RO. Then for the empty leakage function* $\mathcal{L} : isk \to \bot$*, BBS-BlindBLS is SE-Obliv.*

Proof (sketch). SE obliviousness is justified by the statistical blinding of blind BLS – the adversary learns nothing from the output of $\mathcal{O}_{\mathsf{ShowUser},1}$ except in the negligibly unlikely case that $\mathsf{H}_2(\overline{dpk}, ctx) = 0$. Therefore, we may simulate $\mathcal{O}_{\mathsf{ShowUser},1}$ with a "blinded" random element B. Sim.Check can still ensure an honest SE action by testing $\mathsf{e}(G_1, (bst)^{-1} \cdot smsg) \stackrel{?}{=} \mathsf{e}(dpk, B)$. After this change,

since we evidently did not commit to any particular presentation previously, we can simply compute a fresh presentation in $\mathcal{O}_{\mathsf{ShowUser},2}$. The full proof can be found in Appendix G.3.

We build on this result to show UNLINK-3 next. This proof solely repeats arguments from prior proofs, so we omit the full specification.

Theorem 7 (BBS-BlindBLS UNLINK-3). *If* $\mathsf{BBS}[\mathsf{H}_1]$ *is zero-knowledge and* H_1 *and* H_2 *are ROs, then for leakage function* $\mathcal{L} : isk \to (U \leftarrow\!\!\$\ \mathbb{G}_1, isk \cdot U)$, *BBS-BlindBLS is UNLINK-3.*

Proof (sketch). Since BBS-BlindBLS achieves SE-OBLIV, in the real world of UNLINK-3 ($b = 1$) we can replace the creation of *umsg* and verification of *smsg* by the simulation used in Theorem 6, such that *umsg* is completely unrelated to the final presentation. Then we create a presentation τ_1 from scratch in $\mathcal{O}_{\mathsf{ShowUser},2}$. Next, we repeat the strategy from Theorem 4 to transition from a real presentation to one independent of the actual dpk, $\boldsymbol{a}$ and σ. Once the presentation is entirely simulatable, we switch to the ideal case in $\mathcal{O}_{\mathsf{ShowUser},2}$ and output the simulated τ_0. Finally, to get to the random world of UNLINK-3 ($b = 0$), we switch $\mathcal{O}_{\mathsf{ShowUser},1}$ back to normal. At this point, the $(\overline{dpk}, ctx)$ created in $\mathcal{O}_{\mathsf{ShowUser},1}$ is no longer in use by $\mathcal{O}_{\mathsf{ShowUser},2}$ and, in particular, never output, so $\mathsf{H}_2(\overline{dpk}, ctx)$ is distributed like a random message. Simultaneously, we change the check on the response *smsg* back to the original signature verification, such that the behavior of $\mathcal{O}_{\mathsf{ShowUser},2}$ remains the same.

7 Conclusion

In this work we have formalized and designed device-bound anonymous credentials, considering different levels of trust put into the SE. Whereas BBS-Schnorr requires the SE to be honest for privacy, BBS-BLS remains secure even for subverted hardware and BBS-BlindBLS even tolerates fully malicious SEs. This makes the latter a suitable candidate for the remote HSM setting currently considered for the EUDI wallet in Europe. An advantage of our BBS-Schnorr construction is that it only requires computations in $\mathbb{G}_1$ for the SE (during presentations), whereas both BLS-based schemes work in $\mathbb{G}_2$. Thus, we expect BBS-Schnorr to have a slightly better performance on constraint hardware.

Finally, we note that there are also alternative approaches that do not technically enforce non-transferability, but rather disincentivize it through "all-or-nothing" sharing [**?**,**?**]. That is, the only enforcement is that all credentials are bound to a single high-value secret of the user, and sharing of any credential would then reveal this secret too. This mainly targets malicious users though, but does not protect honest users from compromise of their credentials.

Acknowledgements

Franklin Harding and Anna Lysyanskaya are supported by NSF Grants 2312241, 2154170, and 2247305, as well as an award from Facebook. Anja Lehmann thanks

Emil Lundberg for fruitful discussions on practical device binding for BBS credentials which has inspired this work. Her work was partially funded by SPRIND, the Federal Agency for Breakthrough Innovation. The authors are solely responsible for the content of this publication; the positions presented here do not reflect the views of SPRIND.

A Extended Preliminaries

Algebraic adversary. Let Π be any cryptographic system or protocol that requires a setup step for an algebraic group; in the bilinear setting, this would be $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, G_1, G_2, \mathsf{e}) \leftarrow \mathsf{BGGen}(1^\lambda)$. (In the non-bilinear setting, some of these would be $\perp$.) Suppose an adversary $\mathcal{A}$ is playing a security game with a challenger for a Π. We say that $\mathcal{A}$ is algebraic with respect to BGGen if, whenever it sends a group element $H \in \mathbb{G}_1$ (resp., $\in \mathbb{G}_2$) to the challenger, it also produces an explanation $c_1, \ldots, c_n \in \mathbb{Z}_p$ such that $H = \sum_{i=1}^n c_i H_i$ (resp. $H = \sum_{i=1}^n c_i K_i$), where $H_1, \ldots H_n \in \mathbb{G}_1$ (resp. $K_1, \ldots, K_m \in \mathbb{G}_2$) are elements of $\mathbb{G}_1$ (resp. $\mathbb{G}_2$) the adversary has received from the challenger so far.

Cryptographic assumptions. The figure below formally defines the discrete logarithm (DL) and $\mathsf{rel\text{-}DL}$ games which we use in our analyses. Note that we define DL for either group $\mathbb{G}_1$ and $\mathbb{G}_2$ at once, where the global parameter i decides which group the challenge is from. Jaeger and Tessaro showed the $\mathsf{rel\text{-}DL}$ assumption to be tightly equivalent to DL [**?**].

Game $\mathsf{DL}^{\mathcal{A}}_{\mathsf{BGGen},i}(\lambda)$	Game $\mathsf{rel\text{-}DL}^{\mathcal{A}}_{\mathsf{BGGen},n}(\lambda)$
$par \leftarrow (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, G_1, G_2, \mathsf{e}) \leftarrow \mathsf{BGGen}(1^\lambda)$ $x \leftarrow\!\!\$\ \mathbb{Z}_p$ $X \leftarrow x\mathbb{G}_i$ $x' \leftarrow \mathcal{A}(1^\lambda, par, i, X)$ return $x'G_i \stackrel{?}{=} X$	$par \leftarrow (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, G_1, G_2, \mathsf{e}) \leftarrow \mathsf{BGGen}(1^\lambda)$ $(X_i)_{i=0}^n \leftarrow\!\!\$\ \mathbb{G}_1^n$ $(\hat{y}, (y_i)_{i=0}^n) \leftarrow \mathcal{A}(par, (X_i)_{i=0}^n)$ if $(y_i)_{i=0}^n = 0^n$: return 0 return $\sum_{i=0}^n y_i X_i \stackrel{?}{=} \hat{y}G_1$,

Fig. 6. The discrete logarithm (DL) and $\mathsf{rel\text{-}DL}$ games, defined with respect to a bilinear group generator BGGen.

Signature scheme correctness and unforgeability. For correctness, we require that for any parameters *par* output from $\mathsf{Sig.Setup}$, key pair (sk, pk) output from $\mathsf{Sig.KGen}$, and messages $\boldsymbol{m} \in \mathcal{M}^\ell$, we have that $sig \leftarrow \mathsf{Sig.Sign}(sk, \boldsymbol{m})$ always satisfies $\mathsf{Sig.Verify}(pk, sig, \boldsymbol{m}) = 1$. The standard notion of security for signatures is existential unforgeability under chosen message attacks (EUF-CMA). For an adversary $\mathcal{A}$ we define $\mathsf{Adv}^{\text{EUF-CMA}}_{\mathsf{GGen},\mathsf{Sig},\ell}(\mathcal{A}, \lambda)$ as the probability that the following experiment outputs 1; unforgeability requires that $\mathsf{Adv}^{\text{EUF-CMA}}_{\mathsf{GGen},\mathsf{Sig},\ell}(\mathcal{A}, \lambda)$ be negligible:

1. $par \leftarrow \mathsf{Sig.Setup}(1^\lambda, \ell)$ and set $S \leftarrow \emptyset$
2. $(sk, pk) \leftarrow \mathsf{Sig.KGen}(par)$
3. $(\boldsymbol{m}^*, sig^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Sign}}}(1^\lambda, pk)$, where $\mathcal{O}_{\mathsf{Sign}}(\boldsymbol{m})$ first sets $S \leftarrow S \cup \{\boldsymbol{m}\}$ and then returns $\mathsf{Sig.Sign}(sk, \boldsymbol{m})$
4. output $(\mathsf{Sig.Verify}(pk, sig^*, \boldsymbol{m}^*) \wedge \boldsymbol{m}^* \notin S)$

For correctness we similarly require that for any *par*, (sk, pk) and $\boldsymbol{m}$, the following experiment always outputs 1:

1. $(bst, \overline{m}) \leftarrow \mathsf{BSig.Blind}(pk, \boldsymbol{m})$
2. $\overline{sig} \leftarrow \mathsf{BSig.Sign}(sk, \overline{m})$
3. $sig \leftarrow \mathsf{BSig.Unblind}(bst, \overline{sig})$
4. output $\mathsf{BSig.Verify}(pk, sig, \boldsymbol{m})$

Blind signature scheme unforgeability. The typical notion of (weak) unforgeability for a round-optimal blind signature scheme is (weak) one-more unforgeability (OM-Unf). For an adversary $\mathcal{A}$ we define $\mathsf{Adv}^{\text{OM-Unf}}_{\mathsf{GGen},\mathsf{BSig},\ell}(\mathcal{A}, \lambda)$ as the probability that the following experiment outputs 1; OM-Unf requires that this advantage be negligible:

1. $par \leftarrow \mathsf{Sig.Setup}(1^\lambda, \ell)$ and $k \leftarrow 0$
2. $(sk, pk) \leftarrow \mathsf{Sig.KGen}(par)$
3. $(\boldsymbol{m}_i^*, sig_i^*)_{i=1}^n \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{BSign}}}(1^\lambda, pk)$, where $\mathcal{O}_{\mathsf{BSign}}(\overline{m})$ first sets $k \leftarrow k+1$ and then outputs $\mathsf{BSign}(sk, \overline{m})$
4. if $\exists i \neq j : \boldsymbol{m}_i^* = \boldsymbol{m}_j^*$ then output 0
5. if $\exists i : \mathsf{Verify}(pk, \boldsymbol{m}_i^*, sig_i^*) = 0$ then output 0
6. output $(k < n)$

B Correctness of DBAC

Formally, a (B)DBAC scheme is η-correct if for any $\lambda, \ell \in \mathbb{N}$; $par \leftarrow \mathsf{Setup}(1^\lambda, \ell)$; $(isk, ipk) \leftarrow \mathsf{IssKGen}(par)$; $(dsk, dpk) \leftarrow \mathsf{DevKGen}(par)$; $ctx \in \{0,1\}^*$; $\boldsymbol{a} \in \mathcal{M}$ and $\phi \in \Phi$ with $\phi(\boldsymbol{a}) = 1$ the following experiment outputs 1 with probability of at least $1 - \eta(\lambda)$ (ctx in ShowSE is omitted for BDBAC):

1. $\sigma \leftarrow \mathsf{Issue}(isk, dpk, \boldsymbol{a})$
2. $(\tau, \perp) \leftarrow \langle \mathsf{ShowUser}(ipk, dpk, \sigma, \boldsymbol{a}, ctx, \phi) \rightleftharpoons \mathsf{ShowSE}(dsk, ipk, ctx) \rangle$
3. output $\mathsf{VfCred}(ipk, \sigma, dpk, \boldsymbol{a}) \wedge \mathsf{Verify}(ipk, \tau, ctx, \phi)$

C Extended Privacy Definitions

In the following, we extend our privacy notions over multiple users that can each hold multiple credentials. We show that the simple single-user, single-credential definitions imply the extended definitions in most cases and describe under what circumstances the implication might not hold. Finally, we want to increase our model's re-usability and provide a more generic definition applicable to schemes with an arbitrary number of rounds in the interaction between SE and user.

C.1 Extended Unlink-0

For presentation purposes we add the extensions in two steps. First, we keep the single-user setting, but allow the adversary to use multiple credentials throughout the game. Then, we switch to a multi-user setting. Figure 7 shows both extended versions of Unlink-0.

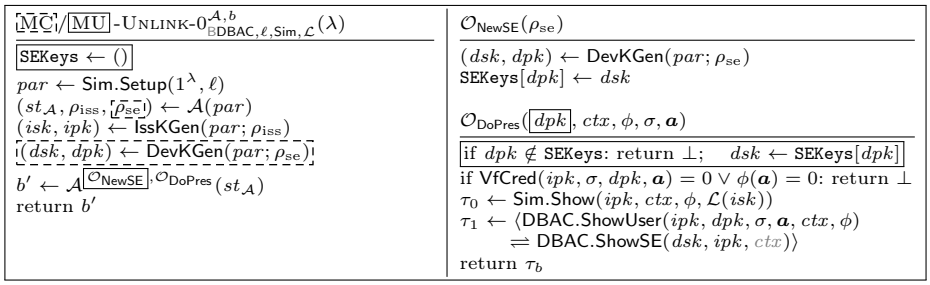

Fig. 7. Extended UNLINK-0: a single-user, but multi-credential version (plain code and dashed boxes), and a multi-user, multi-credential version (plain code and solid boxes). For BDBAC, the *ctx* is not sent to the SE, indicated by gray text color.

Multi-Credential UNLINK-0. Permitting multiple credentials adds the arguments $(\sigma, \boldsymbol{a})$ to the $\mathcal{O}_{\mathsf{DoPres}}$ oracle. We, again, require that the parameters generated by $\mathsf{Sim.Setup}$ are indistinguishable from honest ones. The corresponding advantage is defined analogously to the simple property as:

$$\mathsf{Adv}^{\text{MC-UNLINK-0}}_{\mathbb{B}\mathsf{DBAC},\ell,\mathsf{Sim},\mathcal{L}}(\mathcal{A}, \lambda) :=$$
$$|\Pr[\text{MC-UNLINK-0}^{\mathcal{A},b=0}_{\mathbb{B}\mathsf{DBAC},\ell,\mathsf{Sim},\mathcal{L}} = 1] - \Pr[\text{MC-UNLINK-0}^{\mathcal{A},b=1}_{\mathbb{B}\mathsf{DBAC},\ell,\mathsf{Sim},\mathcal{L}} = 1]|$$

We show that plain UNLINK-0 implies MC-UNLINK-0 through a hybrid argument in the following.

Theorem 8. *Let Π be a* (B)DBAC *scheme that achieves UNLINK-0 with respect to a simulator* $\mathsf{Sim} = (\mathsf{Setup}, \mathsf{Show})$ *such that parameters generation by* $\mathsf{Sim.Setup}$ *is indistinguishable from Π.*Setup*, and a leakage function $\mathcal{L}$. For any adversary $\mathcal{A}_{mc}$ playing MC-UNLINK-0 with* Sim *and $\mathcal{L}$ that uses q_σ distinct credentials in their calls to* $\mathcal{O}_{\mathsf{DoPres}}$*, there exists an adversary $\mathcal{A}$ playing UNLINK-0 with* Sim *and $\mathcal{L}$ such that*

$$\mathsf{Adv}^{\text{MC-UNLINK-0}}_{\Pi,\ell,\mathsf{Sim},\mathcal{L}}(\mathcal{A}_{mc}, \lambda) \leq q_\sigma \cdot \mathsf{Adv}^{\text{UNLINK-0}}_{\Pi,\ell,\mathsf{Sim},\mathcal{L}}(\mathcal{A}, \lambda)$$

Proof. Let $\mathcal{A}_{\text{mc}}$ be an adversary playing MC-UNLINK-0, where it uses q_σ distinct pairs $(\sigma, \boldsymbol{a})$ at the oracle $\mathcal{O}_{\mathsf{DoPres}}$. In the following we will only argue about distinct credentials, since there cannot be two sets of attributes matching the same credential. We prove the theorem via a hybrid argument and will transform the real world of MC-UNLINK-0 into its random world in q_σ steps.

For $i = 0, ..., q_\sigma$, define Game_i as follows:

- Game setup and structure are identical to UNLINK-0.
- In $\mathcal{O}_{\mathsf{DoPres}}$, we observe whether we saw the input credential before or not. Once we see the i^{th} distinct credential, the oracle behavior changes. For the j^{th} distinct credential, $\mathcal{O}_{\mathsf{DoPres}}$ proceeds as follows:
  - If $j \leq i$: Let Sim create the presentation, as in $\text{UNLINK-0}^{\mathcal{A},b=0}_{\mathsf{DBAC},\ell,\mathsf{Sim},\mathcal{L}}$.

- If $j > i$: Create the presentation using the DBAC interaction between SE and user, as in $\text{UNLINK-0}^{\mathcal{A},b=1}_{\mathsf{DBAC},\ell,\mathsf{Sim},\mathcal{L}}$.

Thus, Game_0 is exactly the real world of MC-UNLINK-0, and Game_{q_σ} is exactly the random world of MC-UNLINK-0.

For $i = 0, ..., q_\sigma - 1$, we bound the hop $\mathsf{Game}_i \to \mathsf{Game}_{i+1}$ by a reduction $\mathcal{A}_i$ to single-credential UNLINK-0. The reduction $\mathcal{A}_i$ works as follows.

- Initially, it receives *par* from UNLINK-0 and forwards these parameters to $\mathcal{A}_{\text{mc}}$, who responds with ρ_{iss} and ρ_{se} (we disregard the adversarial state $st_{\mathcal{A}}$ for simplicity, this also needs to be forwarded and $\mathcal{A}_i$ saves all its computations to their own state). $\mathcal{A}_i$ forwards $\rho_{\text{iss}}, \rho_{\text{se}}$ to UNLINK-0 and computes the issuer keys (isk, ipk) and device keys (dsk, dpk) itself depending on randomness ρ_{iss} and ρ_{se}, respectively.
- In $\mathcal{O}_{\mathsf{DoPres}}$, $\mathcal{A}_i$ first runs the expected VfCred check on the inputs. For all the simulated presentations for $\mathcal{O}_{\mathsf{DoPres}}$ calls with credentials up to the i^{th} credential, $\mathcal{A}_i$ runs $\mathsf{Sim.Show}$ itself. Note that it knows all necessary inputs $(ipk, ctx, \phi, \mathcal{L}(isk))$. For all the real presentations for $\mathcal{O}_{\mathsf{DoPres}}$ calls with credentials after the i^{th} credential, $\mathcal{A}_i$ runs the SE-user exchange itself. Again, it knows all necessary inputs $(ipk, dsk, dpk, \sigma, \boldsymbol{a}, ctx, \phi)$. $\mathcal{O}_{\mathsf{DoPres}}$ calls with the i^{th} credential constitute a special case: here, $\mathcal{A}_i$ answers the query with a call to its own UNLINK-0 oracle.
- The output of $\mathcal{A}_{\text{mc}}$ is forwarded to UNLINK-0.

This approach ensures that $\mathcal{A}_i$ simulates either Game_i or Game_{i+1} depending on the responses UNLINK-0 returns for the i^{th} credential. Presentations for all other credentials are computed exactly as expected. Hence, if $\mathcal{A}_{\text{mc}}$ outputs different bits b' in the two cases, then $\mathcal{A}_i$ is able to distinguish the real and random world of UNLINK-0. Summing up the q_σ reductions to single-credential UNLINK-0, we get the above bound for MC-UNLINK-0. □

Multi-user UNLINK-0. Secondly, we also allow the adversary to manipulate multiple users at once. In MU-UNLINK-0, $\mathcal{A}$ gets an oracle $\mathcal{O}_{\mathsf{NewSE}}$ to add more users to the game, for each of which it can influence the device key. $\mathcal{O}_{\mathsf{DoPres}}$ takes the intended *dpk* as an additional argument, but *dpk* must have been honestly generated beforehand. The advantage for MU-UNLINK-0 is defined as:

$$\mathsf{Adv}^{\text{MU-UNLINK-0}}_{\mathbb{B}\mathsf{DBAC},\ell,\mathsf{Sim},\mathcal{L}}(\mathcal{A},\lambda) :=$$
$$|\Pr[\text{MU-UNLINK-0}^{\mathcal{A},b=0}_{\mathbb{B}\mathsf{DBAC},\ell,\mathsf{Sim},\mathcal{L}} = 1] - \Pr[\text{MU-UNLINK-0}^{\mathcal{A},b=1}_{\mathbb{B}\mathsf{DBAC},\ell,\mathsf{Sim},\mathcal{L}} = 1]|$$

We show that MC-UNLINK-0 implies MU-UNLINK-0 through a similar argument as above.

Theorem 9. *Let Π be a* (B)DBAC *scheme that achieves MC-UNLINK-0 with respect to a simulator* $\mathsf{Sim} = (\mathsf{Setup}, \mathsf{Show})$ *such that parameters generation by* $\mathsf{Sim.Setup}$ *is indistinguishable from Π.*Setup*, and a leakage function $\mathcal{L}$. For any adversary $\mathcal{A}_{mu}$ playing MU-UNLINK-0 with* Sim *and $\mathcal{L}$ that makes q_u calls to*

$\mathcal{O}_{\mathsf{NewSE}}$*, there exists an adversary* $\mathcal{A}_{mc}$ *playing MC-*UNLINK-0 *with* Sim *and* $\mathcal{L}$ *such that*

$$\mathsf{Adv}^{\textit{MU-UNLINK-0}}_{\Pi,\ell,\mathsf{Sim},\mathcal{L}}(\mathcal{A}_{mu},\lambda) \leq q_u \cdot \mathsf{Adv}^{\textit{MC-UNLINK-0}}_{\Pi,\ell,\mathsf{Sim},\mathcal{L}}(\mathcal{A}_{mc},\lambda)$$

Proof. Again, we transform the real world of MU-UNLINK-0 into its ideal world in q_u steps, switching the presentation of one dpk from real to ideal in each game hop. Let us first define the games Game_i for $i = 0, ..., q_u$:

- Game setup and structure are identical to MC-UNLINK-0.
- We count the keys generated by $\mathcal{O}_{\mathsf{NewSE}}$ and refer to the key generated upon the j^{th} call as dpk_j in the following.
- The $\mathcal{O}_{\mathsf{DoPres}}$ oracle is operated depending on the input key dpk_j:
  - If $j \leq i$: Let Sim create the presentation, as in MC-UNLINK-$0^{\mathcal{A},b=0}_{\mathsf{DBAC},\ell,\mathsf{Sim},\mathcal{L}}$.
  - If $j > i$: Create the presentation using the DBAC interaction between SE and user, as in MC-UNLINK-$0^{\mathcal{A},b=1}_{\mathsf{DBAC},\ell,\mathsf{Sim},\mathcal{L}}$.

Thus, Game_0 is identical to the real world of MU-UNLINK-0 and Game_{q_u} is identical to the ideal world of MU-UNLINK-0.
Next, we build a reduction $\mathcal{A}_{\mathrm{mc},i}$ of each game hop to MC-UNLINK-0:

- Initially, $\mathcal{A}_{\mathrm{mc},i}$ receives *par* from MC-UNLINK-0 and forwards these parameters to $\mathcal{A}_{\mathrm{mu}}$, who responds with ρ_{iss} (we again disregard the adversarial states for simplicity). $\mathcal{A}_i$ computes the issuer keys (isk, ipk) depending on randomness ρ_{iss}. It does not yet fix the keys to MC-UNLINK-0 but first starts $\mathcal{A}_{\mathrm{mu}}$.
- When $\mathcal{A}_{\mathrm{mu}}$ makes a call to $\mathcal{O}_{\mathsf{NewSE}}$ with input ρ_{se}, $\mathcal{A}_{\mathrm{mc},i}$ generates the device keys, saves the secret key and returns dpk, just like in the MU-UNLINK-0 game. If this is the i^{th} call, $\mathcal{A}_{\mathrm{mc},i}$ additionally sends ρ_{iss} and ρ_{se} to MC-UNLINK-0 and subsequently gains access to its own $\mathcal{O}_{\mathsf{DoPres}}$.
- In $\mathcal{O}_{\mathsf{DoPres}}$, $\mathcal{A}_{\mathrm{mc},i}$ first makes the expected checks on the inputs. For input dpk_j with $j < i$, $\mathcal{A}_{\mathrm{mc},i}$ runs the SE-user exchange itself and for a dpk_j with $j > i$, $\mathcal{A}_{\mathrm{mc},i}$ calls simulator $\mathsf{Sim.Show}$. A call with dpk_i is the special case where we want to reduce to MC-UNLINK-0. Observe that if $\mathcal{A}_{\mathrm{mu}}$ makes a call for dpk_i and the initial check of $\mathcal{O}_{\mathsf{DoPres}}$ do not fail, then we are sure dpk_i was already registered at $\mathcal{O}_{\mathsf{NewSE}}$, and we have initialized MC-UNLINK-0. Hence, $\mathcal{A}_{\mathrm{mc},i}$ is able to forward the inputs $(ctx, \phi, \sigma, \boldsymbol{a})$ to its own $\mathcal{O}_{\mathsf{DoPres}}$ and forward the response to $\mathcal{A}_{\mathrm{mu}}$.
- The output of $\mathcal{A}_{\mathrm{mu}}$ is forwarded to MC-UNLINK-0.

This way of simulation by $\mathcal{A}_{\mathrm{mc},i}$ yields Game_i if the respective MC-UNLINK-0 instance is in the real world, and Game_{i+1} if the respective MC-UNLINK-0 instance is in the ideal world, and if $\mathcal{A}_{\mathrm{mu}}$ could distinguish between Game_i and Game_{i+1} then MC-UNLINK-0 would be broken. Finally, we sum up the bounds of the q_u game hops and get the advantage bound stated in the theorem.

Implication depends on the chosen ideal model. In the above proofs, we brushed over a technical detail which could hinder our chosen strategy. The hybrid arguments used reductions that generate some real and ideal presentations themselves, and obtain one crucial challenge from the single-credential or single-user version. While creating presentations according to the DBAC schemes is guaranteed to be the same no matter who executes the algorithms, the simulator requires some care. In particular, whether the simulator from the single-credential / single-user game and the reduction-run simulator create identically distributed artifacts depends on the idealized abilities of the simulator. We did not fix specific abilities in our privacy model but rather wanted to make the definition oblivious to the employed ideal model, which could be ROM, CRS, or even more involved models. For our argument it is important that the two simulator instances share the same resources, such as random oracles or trapdoors. Since one of the simulators is run by the reduction to the single-credential / single-user game, this implies that the adversary must also have access to these resources. For a CRS-based construction, this can be problematic, as passing the trapdoor to the adversary might make the privacy notion unachievable. In such a case, the relation between the simple and the extended version of our definitions stays unclear, and we advise to revert to the extended versions to ensure security under composition.

For the remaining privacy properties, we simply state their extended versions, the advantage terms are defined analogously. The argument of implication by the simple versions presented in Section 5 follows the same proof technique with straightforward adaptions to the additional oracles, so we only provide brief sketches of how to adapt the argument given for Unlink-0.

C.2 Extended Unlink-1

A multi-credential and multi-user version of Unlink-1 is shown in Figure 8. Here, the $\mathcal{O}_{\mathsf{ShowSE}}$ oracle is also updated to account for an arbitrary number of rounds between user and SE, where we allow the adversary to interleave multiple presentation sessions arbitrarily. The dictionary `Round` logs which rounds have been completed for a specific session id and records the rounds' states. Analogously to Unlink-0, this extension is implied by the simple definition given in the main body of this work. The proof strategy is the same, we only need to additionally provide the $\mathcal{O}_{\mathsf{ShowSE}}$ oracle in each reduction, which is easily achieved by running the algorithm as defined in the original game – the necessary secret key(s) *dsk* are known to the reduction.

Theorem 10. *Let Π be a* (B)DBAC *scheme that achieves Unlink-1 with respect to a simulator* $\mathsf{Sim} = (\mathsf{Setup}, \mathsf{Show})$ *such that parameters generation by* $\mathsf{Sim.Setup}$ *is indistinguishable from Π.*Setup *and a leakage function $\mathcal{L}$. For any adversary $\mathcal{A}_{mc}$ playing MC-Unlink-1 with* Sim *and $\mathcal{L}$ that uses q_σ distinct credentials in their calls to $\mathcal{O}_{\mathsf{DoPres}}$, there exists an adversary $\mathcal{A}$ playing Unlink-1 with* Sim *and $\mathcal{L}$ such that*

$$\mathsf{Adv}^{\text{MC-Unlink-1}}_{\Pi,\ell,\mathsf{Sim},\mathcal{L}}(\mathcal{A}_{mc},\lambda) \leq q_\sigma \cdot \mathsf{Adv}^{\text{Unlink-1}}_{\Pi,\ell,\mathsf{Sim},\mathcal{L}}(\mathcal{A},\lambda)$$

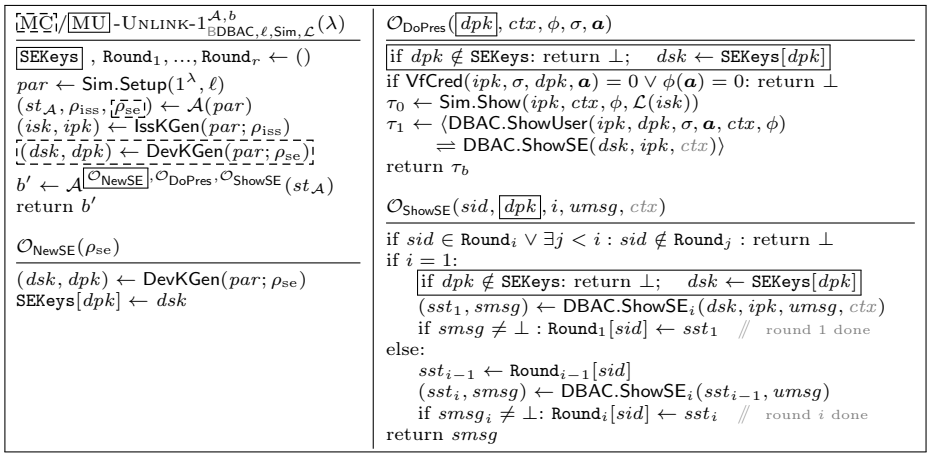

Fig. 8. Extended UNLINK-1: a single-user, but multi-credential version (plain code and dashed boxes), and a multi-user, multi-credential version (plain code and solid boxes). Additionally, the $\mathcal{O}_{\text{ShowSE}}$ oracle is kept abstract to account for schemes with r rounds of interaction for jointly computing a presentation. For BDBAC, the *ctx* is not sent to the SE, indicated by gray text color.

Further, for any $\mathcal{A}_{mu}$ playing MU-UNLINK-1 against DBAC *with* Sim *and $\mathcal{L}$ that makes q_u calls to $\mathcal{O}_{\text{NewSE}}$, there exists an adversary $\mathcal{A}_{mc}$ playing MC-UNLINK-1 with* Sim *and $\mathcal{L}$ such that*

$$\text{Adv}^{\text{MU-UNLINK-1}}_{\Pi,\ell,\text{Sim},\mathcal{L}}(\mathcal{A}_{mu}, \lambda) \leq q_u \cdot \text{Adv}^{\text{MC-UNLINK-1}}_{\Pi,\ell,\text{Sim},\mathcal{L}}(\mathcal{A}_{mc}, \lambda)$$

C.3 Extended UNLINK-2

The extension to multiple credentials, users and ShowSE rounds of UNLINK-2 is given in Figure 9. The extensions are as for the prior privacy notions. Proving the implication from the simple definition is also largely unchanged; the reductions now simply must run $\widetilde{\text{ShowSE}}$ in place of the honest algorithms.

Theorem 11. *Let Π be a* (B)DBAC *scheme that achieves UNLINK-2 with respect to a simulator* Sim = (Setup, Show) *such that parameters generation by* Sim.Setup *is indistinguishable from Π.*Setup*, and a leakage function $\mathcal{L}$. For any adversary $\mathcal{A}_{mc}$ playing MC-UNLINK-2 with* Sim *and $\mathcal{L}$ that uses q_σ distinct credentials in their calls to $\mathcal{O}_{\text{DoPres}}$, there exists an adversary $\mathcal{A}$ playing UNLINK-2 with* Sim *and $\mathcal{L}$ such that*

$$\text{Adv}^{\text{MC-UNLINK-2}}_{\Pi,\ell,\text{Sim},\mathcal{L}}(\mathcal{A}_{mc}, \lambda) \leq q_\sigma \cdot \text{Adv}^{\text{UNLINK-2}}_{\Pi,\ell,\text{Sim},\mathcal{L}}(\mathcal{A}, \lambda)$$

Further, for any $\mathcal{A}_{mu}$ playing MU-UNLINK-2 against DBAC *with* Sim *and $\mathcal{L}$ that makes q_u calls to $\mathcal{O}_{\text{NewSE}}$, there exists an adversary $\mathcal{A}_{mc}$ playing MC-UNLINK-2*

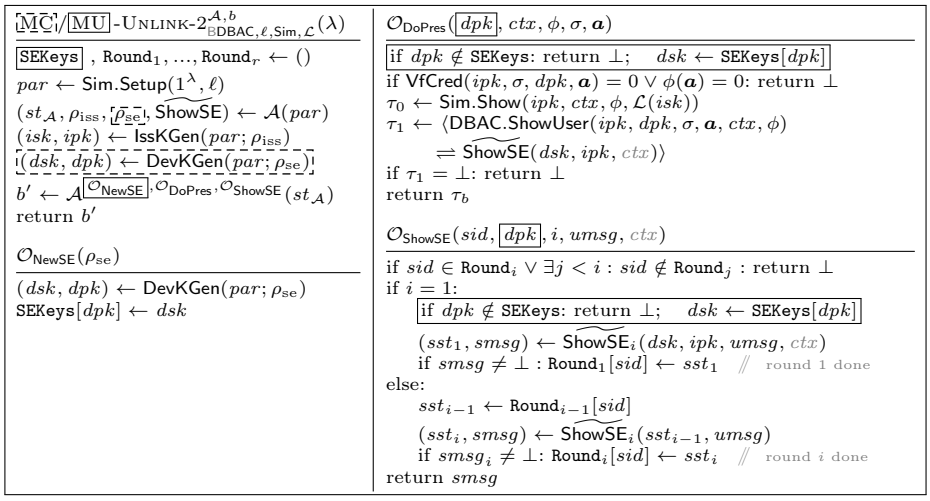

Fig. 9. Extended UNLINK-2: a single-user, but multi-credential version (plain code and dashed boxes), and a multi-user, multi-credential version (plain code and solid boxes). Additionally, the $\mathcal{O}_{\mathsf{ShowSE}}$ oracle is kept abstract to account for schemes with r rounds of interaction for jointly computing a presentation. For BDBAC, the ctx is not sent to the SE, indicated by gray text color.

with Sim *and* $\mathcal{L}$ *such that*

$$\mathsf{Adv}^{MU\text{-}\textsc{Unlink}\text{-}2}_{\Pi,\ell,\mathsf{Sim},\mathcal{L}}(\mathcal{A}_{mu}, \lambda) \leq q_u \cdot \mathsf{Adv}^{MC\text{-}\textsc{Unlink}\text{-}2}_{\Pi,\ell,\mathsf{Sim},\mathcal{L}}(\mathcal{A}_{mc}, \lambda)$$

C.4 Extended UNLINK-3

Finally, we extend UNLINK-3 (Figure 10). Switching to the multi-credential and multi-user setting is analogous to the previous definitions, only having an arbitrary number of rounds in the showing interaction makes a bigger visual change to the game. We chose to separate the first round of the interaction from the rest for readability. To show the implication, we will now have to maintain some state for each interaction to remember whether at the end we need to call Sim or use DBAC, but the principle does not change – we do repeated game hops where in each hop, all interactions concerning one credential / dpk are switched from real to random.

Theorem 12. *Let* Π *be a* (B)DBAC *scheme that achieves* UNLINK-3 *with respect to a simulator* $\mathsf{Sim} = (\mathsf{Setup}, \mathsf{Show})$*, such that parameters generation by* Sim.Setup *is indistinguishable from* Π.Setup*, and a leakage function* $\mathcal{L}$*. For any adversary* $\mathcal{A}_{mc}$ *playing MC-*UNLINK-3 *with* Sim *and* $\mathcal{L}$ *that uses* q_σ *distinct credentials in their calls to* $\mathcal{O}_{\mathsf{ShowUser},1}$*, there exists an adversary* $\mathcal{A}$ *playing* UNLINK-3 *with* Sim *and* $\mathcal{L}$ *such that*

$$\mathsf{Adv}^{MC\text{-}\textsc{Unlink}\text{-}3}_{\Pi,\ell,\mathsf{Sim},\mathcal{L}}(\mathcal{A}_{mc}, \lambda) \leq q_\sigma \cdot \mathsf{Adv}^{\textsc{Unlink}\text{-}3}_{\Pi,\ell,\mathsf{Sim},\mathcal{L}}(\mathcal{A}, \lambda)$$

Further, for any $\mathcal{A}_{mu}$ playing MU-UNLINK-3 against DBAC *with* Sim *and $\mathcal{L}$ that makes q_u calls to $\mathcal{O}_{\mathsf{NewSE}}$, there exists an adversary $\mathcal{A}_{mc}$ playing MC-UNLINK-3 with* Sim *and $\mathcal{L}$ such that*

$$\mathsf{Adv}^{\text{MU-UNLINK-3}}_{\Pi,\ell,\mathsf{Sim},\mathcal{L}}(\mathcal{A}_{mu},\lambda) \le q_u \cdot \mathsf{Adv}^{\text{MC-UNLINK-3}}_{\Pi,\ell,\mathsf{Sim},\mathcal{L}}(\mathcal{A}_{mc},\lambda)$$

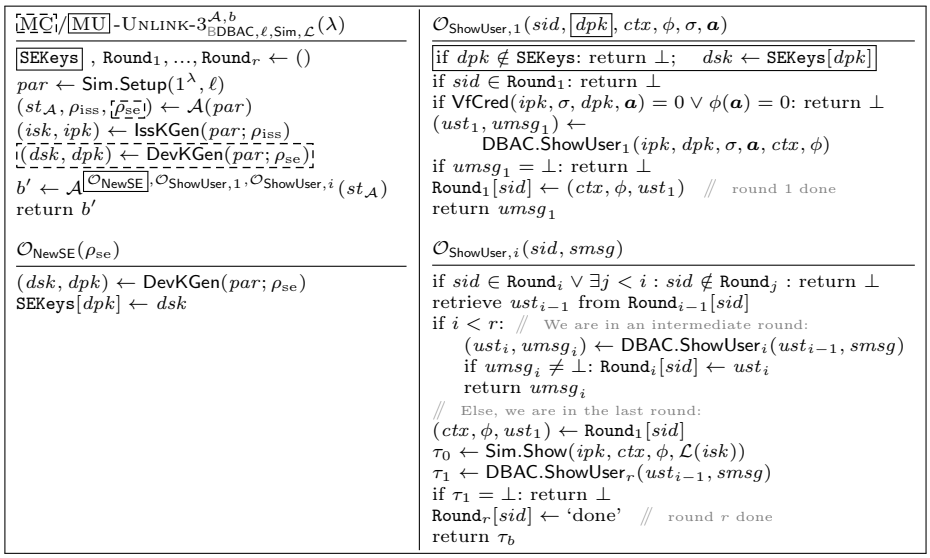

Fig. 10. Extended UNLINK-3: a single-user, but multi-credential version (plain code and dashed boxes), and a multi-user, multi-credential version (plain code and solid boxes). Additionally, the $\mathcal{O}_{\mathsf{ShowUser}}$ oracle is kept abstract to account for schemes with r rounds of interaction for jointly computing a presentation.

C.5 Extended SE-OBLIV

The multi-credential, multi-user, and multi-round version of SE-OBLIV is given in Figure 11. The principle of allowing multiple credentials and users is identical to the UNLINK notions, but note how we extend the property to many rounds: since SE-OBLIV is about intermediate messages not leaking anything to the SE, we need to continue the real-or-ideal paradigm for all rounds up to the last one, where we return a DBAC presentation as before. The simulator gets its own state from the prior round as input to ensure consistency, and Sim.Check inspects the final simulator state $ust_{r-1,0}$. The principle of showing implication from the simple to the extended version we used for UNLINK applies here equivalently

Theorem 13. *Let Π be a* (B)DBAC *scheme that achieves SE-OBLIV with respect to a simulator* $\mathsf{Sim} = (\mathsf{Setup}, \mathsf{ShowUser}_1, \mathsf{Check})$*, such that parameters generation by* Sim.Setup *is indistinguishable from*

DBAC.Setup*, and a leakage function* $\mathcal{L}$*. For any adversary* $\mathcal{A}_{mc}$ *playing MC-SE-*Obliv *with* Sim*' (that is identical to* Sim *but has arbitrary additional algorithms* $\mathsf{ShowUser}_i$*) and* $\mathcal{L}$ *that uses* q_σ *distinct credentials in their calls to* $\mathcal{O}_{\mathsf{ShowUser},1}$*, there exists an adversary* $\mathcal{A}$ *playing SE-*Obliv *with* Sim *and* $\mathcal{L}$ *such that*

$$\mathsf{Adv}^{\textit{MC-SE-Obliv}}_{\mathsf{DBAC},\ell,\mathsf{Sim}',\mathcal{L}}(\mathcal{A}_{mc},\lambda) \le q_\sigma \cdot \mathsf{Adv}^{\textit{SE-Obliv}}_{\Pi,\ell,\mathsf{Sim},\mathcal{L}}(\mathcal{A},\lambda)$$

Further, for any $\mathcal{A}_{mu}$ *playing MU-SE-*Obliv *against* DBAC *with* Sim′ *and* $\mathcal{L}$ *that makes* q_u *calls to* $\mathcal{O}_{\mathsf{NewSE}}$*, there exists an adversary* $\mathcal{A}_{mc}$ *playing MC-SE-*Obliv *with* Sim′ *and* $\mathcal{L}$ *such that*

$$\mathsf{Adv}^{\textit{MU-SE-Obliv}}_{\mathsf{DBAC},\ell,\mathsf{Sim}',\mathcal{L}}(\mathcal{A}_{mu},\lambda) \le q_u \cdot \mathsf{Adv}^{\textit{MC-SE-Obliv}}_{\mathsf{DBAC},\ell,\mathsf{Sim}',\mathcal{L}}(\mathcal{A}_{mc},\lambda)$$

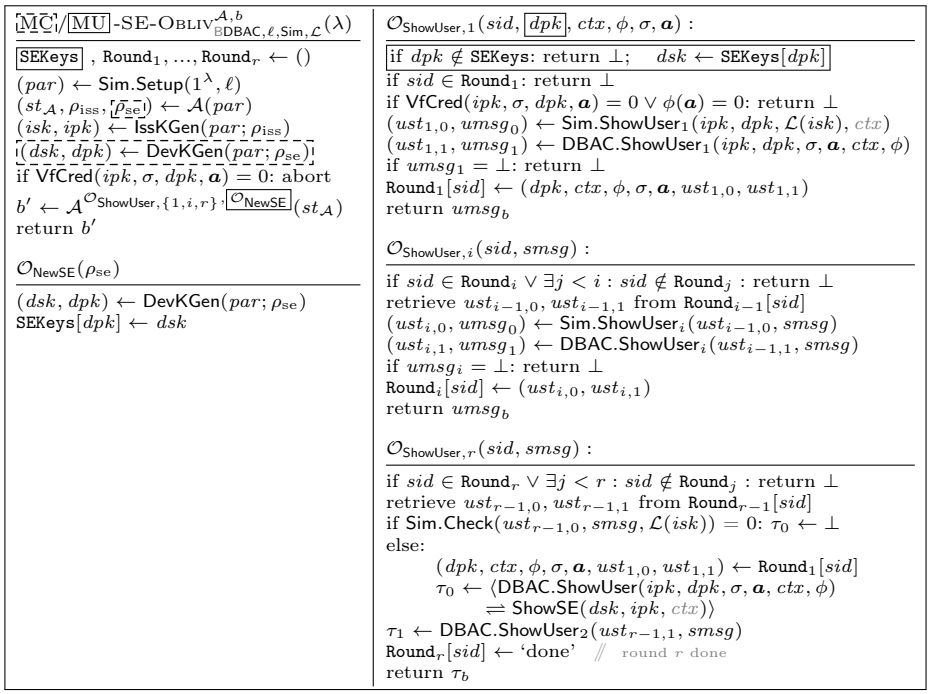

Fig. 11. Extended SE-Obliv: a single-user, but multi-credential version (plain code and dashed boxes), and a multi-user, multi-credential version (plain code and solid boxes). Additionally, the $\mathcal{O}_{\mathsf{ShowUser}}$ oracle is kept abstract to account for schemes with r rounds of interaction for jointly computing a presentation. For BDBAC, the *ctx* is not sent to the SE, indicated by gray text color.

D Relation of Privacy Properties

Let us now formalize the relations between the privacy properties.

Lemma 1 (Relation of UNLINK properties). *UNLINK-3 is strictly stronger than UNLINK-2, UNLINK-2 is strictly stronger than UNLINK-1, which in turn is strictly stronger than UNLINK-0.*

D.1 UNLINK-1 Is Strictly Stronger Than UNLINK-0

Proof (sketch). It is trivial to show UNLINK-1 implies UNLINK-0, so we omit this step here. We construct a DBAC scheme that is UNLINK-0, but not UNLINK-1, to show the inverse implication does not hold.

We start from a scheme DBAC that achieves UNLINK-0 and modify the algorithms for presentation and verification slightly. We make use of a pseudorandom function (PRF) that takes arbitrary bit strings $\in \{0,1\}^*$ as input and uses keys of the same domain as the device secret keys:

- The modified $\mathsf{ShowSE}'_1(dsk, umsg_1)$ executes the original ShowSE_1 algorithm on the same inputs, but also executes the PRF on the dpk and a random $nonce \leftarrow\!\!\$\ \{0,1\}^{2\lambda}$: $s' \leftarrow \mathrm{PRF}(dsk, dpk \| nonce)$. To the message sent back to the user it then appends dsk and $s \leftarrow (s', nonce)$.
- $\mathsf{ShowUser}_2$ first ignores the extra inputs and executes the original algorithm, then appends s to the presentation.
- Verify' ignores s and executes the original Verify.

Presentations of this modified scheme DBAC' now contain a pseudorandom s' and *nonce* which are fresh in each presentation. Thus, UNLINK-0 is still achieved, a simulator could simply output two random values of the same size in place of $(s', nonce)$. Only if the same nonce is chosen twice this is distinguishable, but this happens with negligible probability for $nonce \in \{0,1\}^{2\lambda}$. But UNLINK-1 allows the adversary to see intermediate SE messages, therefore it obtains dsk and can distinguish truly random and pseudorandom values easily.

D.2 UNLINK-2 Is Strictly Stronger Than UNLINK-1

Proof (sketch). To show that UNLINK-2 implies UNLINK-1, we build a straightforward reduction $\mathcal{A}_2$ that uses a successful adversary $\mathcal{A}_1$ against UNLINK-1 to win UNLINK-2: $\mathcal{A}_2$ defines the "subverted" algorithm $\widetilde{\mathsf{ShowSE}}_1$ to execute the honest ShowSE_1. To initialize the game, $\mathcal{A}_2$ forwards $\mathcal{A}_1$'s inputs $st_{\mathcal{A}}, \rho_{\mathrm{iss}}, \rho_{\mathrm{se}}, \sigma, \boldsymbol{a}$ as well as $\widetilde{\mathsf{ShowSE}}_1$ to the UNLINK-2 game.

During the game, the reduction $\mathcal{A}_2$ can simply forward all oracle calls without modification to their own game, since the subverted algorithm is now identical to the honest one. Once $\mathcal{A}_1$ outputs b', $\mathcal{A}_2$ forwards this bit to UNLINK-2. Clearly, if $\mathcal{A}_1$ guessed correctly, then b' is also the correct output for UNLINK-1, as we forwarded the challenges from $\mathcal{O}_{\mathsf{DoPres}}$. This concludes the reduction.

Finally, we argued in Section 6.2 how the BBS-Schnorr construction achieves UNLINK-1 and how UNLINK-2 is not achieved, since Schnorr signatures are susceptible to subliminal channels. Therefore, BBS-Schnorr serves as an example that UNLINK-1 cannot imply UNLINK-2.

D.3 Unlink-3 Is Strictly Stronger Than Unlink-2.

Proof (sketch). We can build a reduction $\mathcal{A}_3$ from Unlink-3 to Unlink-2 as follows: from $\mathcal{A}_2$, we receive $(st_{\mathcal{A}}, \rho_{\text{iss}}, \rho_{\text{se}}, \sigma, \boldsymbol{a}, \widetilde{\mathsf{ShowSE}})$. We can then simulate the Unlink-2 game perfectly by running the subverted $\widetilde{\mathsf{ShowSE}}$ ourselves. A call to $\mathcal{O}_{\mathsf{DoPres}}$ is answered by sending the inputs (ctx, ϕ) with a fresh sid to our own $\mathcal{O}_{\mathsf{ShowUser},1}$, executing $\widetilde{\mathsf{ShowSE}}$ on the returned user message $umsg$, and then obtaining the presentation from $\mathcal{O}_{\mathsf{ShowUser},2}$. Note that if Unlink-3 is in the ideal world, $\mathcal{O}_{\mathsf{ShowUser},2}$ will return a presentation simulated exactly as in Unlink-2, and if Unlink-3 is in the real world, we executed exactly the interaction between honest user and subverted SE as foreseen by Unlink-2. We conclude this simulation is perfect and $\mathcal{A}_3$ wins if $\mathcal{A}_2$ wins.

As shown in Section 6.3, BBS-BLS is an example of a scheme that achieves Unlink-2 but leaks $\overline{dpk}$ as a somewhat unique identifier of a presentation towards the SE, so Unlink-3 is not achieved.

E Privacy of Issuance

Our privacy definitions share one limitation regarding the device public key and issuance. By our definition, issuance is non-blind and reveals a device's dpk, making it inherently not anonymous. This leakage is aggravated with static and full corruption of SEs: it is only realistic to assume that the adversary can influence the choice of device key, for example, select a fixed or low-entropy key. As a consequence, the issuer could clearly distinguish a corrupt device, or the key itself could serve as a subliminal channel. We argue that this limitation has a negligible impact on privacy. Issuance is a rarely occurring process and reveals nothing about a user's dynamic behavior, so it seems fine to reveal the dpk here in general. Secondly, the subliminal channel provided by the corrupt choice is one-time per SE and can thus only transport limited information – in particular, nothing about the presentations a device generated. Besides, it is unclear how subtler changes to a key's distribution could be prevented.

F Security: Full Proofs

In this section we formally prove unforgeability for our three constructions: BBS-Schnorr, BBS-BLS, and BBS-BlindBLS. The proofs largely follow the same outline, and the majority of the technical difficulties are encountered in the BLS setting, i.e., with BBS-BLS and BBS-BlindBLS. For this reason, we give a complete proof for BBS-BLS and then describe only the differences in the BBS-Schnorr and BBS-BlindBLS settings.

F.1 Schnorr as a Proof System for Linear Relations

For the following proofs, we first introduce additional syntax. So far, we kept the NIZK over a BBS credential employed in the construction relatively abstract,

but we now require a more rigorous syntax. The NIZK is an instantiation of Schnorr as a straight-line extractable non-interactive proof system for certain linear relations. Let us reformulate Schnorr signatures as signatures of knowledge for the relation $\mathcal{R}_{\text{linear}} := \{(M \in \mathbb{G}_1^{n\times m}, \boldsymbol{Y} \in \mathbb{G}_1^n) : M\boldsymbol{x} = \boldsymbol{Y}\}$ which, in the case where $M = \begin{pmatrix} G_1 \end{pmatrix}$ and $\boldsymbol{Y} = (pk)$, is the ordinary Schnorr signature scheme. We introduce additional parametrization via $n, m \in \mathbb{N}$, the scheme is now denoted as $\mathsf{Schnorr}[\mathsf{BGGen}, \mathsf{H}, n, m]$ and defined as follows:

$\mathsf{Schnorr.Setup}$**:** generate and output group parameters $par \leftarrow \mathsf{BGGen}(1^\lambda)$

$\mathsf{Schnorr.Sign}((M, \boldsymbol{Y}), \boldsymbol{x}, m)$**:** sample $\boldsymbol{\omega} \leftarrow\!\!\$\ \mathbb{Z}_p^m$, set $\boldsymbol{R} \leftarrow M\boldsymbol{\omega}$, compute $c = \mathsf{H}(M, \boldsymbol{Y}, \boldsymbol{R}, m)$ and $\boldsymbol{s} \leftarrow \boldsymbol{\omega} + c\boldsymbol{x}$. Output $(c, \boldsymbol{s})$.

$\mathsf{Schnorr.Verify}((M, \boldsymbol{Y}), sig, m)$**:** parse $(c, \boldsymbol{s}) \leftarrow sig$ and output 0 if this fails. Then output $\mathsf{H}(M, \boldsymbol{Y}, M\boldsymbol{s} - c\boldsymbol{Y}, m) \stackrel{?}{=} c$.

In the unforgeability proofs, we instantiate the NIZK proof system for BBS with Schnorr, using the following helper matrix for the relation:

$$M^{\text{bbs}}_{\overline{C},(\delta_j)_{j=1}^d,\overline{A}} := \begin{pmatrix} \overline{C} & H_0 & -H_{\delta_1} & \dots & -H_{\delta_d} & 0 & 0 \\ 0 & 0 & & \dots & 0 & \overline{C} & -\overline{A} \end{pmatrix}.$$

F.2 BBS-BLS

Let us first state Theorem 3 in more detail.

Theorem 14 (BBS-BLS UNF, extended theorem). *Let* BGGen *be a bilinear group generator. Let* $\mathcal{A}$ *be an algebraic adversary against the unforgeability of BBS-BLS making at most* $q_{\mathsf{H}_1}, q_{\mathsf{H}_2}, \mathsf{nk}, q_s$ *queries to* H_1, H_2, $\mathcal{O}_{\mathsf{NewSE}}$, *and* $\mathcal{O}_{\mathsf{hShow}}$ *respectively. Define* $\mathsf{Ext} = (\mathsf{Setup}, \mathsf{EKey})$ *as follows:*

1. $\mathsf{Ext.Setup}$*: run* $par \leftarrow \mathsf{Setup}(1^\lambda, \ell)$ *and output* $(\bot, par)$ *(there is no trapdoor).*
2. $\mathsf{Ext.EKey}$*: on input* $(td, \tau^*, ctx^*, \phi^*_{I,\boldsymbol{d}})$*:*
   (a) *let* $(dsk_i, dpk_i)_{i=1}^{\mathsf{nk}}$ *be all of the entries in* $\mathtt{SEKeys}$
   (b) *parse* $(\overline{dpk}, \pi_{se}, \overline{A}, \overline{B}, \overline{C}, \pi_{bbs}) \leftarrow \tau^*$
   (c) *parse* $(c, \boldsymbol{s}) \leftarrow \pi_{bbs}$*. Set* $Y \leftarrow G_1 + \overline{dpk} + \sum_{i\in I} d_i H_i$*. Set* $M \leftarrow M^{bbs}_{\overline{C},I,\overline{A}}$*,* $\boldsymbol{Y} \leftarrow (Y, \overline{B})$*, and* $\boldsymbol{R} \leftarrow M\boldsymbol{s} - c\boldsymbol{Y}$*. If no query* $(M, (Y, \overline{B}), \boldsymbol{R}, \pi_{se})$ *has been made to random oracle* H_1*, then abort and output* $\bot$*. Set* $(R_1, R_2) \leftarrow \boldsymbol{R}$*. Let* $\hat{\alpha}, (\alpha_i)_{i=0}^\ell$ *be the coefficients output by* $\mathcal{A}$*, s.t.* $\overline{A} = \hat{\alpha}G_1 \sum_{i=0}^\ell \alpha_i H_i$*, and respectively* $\hat{\beta}, (\beta_i)$ *for* $\overline{B}$*,* $\hat{\gamma}, (\gamma_i)$ *for* $\overline{C}$*, and* $\hat{\rho}_i, (\rho_{i,j})$ *for* R_i*. Solve for coefficients* $\xi_1, (\xi_{1,i})$ *such that* $R_1 = \xi_1\overline{C} + \xi_{1,0}H_0 + \sum_{j\in[\ell]\setminus I} \xi_{1,j}H_j$*, or abort if no such solution to the linear system exists. Solve for coefficients* $\xi_{2,0}, \xi_{2,1}$ *such that* $R_2 = \xi_{2,0}\overline{C} + \xi_{2,1}\overline{A}$*, or abort if there is no solution. Abort if* $c = 0$ *or* $M\boldsymbol{s} - c\boldsymbol{Y} \neq \boldsymbol{R}$*. Otherwise, we can compute* $\boldsymbol{x} = (r_3, \hat{\boldsymbol{m}}, r_{key}, r_2, e)$ *such that* $M\boldsymbol{x} = (Y, \overline{B})$*. Output* $\overline{dpk} - r_{key}H_0$*.*

Parameter indistinguishability follows trivially. There exist adversaries $\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3, \mathcal{B}_4, \mathcal{B}_5, \mathcal{B}_6, \mathcal{B}_7$ *running in time roughly that of* $\mathcal{A}$ *such that*

$$\begin{aligned}\mathsf{Adv}^{U_{NF}}_{\mathit{BBS\text{-}Schnorr},\ell,\mathsf{Ext}}(\mathcal{A},\lambda) \leq\ & \mathsf{Adv}^{\mathit{rel\text{-}DL}}_{\mathsf{BGGen},\ell}(\mathcal{B}_1,\lambda) \\ & + q_{\mathsf{H}_2}\cdot\left(\mathsf{Adv}^{\mathit{DL}}_{\mathsf{BGGen},1}(\mathcal{B}_2,\lambda) + \mathsf{Adv}^{\mathit{rel\text{-}DL}}_{\mathsf{BGGen},2}(\mathcal{B}_3,\lambda) + \frac{1}{p}\right) \\ & + \mathsf{Adv}^{\mathit{rel\text{-}DL}}_{\mathsf{BGGen},\ell}(\mathcal{B}_7,\lambda) \\ & + \mathsf{Adv}^{\mathit{EUF\text{-}CMA}}_{\mathsf{BGGen},\mathsf{BBS},\ell}(\mathcal{B}_4,\lambda) \\ & + \mathsf{nk}\cdot\mathsf{Adv}^{\mathit{rel\text{-}DL}}_{\mathsf{BGGen},\ell+1}(\mathcal{B}_5,\lambda) \\ & + \mathsf{nk}\cdot\mathsf{Adv}^{\mathit{EUF\text{-}CMA}}_{\mathsf{BGGen},\mathsf{BLS},0}(\mathcal{B}_6,\lambda) \\ & + \frac{q_s(q_s+q_{\mathsf{H}_1})+1}{p}\end{aligned}$$

For our security analysis we first introduce the new cr-co-CDH assumption, defined in the following game $\mathsf{cr\text{-}co\text{-}CDH}^{\mathcal{A}}_{\mathsf{BGGen},\ell}$ and parameterized by a bilinear group generator BGGen and $\ell \in \mathbb{N}$:

1. $par := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, G_1, G_2, \mathsf{e}) \leftarrow \mathsf{BGGen}(1^\lambda)$
2. $H_0, \ldots, H_\ell \leftarrow\!\!\$\ \mathbb{G}_1$
3. $(st, \hat{a}, (a_i)_{i=0}^{\ell}) \leftarrow \mathcal{A}(par, H_0, \ldots H_\ell)$
4. $X \leftarrow \hat{a}G_1 + \sum_{i=0}^{\ell} a_i H_i$
5. $Y \leftarrow\!\!\$\ \mathbb{G}_2$
6. $\sigma \leftarrow \mathcal{A}(st, Y)$
7. If $\mathsf{e}(H_0, \sigma) = \mathsf{e}(X, Y)$ and $a \neq 0$ for some $a \in \{\hat{a}, (a_i)_{i=1}^{\ell}\}$ then output 1

For any $\mathcal{A}$, define $\mathsf{Adv}^{\mathsf{cr\text{-}co\text{-}CDH}}_{\mathsf{BGGen},\ell}(\mathcal{A},\lambda)$ to be the probability that the experiment above outputs 1.

Lemma 2 (Chosen-Representation co-CDH). *Let* BGGen *be a bilinear group generator. In the AGM, for any* $\mathcal{A}$*, there exist* $\mathcal{B}_1, \mathcal{B}_2$ *running in time similar to that of* $\mathcal{A}$ *such that:*

$$\mathsf{Adv}^{\mathit{cr\text{-}co\text{-}CDH}}_{\mathsf{BGGen},\mathsf{H},0}(\mathcal{A},\lambda) \leq \mathsf{Adv}^{\mathit{DL}}_{\mathsf{BGGen},1}(\mathcal{B}_1,\lambda) + \mathsf{Adv}^{\mathit{DL}}_{\mathsf{BGGen},2}(\mathcal{B}_2,\lambda).$$

Moreover, for any $\mathcal{A}$ *and* $\ell \in \mathbb{N}$ *there exists* $\mathcal{B}$ *such that*

$$\mathsf{Adv}^{\mathit{cr\text{-}co\text{-}CDH}}_{\mathsf{BGGen},\mathsf{H},\ell}(\mathcal{A},\lambda) \leq \mathsf{Adv}^{\mathit{cr\text{-}co\text{-}CDH}}_{\mathsf{BGGen},\mathsf{H},0}(\mathcal{B},\lambda) + \frac{1}{p}$$

Proof. We prove the first inequality first. Since $\mathcal{A}$ is algebraic, when they output σ they also output an algebraic representation in terms of all $\mathbb{G}_2$ elements they have seen so far, so we have $\sigma = \hat{s}G_2 + s_0 Y$. The proof is by a sequence of hybrids:

Game_0 This is exactly the AGM version of the game (where $\ell = 0$ as in the lemma) with some variables renamed for readability.

1. $par := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, G_1, G_2, \mathsf{e}) \leftarrow \mathsf{BGGen}(1^\lambda)$
2. $h \leftarrow\!\$\ \mathbb{Z}_p; H_0 \leftarrow hG_1$
3. $(st, (\hat{c}, c_0)) \leftarrow \mathcal{A}(par, H_0)$
4. $X = \hat{c}G_1 + c_0H_0$
5. $y \leftarrow\!\$\ \mathbb{Z}_p; Y \leftarrow yG_2$
6. $(\sigma, (\hat{s}, s_0)) \leftarrow \mathcal{A}(st, Y)$, such that $\sigma = \hat{s}G_2 + s_0Y$
7. If $\mathsf{e}(H_0, \sigma) = \mathsf{e}(X, Y)$ and $\hat{c} \neq 0$ then output 1.

Simulation is perfect and the advantage does not change.

Game_1 We say that the adversary loses if $\hat{c} + c_0h - s_0h \neq 0$. Consider the reduction $\mathcal{B}_1$ playing the DL game in $\mathbb{G}_2$. On challenge input $Z = zG_2$, $\mathcal{B}_1$ simulates the game to $\mathcal{A}$ with $Y \leftarrow Z$ instead of randomly sampling it. This makes no external change. From the pairing equality, we have

$$\begin{aligned} h(\hat{s}G_2 + s_0Y) &= (\hat{c} + c_0h)Y \\ h\hat{s}G_2 &= (\hat{c} + c_0h - hs_0)Y \end{aligned}$$

Clearly $\mathcal{B}_1$ can recover the discrete logarithm of Y.

Game_2 We say that the adversary loses if $s_0 - c_0 \neq 0$. Consider the reduction $\mathcal{B}_2$ playing the DL game in $\mathbb{G}_1$. Given challenge $Z = zG_1$, $\mathcal{B}_2$ simulates the game to $\mathcal{A}$, setting $H_0 = Z$. From the last hop we know that $\hat{c} + c_0h - s_0h = 0$, which we can rewrite to $\hat{c} = h(s_0 - c_0)$. Therefore, at the end of the game the adversary $\mathcal{B}_2$ knows that $\hat{c}G_1 = (s_0 - c_0)H_0$, and extracts the discrete logarithm of $H_0 = Z$.

Game_3 The adversary always loses. Substituting $s_0 = c_0$ into $\hat{c} + c_0h - s_0h = 0$ we have $\hat{c} = 0$, but this contradicts the winning condition of $\mathcal{A}$.

Now for the second part of the proof, where we show the self-reducibility given in the lemma. Consider the reduction $\mathcal{B}$ for a fixed $\ell \in \mathbb{N}$. On input par, H_0, it proceeds as follows:

1. Sample $\beta_1, \gamma_1, \ldots, \beta_\ell, \gamma_\ell \leftarrow\!\$\ \mathbb{Z}_p$. For all $i \in [\ell]$, set $H_i \leftarrow \beta_iG_1 + \gamma_iH_0$.
2. Run $(st, \hat{a}, (a_i)_{i=0}^{\ell}) \leftarrow \mathcal{A}(par, H_0, \ldots, H_\ell)$
3. We have $X = \hat{a}G_1 + \sum_{i=0}^{\ell} a_iH_i = \hat{a}G_1 + a_0H_0 + \sum_{i=1}^{\ell} a_i(\beta_iG_1 + \gamma_iH_0) = (\hat{a} + 0 + \sum_{i=1}^{\ell} a_i\beta_i)G_1 + (0 + a_0 + \sum_{i=1}^{\ell} a_i\gamma_i)H_0$. Output the coefficients on G_1 and H_0 in the latter equation and we receive $Y \in \mathbb{G}_2$ from the challenger.
4. Run $\sigma_{(s_i)} \leftarrow \mathcal{A}(st, Y)$
5. Output $\sigma_{(s_i)}$

$\mathcal{A}$'s winning condition $\mathsf{e}(H_0, \sigma) = \mathsf{e}(X, Y)$ holds for $\mathcal{B}$ too since H_0 is identical. The additional winning condition that there is some $a \in \{\hat{a}, (a_i)_{i=1}^{\ell}\}$ such that $a \neq 0$, together with the information-theoretic hiding of (β_i) implies $\hat{a} + \sum_{i=1}^{\ell} a_i\beta_i = 0$ holds with probability at most $1/p$.

With this lemma we are now able to continue with the full proof of unforgeability for BBS-BLS (Theorem 14).

Proof. We assume without loss of generality that $\mathcal{A}$ never queries a random oracle with the same input twice, and similarly never queries $\mathcal{O}_{\mathsf{NewSE}}$ with the same non-$\perp$ input twice. We assume that the adversary does not output algebraic representations in terms of dpk or the signatures output by the Issue oracle. This is without loss of generality since a valid signature (A, e) on attributes $\boldsymbol{a}$ satisfies $A = \frac{1}{isk+e}(G_1 + dpk + \sum_{i=1}^{\ell} a_i \cdot H_i)$, and the discrete logarithm of every dpk for which we issue credentials is known. The proof is by a sequence of hybrids:

Game_0 This is identically the unforgeability game.

Game_1 We introduce a table O_1 used for simulating H_1 via lazy sampling. This makes no external change.

Game_2 $\mathcal{A}$ can no longer win by getting $\mathsf{Ext.EKey}$ to output $\perp$ due to a hash query $\mathsf{H}_1(M, (Y, \overline{B}), \boldsymbol{R}, \pi_{\mathrm{se}})$ never being made. In this case we have that π_{bbs} is valid with probability $1/p$.

Game_3 $\mathcal{A}$ can no longer win by getting $\mathsf{Ext.EKey}$ to output $\perp$ due to there being no solution to the R_1 linear system, or there being no solution to the R_2 linear system. Consider the following algorithm $\mathcal{B}_1$ playing the $\mathsf{rel\text{-}DL}$ game:

1. On input $(X_i)_{i=0}^{\ell}$, set $H_i \leftarrow X_i$ for all $i \in [0, \ell]$ and simulate the rest of the game to $\mathcal{A}$ with no changes.
2. If there is no solution to the R_1 linear system, then for all $(\overline{\xi}_1, \xi_{1,0}, \xi_{1,j})$ we have

$$R_1 \neq \overline{\xi}_1 \overline{C} + \xi_{1,0} H_0 + \sum_{j \in [\ell] \setminus I} \xi_{1,j} H_j.$$

   Also, we have representations of R_1 and $\overline{C}$ in terms of $G_1, H_0, \ldots, H_\ell$. This means that the equation above is a non-trivial discrete logarithm relationship.
   Alternatively, if there is no solution to the R_2 linear system, then for all $\xi_{2,0}, \xi_{2,1}$ we have $R_2 \neq \xi_{2,0}\overline{C} + \xi_{2,1}\overline{A}$, and similarly we have representations of R_2, $\overline{C}$, and $\overline{A}$ in terms of $G_1, H_0, \ldots, H_\ell$, so the equation yields a non-trivial discrete logarithm relationship. In either case, $\mathcal{B}_1$ wins the $\mathsf{rel\text{-}DL}$ game.

Game_4 We modify $\mathcal{O}_{\mathsf{hShow}}$ to simulate π_{bbs} using standard random oracle programming techniques. Since part of $\mathcal{A}$'s winning conditions ensures that it does not replay an honest showing, this does not affect the extractor. The only external change is if there is a collision when programming the random oracle, but due to the uniform randomness of $\boldsymbol{R}$ this occurs with probability at most $\frac{q_s(q_s+q_{\mathsf{H}_1})}{p}$.

Game_5 We make a change to how the extractor works, and the goal is to learn the discrete logarithm of dpk_1^* relative to H_0. First, in the event that $\overline{dpk} = dpk_i$ for any $i \in \mathsf{nk}$, then we know that $dpk_1^* = (dsk_i + r_{\mathrm{key}})H_0$. Write $dsk^* \leftarrow dsk_i + r_{\mathrm{key}}$. Otherwise, we need a more technical argument. Let $(\hat{\zeta}, \zeta_i)$ be the coefficients output by $\mathcal{A}$ such that $\overline{dpk} = \hat{\zeta}G_1 + \sum_{i=0}^{\ell} \zeta_i H_i$. Abort if $\zeta_i \neq 0$ for some $\zeta \in \{\hat{\zeta}, \zeta_1, ..., \zeta_\ell\}$, and otherwise set $dsk^* \leftarrow \zeta_0 + r_{\mathrm{key}}$. This only makes a change if the extractor now aborts. Consider the reduction $\mathcal{B}$

to cr-co-CDH which proceeds by simulating the entire game to $\mathcal{A}$ with the following changes:

1. Use the cr-co-CDH challenge generators $H_0, \ldots, H_\ell$ instead of randomly sampling them.
2. Guess $i^* \in q_{\mathsf{H}_2}$ and set up a table $O_2 \leftarrow (\,)$.
3. On the i^*-th query to q_{H_2}, which is of the form $\mathsf{H}_2(\overline{dpk}, ctx)$, let $(\hat{c})$ be the algebraic representation output by $\mathcal{A}$ so that $\overline{dpk} = \hat{c}G_1 + \sum_{i=0}^{\ell} H_i$. Give this input to $\mathcal{B}$'s challenger to get a group element $Y \in \mathbb{G}_2$ in return and output this group element.
4. On all other queries to q_{H_2}, sample $z \leftarrow \mathbb{Z}_p$ and set $O_2(m) \leftarrow z$ and output zG_2.

When $\mathcal{A}$ is finished, if $\overline{dpk}, ctx$ in their showing are such that $\mathsf{H}_2(\overline{dpk}, ctx)$ was never queried then we know that the showing can be valid with probability at most $1/p$. Otherwise, assuming that we correctly guessed that this hash query was the i^*-th, then their output satisfies $\mathsf{e}(H_0, \pi_{\mathrm{se}}) = \mathsf{e}(\overline{dpk}, \mathsf{H}_2(\overline{dpk}, ctx))$. Output π_{se} to $\mathcal{B}$'s challenger. By Lemma 2, we have that there are adversaries $\mathcal{B}_2, \mathcal{B}_3$ for the DL problem with advantages shown in the lemma statement.

Game_6 We abort if the extracted $r_3 = 0$. In this case we have $0_{\mathbb{G}_1} = G_1 + dpk_i + \sum_{i \in I} \boldsymbol{d}_i H_i + \sum_{i \in [\ell] \setminus I} \boldsymbol{u}_i H_i$ for some $i \in [\mathsf{nk}]$ and can clearly construct $\mathcal{B}_7$ which wins the rel-DL game with $n = \ell$ just by setting $H_i \leftarrow X_i$ for all $i \in [0, \ell]$.

Game_7 We add credential forgeries to the set of trivial showings, as well as showings for invalid keys and malleability forgeries. Define

$$\begin{aligned} T_C &:= \{(dpk, ctx, \phi) : \mathtt{SEKeys}(dpk) \neq \bot \wedge dpk \notin \mathtt{HUsers} \wedge \\ &\qquad (dpk \notin \mathtt{CSEs} \implies (dpk, ctx) \in \mathtt{Granted})\} \\ T_I &:= \{(dpk, ctx, \phi) : \mathtt{SEKeys}(dpk) = \bot, ctx \in \{0,1\}^*, \phi \in \Phi\} \\ T_M &:= \{(dpk, ctx, \phi) : \mathtt{SEKeys}(dpk) \neq \bot \wedge dpk \in \mathtt{HUsers} \wedge \\ &\qquad (ctx, \phi) \notin \mathtt{HShown}\} \end{aligned}$$

and set $T \leftarrow T \cup T_I \cup T_C$. Consider the reduction $\mathcal{B}_4$ playing the BBS EUF-CMA game:

1. On input par, generators $H_0, \ldots, H_\ell$, public key $pk = xG_2$, and with oracle access to $\mathcal{O}_{\mathsf{Sign}}$, run $\mathcal{A}$ with the following changes:
   (a) Instead of IssKGen just set $ipk \leftarrow pk$.
   (b) In $\mathcal{O}_{\mathsf{Issue}}$ and $\mathcal{O}_{\mathsf{hIssue}}$, look up $dsk \leftarrow \mathtt{SEKeys}(dpk)$ and instead of computing $\mathsf{Issue}(isk, dpk, \boldsymbol{a})$ (which can't be done without knowledge of isk), just query $\mathcal{O}_{\mathsf{Sign}}(\boldsymbol{a}, dsk)$.
2. When $\mathcal{A}$ is done, we have from the extractor that $\zeta_0 H_0 = \overline{dpk}$ and we have $dpk_1^* = \overline{dpk} - r_{\mathrm{key}} H_0 = (\zeta_0 - r_{\mathrm{key}}) H_0$. Additionally, the extractor computed $\boldsymbol{x} = (r_3, \boldsymbol{u}, r_{\mathrm{key}}, r_2, e)$ such that $M\boldsymbol{x} = (Y, \overline{B})$. This tells us that
$$r_3 \overline{C} + r_{\mathrm{key}} H_0 - \sum_{i \in [\ell] \setminus I} \boldsymbol{u}_i H_i = G_1 + \overline{dpk} + \sum_{i \in I} \boldsymbol{d}_i H_i$$
and $r_2 \overline{C} - e\overline{A} = \overline{B}$. Note that verification means $\mathsf{e}(\overline{A}, ipk) = \mathsf{e}(\overline{B}, G_2)$ which implies $\overline{B} = isk \cdot \overline{A}$, so if we have $r_2 = 0$ then we have $-e\overline{A} = isk \cdot \overline{A}$

which implies $isk = -e$. In that case we have recovered the BBS secret key and we can just create a signature on a message we haven't queried to $\mathcal{O}_{\mathsf{Sign}}$. Otherwise, let us proceed assuming that $r_2 \neq 0$. Compute $A \leftarrow r_3^{-1} r_2^{-1} \overline{A}$ and $C \leftarrow r_3^{-1} \overline{C}$. By bilinearity and using $r_2 \overline{C} - e\overline{A} = \overline{B}$, we have that $\mathsf{e}(\overline{A}, ipk) = \mathsf{e}(\overline{B}, G_2) \iff \mathsf{e}(A, ipk + eG_2) = \mathsf{e}(C, G_2)$. Let $\boldsymbol{m}$ be the union of the undisclosed attributes $\boldsymbol{u}$ and disclosed attributes $\boldsymbol{d}$ so that $C = G_1 + (\zeta_0 - r_{\text{key}})H_0 + \sum_{i=1}^{\ell} m_i H_i$. Finally, output $\boldsymbol{m} \| (\zeta_0 - r_{\text{key}}), (A, e)$. First, if dpk_1^* is a completely invalid key (i.e. $\texttt{SEKeys}(dpk_1^*) = \bot$), then clearly the H_0 attribute is fresh. If dpk_1^* is a valid key and $dpk \notin \texttt{HUsers}$, then we know that there is no $\boldsymbol{a}$ such that $(dpk, \boldsymbol{a}) \in \texttt{CCreds}$ and $\phi(\boldsymbol{a}) = 1$, as this case was covered by the previous game, and hence we know that one of the attributes corresponding to $H_1, \ldots, H_\ell$ is fresh. Lastly, in the event that the forgery is a "malleability" forgery, i.e., $\texttt{SEKeys}(dpk) \neq \bot$ and $dpk \in \texttt{HUsers}$ and $(ctx, \phi) \notin \texttt{HShown}$, then we also have a forgery since we never asked the signing oracle for a credential with attribute $\texttt{SEKeys}(dpk)$ for the 0-th attribute.

Game_8 Abort if all SEs are corrupted at the end of the game. This cannot be the case because the only possible forgeries would be one of the BBS forgeries that we ruled out in the previous game.

Game_9 Note that up until this point the extractor has assumed that $\mathcal{A}$ will output algebraic representations in terms of $G_1, H_0, \ldots, H_\ell$ only, and in particular not in terms of any the (dpk_i). This has been fine because we've known dsk_i for all dpk_i, but that will no longer be the case in the next hybrid. Now we say that the extractor loses if $\mathcal{A}$ gives the extractor representations of $R_1, \overline{C}$ or $R_2, \overline{C}, \overline{A}$ in terms of any dpk_i which is uncorrupted at the end of the game, such that there is no solution to the R_1 and R_2 linear systems (without "unrolling" the fact that $dpk_i = dsk_i H_0$ for all of the uncorrupted SEs). Clearly there exists a reduction $\mathcal{B}_5$ playing $\mathsf{rel\text{-}DL}$ (parametrized to $n = \ell + 1$ to have $\ell + 2$ generators) which succeeds with probability $1/\mathsf{nk}$ times the advantage of $\mathcal{A}$ by guessing $i^* \in [\mathsf{nk}]$ such that dpk_{i^*} is uncorrupted at the end of the game.

Game_{10} We add presentation forgeries to the set of trivial showings. Define

$$T_P := \{(dpk, ctx, \phi) | \texttt{SEKeys}(dpk) \neq \bot \ \wedge \ dpk \notin \texttt{CSEs} \wedge (dpk, ctx) \notin \texttt{Granted}\}$$

and set $T \leftarrow T \cup T_P$. Consider the following algorithm $\mathcal{B}_6$ which, given par, pk, and oracle access to $\mathcal{O}_{\mathsf{Sign}}$, plays the BLS EUF-CMA game in the following way:

1. First, guess $i^* \leftarrow\!\!\$\ [\mathsf{nk}]$.
2. Simulate the entire game to $\mathcal{A}$ with the following changes:
   (a) Use par as the public parameters instead of calling BGGen.
   (b) On the i^*-th query to $\mathcal{O}_{\mathsf{NewSE}}$, respond with pk instead of a randomly chosen key.
   (c) On a query $\mathcal{O}_{\mathsf{ShowSE}}(dpk, umsg, ctx)$ where $dpk = pk$, instead of $\mathsf{ShowSE}_1(dsk, ipk, ctx, umsg)$ (which could not be computed because dsk is not known), return $\mathcal{O}_{\mathsf{Sign}}(ctx)$ instead.

3. Regarding $\mathcal{A}$'s output, first check if $dpk_1^* = pk$ and abort if not.
4. Otherwise, parse $(\overline{dpk}, \pi_{\text{se}}, \overline{A}, \overline{B}, \overline{C}, \pi_{\text{bbs}}) \leftarrow \tau_1^*$ and output $(ctx_1^*, \pi_{\text{se}})$.

Considering $\mathcal{A}$'s final showing $(dpk_1^*, ctx_1^*, \phi_1^*)$, the only case that concerns this game change is that dpk_1^* is an uncorrupted key and $\mathcal{O}_{\mathsf{ShowSE}}$ was never queried for dpk_1^* and ctx_1^*. Since dpk_1^* is guaranteed to be in the set of uncorrupted keys, and $\mathcal{B}_6$ randomly chose one uncorrupted key as pk, there is a $1/\mathsf{nk}$ chance that $\mathcal{B}_6$ guessed correctly and does not abort. Since $\mathcal{A}$'s final showing is guaranteed to be valid we have that π_{se} is a valid signature under pk on message ctx_1^*. Additionally, since we have that $\mathcal{O}_{\mathsf{ShowSE}}$ was never queried for dpk_1^* and ctx_1^* (since $(dpk_1^*, ctx_1^*) \notin \texttt{Granted}$), we never queried $\mathcal{O}_{\mathsf{Sign}}$ with ctx_1^*. We can conclude that $\mathcal{B}_6$ wins the BLS EUF-CMA game.

Game_{11} The adversary always loses. In T_I we ruled out the possibility of a key being invalid, i.e. $\texttt{SEKeys}(dpk) = \bot$, so we can assume $\texttt{SEKeys}(dpk) \neq \bot$. If $dpk \in \texttt{CSEs}$ and $dpk \in \texttt{HUsers}$, then the set of trivial showings combined with T_M rule out any showing. If $dpk \in \texttt{CSEs}$ and $dpk \notin \texttt{HUsers}$, then T_C rules out any showing. If $dpk \notin \texttt{CSEs}$ and $dpk \in \texttt{HUsers}$, then again T_M and the trivial showings rule out any showing. If $dpk \notin \texttt{CSEs}$ and $dpk \notin \texttt{HUsers}$, then T_C rules out the possibility of showing for any predicate where $(dpk, ctx) \in \texttt{Granted}$, and combined with T_P all showings are ruled out. Lastly, if $dpk \notin \texttt{CSEs}$ and $dpk \in \texttt{HUsers}$, then T_M combined with the trivial showings rules out any showing. By this point in the game T covers all possible showings.

F.3 BBS-Schnorr

We provide the extended version of Theorem 1 and prove it.

Theorem 15 (BBS-Schnorr UNF, extended theorem). *Let* BGGen *be a bilinear group generator. Let $\mathcal{A}$ be an algebraic adversary against the unforgeability of BBS-Schnorr making at most $q_{\mathsf{H}_1}, q_{\mathsf{H}_2}, \mathsf{nk}, q_s$ queries to H_1, H_2, $\mathcal{O}_{\mathsf{NewSE}}$, and $\mathcal{O}_{\mathsf{hShow}}$ respectively. Define $\mathsf{Ext} = (\mathsf{Setup}, \mathsf{EKey})$ as in Theorem 14. Parameter indistinguishability follows trivially. There exist adversaries $\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3, \mathcal{B}_4, \mathcal{B}_5, \mathcal{B}_6$ running in time roughly that of $\mathcal{A}$ such that*

$$\begin{aligned}
\mathsf{Adv}^{\mathrm{UNF}}_{\mathit{BBS\text{-}Schnorr},\ell,\mathsf{Ext}}(\mathcal{A},\lambda) \leq\ & \mathsf{Adv}^{\mathit{rel\text{-}DL}}_{\mathsf{BGGen},\ell}(\mathcal{B}_1,\lambda) \\
& + \mathsf{Adv}^{\mathit{rel\text{-}DL}}_{\mathsf{BGGen},\ell}(\mathcal{B}_2,\lambda) \\
& + \mathsf{Adv}^{\mathit{rel\text{-}DL}}_{\mathsf{BGGen},\ell}(\mathcal{B}_6,\lambda) \\
& + \mathsf{Adv}^{\mathit{EUF\text{-}CMA}}_{\mathsf{BGGen},\mathsf{BBS},\ell}(\mathcal{B}_3,\lambda) \\
& + \mathsf{nk} \cdot \mathsf{Adv}^{\mathit{rel\text{-}DL}}_{\mathsf{BGGen},\ell+1}(\mathcal{B}_4,\lambda) \\
& + \mathsf{nk} \cdot \mathsf{Adv}^{\mathit{EUF\text{-}CMA}}_{\mathsf{BGGen},\mathsf{Schnorr},0}(\mathcal{B}_5,\lambda) \\
& + \frac{q_s(q_s + q_{\mathsf{H}_1}) + 2}{p}
\end{aligned}$$

Proof (sketch). The main technical differences compared to the proof for BBS-BLS are highlighted below:

1. In Game_{10}, the reduction plays the EUF-CMA game for Schnorr instead of BLS.
2. In Game_5, we need a different argument to show that $\zeta_i = 0$ for all $\zeta \in \{\hat{\zeta}, \zeta_1, ..., \zeta_\ell\}$. Considering $\pi_{\text{se}} = (c, \boldsymbol{s})$, the signature is valid if $c = \mathsf{H}_2(sH_0 - c \cdot \overline{dpk}, \overline{dpk}, ctx_1^*)$. We already handled the case where $\overline{dpk} = dpk_i$ for some $i \in [\mathsf{nk}]$, so we can assume this does not hold. As this is the case, we know that the adversary must have made the hash query themselves, otherwise verification holds with probability at most $1/p$. We can therefore view the algebraic coefficients ζ as being fixed when this hash query was made. The reduction $\mathcal{B}$ plays rel-DL with $n = \ell$ generators and simulates the game to $\mathcal{A}$ using the rel-DL challenge generators instead of randomly sampling them. When $\mathcal{A}$ makes the hash query $\mathsf{H}_2(R, \overline{dpk}, ctx_1^*)$, they provide algebraic coefficients $\{\hat{\zeta}, \zeta_1, ..., \zeta_\ell\}$ such that $\overline{dpk} = \hat{\zeta} G_1 + \sum_{i=0}^{\ell} \zeta_i H_i$ and $R = \hat{a} G_0 + \sum_{i=0}^{\ell} a_i H_i$. After both of those coefficients are fixed, the game randomly samples c, and $\mathcal{A}$'s output has to satisfy $sH_0 - c \cdot \overline{dpk} = R$. Equivalently,

$$sH_0 = (\hat{a} + c \cdot \hat{\zeta}) G_1 + \sum_{i=0}^{\ell} (a_i + c \cdot \zeta_i) H_i,$$

which yields a non-trivial rel-DL solution with probability $1 - 1/p$ unless $\zeta_i = 0$ for all $\zeta \in \{\hat{\zeta}, \zeta_1, ..., \zeta_\ell\}$.

F.4 BBS-BlindBLS

Next, we turn to the extended version and proof of Theorem 5.

Theorem 16 (BBS-BlindBLS UNF, extended theorem). *Let* BGGen *be a bilinear group generator. Let $\mathcal{A}$ be an algebraic adversary against the unforgeability of BBS-BlindBLS making at most $q_{\mathsf{H}_1}, q_{\mathsf{H}_2}, \mathsf{nk}, q_s$ queries to H_1, H_2, $\mathcal{O}_{\mathsf{NewSE}}$, and $\mathcal{O}_{\mathsf{hShow}}$ respectively. Define $\mathsf{Ext} = (\mathsf{Setup}, \mathsf{EKey})$ as in Theorem 14. Parameter indistinguishability follows trivially. There exist adversaries $\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3$,*

$\mathcal{B}_4, \mathcal{B}_5, \mathcal{B}_6, \mathcal{B}_7, \mathcal{B}_8, \mathcal{B}_9$ *running in time roughly that of* $\mathcal{A}$ *such that*

$$\begin{aligned}
\mathsf{Adv}^{\mathit{UNF}}_{\mathit{BBS\text{-}BlindBLS},\ell,\mathsf{Ext}}(\mathcal{A},\lambda) \leq\ & \mathsf{Adv}^{\mathit{rel\text{-}DL}}_{\mathsf{BGGen},\ell}(\mathcal{B}_1,\lambda) \\
& + q_{\mathsf{H}_2} \cdot \left(\mathsf{Adv}^{\mathit{DL}}_{\mathsf{BGGen},1}(\mathcal{B}_2,\lambda) + \mathsf{Adv}^{\mathit{DL}}_{\mathsf{BGGen},2}(\mathcal{B}_3,\lambda) + \frac{1}{p}\right) \\
& + \mathsf{Adv}^{\mathit{rel\text{-}DL}}_{\mathsf{BGGen},\ell+1}(\mathcal{B}_8,\lambda) \\
& + \mathsf{Adv}^{\mathit{rel\text{-}DL}}_{\mathsf{BGGen},\ell}(\mathcal{B}_9,\lambda) \\
& + \mathsf{Adv}^{\mathit{EUF\text{-}CMA}}_{\mathsf{BGGen},\mathsf{BBS},\ell}(\mathcal{B}_4,\lambda) \\
& + \mathsf{nk} \cdot \mathsf{Adv}^{\mathit{rel\text{-}DL}}_{\mathsf{BGGen},\ell+1}(\mathcal{B}_5,\lambda) \\
& + \mathsf{nk} \cdot \mathsf{Adv}^{\mathit{EUF\text{-}CMA}}_{\mathsf{BGGen},\mathsf{BLS},0}(\mathcal{B}_6,\lambda) \\
& + \mathsf{nk} \cdot \mathsf{Adv}^{\mathit{OM\text{-}UNF}}_{\mathsf{BGGen},\mathsf{BlindBLS},0}(\mathcal{B}_7,\lambda) \\
& + \frac{q_s(q_s + q_{\mathsf{H}_1}) + 1}{p}
\end{aligned}$$

Proof (sketch). The main technical differences compared to the proof for BBS-BLS are highlighted below:

1. There is no need for Game_5, because these kinds of forgeries are already ruled out in the blind unforgeability definition.
2. In the sequence of hybrids, we need to rule out the possibility of a one-more forgery, and for this we reduce to the (weak) one-more unforgeability of BLS. Consider the reduction $\mathcal{B}$ which, given $par, H_0, \ldots, H_\ell, pk$, and oracle access to $\mathcal{O}_{\mathsf{Sign}}(\boldsymbol{a})$, proceeds as follows:
   (a) Guess $i^* \in [\mathsf{nk}]$
   (b) Simulate the entire game to $\mathcal{A}$, with the following changes:
      i. On the (i^*)-th query to $\mathcal{O}_{\mathsf{NewSE}}$, abort if $\rho \neq \bot$. Otherwise, output pk.
      ii. On a query $\mathcal{O}_{\mathsf{ShowSE}}(dpk, \overline{m})$ with $dpk = pk$, query $smsg \leftarrow \mathcal{O}_{\mathsf{Sign}}(\overline{m})$ and return $smsg$
   (c) Parse $(\overline{dpk}^{i^*}, \pi_{\mathrm{se}}^{i^*}, \overline{A}^{i^*}, \overline{B}^{i^*}, \overline{C}^{i^*}, \pi_{\mathrm{bbs}}^{i^*}) \leftarrow \tau_i^*$ for all $i \in [k+1]$
   (d) Output $((\overline{dpk}^{i^*}, ctx_{i^*}), \pi_{\mathrm{se}}^{i^*})_{i=1}^{k}$.

   Note that we already ruled out the possibility that $\texttt{SEKeys}(dpk_1^*) = \bot$ in a previous hybrid. Hence there is a $1/\mathsf{nk}$ probability that $\mathcal{B}$ correctly guesses i^* and consequently $dpk_1^* = pk$. From $\mathcal{A}$'s winning conditions we have that $pk \notin \texttt{CSEs}$, which is good because we would not be able to simulate this, $pk \notin \texttt{HUsers}$, which again is good because while we could simulate this, it might mean that there were more $\mathcal{O}_{\mathsf{Sign}}$ queries than forgeries, and lastly we have that $\texttt{GrantedCount}(pk) < k$ which means we queried $\mathcal{O}_{\mathsf{Sign}}$ fewer than k times. On the other hand we have that $ctx_i^* \neq ctx_j^*$ for all $i \neq j \in [k]$ and that all the signatures are valid, so $\mathcal{B}$ wins the one-more unforgeability game.

G Privacy: Full Proofs

G.1 BBS-Schnorr

Let us first state the extended version of Theorem 2 before proving that construction BBS-Schnorr satisfies UNLINK-1.

Theorem 17 (BBS-Schnorr UNLINK-1, extended theorem). *Let* $\mathsf{Sim}_{\mathsf{BBS}} = (\mathsf{Prove}, \mathsf{H}_1)$ *be a zero-knowledge simulator for* $\mathsf{BBS}[\mathsf{H}_1]$ *and* H_1 *be a RO. Then for leakage function* $\mathcal{L} : isk \to (U \leftarrow\!\$\ \mathbb{G}_1, isk \cdot U)$ *and simulator* $\mathsf{Sim} = (\mathsf{Setup}, \mathsf{Show}, \mathsf{H}_1)$ *given in Figure 12,* BBS-Schnorr *is* UNLINK-1.

Proof. We show that the "real world" $\text{UNLINK-1}^{\mathcal{A},b=1}_{\mathsf{DBAC},\ell,\mathsf{Sim},\mathcal{L}}$, where presentations are created using BBS-Schnorr, is indistinguishable from the "ideal world" $\text{UNLINK-1}^{\mathcal{A},b=0}_{\mathsf{DBAC},\ell,\mathsf{Sim},\mathcal{L}}$, where presentations are created by a simulator, without any identifying information. On a high level, in a series of game hops we will successively remove device-identifying information from the presentation until a simulator can run the procedure. The intermediate games are shown in Figure 12. Game_0 will be the real world of UNLINK-1.

Game_1. First, we detach π_{se} from the dsk. To this end, we generate independent keys (dsk', dpk'). We want to use these keys to create π_{se}, but maintain the same distribution, so we also adapt the re-randomization of the signature: the blinding r'_{key} is set to be a transform from keys (dsk', dpk') to $(\overline{dsk}, \overline{dpk})$.

$\mathsf{Game}_0 \to \mathsf{Game}_1$. The games are indistinguishable from the adversarial point of view, since despite changing how $\overline{dpk}$ and π_{se} are constructed, the results are identical. Re-randomizing dpk' with r'_{key} results exactly in $\overline{dpk}$ from Game_0:

$$dpk' + r'_{\text{key}} H_0 = dsk' \cdot H_0 + (dsk + r_{\text{key}} - dsk')H_0 = dpk + r_{\text{key}} H_0$$

Analogously, π_{se} is identical in both games. In Game_1, we adapt the signature signed by dsk', which has the form $(c, s' = c \cdot dsk' + \omega')$, with r'_{key}. This yields a π_{se} identical to adapting a signature signed by dsk with r_{key} as in Game_0:

$$s' + c \cdot r'_{\text{key}} = c \cdot dsk' + \omega' + c \cdot (dsk + r_{\text{key}} - dsk') = c \cdot (dsk + r_{\text{key}}) + \omega'$$

Game_2. Next we also change how the π_{bbs} part of the presentation is generated; in particular, we get rid of the credential and the non-disclosed attributes. Therefore, we make use of the zero-knowledge property of BBS and invoke the corresponding simulator $\mathsf{Sim}_{\mathsf{BBS}}$ to simulate a proof of knowledge of a credential. As noted by [**?**], this simulation is not possible solely based on the issuer public key ipk and parameters, but an additional leakage $U \leftarrow\!\$\ \mathbb{G}_1, isk \cdot U$ is necessary, which can be roughly interpreted as a public key version for the group $\mathbb{G}_1$. This requirement is a consequence of the type-3 pairing, where the $ipk \in \mathbb{G}_2$ cannot easily be translated into $\mathbb{G}_1$. The authors also justify the aforementioned leakage function by observing that it does not leak more about the isk than seeing

Fig. 12. Games for Unlink-1 of BBS-Schnorr. Changes in each game hop are boxed; larger chunks of unmodified code are truncated. At the bottom right, we also give the simulator Sim used in the proof.

an arbitrary signature. We construct $\overline{A}$ and $\overline{B}$ by re-randomizing the leakage $(U, isk \cdot U)$ and choose $\overline{C}$ randomly. Y is as before since it only involves public elements, i.e., the blinded key and disclosed attributes. Additionally, the $\mathsf{Sim}_{\mathsf{BBS}}$ gets the ability to program the corresponding random oracle H_1.

$\mathsf{Game}_1 \rightarrow \mathsf{Game}_2$. Let us justify how $\overline{A}$, $\overline{B}$, and $\overline{C}$ are indistinguishable between Game_1 and Game_2. In Game_1, $\overline{A}$ and $\overline{C}$ are re-randomized versions of A and C with truly random and independent factors $r_2 r_1$ and r_1, respectively. They can, therefore, not be distinguished from the truly random group elements we chose in Game_2. Finally, $\overline{B}$ is originally constructed as $r_2\overline{C} - e\overline{A}$, which equals $isk \cdot \overline{A}$. By setting $\overline{A}$ to δU and $\overline{B}$ to $isk \cdot \delta U$ we ensured this same relation in Game_2. So after constructing a correctly distributed input, the simulated π_{bbs} should also be indistinguishable from a real proof by the zero-knowledge property of BBS. We conclude that Game_1 and Game_2 are indistinguishable.

Game_3. Already at Game_2, we were creating the presentation from an independent key and without use of the credential or non-disclosed attributes. The only remaining issue to have the entire presentation be simulated is r'_{key}. So far, we still pretended that the blinded key $\overline{dpk}$ must be a re-randomization of the original dpk, but since r'_{key} was always an information-theoretical blinding factor, we might as well sample it freshly from $\mathbb{Z}_p$. The distribution remains identical, so Game_2 and Game_3 are identically distributed.

Game_4. The way we create presentations is now independent of any user-identifying information, so in Game_4 we re-organize the code to have a simulator Sim generate the entire presentation. It uses $\mathsf{Sim}_{\mathsf{BBS}}$ as a subroutine. As this is a purely conceptual change, Game_3 and Game_4 are identical. Adding the bounds of the prior game hops, we obtain exactly the advantage from Theorem 2 for any PPT adversary in the $\textsc{Unlink}\text{-}1^{\mathcal{A},b}_{\mathsf{DBAC},\ell,\mathsf{Sim},\mathcal{L}}$ game. □

No Privacy Against Malicious SEs In the following, we describe a formal attack to demonstrate that the BBS-Schnorr construction does not achieve the stronger privacy property Unlink-2.

BBS-Schnorr uses standard Schnorr signatures, which are not secure against algorithm substitution attacks [**?**]. For instance, the simple and efficient technique by [**?**] can be used to exfiltrate information from the SE to the outside, which an outside adversary can recover from as little as two signatures with different messages. The authors embed information in the randomness used the signature by employing an adversarially-keyed PRF instead of sampling honestly, then make use of the scheme's extractability to recover both the signing key and a second freely chosen value. All subverted signatures are still valid, so this substitution attack evidently allows us to break Unlink-2 efficiently, for instance, embedding the dpk into a presentation.

G.2 BBS-BLS

We give the full version of and prove Theorem 4, stating that construction BBS-BLS achieves UNLINK-2.

Theorem 18 (BBS-BLS UNLINK-2, extended version). *Let* $\mathsf{Sim}_{\mathsf{BBS}} = (\mathsf{Prove}, \mathsf{H}_1)$ *be a zero-knowledge simulator for* $\mathsf{BBS}[\mathsf{H}_1]$ *and* H_1 *be a RO. Then for leakage function* $\mathcal{L} : isk \to (U \leftarrow\!\!\$\ \mathbb{G}_1, isk{\cdot}U)$ *and simulator* $\mathsf{Sim} = (\mathsf{Setup}, \mathsf{Show}, \mathsf{H}_1)$ *given in Figure 13,* BBS-BLS *is* UNLINK-2.

Proof. The proof follows similar steps as the prior proof of UNLINK-1 for BBS-Schnorr, but we need to additionally exclude that the subverted $\widetilde{\mathsf{ShowSE}}$ outputs something else than an honest signature, so we prepend another game hop.

Game_1. BBS-BLS makes the user verify the BLS signature created by the SE and abort the showing in case verification fails. Since BLS signatures are deterministic, only a single signature will pass this test. We make use of this fact by changing $\mathcal{O}_{\mathsf{DoPres}}$ as follows: after checking that the subverted SE's *smsg* passes the test, we ignore *smsg* and simply replace it by a fresh, honestly generated signature. Since this unique valid BLS signature the only possible value that *smsg* can have at this point of the algorithm, the distributions of Game_0 and Game_1 are identical.

Game_2. Next, we apply the same trick as for BBS-Schnorr to build an SE signature independent of the actual device key: we sample a fresh key pair (dsk', dpk') and set the re-randomization r'_{key} to map from the fresh keys to the original blind $\overline{dpk}$. Then, we use dsk' for the honest signature and re-randomize HonSig' with r'_{key}. Thus, the resulting π_{se} is identical to its counterpart in Game_1.

For the rest of the proof, we repeat exactly the game hops that we did for Theorem 2. In short, for in Game_3 (called Game_2 for Theorem 2) we choose random $\overline{A}, \overline{B}$ and $\overline{C}$ with the help of $\mathcal{L}(isk)$, and let the BBS zero-knowledge simulator create π_{bbs}. In Game_4, we sample ${r_{\text{key}}}'$ freshly from $\mathbb{Z}_p$ to make it independent of *dsk*, and in Game_5, we switch from outputting the *real* presentation τ_1 to outputting the simulated τ_0, as the entire code to produce τ_1 can now we identically run by Sim. We refer the reader to Section 6.2 for a more detailed description and analysis. □

G.3 BBS-BlindBLS

We give the extended version of Theorem 6 stating that construction BBS-BlindBLS satisfies SE-OBLIV, and provide the full proof.

Theorem 19 (BBS-BlindBLS SE-OBLIV, extended theorem). *Let* H_2 *be a RO. Then for the empty leakage function* $\mathcal{L} : isk \to \bot$ *and simulator* $\mathsf{Sim} = (\mathsf{Setup}, \mathsf{ShowUser}_1, \mathsf{Check})$ *given in Figure 14,* BBS-BlindBLS *is SE-OBLIV.*

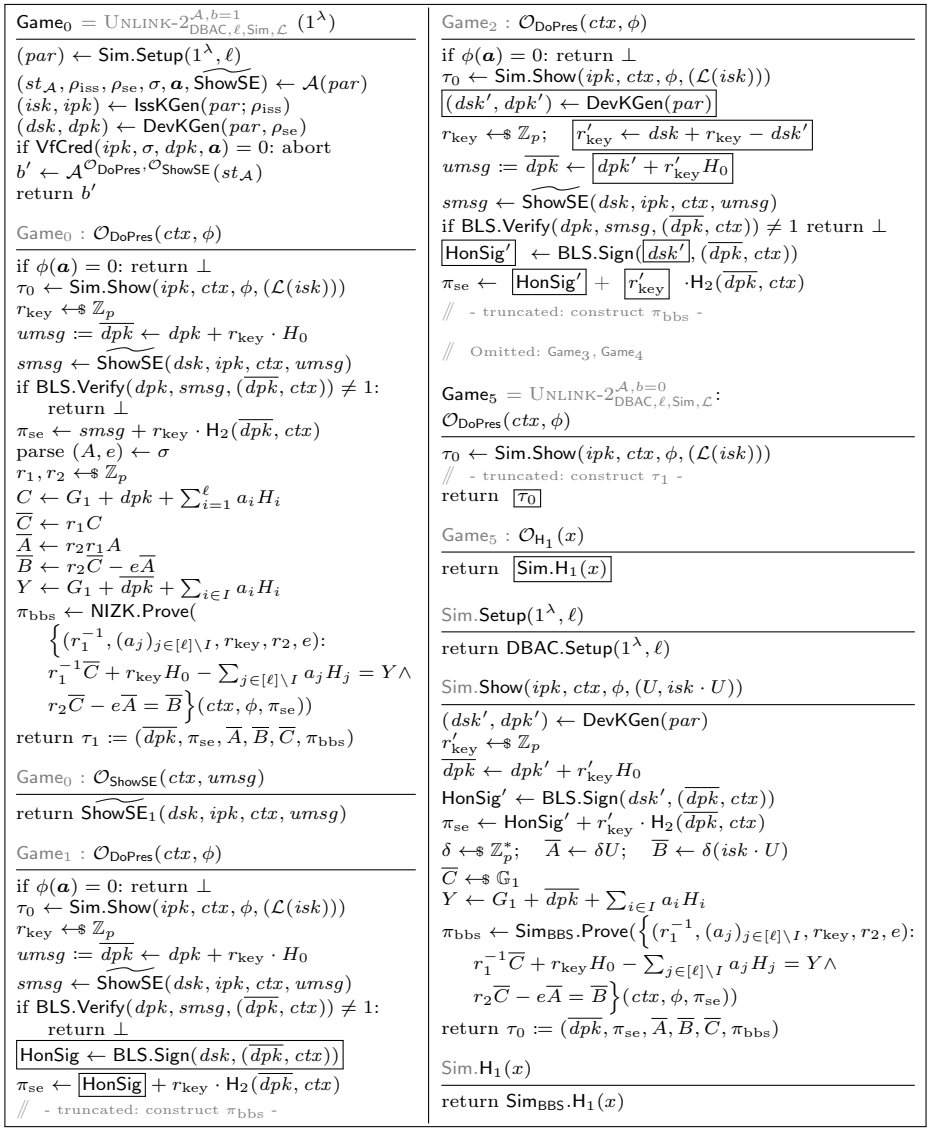

Fig. 13. Games for UNLINK-2 of BBS-BLS. Changes in each game hop are boxed; larger chunks of unmodified code are truncated.

$\text{SE-OBLIV}^{\mathcal{A},b}_{\mathsf{BDBAC},\ell,\mathsf{Sim},\mathcal{L}}$: $\mathsf{Sim.Setup}(1^\lambda, \ell)$	$\mathsf{Sim.ShowUser}_1(ipk, dpk, \mathcal{L}(isk))$
$par \leftarrow \mathsf{BDBAC.Setup}(1^\lambda, \ell)$ return (par)	$bst \leftarrow\!\!\$\ \mathbb{Z}_p$; $\quad B \leftarrow\!\!\$\ \mathbb{G}_2$ $umsg_{sid,0} \leftarrow bst \cdot B$ $ust_{sid,0} \leftarrow (dpk, B, bst)$ return $(ust_{sid,0}, umsg_{sid,0})$
$\mathsf{Sim.Check}(ust_{sid,0}, smsg, \mathcal{L}(isk))$	
parse $(dpk, B, bst) \leftarrow ust_{sid,0}$ return $\mathsf{e}(G_1, bst^{-1} \cdot smsg) \stackrel{?}{=} \mathsf{e}(dpk, B)$	

Fig. 14. Simulator for context-blind SE-OBLIV of BBS-BlindBLS.

Proof. We first make use of the statistical blinding and random oracle H_2 to argue that $\mathcal{O}_{\mathsf{ShowUser},1}$ reveals nothing and we must not include $\overline{dpk}$ and ctx into the user message. Then we argue that if $umsg$ does not "commit" to a specific content of the SE signature, we might as well create it from scratch in $\mathcal{O}_{\mathsf{ShowUser},2}$. At that point, the remaining computation steps can be executed by a simulator.

Game_1. Let us first modify the user messages $umsg$ output by $\mathcal{O}_{\mathsf{ShowUser},1}$. Instead of sampling $bst \leftarrow\!\!\$\ \mathbb{Z}_p$ and blinding $\overline{dpk}$ and ctx as $bst \cdot \mathsf{H}_2(\overline{dpk}, ctx)$, we simply choose a random $B \leftarrow\!\!\$\ \mathbb{G}_2$ and set $bst \cdot B$ as the blind "message", which contains no information. Accordingly, we need to fix the signature verification in $\mathcal{O}_{\mathsf{ShowUser},2}$ so that we still only accept the unique, honestly created $smsg$: we save B additionally to the user state and in $\mathcal{O}_{\mathsf{ShowUser},2}$, we replace signature verification by $\mathsf{e}(G_1, (bst)^{-1} \cdot smsg) \stackrel{?}{=} \mathsf{e}(dpk, B)$. If the verification passes, we set $smsg \leftarrow \mathsf{BLS.Sign}(dsk, (\overline{dpk}, ctx))$ and continue.

$\mathsf{Game}_1 \to \mathsf{Game}_2$. The factor bst blinds $umsg$ with statistical security, it only fails when blinding element 0, but a preimage of 0 for random oracle H_2 is hard to find. The adversary gets only one attempt at finding $(\overline{dpk}, ctx)$ that is mapped to 0 per query to both $\mathcal{O}_{\mathsf{ShowUser},1}$ and $\mathcal{O}_{\mathsf{ShowUser},2}$ ($\mathcal{A}$ has no control over which $\overline{dpk}$ $\mathcal{O}_{\mathsf{ShowUser},1}$ chooses), so with q_{H_2} such queries a preimage is found only with probability $q_{\mathsf{H}_2}/|\mathbb{G}_2|$. Therefore, we can replace the hashed value with a random B without the $\mathcal{A}$ noticing. The updated check ensures that only accept $smsg = dsk \cdot umsg$, as before, and we still build the presentation τ_1 from the unique honest SE signature. Consequently, the games are indistinguishable except with negligible probability.

Game_2. Instead of creating an honest signature HonSig on the $\overline{dpk}$ that was pre-computed in $\mathcal{O}_{\mathsf{ShowUser},1}$ and stored in ust, we select a fresh $\overline{dpk}$ on the spot as a re-randomization of dpk. The rest of the presentation generation remains the same. Since the adversary never got to see any artifact related to $\overline{dpk}$ before, the change is unnoticeable.

Game_3. Finally, we switch to $b = 0$ and let the simulator compute $umsg$ and perform the check on $smsg$ as described above. This is a purely structural change, so there is no change. We show the final simulator in Figure 14. □