Foundations of Anonymous Signatures: Formal Definitions, Simplified Requirements, and a Construction Based on General Assumptions

Jan Bobolz[1], Jesus Diaz[2], and Markulf Kohlweiss[3]

[1] University of Edinburgh, UK – jan.bobolz@ed.ac.uk
[2] Input Output, Spain – jesus.diazvico@iohk.io
[3] University of Edinburgh and Input Output, UK – markulf.kohlweiss@iohk.io

Abstract. In today's systems, privacy is often at odds with utility: users that reveal little information about themselves get restricted functionality, and service providers mistrust them. In practice, systems tip to either full anonymity (e.g. Monero), or full utility (e.g. Bitcoin). Well-known cryptographic primitives for bridging this gap exist: anonymous credentials (AC) let users disclose a subset of their credentials' attributes, revealing to service providers "just what they need"; group signatures (GS) allow users to authenticate anonymously, to be de-anonymized "just when deemed necessary". However, these primitives are hard to deploy. Current AC and GS variants reach specific points in the privacy-utility tradeoff, which we point as counter-productive engineering-wise, as it requires full and error-prone re-engineering to adjust the tradeoff. Also, so far, GS and AC have been studied separately by theoretical research. We take the first steps toward unifying and generalizing both domains, with the goal of bringing their benefits to practice, in a flexible way. We give a common model capturing their core properties, and use functional placeholders to subsume intermediate instantiations of the privacy-utility tradeoff under the same model. To prove its flexibility, we show how concrete variants of GS, AC (and others, like ring signatures) can be seen as special cases of our scheme – to which we refer as *universal anonymous signatures* (UAS). In practice, this means that instantiations following our construction can be configured to behave as variant X of a GS scheme, or as variant Y of an AC scheme, by tweaking a few functions.

1 Introduction

Anonymous signatures aim to strike a balance between the utility of authenticating identity information and the privacy offered by unlinkability. Exploring different privacy-vs-utility tradeoffs has been at the core of well-known cryptographic primitives for decades. Anonymous credentials (AC) [17] allow users to selectively disclose attributes to verifiers. Group signatures (GS) [18] let users prove group membership, preserving anonymity unless an authority de-anonymizes them. Related schemes, like ring signatures or direct anonymous attestation, share their goal: preserving privacy while authenticating some useful information.

Utility at authentication time. AC schemes focus on offering utility at the moment in which a user shows possession of a credential, often enabling selective attribute disclosure, or arbitrary predicates – e.g. "I am over 18 and from an EU country." In contrast, GS schemes create signatures over arbitrary messages that, in addition, prove the statement "I am a valid member of this group."

Utility after authentication time. Camenisch et al. [14,11] mention AC schemes with conditional release of information after authentication – but, to the best of our knowledge, no formal model is provided, and this type of utility is not frequent in AC. In contrast, GS schemes emphasize utility after signing time, typically allowing trusted parties to open signatures for signer identification.

Utility at issuance time. Frequently, the term utility refers to information revealed by signatures or the authentication process, but some AC schemes add extra semantics to issuance. For instance, some schemes support using previously obtained credentials to request new ones [11]. In the GS literature, as far as we know, extended behavior at issuance time has not been considered so far.

Why is this not ideal? Many AC and GS variants with different privacy-vs-utility tradeoffs exist, covering many use cases. Yet, real world adoption is limited, with exceptions like Hyperledger Anoncreds [35]. We explore potential reasons:

From a security point of view, while reference models exist such as the *foundations of group signatures* series [3,4,8], each variant requires a slightly different model to capture privacy and unforgeability properties that deviate from the established tradeoff. That is: whenever we aim for a new privacy-vs-utility tradeoff, a slightly different security model needs to be created, which is not a trivial task.

Engineering-wise, GS or AC schemes usually seem a good fit for privacy problems. But often, the privacy-vs-utility tradeoff needed is not exactly what existing schemes and implementations offer. Then, engineers face a trilemma: (1) if, luckily, a model and provably secure construction – but no implementation – exist for the desired tradeoff, they can implement it from scratch; (2) if an implementation for a closely related scheme exists, they can adapt it *ad hoc*; or (3) they may be forced to abandon the privacy-enhancing approach, to achieve the needed utility. (1) and (2) are error-prone and discouraged for production-ready systems; and (3) is bad for privacy. In another frequent setting, what engineers are demanded for v1 of their product may differ from what their v2 will need. While a flexible system may not always offer the most efficient implementation, it may still be more acceptable than rebuilding the solution from scratch.

Given these concerns, can we create a unified *tradeoff-dynamic* model spanning privacy and security of AC and GS schemes? Is there a generic construction for such a model, offering engineers flexibility to choose their desired privacy-vs-utility tradeoff without requiring full reimplementation, redesign, and proof?

1.1 Our contributions

A model with functional placeholders for dynamic privacy-vs-utility tradeoffs (Section 4). Customization is desirable during credential issuance, authentication, and after authentication. We use functional placeholders to capture possible tradeoffs, bringing the expressiveness of AC show predicates to issuance and opening. Modeling this flexibility requires abstracting anonymity, unforgeability, and non-frameability for issuance and signing. Usually, schemes prove knowledge of a key pair and credential(s), plus a tradeoff-dependent claim that can be captured by a function. Yet, capturing security and privacy with dynamic tradeoffs is harder than with static ones. E.g., in the dynamic case, one can define a function f such that, for two users with keys upk_1, upk_2, and credentials $\mathbf{crd}_1, \mathbf{crd}_2$, $y = f(upk_1, \mathbf{crd}_1) = f(upk_2, \mathbf{crd}_2)$. Thus, user upk_1 producing a signature that opens to y may not be a framing of user upk_2. To address it, we use extraction, and check that the extracted values are consistent with what is expected. We call the resulting scheme *Universal Anonymous Signatures* (UAS).

A generic construction (Section 5). We present a generic construction, Π_{UAS}, that we prove secure under our UAS model. We use BBS+ [1,10,34], a variant of CL signatures [15], providing randomizable attribute-based credentials. This signature scheme has been used before (e.g.[23]) to build Sign-Randomize-Proof GS schemes [6]. We also draw inspiration from the Sign-Encrypt-Prove approach to GS schemes [4]: we have the signer encrypt a function of their credential's attributes under an opener's public key and prove its correct computation. For issuance-time utility, the user proves, during an interactive protocol with the issuer, that a predefined function of their public key and credentials is acceptable.

Relationships with other schemes (Section 6). We study our UAS model and Π_{UAS} construction as a generalization of privacy-preserving signature and authentication schemes. We define several function combinations that instantiate specific privacy-vs-utility tradeoffs within our UAS framework – which we call Π_{UAS} *restrictions.* We prove that these restrictions imply well-known schemes, including digital signatures, GS, AC, and ring signatures, under their respective reference models. While we establish concrete connections for a few schemes, the space of possible Π_{UAS} restrictions is extensive. We give a glimpse of these more advanced connections in the full version [7], where we sketch how to build GS with message-dependent opening [20], multimodal private signatures [31], and revocable ACs [12]. Further restrictions can be easily imagined, giving schemes such as AC with auditability, ring signatures with some sort of linkability, etc. This allows engineers to adapt privacy-vs-utility tradeoffs by modifying restriction functions, maintaining security and controlling information leakage.

Before presenting our main construction, Section 2 introduces related work on closely related primitives, and Section 3 summarizes our construction's main building blocks. Further details are deferred to the full version [7].

Why do we need flexible privacy-vs-utility tradeoffs? A promising use case for digital identity is compliance for global decentralized financial infrastructures (e.g., *Zcash* or *Monero*), or in Centrally Banked Digital Currencies [26]. As the legal frameworks for such systems is still evolving, it is paramount that asset privacy be configurable [21]. UAS offers a principled way to achieve this.

2 Related Work

There are many anonymous signatures (AC, GS, or related) schemes that aim at achieving a different privacy-vs-utility tradeoff. In GS schemes the focus is on opening and linkability: [32] makes de-anonymization dependent on messages; [23] does not allow de-anonymization, but signatures by the same signer are linkable; in [30] signers are fully identified towards other group members and linkable for non group members; in [29], signers are de-anonymizable only if a predicate of their "identity" and the signed message is not satisfied. In AC schemes, the usual selective disclosure [14] is augmented to revealing arbitrary predicates on the credentials' attributes, e.g., in [19], which is the state of the art in utility at authentication time. Some works consider delegation capabilities [2,16], or revocation [12]. We now focus on schemes that aim at more flexible tradeoffs and at general security models that are scheme independent.

Related work on flexible tradeoffs. To the best of our knowledge, our work is the first to model flexible tradeoffs at all steps of the "credential and signature lifecycle". However, some works already pushed towards achieving more flexibility. Benoît et al. and Nguyen et al. [29,31] introduce the notion of what can be called "functional opening". That is, the information that the opener can learn is *a function of* the signer's identity, rather than the identity itself. However, their notion of "identity" is left abstract, which makes it hard to apply in real world settings. We give a concrete definition of identity, via credentials with attributes. Kohlweiss et al. [28] introduces generic functions for utility at opening time, that allow an auditor to learn a function of the user's credential and private information fixed by the auditor in advance.

Related work on general security models. Probably, the most relevant works towards achieving a common and generic model are those in the "Foundations of Group Signatures" line [3,4,9], with [27] proposed in parallel to [4]. Before them, many different models coexisted, each focusing on similar but slightly different security and privacy properties. In the AC domain no similar foundational line exists as far as we know, although [13] does a great job in subsuming previous works and proposing a modular approach towards AC schemes. In some sense, our goal is similar to these unifying works, but focusing on the achieved privacy-vs-utility tradeoff. As mentioned, there are currently many variants of both GS and AC schemes offering similar but slightly different tradeoffs, at the cost of introducing many similar but slightly different models. Our goal is to avoid that.

3 Preliminaries

Public-Key Encryption has Setup, KG, Enc, and Dec algorithms. $\mathrm{Setup}(1^\kappa)$ produces public parameters par. $\mathrm{KG}(par)$ generates a encryption-decryption key pair (ek, dk), $\mathrm{Enc}(ek, m)$ encrypts message m with ek and outputs ciphertext c. $\mathrm{Dec}(dk, c)$ decrypts ciphertexts using dk to retrieve message m. We rely on chosen plaintext secure public-key encryption schemes (IND-CPA).

Non-Interactive Zero-Knowledge (NIZK). A NIZK scheme [33] for a NP relation $\mathcal{R}$ has three algorithms: $\text{Setup}^{\mathcal{R}}$, $\text{Prove}^{\mathcal{R}}$, $\text{Verify}^{\mathcal{R}}$. Algorithm $\text{Setup}^{\mathcal{R}}(1^{\kappa})$ produces the common reference string crs. $\text{Prove}^{\mathcal{R}}(crs, x, w)$ creates a NIZK proof π of knowledge of witness w for x such that $(w, x) \in \mathcal{R}$. $1/0 \leftarrow \text{Verify}^{\mathcal{R}}(crs, x, \pi)$ verifies the proof. The properties we build on are completeness, soundness, zero-knowledgeness, and extractability. We require extractability to hold in the presence of a simulator (simulation extractability), and zero-knowledge to hold in the presence of an extractor (extraction zero-knowledge) [25].

Signatures over Blocks of Committed Messages, with proofs. We use schemes that allow signing blocks of messages, and commitments to blocks of messages, and which are also compatible with proof systems over the produced signature and signed (commitments to) messages. An SBCM scheme is as a tuple (Setup, KG,Blind, Sign, Unblind,Verify). $par \leftarrow \text{Setup}(1^{\kappa})$ produces some public parameters. $(vk, sk) \leftarrow \text{KG}(par)$ produces a verification-signing key pair. $c \leftarrow \text{Blind}(vk, \overline{\boldsymbol{msg}}, \boldsymbol{msg}, r)$ is run by a user to request a signature over $\overline{\boldsymbol{msg}} \cup \boldsymbol{msg}$, where $\boldsymbol{msg}$ are revealed to the signer, but $\overline{\boldsymbol{msg}}$ are signed in committed form. $\beta \leftarrow \text{Sign}(sk, c, \pi, \boldsymbol{msg})$ is run by the signer, to produce a partial signature β over a set of committed messages $\overline{\boldsymbol{msg}}$ and set $\boldsymbol{msg}$. $\sigma \leftarrow \text{Unblind}(vk, \beta, c, r, \overline{\boldsymbol{msg}}, \boldsymbol{msg})$ is run by the user to complete the signer's partial signature β. Finally, $1/0 \leftarrow \text{Verify}(vk, \sigma, \overline{\boldsymbol{msg}}, \boldsymbol{msg})$ verifies a signature σ over message vector $\overline{\boldsymbol{msg}} \cup \boldsymbol{msg}$. An SBCM scheme must be unforgeable and blind.

4 Formalizing UAS

In a UAS scheme, users generate their key pair, and optionally advertise their public key and conditions for issuance in order to become issuers. Openers first generate their key pairs, and then advertise what information they expect to learn from signatures. Users may later use their credentials to request new ones, or to produce a signature. Issuance only succeeds if the user's credentials (if any) meet the issuer's requirements. Signatures may directly output some information derived from the user's data. Also, each signature has a selected opener, who can later *learn only the information included by the user*. When openers learn their information, they also prove the correctness of the result, which can be verified by any interested party. This prevents openers from framing innocent users.

4.1 Syntax

In detail, a UAS scheme is composed of the following PPT algorithms:

$\text{Setup}(1^{\kappa}) \rightarrow par$**.** Given security parameter 1^{κ}, returns global system parameters par. We assume that par are passed implicitly to all other functions.

$\text{KG}(par) \rightarrow (upk, usk)$**.** Given par, a user generates a key pair (upk, usk). An *issuer* is a user who defines an issuance function f_{is}. We denote such issuance keys by $ipk = (upk, f_{\text{is}})$ and $isk = (ipk, usk)$.[4]

[4] The (pk, f) tuple simplifies our notation, while simultaneously guaranteeing that users can easily inspect the functions picked by issuers and openers. The second

$\mathrm{OKG}(par) \to (preopk, preosk)$. An opener runs OKG to generate its pre-opener keys. The opener externally extends the keys with an opening function f_{op}. We denote such opening keys $opk = (preopk, f_{\mathrm{op}})$ and $osk = (opk, preosk)$.

$\langle \mathrm{Obt}(upk, usk, ipk, \boldsymbol{C}, \boldsymbol{a}), \mathrm{Iss}(isk, \boldsymbol{ipk}, \boldsymbol{a}, y_{\mathrm{is}}) \rangle \to \langle C/\bot, R/\bot \rangle$. Lets a user with key usk obtain a credential $C = (cid, \boldsymbol{a}, crd, ipk)$ from an issuer with key (ipk, isk). cid is a unique identifier for the credential crd, on attribute set $\boldsymbol{a}$. The user may employ previously obtained credentials $\boldsymbol{C} = \{(cid_i, \boldsymbol{a}_i, crd_i, ipk_i)\}_{i \in [n]}$, from which we may omit the ipk_i for readability. The y_{is} value received by the issuer is the claimed output of f_{is}, over the user's data. Note that the issuer can reject initiating the protocol if y_{is} is not acceptable. The user outputs the issued credential C, and the issuer outputs $R \leftarrow (reg, cid)$, where reg is the protocol transcript.

$\mathrm{Sign}(upk, usk, opk, \boldsymbol{C}, m, f_{\mathrm{ev}}) \to (\sigma, y_{\mathrm{ev}})$. Upon receiving user secret key usk, opener public key opk, credentials $\boldsymbol{C}$, message m and evaluation function f_{ev}, returns signature σ, and a value y_{ev}. We use Σ to denote the tuple $(\sigma, y_{\mathrm{ev}})$.

$\mathrm{Verify}(opk, \boldsymbol{ipk}, \Sigma, m, f_{\mathrm{ev}}) \to 1/0$. Checks whether $\Sigma = (\sigma, y_{\mathrm{ev}})$ is a valid signature over message m, from a user with credentials issued by issuers with public keys in $\boldsymbol{ipk}$, for evaluation function f_{ev} and opener key opk.

$\mathrm{Open}(osk, \boldsymbol{ipk}, \Sigma, m, f_{\mathrm{ev}}) \to (y_{\mathrm{op}}, \pi)/\bot$. Run by the opener with private key osk. Receives a signature $\Sigma = (\sigma, y_{\mathrm{ev}})$ over message m and evaluation function f_{ev}, generated using credentials by issuers with public keys in $\boldsymbol{ipk}$. If Σ is valid, the function outputs a value y_{op}, and a proof of correct opening π.

$\mathrm{Judge}(opk, \boldsymbol{ipk}, y_{\mathrm{op}}, \pi, \Sigma, m, f_{\mathrm{ev}}) \to 1/0$. Checks if π is a valid opening correctness proof for the value y_{op}, obtained by applying Open to the the signature $\Sigma = (\sigma, y_{\mathrm{ev}})$ over message m, and for evaluation function f_{ev}.

Issuance, evaluation, and opening functions. These are the functional placeholders modulating the behavior of UAS instantiations. They control the conditions for issuing credentials, the information revealed alongside signatures, and the information revealed when opening signatures.

$f_{\mathrm{is}} : (upk, \boldsymbol{a}, \{(cid_i, \boldsymbol{a}_i)\}_{i \in [n]}) \to y_{\mathrm{is}}$. Chosen by each issuer within a family of functions $\mathcal{F}_{\mathrm{is}}$, the issuance function defines the conditions required by the issuer to grant a credential over attributes $\boldsymbol{a}$, when requested by a user with public key upk, optionally using a set of n endorsement credentials with identifiers and attributes given by $\{(cid_i, \boldsymbol{a}_i)\}_{i \in [n]}$. The range of f_{is} is R_{is}.

$f_{\mathrm{ev}} : (upk, \{(cid_i, \boldsymbol{a}_i)\}_{i \in [n]}, m) \to y_{\mathrm{ev}}$. Signing evaluation functions (or, simply, evaluation functions), from a family of functions $\mathcal{F}_{\mathrm{ev}}$, can be set on a per-signature basis. They receive the user public key upk, a set of credential identifiers and attributes $\{(cid_i, \boldsymbol{a}_i)\}_{i \in [n]}$ (where n may be 0), and the message to be signed m. f_{ev} outputs a value y_{ev} from a well defined set R_{ev}.

$f_{\mathrm{op}} : (upk, \{(cid_i, \boldsymbol{a}_i)\}_{i \in [n]}, m) \to y_{\mathrm{op}}$. Chosen by openers from a family of functions $\mathcal{F}_{\mathrm{op}}$. The opening functions define the utility value extractable from

tuple might appear surprising, but it is natural that the secret key in a public key scheme contains at least the information of the public key, e.g., RSA.

signatures. This value is derived from the user's upk, credentials' identifiers and attributes $\{(cid_i, \boldsymbol{a}_i)\}_{i \in [n]}$ $(n \geq 0)$ used for signing, and signed message m. It outputs a value y_{op} from a well defined set R_{op}.

For instance, let f_{is} (resp. f_{ev}) output the requesting user's (resp. signer's) public key, and f_{ev} output always 0. The corresponding R_{is} and R_{op} are the set of all possible user public keys, and R_{ev} is $\{0\}$. This combination leads to a $\mathit{\Pi}_{\mathsf{UAS}}$ restriction that behaves like a group signature scheme. See Section 6 for details.

Correctness. An UAS scheme is correct if a signature produced by an honest user, who chooses an honest opener and leverages only credentials obtained from honest issuers, is accepted by an honest verifier, and any opening proof honestly computed from such a valid signature is accepted by an honest judge. Moreover, the y_{ev} (resp. y_{op}) value attached to the signature must match the output of f_{ev} (resp. f_{op}) when evaluated on the endorsement credentials, signed message, and the user's upk. Similarly, the y_{is} value in the transcripts of all involved credentials used to produce the signature must match the output of f_{is} when computed from the requested attributes, user's upk, and any further endorsement credential involved in its issuance. Correctness is formalized in the full version [7].

4.2 Security Model

A UAS scheme must satisfy privacy and security properties, both for the issuance protocol, and for the produced signatures. We introduce them semi-formally here. The full formalization as experiments $\mathsf{Exp}^{\text{iss-anon-}b}_{\mathsf{UAS},\mathcal{A}}$, $\mathsf{Exp}^{\text{sig-anon-}b}_{\mathsf{UAS},\mathcal{A}}$, $\mathsf{Exp}^{\text{iss-forge}}_{\mathsf{UAS},\mathcal{A}}$, $\mathsf{Exp}^{\text{sig-forge}}_{\mathsf{UAS},\mathcal{A}}$, and $\mathsf{Exp}^{\text{frame}}_{\mathsf{UAS},\mathcal{A}}$ can be found in the full version.

Issuance anonymity. User obtains credentials by running an interactive $\langle \mathrm{Obt}, \mathrm{Iss} \rangle$ protocol with an issuer. The authorization of the issuing can employ previously obtained endorsement credentials to prove that the request meets the issuers requirements—captured by function f_{is} defined by the issuer. We model via the issuance anonymity property that no information about the endorsement credentials besides the output of f_{is} is revealed.

Formally, we define a left-or-right game. The adversary repeatedly plays the issuer in the $\langle \mathrm{Obt}, \mathrm{Iss} \rangle$ protocols against one out of two challenges selected by a random bit b that the adversary needs to guess. Each challenge specifies an honest user and its endorsement credentials. In addition, the adversary can obtain new (non-challenge) credentials for honest users that can be used as endorsement credentials, create signatures with the challenge and non-challenge credentials, open non-challenge signatures, and corrupt users, issuers, and openers at will. To avoid trivial wins, the adversary cannot mix challenge with non-challenge credentials when signing, and the output of the f_{is} function when obtaining challenge credentials and the outputs of f_{ev} and f_{op} functions when signing with challenge credentials need to be the same for both challenge users. To see why this is needed, consider a simple f_{is} function that outputs the public key of the user – which trivially allows an adversarial issuer to distinguish $\langle \mathrm{Obt}, \mathrm{Iss} \rangle$ runs.

Signature anonymity. In UAS, signatures come with signature utility information y_{ev}, computed by f_{ev}, and opener utility information y_{op} computed by f_{op} and only retrievable by the chosen opener. Signature anonymity captures that signatures do not leak any more information than specified by these functions.

Formally, we define a left-or-right game. The adversary can add honest and corrupt users at will, request signatures using arbitrary credentials, and open the resulting signatures. The adversary is given the capability to repeatedly request signatures for an opener, an evaluation function f_{ev}, a message, and one out of two challenges selected by a random bit b that the adversary needs to guess. Each challenge specifies a challenge users and a set of credentials. To avoid trivial wins, among other simple checks, the y_{ev} value output by f_{ev} has to be the same for both challenges. Similarly, if the opener is corrupt, the value output by f_{op} has to be the same for both challenges. The adversary can open signatures, but only if they are not challenge ones. This is to avoid trivial wins in which the opener was not initially corrupt, and thus the f_{op} equality check did not apply.

Issuance unforgeability. Honest issuers in the interactive $\langle \text{Obt}, \text{Iss} \rangle$ protocol to request new credentials specify an issuing policy f_{is} that involves endorsement credential attributes and identifiers (if any), the requesting user's public key, and the requested attributes. Issuance unforgeability captures that f_{is} must be met, and that the endorsement credentials (if any) have been legitimately obtained.

In a nutshell, the adversary can corrupt users, issuers, and openers, obtain credentials on behalf of honest or corrupt users, and produce signatures leveraging any of these credentials. Eventually, the adversary has to output a credential identifier, which must belong to a credential issued by an honest issuer. The adversary wins if the y_{is} value claimed by the user as an input to Iss does not match the expected output from f_{is}, or if there is a mismatch in the endorsement credentials' attributes or the associated user public key. Note that we require that the credential identifier output by the adversary corresponds to a credential produced by an honest issuer. While we have access to the issuance transcript, we cannot otherwise assume that we know the identifiers of the adversary's endorsement credentials, their attributes, or user public keys needed for evaluating f_{is} and running the required tests. Thus, we resort to extraction: the issuance transcript must allow for the extraction of these otherwise private values.

Note that the issuance unforgeability requirements apply recursively to endorsement credentials. To see this consider adversaries that perform the same oracle queries but output the credential identifiers of endorsement credentials. This excludes construction that allows the use of fraudulently obtained credentials when honestly obtaining a new credential.

Signature unforgeability. Signatures carry two types of utility: y_{ev}, produced by computing f_{ev}, and revealed alongside the signature; and y_{op}, produced by computing f_{op}, and learned by the chosen opener. No adversary should be capable of producing a signature that is accepted by Verify, yet contains *wrong* y_{ev} and y_{op} values. To check this, we let the adversary add corrupt users, issuers, and openers, obtain credentials on behalf of any existing user and issuer, produce signatures by honest users, as well as open any signature. The adversary is challenged to produce a signature, and wins if the signature is accepted by Verify, yet the y_{ev} value associated to it is *wrong*; or if an honest opener cannot produce a proof that is accepted by Judge, or for which the associated y_{op} value is *wrong*. The adversary also wins if any of the credentials used to produce the signature was fraudulently obtained. We define *wrong* by recomputing the f_{ev} and f_{op} functions: as in issuance unforgeability, we extract the necessary inputs from the adversary's signature forgery. We also extract from the issuance transcripts of honestly issued endorsement credentials to check that they were correctly issued.

Figure Fig. 1 explains the need for both issuance and signature unforgeability.

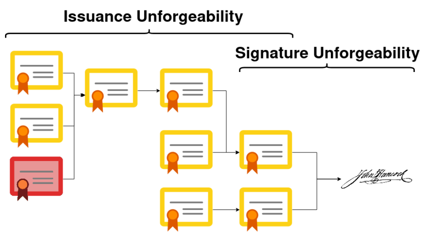

Fig. 1. Credential chain segments covered by each unforgeability property. Issuance unforgeability prevents forged credentials at any point, but is agnostic to signatures. Signature unforgeability prevents last-layer credential forgeries, and signature forgeries. E.g., only issuance unforgeability detects if the credential in red is a forgery.

Non-frameability. While the unforgeability properties capture the security expectations of verifiers and openers, non-frameability captures what expectations honest users can still have even when both issuers and openers are corrupt. Concretely, note that in the unforgeability properties, we open using the official opening algorithm. In contrast, in non-frameability the adversary has to produce both a valid signature and valid opening and proof. It wins if this signature was not produced via some of the oracles, if the signature and opening proof are valid, yet the output y_{op} values is *wrong*, or if the *upk* associated to the signature belongs to an honest user. This protects honest users from being framed, either by their *upk* being used to compute y_{op} or by y_{op} being made up in the first place. Again, we make use of extraction techniques to recover all the needed information for attesting correct computation f_{op}, and to learn the signer's *upk*.

5 Π_{UAS}: A Generic UAS Construction

We give a generic construction of a UAS scheme from generic building blocks. We use three NP relations: $\mathcal{R}_{\text{is}}$, $\mathcal{R}_{\text{ev}}$, and $\mathcal{R}_{\text{op}}$, described next. These relations include verifying correct computation of the f_{is}, f_{ev} and f_{op} functions.

$\mathcal{R}_{\text{is}}$**:** For NIZK proofs at issuance time. Requires users to prove knowledge of their (usk, upk) pair, and the requested credential is bound to usk. It also requires any endorsement credential to be a valid credential (signed by some issuer) and bound to usk, and enforces the corresponding f_{is} policy.

$\mathcal{R}_{\mathrm{ev}}$: For NIZK proofs at signing time. Ensures that signatures encode the correct signature evaluation (computed via f_{ev}) and opening values (computed via f_{op}), and all the involved credentials are bound to the same usk.

$\mathcal{R}_{\mathrm{op}}$: For NIZK proofs at opening time. Ensures that the utility information revealed by the opener, via the Open algorithm, is correct.

$$\mathcal{R}_{\mathrm{is}} = \left\{ \begin{array}{c} (f_{\mathrm{is}}, c, cid, \boldsymbol{a}, ipk, \{ipk_i\}_{i\in[n]}, y_{\mathrm{is}}), (upk, usk, \{(cid_i, \boldsymbol{a}_i, crd_i)\}_{i\in[n]}, r) : \\ (upk, usk) \in [\mathrm{KG}(par)] \wedge \\ c = \mathrm{SBCM.Blind}(ipk, usk, (cid, \boldsymbol{a}), r) \wedge \\ f_{\mathrm{is}}(upk, \boldsymbol{a}, \{(cid_i, \boldsymbol{a}_i)\}_{i\in[n]}) = y_{\mathrm{is}} \wedge \\ \forall i \in [n] :\ \mathrm{SBCM.Verify}(ipk_i, crd_i, usk, (cid_i, \boldsymbol{a}_i)) = 1 \end{array} \right\}$$

$$\mathcal{R}_{\mathrm{ev}} = \left\{ \begin{array}{c} (m, f_{\mathrm{ev}}, y_{\mathrm{ev}}, f_{\mathrm{op}}, c_{\mathrm{op}}, \{ipk_i\}_{i\in[n]}, ek), (upk, usk, \{(cid_i, \boldsymbol{a}_i, crd_i)\}_{i\in[n]}, y_{\mathrm{op}}, r) : \\ (upk, usk) \in [\mathrm{KG}(par)] \wedge c_{\mathrm{op}} = \mathrm{E.Enc}(ek, y_{\mathrm{op}}; r) \wedge \\ y_{\mathrm{ev}} = f_{\mathrm{ev}}(upk, \{(cid_i, \boldsymbol{a}_i)\}_{i\in[n]}, m) \wedge \\ y_{\mathrm{op}} = f_{\mathrm{op}}(upk, \{(cid_i, \boldsymbol{a}_i)\}_{i\in[n]}, m) \wedge \\ \forall i \in [n] :\ \mathrm{SBCM.Verify}(ipk_i, crd_i, usk, (cid_i, \boldsymbol{a}_i)) = 1 \end{array} \right\}$$

$$\mathcal{R}_{\mathrm{op}} = \left\{ \begin{array}{c} (ek, c, y_{\mathrm{op}}), (dk) : \\ (ek, dk) \in [\mathrm{OKG}(par, \cdot)] \wedge \\ y_{\mathrm{op}} = \mathrm{E.Dec}(dk, c) \end{array} \right\}$$

Fig. 2. Specification of the NP relations used in $\mathit{\Pi}_{\mathsf{UAS}}$. $\mathcal{R} = \{x, w : \mathrm{predicate}(x, w)\}$, where x is the public statement and w is the prover's secret witness.

We build $\mathit{\Pi}_{\mathsf{UAS}}$ as defined in Fig. 3 and Fig. 4. Briefly, Setup computes the public parameters for SBCM, Enc, and the three NIZK systems. KG generates an SBCM user key pair, and OKG an Enc key pair (ek, dk). If a user wishes to *upgrade* itself to an issuer, it sets $(ipk = (upk, f_{\mathrm{is}}), isk = (ipk, usk))$ for its chosen f_{is}. Similarly, an opener can upgrade (ek, dk) into $(opk = (ek, f_{\mathrm{op}}), osk = (opk, dk))$ by advertising f_{op} in a reliable manner, defining the utility it will accept to extract from signatures. In the $\langle \mathrm{Obt}, \mathrm{Iss}\rangle$ protocol, the user computes f_{is} over the requested attributes, upk, and the endorsement credentials and runs an SBCM blind signature protocol with the issuer, augmented with $\mathrm{NIZK.Prove}^{\mathcal{R}_{\mathrm{is}}}$. In Sign, the user computes f_{ev} and f_{op} over the message, attributes, and upk, encrypts y_{op} with the chosen opk, and proves correct computation via $\mathrm{NIZK.Prove}^{\mathcal{R}_{\mathrm{ev}}}$. Verify simply verifies the NIZK. In Open, if Verify accepts the signature, the opener decrypts c_{op} to get y_{op}, and outputs it along with a proof obtained via $\mathrm{NIZK.Prove}^{\mathcal{R}_{\mathrm{op}}}$. Judge verifies both the signature and the opening proof.

5.1 Correctness and Security of $\mathit{\Pi}_{\mathsf{UAS}}$

Correctness is by inspection of the honest algorithms. The uniqueness of credentials for honestly issued and obtained credential identifiers is a sub-property of

```
Setup(1^κ)
    $par_{\mathrm{SBCM}} \leftarrow \mathrm{SBCM.Setup}(1^\kappa)$
    $par_{\mathrm{E}} \leftarrow \mathrm{E.Setup}(1^\kappa)$
    $crs_{\mathrm{is}} \leftarrow \mathrm{NIZK.Setup}^{\mathcal{R}_{\mathrm{is}}}(1^\kappa)$
    $crs_{\mathrm{ev}} \leftarrow \mathrm{NIZK.Setup}^{\mathcal{R}_{\mathrm{ev}}}(1^\kappa)$
    $crs_{\mathrm{op}} \leftarrow \mathrm{NIZK.Setup}^{\mathcal{R}_{\mathrm{op}}}(1^\kappa)$
    return $par = (par_{\mathrm{SBCM}}, par_{\mathrm{E}},$
            $crs_{\mathrm{is}}, crs_{\mathrm{ev}}, crs_{\mathrm{op}})$

KG(par)
    $(par_{\mathrm{SBCM}}, \cdot, \cdot, \cdot, \cdot) \leftarrow par$
    $(vk, sk) \leftarrow \mathrm{SBCM.KG}(par_{\mathrm{SBCM}})$
    $upk \leftarrow (par, vk)$
    $usk \leftarrow sk$
    return $(upk, usk)$

OKG(par)
    $(\cdot, par_{\mathrm{E}}, \cdot, \cdot, \cdot) \leftarrow par$
    $(ek, dk) \leftarrow \mathrm{E.KG}(par_{\mathrm{E}})$
    $preopk \leftarrow ek$
    $preosk \leftarrow dk$
    return $(preopk, preosk)$

Sign(upk, usk, opk, ({(cid_i, a_i, crd_i, ipk_i)}_{i∈[n]}), m, f_ev)
    $y_{\mathrm{ev}} \leftarrow f_{\mathrm{ev}}(upk, \{(cid_i, \boldsymbol{a}_i)\}_{i\in[n]}, m)$
    $(ek, f_{\mathrm{op}}) \leftarrow opk$
    $y_{\mathrm{op}} \leftarrow f_{\mathrm{op}}(upk, \{(cid_i, \boldsymbol{a}_i)\}_{i\in[n]}, m)$
    $c_{\mathrm{op}} \leftarrow \mathrm{E.Enc}(ek, y_{\mathrm{op}}; r)$
    $\pi_{\mathrm{ev}} \leftarrow \mathrm{NIZK.Prove}^{\mathcal{R}_{\mathrm{ev}}}(crs_{\mathrm{ev}},$
            $(m, f_{\mathrm{ev}}, y_{\mathrm{ev}}, f_{\mathrm{op}}, c_{\mathrm{op}}, \{ipk_i\}_{i\in[n]}, ek),$
            $(upk, usk, \{(cid_i, \boldsymbol{a}_i, crd_i)\}_{i\in[n]}, y_{\mathrm{op}}, r))$
    return $\Sigma = (\sigma = (\pi_{\mathrm{ev}}, c_{\mathrm{op}}), y_{\mathrm{ev}})$

Verify(opk, ipk = {ipk_i}_{i∈[n]}, Σ, m, f_ev)
    $(\pi_{\mathrm{ev}}, c_{\mathrm{op}}), y_{\mathrm{ev}}) \leftarrow \Sigma$
    $(ek, f_{\mathrm{op}}) \leftarrow opk$
    return $\mathrm{NIZK.Verify}^{\mathcal{R}_{\mathrm{ev}}}(crs_{\mathrm{ev}}, \pi_{\mathrm{ev}},$
            $(m, f_{\mathrm{ev}}, y_{\mathrm{ev}}, f_{\mathrm{op}}, c_{\mathrm{op}}, \{ipk_i\}_{i\in[n]}, ek)$
        )

Open(osk, ipk, Σ, m, f_ev)
    $(opk, preosk = dk) \leftarrow osk; (ek, \cdot) \leftarrow opk$
    if $\mathrm{Verify}(opk, \boldsymbol{ipk}, \Sigma, m, f_{\mathrm{ev}}) = 0$ : return $\bot$
    $((\pi_{\mathrm{ev}}, c_{\mathrm{op}}), y_{\mathrm{ev}}) \leftarrow \Sigma$
    $y_{\mathrm{op}} \leftarrow \mathrm{E.Dec}(dk, c_{\mathrm{op}})$
    $\pi_{\mathrm{op}} \leftarrow \mathrm{NIZK.Prove}^{\mathcal{R}_{\mathrm{op}}}(crs_{\mathrm{op}}, (ek, c, y_{\mathrm{op}}), (dk))$
    return $(y_{\mathrm{op}}, \pi_{\mathrm{op}})$

Judge(opk, y_op, π_op, Σ, m)
    if $\mathrm{Verify}(opk, \boldsymbol{ipk}, \Sigma, m, f_{\mathrm{ev}}) = 0$ : return 0
    $((\cdot, c_{\mathrm{op}}), \cdot) \leftarrow \Sigma; (ek, \cdot) \leftarrow opk$
    return $\mathrm{NIZK.Verify}^{\mathcal{R}_{\mathrm{op}}}(crs_{\mathrm{op}}, \pi_{\mathrm{op}}, (ek, c, y_{\mathrm{op}}))$
```

Fig. 3. Π_{UAS} algorithms 1/2: everything except issuing.

correctness. We observe that both parties contribute uniformly random nonces to this identifier. We give theorems and intuition, but defer formal definitions and proofs to the full version [7].

Theorem 1 (Issuance anonymity of Π_{UAS}). *If the* SBCM *scheme is blinding, the NIZK system is zero-knowledge and simulation-extractable, and the public-key encryption scheme is correct and IND-CPA secure, then Π_{UAS} satisfies issuance anonymity.*

Theorem 2 (Signature anonymity of Π_{UAS}). *If the NIZK system is zero-knowledge and simulation extractable, and the public-key encryption scheme is correct and IND-CPA secure, then Π_{UAS} satisfies signature anonymity.*

For the two anonymity properties, we perform game hops until we obtain a game independent of bit b. For this, we simulate the NIZK proofs and replace the encryption of y_{op} with an encryption of 0. This is justified by IND-CPA security. Notably simulation extraction is needed to simulate decryption. For issuing anonymity we additionally assign all challenge credentials to the same virtual user. This is justified by the blinding property of the SBCM scheme.

```
$\mathrm{Obt}(upk, usk, ipk, \{(cid_i, \boldsymbol{a}_i, crd_i)\}_{i \in [n]}, \boldsymbol{a})$                    $\mathrm{Iss}(isk, \{ipk_i\}_{i \in [n]}, \boldsymbol{a}, y_{\mathrm{is}})$
                                                            $\xleftarrow{cid^{\mathrm{I}}}$    $cid^{\mathrm{I}} \leftarrow\!\$\ \mathcal{AS}$
$cid^{\mathrm{U}} \leftarrow\!\$\ \mathcal{AS}$
$cid \leftarrow (cid^{\mathrm{I}}, cid^{\mathrm{U}})$
$y'_{\mathrm{is}} = f_{\mathrm{is}}(upk, \boldsymbol{a}, \{(cid_i, \boldsymbol{a}_i)\}_{i \in [n]})$
$r \leftarrow \mathit{\Gamma}_{\mathrm{SBCM}}$
$c \leftarrow \mathrm{SBCM.Blind}(ipk, usk, (cid, \boldsymbol{a}), r)$
$\pi \leftarrow \mathrm{NIZK.Prove}^{\mathcal{R}_{\mathrm{is}}}(crs_{\mathrm{is}},$
        $(f_{\mathrm{is}}, c, cid, \boldsymbol{a}, ipk, \{ipk_i\}_{i \in [n]}, y'_{\mathrm{is}}),$
        $(upk, usk, \{(cid_i, \boldsymbol{a}_i, crd_i)\}_{i \in [n]}, r))$    $\xrightarrow{(cid^{\mathrm{U}}, c, \pi)}$
                                                                   $cid \leftarrow (cid^{\mathrm{I}}, cid^{\mathrm{U}})$
                                                                   $b \leftarrow \mathrm{NIZK.Verify}(crs_{\mathrm{is}}, \pi,$
                                                                     $(f_{\mathrm{is}}, c, cid, \boldsymbol{a}, ipk, \{ipk_i\}_{i \in [n]}, y_{\mathrm{is}}))$
                                                                   if $b = 0$ : return $\perp$
                                                            $\xleftarrow{\beta}$    $\beta \leftarrow \mathrm{SBCM.Sign}(isk, c, (cid, \boldsymbol{a}))$
                                                                   $reg = ((\{ipk_i\}_{i \in [n]}, \boldsymbol{a}, y_{\mathrm{is}}),$
                                                                         $cid^{\mathrm{I}}, (cid^{\mathrm{U}}, c, \pi), \beta)$
                                                                   return $R = (reg, cid)$
$\sigma \leftarrow \mathrm{SBCM.Unblind}(ipk, \beta, c, r, usk, (cid, \boldsymbol{a}))$
if $\mathrm{SBCM.Verify}(ipk, \sigma, usk, (cid, \boldsymbol{a})) = 0$ :
  return $\perp$
return $(cid, \boldsymbol{a}, crd = \sigma, ipk)$
```

Fig. 4. $\mathit{\Pi}_{\mathsf{UAS}}$ algorithms 2/2: issuing protocol. $\mathcal{AS}$ is an assumed attribute space.

Theorem 3 (Issuance unforgeability of $\mathit{\Pi}_{\mathsf{UAS}}$). *If the* NIZK *scheme is extraction zero-knowledge and simulation extractable, and the* SBCM *scheme is correct, unforgeable, and has deterministically derived public keys, then* $\mathit{\Pi}_{\mathsf{UAS}}$ *satisfies issuance unforgeability.*

Theorem 4 (Signature unforgeability of $\mathit{\Pi}_{\mathsf{UAS}}$). *If the underlying NIZK scheme is complete, extraction zero-knowledge and simulation extractable, the public key encryption scheme is correct, and the* SBCM *scheme is correct, unforgeable, and has deterministically derived public keys, then* $\mathit{\Pi}_{\mathsf{UAS}}$ *satisfies signing unforgeability.*

Theorem 5 (Non-frameability of $\mathit{\Pi}_{\mathsf{UAS}}$). *If the NIZK system is extraction zero-knowledge and simulation extractable and the* SBCM *scheme is correct, blind, and unforgeable, then* $\mathit{\Pi}_{\mathsf{UAS}}$ *satisfies non-frameability.*

The three unforgeability and non-frameability proofs share the intuition: Using simulation, we ensure via a series of game hops that we reach a game where we can embed an SBCM unforgeability challenge. This requires that the SBCM secret key is only used in operations that can be simulated using SBCM challenge oracles. Then, if the adversary wins the UAS game, we leverage its output to break SBCM unforgeability. Here, the main complexity is in alternating extraction and simulation: as our initial games already extract, we rely on extraction zero-knowledge NIZKs. Moreover, we need to ensure that we only extract from non-simulated proofs (and never attempt extraction of simulated ones).

6 Building Related Schemes from UAS

$\mathit{\Pi}_{\mathsf{UAS}}$ *restrictions.* Given a generic UAS construction, $\mathit{\Pi}_{\mathsf{UAS}}$, we can restrict the achieved privacy-vs-utility tradeoff by requiring it to use concrete f^{a}_{is}, f^{b}_{ev} and f^{c}_{op} functions. We refer to the result as the $(f^{a}_{\text{is}}, f^{b}_{\text{ev}}, f^{c}_{\text{op}})$-$\mathit{\Pi}_{\mathsf{UAS}}$ *restriction.* Note that security of $\mathit{\Pi}_{\mathsf{UAS}}$ implies security of its restrictions.

To showcase the generality of UAS, we briefly describe concrete $\mathit{\Pi}_{\mathsf{UAS}}$ restrictions that instantiate vanilla digital signatures, group signatures, anonymous credentials, and ring signatures, using the functions defined in Fig. 5. Fig. 6 graphically depicts these connections. The instantiations based on our $\mathit{\Pi}_{\mathsf{UAS}}$ generic construction are probably not the most efficient approach to build the corresponding related scheme. Still, we see it as an initial feasibility result, from which to build more efficient instantiations – perhaps relying on alternative UAS constructions for more restricted but still expressive enough function classes. We defer security models and proofs to the full version [7].

ISSUANCE FUNCTIONS

$f^{\mathsf{upk}}_{\text{is}}(upk, \cdot, \cdot) \coloneqq$ **return** upk

OPEN FUNCTIONS

$f^{\mathsf{0}}_{\text{op}}(\cdot, \cdot, \cdot) \coloneqq$ **return** 0

$f^{\mathsf{upk}}_{\text{op}}(upk, \cdot, \cdot) \coloneqq$ **return** upk

SIGNATURE EVALUATION FUNCTIONS

$f^{\mathsf{upk}}_{\text{ev}}(upk, \cdot, \cdot) \coloneqq$ **return** upk

$f^{\mathsf{0}}_{\text{ev}}(\cdot, \cdot, \cdot) \coloneqq$ **return** 0

$f^{\boldsymbol{d}}_{\text{ev}}(\cdot, (\cdot, \boldsymbol{a}), \cdot) \coloneqq$ **return** $(attr_i)_{i \in \boldsymbol{d}}$

```
f_ev^ring(upk, ·, ·) := if upk ∈ ring : return 1
        else return 0
```

Fig. 5. Functions for the $\mathit{\Pi}_{\mathsf{UAS}}$ restrictions described next. "·" denotes ignored arguments. f^{a} is a function named "a"; f^{a} is a function parameterized with a.

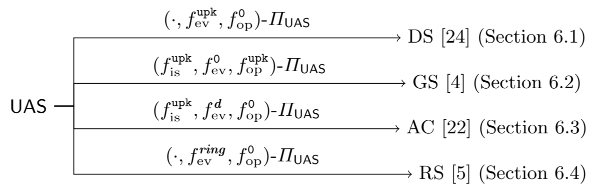
$(\cdot, f^{\mathsf{upk}}_{\text{ev}}, f^{\mathsf{0}}_{\text{op}})$-$\mathit{\Pi}_{\mathsf{UAS}}$
DS [24] (Section 6.1)
$(f^{\mathsf{upk}}_{\text{is}}, f^{\mathsf{0}}_{\text{ev}}, f^{\mathsf{upk}}_{\text{op}})$-$\mathit{\Pi}_{\mathsf{UAS}}$
GS [4] (Section 6.2)
UAS
$(f^{\mathsf{upk}}_{\text{is}}, f^{\boldsymbol{d}}_{\text{ev}}, f^{\mathsf{0}}_{\text{op}})$-$\mathit{\Pi}_{\mathsf{UAS}}$
AC [22] (Section 6.3)
$(\cdot, f^{\boldsymbol{ring}}_{\text{ev}}, f^{\mathsf{0}}_{\text{op}})$-$\mathit{\Pi}_{\mathsf{UAS}}$
RS [5] (Section 6.4)

Fig. 6. $\mathit{\Pi}_{\mathsf{UAS}}$-restrictions instantiating related schemes.

6.1 Digital Signatures

As a warm up, we show that a $(\cdot, f^{\mathsf{upk}}_{\text{ev}}, f^{\mathsf{0}}_{\text{op}})$-$\mathit{\Pi}_{\mathsf{UAS}}$ restriction realizes a conventional digital signatures satisfying EUF-CMA security [24]. No issuance function is required, as users do not need credentials to sign. The evaluation function outputs the signer's public key, and any opening function works – the output is ignored. A possible optimization would be using the empty string as ciphertext for constant functions (e.g. $f^{\mathsf{0}}_{\mathrm{op}}$) and skip verifiable encryption. Concretely, we create $\mathit{\Pi}^{\mathrm{ds}}_{\mathsf{UAS}}$ from $\mathit{\Pi}_{\mathsf{UAS}}$, where public parameters *par* are passed implicitly:

```
$\underline{\mathrm{Setup}(1^\kappa)}$ $par' \leftarrow \mathit{\Pi}_{\mathsf{UAS}}.\mathrm{Setup}(1^\kappa)$; $(preopk, preosk) \leftarrow \mathit{\Pi}_{\mathsf{UAS}}.\mathrm{OKG}(par)$;
    $opk \leftarrow (preopk, f^{\mathsf{0}}_{\mathrm{op}})$; $osk \leftarrow (opk, preosk)$; $\mathbf{return}$ $par = (par', opk)$.
$\underline{\mathrm{KG}(1^\kappa)}$ $\mathbf{return}$ $(upk, usk) \leftarrow \mathit{\Pi}_{\mathsf{UAS}}.\mathrm{KG}(par')$.
$\underline{\mathrm{Sign}(usk, m)}$ $(\sigma, y_{\mathrm{ev}}) \leftarrow \mathit{\Pi}_{\mathsf{UAS}}.\mathrm{Sign}(upk, usk, opk, \emptyset, m, f^{\mathsf{upk}}_{\mathrm{ev}})$;
    $\mathbf{return}$ $\sigma$. //$y_{\mathrm{ev}} = upk$
$\underline{\mathrm{Verify}(upk, m, \sigma)}$ $\mathbf{return}$ $\mathit{\Pi}_{\mathsf{UAS}}.\mathrm{Verify}(opk, \emptyset, \Sigma = (\sigma, upk), m, f^{\mathsf{upk}}_{\mathrm{ev}})$.
 //Since $f^{\mathrm{upk}}_{\mathrm{ev}}$ outputs the signer's $upk$, we know that $\sigma$ is bound to the owner of $upk$.
```

6.2 Group Signatures

We show that a $(f^{\mathsf{upk}}_{\mathrm{is}}, f^{\mathsf{0}}_{\mathrm{ev}}, f^{\mathsf{upk}}_{\mathrm{op}})$-$\mathit{\Pi}_{\mathsf{UAS}}$ restriction is a secure group signature scheme in a definition in the spirit of [4]. First, the $f^{\mathsf{upk}}_{\mathrm{is}}$ function outputs the *upk* of the requesting user, allowing the issuer to detect a user requesting multiple credentials – note that, in vanilla group signatures, there is a single issuer and a single membership certificate per user. Thus, linkable issuance is the expected behavior. The evaluation function $f^{\mathsf{0}}_{\mathrm{ev}}$ outputs a constant value. Finally, the opening function $f^{\mathsf{upk}}_{\mathrm{op}}$ outputs the signer's *upk*, allowing the opener to identify the signer of any group signature. Concretely, we create $\mathit{\Pi}^{\mathrm{gs}}_{\mathsf{UAS}}$ from $\mathit{\Pi}_{\mathsf{UAS}}$ as follows:

```
$\underline{\mathrm{KG}(1^\kappa)}$ $par \leftarrow \mathit{\Pi}_{\mathsf{UAS}}.\mathrm{Setup}(1^\kappa)$//implicit; $(preopk, preosk) \leftarrow \mathit{\Pi}_{\mathsf{UAS}}.\mathrm{OKG}(par)$;
    $opk \leftarrow (preopk, f^{\mathsf{upk}}_{\mathrm{op}})$; $osk \leftarrow (opk, preosk)$; $(ipk', isk') \leftarrow \mathit{\Pi}_{\mathsf{UAS}}.\mathrm{KG}(par)$;
    $ipk \leftarrow (ipk', f^{\mathsf{upk}}_{\mathrm{is}})$; $isk \leftarrow (ipk, isk')$; $\mathbf{return}$ $(gpk = (ipk, opk), isk, osk)$.
$\underline{\mathrm{UKG}(1^\kappa)}$ $\mathbf{return}$ $(upk, usk) \leftarrow \mathit{\Pi}_{\mathsf{UAS}}.\mathrm{KG}(par)$.
$\underline{\langle \mathrm{Obt}(usk, ipk), \mathrm{Iss}((isk, opk), upk) \rangle}$
    $\langle C, R \rangle \leftarrow \mathit{\Pi}_{\mathsf{UAS}}.\langle \mathrm{Obt}(upk, usk, ipk, \emptyset, \emptyset), \mathrm{Iss}(isk, \emptyset, \emptyset, y_{\mathrm{is}}{=}upk) \rangle$.
    $\mathbf{return}$ $\langle (usk, C), R \rangle$ //The credential is locally augmented with the user's secret
    key.
$\underline{\mathrm{Sign}(gpk, (usk, C), m)}$ $(ipk, opk) \leftarrow gpk$; $(\sigma, y_{\mathrm{ev}}) \leftarrow \mathit{\Pi}_{\mathsf{UAS}}.\mathrm{Sign}(upk, usk, opk, C,$
    $m, f^{\mathsf{0}}_{\mathrm{ev}})$; $\mathbf{return}$ $\sigma$ //$y_{\mathrm{ev}} = 0$ is the constant output of $f^{0}_{\mathrm{ev}}$.
$\underline{\mathrm{Verify}(gpk, \sigma, m)}$ $(ipk, opk) \leftarrow gpk$; $\mathbf{return}$ $\mathit{\Pi}_{\mathsf{UAS}}.\mathrm{Verify}(opk, ipk, (\sigma, 0), m, f^{\mathsf{0}}_{\mathrm{ev}})$.
$\underline{\mathrm{Open}(gpk{=}(ipk, opk), osk, \sigma, m)}$ $\mathbf{return}$ $\mathit{\Pi}_{\mathsf{UAS}}.\mathrm{Open}(osk, ipk, \sigma, 0, m, f^{\mathsf{0}}_{\mathrm{ev}})$.
$\underline{\mathrm{Judge}(gpk{=}(ipk, opk), \pi, upk, \sigma, m)}$ $\mathbf{return}$ $\mathit{\Pi}_{\mathsf{UAS}}.\mathrm{Judge}(opk, upk, \pi, (\sigma, 0), m)$.
```

6.3 Anonymous Credentials

We show how to build AC systems from UAS signatures. For concreteness, we use a $(f^{\mathsf{upk}}_{\mathrm{is}}, f^{\boldsymbol{d}}_{\mathrm{ev}}, f^{\mathsf{0}}_{\mathrm{op}})$-$\mathit{\Pi}_{\mathsf{UAS}}$ restriction which suffices for the AC scheme of Fuchsbauer et al. [22] which does not have issuance anonymity and supports selective disclosure of attributes. Thus, our issuance function returns the user's public key as for GS and the evaluation function reveals the chosen subset of attributes.

To match the syntax of the target scheme, we implement a simple challenge response protocol via UAS signing. The evaluation function $f_{\text{ev}}^{\boldsymbol{d}}$ returns subset $D = (attr_i)_{i \in \boldsymbol{d}}$ of the attributes. The open function f_{op}^{0} reveals nothing.

```
$\underline{\text{IssKeyGen}(1^\kappa)}$ $par' \leftarrow \Pi_{\mathsf{UAS}}.\text{Setup}(1^\kappa)$; $(preopk, preosk) \leftarrow \Pi_{\mathsf{UAS}}.\text{OKG}(par')$;
    $opk \leftarrow (preopk, f_{\text{op}}^{0})$; $osk \leftarrow (opk, preosk)$; $(ipk', isk') \leftarrow \Pi_{\mathsf{UAS}}.\text{KG}(par')$;
    $par \leftarrow (par', opk)$ //kept implicit ; $\textbf{return}\ (ipk{=}(ipk', f_{\text{is}}^{\text{upk}})), isk{=}(ipk, isk'))$.
$\underline{\text{UserKeyGen}(1^\kappa)}$ $\textbf{return}\ (upk, usk) \leftarrow \Pi_{\mathsf{UAS}}.\text{KG}(par')$.
$\underline{\langle \text{Obt}(usk, ipk, \boldsymbol{a}), \text{Iss}(isk, upk, \boldsymbol{a})\rangle}$
    $\langle C_{\mathsf{UAS}}, R\rangle \leftarrow \Pi_{\mathsf{UAS}}.\langle \text{Obt}(upk, usk, ipk, \boldsymbol{C}{=}\emptyset, \boldsymbol{a}), \text{Iss}(isk, \boldsymbol{ipk}{=}\emptyset, \boldsymbol{a}, y_{\text{is}}{=}upk)\rangle$.
    $\textbf{return}\ \langle C{=}(usk, C_{\mathsf{UAS}}), \textbf{if}\ R \neq \bot : \top\rangle$//Local translation of protocol outputs.
$\underline{\text{Show}(ipk, \boldsymbol{a}, \boldsymbol{d}, C{=}(usk, C_{\mathsf{UAS}})), \text{Verify}(ipk, \boldsymbol{d}, D)\rangle}$
    V: $\textbf{send}\ r \leftarrow \{0,1\}^\kappa\ \textbf{to}\ \text{S}$
    S: $\textbf{send}\ (\sigma, y_{\text{ev}}) = \Sigma \leftarrow \Pi_{\mathsf{UAS}}.\text{Sign}(upk, usk, opk, C_{\mathsf{UAS}}, r, f_{\text{ev}}^{\boldsymbol{d}})\ \textbf{to}\ \text{V}$
    V: $\textbf{return}\ y_{\text{ev}} = D \wedge \Pi_{\mathsf{UAS}}.\text{Verify}(opk, ipk, \Sigma, r, f_{\text{ev}}^{\boldsymbol{d}})$
```

6.4 Ring Signatures

We use a $(\cdot, f_{\text{ev}}^{\boldsymbol{ring}}, f_{\text{op}}^{0})$-$\Pi_{\mathsf{UAS}}$ restriction to build a ring signature scheme, $\Pi_{\mathsf{UAS}}^{\text{ring}}$. For signing, signers compute a $f_{\text{ev}}^{\boldsymbol{ring}}$ function, where $\boldsymbol{ring} = \{upk_i\}_{i \in [n]}$ is an arbitrary set of public keys. This function returns 1 if the signer's $upk \in \boldsymbol{ring}$, and 0 otherwise. A ring signature is a Π_{UAS} signature evaluated on such a $f_{\text{ev}}^{\boldsymbol{ring}}$ function – which does not require any credential. The construction is as follows:

```
$\underline{\text{Setup}(1^\kappa)}$ $par' \leftarrow \Pi_{\mathsf{UAS}}.\text{Setup}(1^\kappa)$; $(preopk, preosk) \leftarrow \Pi_{\mathsf{UAS}}.\text{OKG}(par)$;
    $opk \leftarrow (preopk, f_{\text{op}}^{0})$; $osk \leftarrow (opk, preosk)$; $\textbf{return}\ par = (par', opk)$.
    //We assume that the setup is trusted to compute the correct $opk$.
$\underline{\text{KG}(1^\kappa)}$ $\textbf{return}\ (pk, sk) \leftarrow \Pi_{\mathsf{UAS}}.\text{KG}(par')$.
$\underline{\text{Sign}(usk, \boldsymbol{ring}, m)}$ $(\sigma, y_{\text{ev}}) \leftarrow \Pi_{\mathsf{UAS}}.\text{Sign}(upk, usk, opk, \emptyset, m, f_{\text{ev}}^{\boldsymbol{ring}})$; $\textbf{return}\ \sigma$
    //$y_{\text{ev}} = (upk \in \boldsymbol{ring})$ .
$\underline{\text{Verify}(\boldsymbol{ring}, m, \sigma)}$ $\textbf{return}\ \Pi_{\mathsf{UAS}}.\text{Verify}(opk, \emptyset, \Sigma = (\sigma, y_{\text{ev}} = 1), m, f_{\text{ev}}^{\boldsymbol{ring}})$.
    //Since $f_{\text{ev}}^{\boldsymbol{ring}}$ outputs 1 we know that the signer is in the ring.
```

Note that the issuance function is never used, as no credential takes part in the signing process, so we simply ignore it. The same does not apply to the open function, though. Even if no actual Open function is exposed by the ring signature construction, a malicious party could try to open a signature. Thus, we need to fix it to a function that does not leak information, like f_{op}^{0}.

7 Conclusion and Future Work

We present a general model and construction for anonymous signatures, allowing for different privacy-vs-utility trade-offs. The flexibility of our model stems from functional placeholders that modulate the utility information learned by issuers, verifiers, and openers at credential issuance and authentication time, as well as after authentication. To showcase its generality, we show how to securely instantiate well-known schemes using our construction.

A further natural generalization of our model would allow for issuers and openers to adjust their functions dynamically, or to allow for multiple openers for the same signature. A practical concern are optimized implementations of Π_{UAS} restrictions and optimized constructions for restricted function classes. For example, the instantiation in the full version [7], based on BBS+, ElGamal, and basic sigma proofs, is well suited (and efficient) for cases that need selective disclosure. However, it falls short (or would be inefficient) for others.

References

1. Au, M.H., Susilo, W., Mu, Y.: Constant-size dynamic k-taa. In: Security and Cryptography for Networks, 5th International Conference, SCN 2006, Maiori, Italy, September 6-8, 2006, Proceedings. pp. 111–125 (2006)
2. Belenkiy, M., Camenisch, J., Chase, M., Kohlweiss, M., Lysyanskaya, A., Shacham, H.: Randomizable proofs and delegatable anonymous credentials. In: Halevi, S. (ed.) Advances in Cryptology - CRYPTO 2009, 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings. Lecture Notes in Computer Science, vol. 5677, pp. 108–125. Springer (2009). https://doi.org/10.1007/978-3-642-03356-8_7, https://doi.org/10.1007/978-3-642-03356-8_7
3. Bellare, M., Micciancio, D., Warinschi, B.: Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In: EUROCRYPT 2003, Proceedings. pp. 614–629 (2003)
4. Bellare, M., Shi, H., Zhang, C.: Foundations of group signatures: The case of dynamic groups. In: CT-RSA 2005, Proceedings. pp. 136–153 (2005)
5. Bender, A., Katz, J., Morselli, R.: Ring signatures: Stronger definitions, and constructions without random oracles. In: Halevi, S., Rabin, T. (eds.) Theory of Cryptography, Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006, Proceedings. Lecture Notes in Computer Science, vol. 3876, pp. 60–79. Springer (2006). https://doi.org/10.1007/11681878_4, https://doi.org/10.1007/11681878_4
6. Bichsel, P., Camenisch, J., Neven, G., Smart, N.P., Warinschi, B.: Get shorty via group signatures without encryption. In: Security and Cryptography for Networks, 7th International Conference, SCN 2010, Amalfi, Italy, September 13-15, 2010. Proceedings. pp. 381–398 (2010)
7. Bobolz, J., Diaz, J., Kohlweiss, M.: Foundations of anonymous signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. Cryptology ePrint Archive, Paper 2024/042 (2024), https://eprint.iacr.org/2024/042, https://eprint.iacr.org/2024/042
8. Bootle, J., Cerulli, A., Chaidos, P., Ghadafi, E., Groth, J.: Foundations of fully dynamic group signatures. In: Manulis, M., Sadeghi, A., Schneider, S.A. (eds.) Applied Cryptography and Network Security - 14th International Conference, ACNS 2016, Guildford, UK, June 19-22, 2016. Proceedings. Lecture Notes in Computer Science, vol. 9696, pp. 117–136. Springer (2016). https://doi.org/10.1007/978-3-319-39555-5_7, https://doi.org/10.1007/978-3-319-39555-5_7

9. Bootle, J., Cerulli, A., Chaidos, P., Ghadafi, E., Groth, J., Petit, C.: Short accountable ring signatures based on DDH. In: Pernul, G., Ryan, P.Y.A., Weippl, E.R. (eds.) Computer Security - ESORICS 2015 - 20th European Symposium on Research in Computer Security, Vienna, Austria, September 21-25, 2015, Proceedings, Part I. Lecture Notes in Computer Science, vol. 9326, pp. 243–265. Springer (2015). https://doi.org/10.1007/978-3-319-24174-6_13, https://doi.org/10.1007/978-3-319-24174-6_13
10. Camenisch, J., Drijvers, M., Lehmann, A.: Anonymous attestation using the strong diffie hellman assumption revisited. In: TRUST 2016, Proceedings. pp. 1–20 (2016)
11. Camenisch, J., Dubovitskaya, M., Lehmann, A., Neven, G., Paquin, C., Preiss, F.: Concepts and languages for privacy-preserving attribute-based authentication. In: Fischer-Hübner, S., de Leeuw, E., Mitchell, C.J. (eds.) Policies and Research in Identity Management - Third IFIP WG 11.6 Working Conference, IDMAN 2013, London, UK, April 8-9, 2013. Proceedings. IFIP Advances in Information and Communication Technology, vol. 396, pp. 34–52. Springer (2013). https://doi.org/10.1007/978-3-642-37282-7_4, https://doi.org/10.1007/978-3-642-37282-7_4
12. Camenisch, J., Kohlweiss, M., Soriente, C.: Solving revocation with efficient update of anonymous credentials. In: Garay, J.A., Prisco, R.D. (eds.) Security and Cryptography for Networks, 7th International Conference, SCN 2010, Amalfi, Italy, September 13-15, 2010. Proceedings. Lecture Notes in Computer Science, vol. 6280, pp. 454–471. Springer (2010). https://doi.org/10.1007/978-3-642-15317-4_28, https://doi.org/10.1007/978-3-642-15317-4_28
13. Camenisch, J., Krenn, S., Lehmann, A., Mikkelsen, G.L., Neven, G., Pedersen, M.Ø.: Formal treatment of privacy-enhancing credential systems. In: SAC 2015, Revised Selected Papers. pp. 3–24 (2015)
14. Camenisch, J., Lysyanskaya, A.: An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In: EUROCRYPT 2001, Proceeding. pp. 93–118 (2001)
15. Camenisch, J., Lysyanskaya, A.: A signature scheme with efficient protocols. In: Cimato, S., Galdi, C., Persiano, G. (eds.) Security in Communication Networks, Third International Conference, SCN 2002, Amalfi, Italy, September 11-13, 2002. Revised Papers. Lecture Notes in Computer Science, vol. 2576, pp. 268–289. Springer (2002). https://doi.org/10.1007/3-540-36413-7_20, https://doi.org/10.1007/3-540-36413-7_20
16. Chase, M., Kohlweiss, M., Lysyanskaya, A., Meiklejohn, S.: Malleable signatures: New definitions and delegatable anonymous credentials. In: IEEE 27th Computer Security Foundations Symposium, CSF 2014, Vienna, Austria, 19-22 July, 2014. pp. 199–213. IEEE Computer Society (2014). https://doi.org/10.1109/CSF.2014.22, https://doi.org/10.1109/CSF.2014.22
17. Chaum, D.: Security without identification: Transaction systems to make big brother obsolete. Commun. ACM **28**(10), 1030–1044 (1985). https://doi.org/10.1145/4372.4373, https://doi.org/10.1145/4372.4373
18. Chaum, D., van Heyst, E.: Group signatures. In: Davies, D.W. (ed.) EUROCRYPT '91, Proceedings. pp. 257–265. Springer (1991)
19. Deuber, D., Maffei, M., Malavolta, G., Rabkin, M., Schröder, D., Simkin, M.: Functional credentials. Proc. Priv. Enhancing Technol. **2018**(2), 64–84 (2018). https://doi.org/10.1515/popets-2018-0013, https://doi.org/10.1515/popets-2018-0013

20. Emura, K., Hanaoka, G., Kawai, Y., Matsuda, T., Ohara, K., Omote, K., Sakai, Y.: Group signatures with message-dependent opening: Formal definitions and constructions. Secur. Commun. Networks **2019**, 4872403:1–4872403:36 (2019). https://doi.org/10.1155/2019/4872403, `https://doi.org/10.1155/2019/4872403`
21. Espresso Systems: Configurable Asset Privacy. `https://github.com/EspressoSystems/cap/blob/main/cap-specification.pdf` (2022)
22. Fuchsbauer, G., Hanser, C., Slamanig, D.: Structure-preserving signatures on equivalence classes and constant-size anonymous credentials. J. Cryptol. **32**(2), 498–546 (2019). https://doi.org/10.1007/s00145-018-9281-4, `https://doi.org/10.1007/s00145-018-9281-4`
23. Garms, L., Lehmann, A.: Group signatures with selective linkability. In: PKC 2019, Proceedings. pp. 190–220 (2019)
24. Goldwasser, S., Micali, S., Rivest, R.L.: A digital signature scheme secure against adaptive chosen-message attacks. SIAM J. Comput. **17**(2), 281–308 (1988). https://doi.org/10.1137/0217017, `https://doi.org/10.1137/0217017`
25. Groth, J., Ostrovsky, R.: Cryptography in the multi-string model. J. Cryptol. **27**(3), 506–543 (2014). https://doi.org/10.1007/s00145-013-9152-y, `https://doi.org/10.1007/s00145-013-9152-y`
26. Kiayias, A., Kohlweiss, M., Sarencheh, A.: Peredi: Privacy-enhanced, regulated and distributed central bank digital currencies. In: Yin, H., Stavrou, A., Cremers, C., Shi, E. (eds.) Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022, Los Angeles, CA, USA, November 7-11, 2022. pp. 1739–1752. ACM (2022). https://doi.org/10.1145/3548606.3560707, `https://doi.org/10.1145/3548606.3560707`
27. Kiayias, A., Yung, M.: Secure scalable group signature with dynamic joins and separable authorities. Int. J. Secur. Networks **1**(1/2), 24–45 (2006). https://doi.org/10.1504/IJSN.2006.010821, `https://doi.org/10.1504/IJSN.2006.010821`
28. Kohlweiss, M., Lysyanskaya, A., Nguyen, A.: Privacy-preserving blueprints. In: Hazay, C., Stam, M. (eds.) Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part II. Lecture Notes in Computer Science, vol. 14005, pp. 594–625. Springer (2023). https://doi.org/10.1007/978-3-031-30617-4_20, `https://doi.org/10.1007/978-3-031-30617-4_20`
29. Libert, B., Nguyen, K., Peters, T., Yung, M.: Bifurcated signatures: Folding the accountability vs. anonymity dilemma into a single private signing scheme. In: Canteaut, A., Standaert, F. (eds.) Advances in Cryptology - EUROCRYPT 2021 - 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17-21, 2021, Proceedings, Part III. Lecture Notes in Computer Science, vol. 12698, pp. 521–552. Springer (2021). https://doi.org/10.1007/978-3-030-77883-5_18, `https://doi.org/10.1007/978-3-030-77883-5_18`
30. Manulis, M., Sadeghi, A., Schwenk, J.: Linkable democratic group signatures. In: Information Security Practice and Experience, Second International Conference, ISPEC 2006, Hangzhou, China, April 11-14, 2006, Proceedings. pp. 187–201 (2006)
31. Nguyen, K., Guo, F., Susilo, W., Yang, G.: Multimodal private signatures. Cryptology ePrint Archive, Paper 2022/1008 (2022), `https://eprint.iacr.org/2022/1008`, `https://eprint.iacr.org/2022/1008`

32. Sakai, Y., Emura, K., Hanaoka, G., Kawai, Y., Matsuda, T., Omote, K.: Group signatures with message-dependent opening. In: Abdalla, M., Lange, T. (eds.) Pairing-Based Cryptography - Pairing 2012 - 5th International Conference, Cologne, Germany, May 16-18, 2012, Revised Selected Papers. Lecture Notes in Computer Science, vol. 7708, pp. 270–294. Springer (2012). https://doi.org/10.1007/978-3-642-36334-4_18, https://doi.org/10.1007/978-3-642-36334-4_18
33. Santis, A.D., Crescenzo, G.D., Ostrovsky, R., Persiano, G., Sahai, A.: Robust non-interactive zero knowledge. In: Kilian, J. (ed.) Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings. Lecture Notes in Computer Science, vol. 2139, pp. 566–598. Springer (2001). https://doi.org/10.1007/3-540-44647-8_33, https://doi.org/10.1007/3-540-44647-8_33
34. Tessaro, S., Zhu, C.: Revisiting BBS signatures. In: Hazay, C., Stam, M. (eds.) Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part V. Lecture Notes in Computer Science, vol. 14008, pp. 691–721. Springer (2023). https://doi.org/10.1007/978-3-031-30589-4_24, https://doi.org/10.1007/978-3-031-30589-4_24
35. WG, A.: Hyperledger anoncreds. https://www.hyperledger.org/use/anoncreds (May 2023)