Efficient Signatures of Knowledge and DAA in the Standard Model

David Bernhard, Georg Fuchsbauer and Essam Ghadafi

University of Bristol, UK.
{bernhard,georg,ghadafi}@cs.bris.ac.uk

Abstract. Direct Anonymous Attestation (DAA) is one of the most complex cryptographic protocols deployed in practice. It allows an embedded secure processor known as a Trusted Platform Module (TPM) to attest to the configuration of its host computer without violating the owner's privacy. DAA has been standardized by the Trusted Computing Group.
The security of the DAA standard and all existing schemes is analyzed in the random oracle model. We provide the first constructions of DAA in the standard model, that is, without relying on random oracles.
As a building block for our schemes, we construct the first efficient standard-model signatures of knowledge, which have many applications beyond DAA.

Keywords. DAA, group signatures, signatures of knowledge, standard model.

1 Introduction

Direct Anonymous Attestation (DAA) is a protocol for a secure embedded processor known as a Trusted Platform Module (TPM) to authenticate itself and sign messages attesting to the state of its host while preserving the privacy of its owner. The first DAA protocol by Brickell, Camenisch and Chen [13] was standardized in 2004 by the Trusted Computing Group (TCG) [45] as the TPM 1.2 standard; since then millions of TPMs have been shipped with personal computers. In DAA a party owning a TPM can join a group and then sign messages as a member of this group. A DAA signature is created on a pair consisting of the message and a *basename*, which can be thought of as an identity of the verifier for which the message is intended. Messages signed by the same user under the same basename are linked: a verifier can trace which signatures using his basename were signed by the same user. Apart from this however, signatures are anonymous: two verifiers comparing signatures under their respective basenames cannot tell if a signature under one basename was produced by the same user as another signature under a different basename. Many subsequent papers [14–17, 23–28] have advanced the state of the art in two areas. First, since the discovery of cryptographic applications of bilinear pairings over elliptic curves, authors have proposed more efficient pairing-based DAA protocols, first in the symmetric then in the asymmetric setting. Secondly, the security model for DAA has evolved and improved over the years. The first schemes used a simulation-based model. Starting with Brickell, Chen and Li in 2009 [14], recent papers have switched to game-based models to analyze security of DAA protocols. In this paper, we use the most recent security model by Bernhard et al. [6].

All previous DAA schemes rely on the random-oracle heuristic [30, 4] to reason about their security. However, a series of papers starting with Canetti, Goldreich and Halevi [19] shows that it is impossible to implement a random oracle securely. Since a security "proof" requiring random oracles rests on an assumption known to be unachievable, it is nowadays common practice to strive for provably secure constructions in the *standard model*, where one does not rely on a random oracle. There, numerous computational assumptions have been proposed and their relative merits are hard to assess. Efficient standard-model protocols often use complex assumptions such as "strong" or interactive assumptions. We construct two standard-model DAA protocols in this paper. The first relies on non-interactive assumptions only. Our second protocol, while still in the standard model, uses some components whose security is based on interactive assumptions from the literature to achieve greater efficiency.

A Blueprint for DAA. Our DAA scheme is in the model of Bernhard et al. [6], who argued that all existing DAA schemes follow the same "blueprint" and are built from the same three basic building blocks. We will seek to follow this blueprint in our DAA constructions and explain where it is necessary to make changes.

The first building block for DAA is a Randomizable Weakly Blind Signature (RwBS). Blind signatures allow a user to submit messages to a signer and receive signatures on them such that the signer is unable to tell which messages he signed. The blindness property also ensures that signatures cannot be linked to their signing sessions. This is captured via a game where the adversary freely chooses a pair of messages and then asks the honest user to request signatures on those two messages in an order unknown to him. If both signing sessions complete successfully, the adversary gets the two final signatures and wins if he successfully tells with a probability non-negligibly greater than 1/2 the order in which the messages were signed. RwBS are weaker in the sense that blindness only holds for signers that never see the messages to be signed in the first place: the user may pick a message at random, run the blind-signing protocol and the signer should not be able to discover the message.

The second building block for DAA is a Linkable Indistinguishable Tag (LIT). This tag is similar to a message authentication code but is deterministic and requires different security properties. LITs are linkable in the sense that it is easy to tell if two tags were made with the same signing key on the same data (in fact these two tags will be identical) but given two tags on different data, it is hard to tell whether or not they were produced with the same signing key.

The final building block is a Signature of Knowledge (SoK). A SoK enables signing of messages like a digital signature, but does so w.r.t. an arbitrary NP statement, whose witness serves as the signing key.

In DAA users can join groups and then produce signatures that can be verified as coming from a group member, but they cannot be traced to the signer. To join a group, a user chooses a key and obtains a RwBS as a credential on his key. To sign a message under a basename, the user first randomizes his credential and produces a LIT under his key on the basename. He then makes a SoK on the message proving knowledge of a key on which the randomized credential and the LIT verify.

The security model in [6] considers *pre-DAA* schemes, which are fully functional DAA schemes in which the TPM handles the whole user-side computation without delegating non-security-critical operations to its host computer. This delegation and its security analysis can then be handled separately from that of the scheme itself. In effect, the computations and data that a TPM can safely delegate to its host in the existing schemes are exactly the ones that the adversary controls in the pre-DAA security model. We will construct and analyze a standard-model pre-DAA scheme; delegation to the host can then be realized securely with any of the methods described in the security model from [6] (as delegation operations are independent of a random oracle).

LIT in the Standard Model. A LIT is a deterministic tag under a key on a basename, which should look random, so that tags under different keys are indistinguishable. This property is trivially achievable by using a random oracle. Like Verifiable Random Functions (VRF) [43], LITs are a lot harder to construct in the standard model, in particular for large input spaces. LITs are a somewhat stronger primitive than VRFs, and we do not know of any large-domain VRF that would yield an LIT. We discuss LITs and their relation to VRFs in greater detail in Sect. 4.3.

For DAA, we believe that it is reasonable to postulate that the number of possible basenames is polynomial in the security parameter. While the set of *messages* which users can sign must be large, the number of possible verifiers (corresponding to basenames) will be efficiently enumerable. For polynomial-size domains, we construct a standard-model LIT using the VRF by Dodis and Yampolskiy [29]. In addition, we show how a standard-model LIT for a large basename space could be constructed at the cost of using an interactive assumption. This LIT can then be used in our generic construction, leading to a DAA scheme with large basename spaces, which satisfies thus the full model of [6].

Our Contribution. We present the first constructions of DAA in the standard model. Our paper contains two such constructions: the first in Sect. 5 avoids not only random oracles but also interactive assumptions; the second in Sect. 6 is still in the standard-model and its efficiency exceeds that of the best standard-model instantiations of comparable primitives, such as group signatures.

We start by constructing the first efficient standard-model signatures of knowledge. While SoKs exist for all NP statements [20], these generic constructions are highly inefficient. Our realization is based on Groth-Sahai (GS) proofs [36], which are the only known efficient standard-model Non-Interactive Zero-Knowledge (NIZK) proof systems.

GS proofs can be set up to offer either of two properties known as *simulation* and *extraction* but not both at once, yet this is precisely what is required in the security model for SoK. This is in contrast to the ROM where proof systems typically offer both of these properties simultaneously. Making a SoK out of GS proofs is thus a non-trivial matter that we deal with in detail in Sect. 3. We revert to a known technique, used e.g. in [35]: the signer proves that she knows a witness to the actual statement *or* that she knows a signature on this statement under a public key included in the public parameters. The corresponding secret key can then e used to simulate signatures without knowing a witness.

The obstacles in getting this approach to standard-model DAA to work are the following. First, all statements of the proof used in the SoK, as well as their disjunction, need to be encoded in the language of GS proofs. Secondly, GS proofs only allow the extraction of group elements rather than group exponents, as was common in ROM DAA schemes. Together with forgoing random oracles, this places restrictions on the type of LITs and RwBSs we can use and in particular excludes all previous constructions. Finally, all this needs to be done as efficiently as possible. We stress that we cannot hope to compete for efficiency with instantiations in the ROM, as there, pseudorandomness (for the LIT) and simulation-extractable NIZKs (used to construct the SoK) come essentially for free.

We construct two new standard model RwBSs. The first construction is based on Abe et al.'s signature scheme [1], which we use in our first DAA construction to certify group members' keys upon joining. We make use of its partial randomizability which allows us to include parts of the certificate in the clear in a DAA signature and prove knowledge of the remainder. In our second DAA construction we deploy a new RwBS scheme, which is compatible with GS proofs and satisfies the full model from [6]. We derive it from a recent signature scheme by Ghadafi [32]. By using a novel approach, we moreover show how to avoid expensive SoKs and rely directly on standard GS proofs instead.

As a last building block we derive a standard-model LIT, based on the VRF of Dodis and Yampolskiy [29]. Basing it on a non-interactive assumption, it requires a basename space that has polynomial size. However, if one is willing to assume an interactive assumption, the LIT would be secure even for exponential-sized basename spaces.

2 Preliminaries

Notation. A bilinear group is a tuple $\mathcal{P} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, P_1, P_2)$ where p is a prime, $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are groups of order p; P_1, P_2 are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively and e is a non-degenerate bilinear map $\mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ (i.e., $e(P_1, P_2)$ generates $\mathbb{G}_T$ and for any $Q_1 \in \mathbb{G}_1, Q_2 \in \mathbb{G}_2, x, y \in \mathbb{Z}_p : e([x]Q_1, [y]Q_2) = e(Q_1, Q_2)^{xy}$). We assume that the operations in all groups are efficiently computable and write $[x]P$ to denote the x-fold composition of a group element P with itself. We use *asymmetric* bilinear groups (which are more efficient), meaning that there are no known efficiently computable homomorphisms from $\mathbb{G}_1$ to $\mathbb{G}_2$ or vice versa. We let $\mathbb{G}^\times := \mathbb{G} \setminus \{0_\mathbb{G}\}$.

2.1 Hardness Assumptions

Our schemes rely on different assumptions from the literature, all of which are stated here.

Definition 1 (SXDH Assumption). *For $i = 1, 2$, we have: given $(P_i, [a]P_i, [b]P_i)$ for random $a, b \leftarrow \mathbb{Z}_p$, it is hard to distinguish $[ab]P_i$ from a random element of $\mathbb{G}_i$, i.e. the DDH assumption holds in both groups $\mathbb{G}_1$ and $\mathbb{G}_2$.*

Definition 2 (CDH$^+$ Assumption [21, 7]**).** *Given $(P_1, P_2, [a]P_1, [b]P_1, [a]P_2)$ it is hard to compute $[ab]P_1$. This is identical to CDH in symmetric bilinear groups.*

Definition 3 (q-SDH Assumption [10]**).** *Given $(P_1, [x]P_1 \ldots, [x^q]P_1, P_2, [x]P_2)$ for $x \leftarrow \mathbb{Z}_p^\times$, it is hard to output a pair $(c, [\frac{1}{x+c}]P_1) \in \mathbb{Z}_p \times \mathbb{G}_1$ for an arbitrary $c \in \mathbb{Z}_p \backslash \{-x\}$.*

Definition 4 (q-DDHI Assumption [8, 2]**).** *Given $(P_i, [x]P_i, [x^2]P_i, \ldots, [x^q]P_i)$ where $x \leftarrow \mathbb{Z}_p^\times$ it is hard to distinguish $[\frac{1}{x}]P_i$ from a random element of $\mathbb{G}_i$. Here i can be either 1 or 2.*

Definition 5 (q-SFP Assumption [1]**).** *Given $G_Z, F_Z, G_R, F_U \in \mathbb{G}_2$, $(A, \tilde{A}), (B, \tilde{B}) \in \mathbb{G}_1 \times \mathbb{G}_2$ and q random tuples (Z, R, S, T, U, V, W) each satisfying*

$$e(A, \tilde{A}) = e(Z, G_Z)\, e(R, G_R)\, e(T, S) \qquad e(B, \tilde{B}) = e(Z, F_Z)\, e(U, F_U)\, e(W, V)$$

it is hard to find another such tuple for which Z is neither 1 nor equal to any of the given Z-values.

Definition 6 (DH-LRSW Assumption [32]**).** *Given $([x]P_2, [y]P_2)$ for a random $(x, y) \leftarrow \mathbb{Z}_p^2$ and an oracle that on input a pair $(M_1, M_2) \in \mathbb{G}_1 \times \mathbb{G}_2$ of the form*[1] *$([m]P_1, [m]P_2)$ for some $m \in \mathbb{Z}_p$ picks a random $a \leftarrow \mathbb{Z}_p$ and outputs a DH-LRSW tuple of the form $([a]P_1, [ay]P_1, [ay]M_1, [ax]P_1 + [axy]M_1)$, it is hard to compute $([m']P_1, [m']P_2)$ that was never queried to the oracle and a corresponding DH-LRSW tuple.*

[1] The set of such messages $([m]P_1, [m]P_2)$ was called "Diffie-Hellman pairs" in [1] and can be efficiently decided by checking $e(M_1, P_2) = e(P_1, M_2)$.

2.2 Groth-Sahai Proofs

Groth-Sahai (GS) proofs [36, 37] are non-interactive zero-knowledge proofs (NIZK) in the common reference string (CRS) model. We will use GS proofs that are secure under the SXDH assumption (which as noted by [33] yields the most efficient instantiation of the proof system) and that prove knowledge of witnesses to pairing-product equations. These are equations of the form

$$\prod_{j=1}^{n} e(A_j, \underline{Y_j}) \prod_{i=1}^{m} e(\underline{X_i}, B_i) \prod_{i=1}^{m}\prod_{j=1}^{n} e(\underline{X_i}, \underline{Y_j})^{\gamma_{i,j}} \;=\; \prod_{\ell=1}^{M} e(G_\ell, H_\ell) \tag{1}$$

(the variables are underlined, all other values are constants). The language for these proofs is of the form $\mathcal{L} := \{\mathsf{statement} \mid \exists\, \mathsf{witness} : E(\mathsf{statement}, \mathsf{witness}) \text{ holds }\}$ where E is a set of pairing-product equations.

The GS proof system is formally defined by a tuple of algorithms

$$(\mathsf{GSSetup}, \mathsf{GSProve}, \mathsf{GSVerify}, \mathsf{GSExtract}, \mathsf{GSSimSetup}, \mathsf{GSSimProve}) \ .$$

$\mathsf{GSSetup}$ takes as input the description of a bilinear group $\mathcal{P}$ and outputs a binding reference string crs and an extraction key xk. $\mathsf{GSProve}$ takes as input a set of equations $\mathsf{statement}$, the string crs and a witness, and outputs a proof Ω for the satisfiability of the equations. For ease of notation, we write $\mathsf{GSProve}(\mathsf{crs}, \{\mathsf{witness}\} : \mathsf{statement} \in \mathcal{L})$. We write $\mathsf{GSProve}_{\mathrm{ZK}}$ if the proof to be produced is zero-knowledge, whereas we write $\mathsf{GSProve}_{\mathrm{WI}}$ if it is only witness indistinguishable. $\mathsf{GSVerify}$ takes as input a set of equations, a string crs and a proof Ω and outputs 1 if the proof is valid or 0 otherwise.

$\mathsf{GSExtract}$ takes as input a binding reference string crs, the extraction key xk and a valid proof Ω, and outputs the witness used for the proof. $\mathsf{GSSimSetup}$ takes as input a bilinear group $\mathcal{P}$ and outputs a *hiding* string $\mathsf{crs}_{\mathsf{Sim}}$ and a trapdoor key tr that allows to simulate proofs. $\mathsf{GSSimProve}$ takes as input $\mathsf{crs}_{\mathsf{Sim}}$, a statement and the simulation trapdoor tr and produces a simulated proof Ω_{Sim} without a witness. The distributions of strings crs and $\mathsf{crs}_{\mathsf{Sim}}$ are computationally indistinguishable and simulated proofs are indistinguishable from proofs generated by an honest prover. The proof system has the following properties:

- **Prefect Completeness:** On a correctly generated reference string crs and a proof Ω output by $\mathsf{GSProve}$ on a valid witness, the algorithm $\mathsf{GSVerify}$ always accepts.
- **Perfect Soundness:** On a correctly generated crs, it is impossible to generate a proof unless the equations are satisfiable; moreover, $\mathsf{GSExtract}$ extracts a witness for satisfiability.
- **Composable Witness-Indistinguishability:** The CRS crs output by $\mathsf{GSSetup}$ is computationally indistinguishable from the CRS $\mathsf{crs}_{\mathsf{Sim}}$ output by $\mathsf{GSSimSetup}$. Moreover, for any PPT adversaries $\mathcal{A}$ that is given $(\mathsf{crs}_{\mathsf{Sim}}, \mathsf{tr})$ and is allowed to choose any statement y and two witnesses w_0 and w_1 for the statement y, we have

  $$\Pr\left[b \leftarrow \{0,1\}; \Omega \leftarrow \mathsf{GSProve}(\mathsf{crs}_{\mathsf{Sim}}, w_b, y); b' \leftarrow \mathcal{A}(\Omega) : b = b' \wedge (w_0, y) \in R \wedge (w_1, y) \in R\right] = \frac{1}{2} + \nu(\lambda) \ .$$

- **Composable Zero-Knowledge:** Again, the CRS crs output by $\mathsf{GSSetup}$ is computationally indistinguishable from the CRS $\mathsf{crs}_{\mathsf{Sim}}$ output by $\mathsf{GSSimSetup}$. Moreover, we have for all PPT adversaries $\mathcal{A}$ that

  $$\Pr\begin{bmatrix} (\mathsf{crs}_{\mathsf{Sim}}, \mathsf{tr}) \leftarrow \mathsf{GSSimSetup}(\mathcal{P}); \\ (w, y) \leftarrow \mathcal{A}(\mathsf{crs}_{\mathsf{Sim}}, \mathsf{tr}); \\ \Omega \leftarrow \mathsf{GSProve}(\mathsf{crs}_{\mathsf{Sim}}, w, y) : \mathcal{A}(\Omega) = 1 \end{bmatrix} = \Pr\begin{bmatrix} (\mathsf{crs}_{\mathsf{Sim}}, \mathsf{tr}) \leftarrow \mathsf{GSSimSetup}(\mathcal{P}); \\ (w, y) \leftarrow \mathcal{A}(\mathsf{crs}_{\mathsf{Sim}}, \mathsf{tr}); \\ \Omega_{sim} \leftarrow \mathsf{GSSimProve}(\mathsf{crs}_{\mathsf{Sim}}, \mathsf{tr}, y) : \mathcal{A}(\Omega_{\mathsf{Sim}}) = 1 \end{bmatrix},$$

  where (w, y) output by $\mathcal{A}$ must satisfy $(w, y) \in R$.

2.3 Direct Anonymous Attestation: The pre-DAA Model

The syntax and security model for pre-DAA were defined in [6]. A pre-DAA scheme consists of a tuple of algorithms

$$\big(\mathsf{Setup}, \mathsf{GKg}, \mathsf{UKg}, \langle\mathsf{Join}, \mathsf{Iss}\rangle, \mathsf{GSig}, \mathsf{GVf}, \mathsf{Identify}_{\mathsf{T}}, \mathsf{Identify}_{\mathsf{S}}, \mathsf{Link}\big) \ .$$

All the algorithms (bar Setup, GKg and UKg) take as an implicit input the common public parameters param output by the Setup algorithm.

$\mathsf{Setup}(1^\lambda)$ takes as input the security parameter λ and outputs common public parameters param.

$\mathsf{GKg}(\mathsf{param})$ takes as input param and outputs a pair $(\mathsf{gmpk}, \mathsf{gmsk})$ of public/secret keys for the group issuer. The space of group issuer secret keys gmsk is denoted by GMSK.

$\mathsf{UKg}(\mathsf{param})$ takes as input param and outputs a secret key sk for a user. The space of such keys is denoted SK.

$\langle \mathsf{Join}(\mathsf{sk}_i, \mathsf{gmpk}), \mathsf{Iss}(\mathsf{gmsk}, \mathsf{gmpk}) \rangle$ are the user- and issuer-side procedures for an interactive protocol by means of which a user joins a group. The user takes a secret key sk_i and the issuer's public key gmpk as input, whereas the issuer has as input a key pair $(\mathsf{gmpk}, \mathsf{gmsk})$. For the uniqueness property of transcripts, we refer to the user's random coins in this protocol as R_U and the issuer's as R_I. If this protocol completes successfully, the user obtains a group signing key gsk. We assume without loss of generality that a group signing key thus obtained contains the group manager public key gmpk.

$\mathsf{GSig}(\mathsf{gsk}_i, \mathsf{sk}_i, m, \mathsf{bsn})$ is the signing algorithm. It takes as input a group signing key gsk_i, a user secret key sk_i, a message m and a basename bsn and it outputs a signature σ on the message m under the basename bsn.

$\mathsf{GVf}(\mathsf{gmpk}, \sigma, m, \mathsf{bsn})$ is the verification algorithm. It takes as input the group public key gmpk, a signature σ, a message m and a basename bsn. It returns 1 if the signature verifies with respect to these parameters, otherwise 0.

$\mathsf{Identify}_\mathsf{T}(\mathsf{gmpk}, \mathcal{T}, \mathsf{sk}_i)$ is a transcript tracing algorithm. It is used mainly in the security model rather than the actual DAA protocol, although it could be used to trace dishonest users who reveal their secret key. This algorithm takes as input the group public key gmpk, a transcript $\mathcal{T}$ of a join/issue protocol execution and a secret key sk_i and outputs 1 if this transcript could have been produced by a honest user with secret key sk_i, otherwise 0.

$\mathsf{Identify}_\mathsf{S}(\mathsf{gmpk}, \sigma, m, \mathsf{bsn}, \mathsf{sk}_i)$ is a signature tracing algorithm. Like the transcript tracing algorithm, its use is in the security model and possibly to trace dishonest users. Its inputs are the group public key gmpk, a signature σ, a message m, a basename bsn and a secret key sk_i. It outputs 1 if the signature σ could have been produced by a honest user with the secret key sk_i provided.

$\mathsf{Link}(\mathsf{gmpk}, m_0, \sigma_0, m_1, \sigma_1, \mathsf{bsn})$ is the linking algorithm. Its inputs are a group public key gmpk, two messages and signatures $m_0, m_1, \sigma_0, \sigma_1$ and a basename bsn. The linking algorithm outputs 1 if both signatures were produced by the same user on their respective messages, under the basename bsn, otherwise 0.

2.4 Security Definitions of pre-DAA

We briefly present the formal definitions of pre-DAA scheme security [6]. All oracles (and the underlying experiments) maintain the following global lists: a list HU of initially honest users, a list CU of corrupted users which are controlled by the adversary, a list BU of "bad" users which have been compromised (these are previously honest users which have since been corrupted), a list SL of queries to the signing oracle, and a list CL of queries to the challenge oracle. All the lists are assumed to be initially empty. The lists CL and SL are used to restrict the two relevant oracles so that one cannot trivially win the anonymity or non-frameability games, respectively.

The oracles are given in Fig. 1 and the security games in Fig. 2.

Correctness of Linking. The Link algorithm is supposed to indicate whether two signatures were produced by the same user on the same, non-empty basename. We require that if $\mathsf{Link}(\mathsf{gmpk}, m_0, \sigma_0, m_1, \sigma_1, \mathsf{bsn})$ outputs 1 then the following must hold: bsn is not empty and both signatures verify with respect to bsn i.e. $\mathsf{GVf}(\mathsf{gmpk}, \sigma_0, m_0, \mathsf{bsn}) = 1$ and $\mathsf{GVf}(\mathsf{gmpk}, \sigma_1, m_1, \mathsf{bsn}) = 1$.

Uniquely Identifiable Transcripts. In some of the security games, a formal definition is required for the property that a signature was produced by a certain user. The anonymity game ensures that no one can exploit this property but it must be defined nonetheless.

The solution chosen for the security model in [6] is that each execution of the join/issue protocol and each signature is bound in a mathematical way to a user secret key, i.e. there is a function that takes a transcript or signature and a key and tells if this object was created with the given key. These functions are called $\mathsf{Identify}_\mathsf{T}$ and $\mathsf{Identify}_\mathsf{S}$ respectively.

For the linking and traceability properties, it is essential that these tracing functions cannot trace the same transcript to two different keys. This property is called *uniquely identifiable transcripts* and is formally defined as follows:

```
$\underline{\mathsf{AddU}(i)}$
  • If $i \in \mathsf{HU} \cup \mathsf{CU}$ then return $\perp$.
  • $\mathsf{HU} := \mathsf{HU} \cup \{i\}$.
  • $\mathsf{sk}_i \leftarrow \mathsf{UKg}(\mathsf{param})$.

$\underline{\mathsf{CrptU}(i)}$
  • If $i \in \mathsf{HU} \cup \mathsf{CU}$ then return $\perp$.
  • $\mathsf{CU} := \mathsf{CU} \cup \{i\}$.

$\underline{\mathsf{InitU}(i)}$
  • If $i \notin \mathsf{HU} \setminus \mathsf{BU}$ then return $\perp$.
  • $\mathsf{gsk}_i := \perp$, $\mathsf{dec}_I^i = \mathsf{cont}$.
  • $\mathsf{St}_U^i := (\mathsf{gmpk}, \mathsf{sk}_i)$.
  • $\mathsf{St}_I^i := (\mathsf{gmpk}, \mathsf{gmsk})$.
  • $(\mathsf{St}_U^i, M_I, \mathsf{dec}_U^i) \leftarrow \mathsf{Join}(\mathsf{St}_U^i, \perp)$.
  • While $(\mathsf{dec}_I^i = \mathsf{cont}$ and $\mathsf{dec}_U^i = \mathsf{cont})$ do
      ◦ $(\mathsf{St}_I^i, M_J, \mathsf{dec}_I^i) \leftarrow \mathsf{Iss}(\mathsf{St}_I^i, M_I)$.
      ◦ $(\mathsf{St}_U^i, M_I, \mathsf{dec}_U^i) \leftarrow \mathsf{Join}(\mathsf{St}_U^i, M_J)$.
  • $\mathsf{gsk}_i := \mathsf{St}_U^i$.

$\underline{\mathsf{USK}(i)}$
  • If $i \notin \mathsf{HU}$ or $(i, \star) \in \mathsf{CL}$ then return $\perp$.
  • $\mathsf{BU} := \mathsf{BU} \cup \{i\}$.
  • Return $(\mathsf{sk}_i, \mathsf{gsk}_i)$.

$\underline{\mathsf{GSK}(i)}$
  • If $i \notin \mathsf{HU}$ then return $\perp$.
  • Return $\mathsf{gsk}_i$.

$\underline{\mathsf{Sign}(i, \mathsf{gsk}, m, \mathsf{bsn})}$
  • If $i \in \mathsf{CU} \cup \mathsf{BU}$ then return $\perp$.
  • $\mathsf{SL} := \mathsf{SL} \cup \{(i, m, \mathsf{bsn}, \sigma)\}$.
  • Return $\mathsf{GSig}(\mathsf{gsk}, \mathsf{sk}_i, m, \mathsf{bsn})$.

$\underline{\mathsf{SndToI}(i, M)}$
  • If $i \notin \mathsf{CU} \cup \mathsf{BU}$ or $\mathsf{dec}_I^i \neq \mathsf{cont}$ then return $\perp$.
  • If $\mathsf{St}_I^i$ is undefined then $\mathsf{St}_I^i := (\mathsf{gmpk}, \mathsf{gmsk})$.
  • $(\mathsf{St}_I^i, M', \mathsf{dec}_I^i) \leftarrow \mathsf{Iss}(\mathsf{St}_I^i, M)$.
  • Return $(M', \mathsf{dec}_I^i)$.

$\underline{\mathsf{SndToU}(i, M)}$
  • If $i \notin \mathsf{HU} \setminus \mathsf{BU}$ or $\mathsf{dec}_U^i \neq \mathsf{cont}$ or $\mathsf{gsk}_i \neq \perp$
      ◦ Return $\perp$.
  • If $\mathsf{St}_U^i$ is undefined
      ◦ $\mathsf{St}_U^i := (\mathsf{gmpk}, \mathsf{sk}_i)$.
  • $(\mathsf{St}_U^i, M', \mathsf{dec}_U^i) \leftarrow \mathsf{Join}(\mathsf{St}_U^i, M)$
  • If $\mathsf{dec}_U^i = \mathsf{accept}$ then
      ◦ $\mathsf{gsk}_i := \mathsf{St}_U^i$.
  • Return $(M', \mathsf{dec}_U^i)$.

$\underline{\mathsf{CH}_b(i_0, i_1, \mathsf{bsn}, m)}$
  • If $i_0$ or $i_1 \in \mathsf{CU} \cup \mathsf{BU}$, or $\mathsf{gsk}_{i_0} = \perp$, or $\mathsf{gsk}_{i_1} = \perp$
      ◦ Return $\perp$.
  • $\mathsf{CL} := \{(i_0, \mathsf{bsn}), (i_1, \mathsf{bsn})\}$.
  • $\sigma \leftarrow \mathsf{GSig}(\mathsf{gsk}_{i_b}, \mathsf{sk}_{i_b}, m, \mathsf{bsn})$.
  • Return $\sigma$.
```

Fig. 1. Oracles used in the security games for a pre-DAA scheme

Definition 7. *Let* GMSK *and* SK *be the spaces of all possible issuer and user secret keys, respectively. Also, let* R_I *and* R_U *be the spaces of randomness used by the issuer and the user in the* $\mathsf{Join}/\mathsf{Iss}$ *protocol, respectively. A pre-DAA scheme has* uniquely identifying transcripts *if there exists a predicate*

$$\mathsf{Check}_T : \{0,1\}^* \times \mathsf{GMSK} \times \mathsf{SK} \times \mathsf{R}_I \times \mathsf{R}_U \to \{0,1\}$$

such that the following holds:

- *Suppose both parties are honest and run* $(\mathsf{Join}, \mathsf{Iss})$ *with inputs* sk *and* gmsk*, and randomness* r_U *and* r_I*, respectively. Then the resulting transcript* $\mathcal{T}$ *satisfies* $\mathsf{Check}_T(\mathcal{T}, \mathsf{gmsk}, \mathsf{sk}, r_I, r_U) = 1$.
- *For all (possibly dishonest) protocols* Join' *interacting with* Iss *producing* $\mathcal{T}$*, if the issuer accepts then there is at most one value* $\mathsf{sk} \in \mathsf{SK}$ *(but possibly many values of* r_U*) such that we have* $\mathsf{Check}_T(\mathcal{T}, \mathsf{gmsk}, \mathsf{sk}, r_I, r_U) = 1$. *Here* gmsk *is the issuer's input and* r_I *his random coins; as the user-side algorithm may be dishonest, we let it take arbitrary input.*

Security Requirements. Security of pre-DAA consists of the following requirements whose formal security games are given in Fig. 2.

- **Correctness:** This demands that signatures produced by honest users are accepted by the verifier, and that the user who produced a valid signature can be traced. Moreover, two signatures by the same user and on the same non-empty basename are linked.

```
*Experiment*: $\mathsf{Exp}^{\text{corr}}_{\mathcal{A}}(\lambda)$
  • $\mathsf{param} \leftarrow \mathsf{Setup}(1^\lambda)$;
  • $(\mathsf{gmpk}, \mathsf{gmsk}) \leftarrow \mathsf{GKg}(\mathsf{param})$.
  • $\mathsf{HU} := \emptyset$.
  • $(i, m_0, m_1, \mathsf{bsn}) \leftarrow \mathcal{A}(\mathsf{gmpk} : \mathsf{AddU}, \mathsf{InitU})$.
  • If $\mathsf{gsk}_i = \perp$ then return 0.
  • $\sigma_0 \leftarrow \mathsf{GSig}(\mathsf{gsk}_i, \mathsf{sk}_i, m_0, \mathsf{bsn})$.
  • $\sigma_1 \leftarrow \mathsf{GSig}(\mathsf{gsk}_i, \mathsf{sk}_i, m_1, \mathsf{bsn})$.
  • If $\mathsf{GVf}(\mathsf{gmpk}, \sigma_0, m_0, \mathsf{bsn}) = 0$ then return 1.
  • If $\mathsf{GVf}(\mathsf{gmpk}, \sigma_1, m_1, \mathsf{bsn}) = 0$ then return 1.
  • If $\mathsf{bsn} \neq \perp$ then
      ◦ If $\mathsf{Link}(\mathsf{gmpk}, \sigma_0, m_0, \sigma_1, m_1, \mathsf{bsn}) = 0$
        then return 1.
  • If $\mathsf{Identify}_{\mathsf{S}}(\mathsf{gmpk}, \sigma_0, m_0, \mathsf{bsn}, \mathsf{sk}_i) = 0$ then return 1.
  • Let $\mathcal{T}_i$ denote the $(\mathsf{Join}, \mathsf{Iss})$ transcript for user $i$.
  • If $\mathsf{Identify}_{\mathsf{T}}(\mathsf{gmpk}, \mathcal{T}_i, \mathsf{sk}_i) = 0$ then return 1.
  • Return 0.
```

```
*Experiment*: $\mathsf{Exp}^{\text{trace}}_{\mathcal{A}}(\lambda)$
  • $\mathsf{param} \leftarrow \mathsf{Setup}(1^\lambda)$;
  • $(\mathsf{gmpk}, \mathsf{gmsk}) \leftarrow \mathsf{GKg}(\mathsf{param})$.
  • $\mathsf{CU} := \emptyset$.
  • $(\sigma, m, \mathsf{bsn}, \{\mathsf{sk}'_i\}_{i=1}^{\ell}) \leftarrow \mathcal{A}_1(\mathsf{gmpk} : \mathsf{SndToI}, \mathsf{CrptU})$.
  • Let $\mathbb{T}$ denote the set of all transcripts accepted by the
    honest issuer via use of $\mathsf{SndToI}$.
  • If the following three conditions all hold then return 1
      ◦ $\mathsf{GVf}(\mathsf{gmpk}, \sigma, m, \mathsf{bsn}) = 1$.
      ◦ $\forall \mathcal{T} \in \mathbb{T}\ \exists i \in [1, \ell]$: $\mathsf{Identify}_{\mathsf{T}}(\mathsf{gmpk}, \mathcal{T}, \mathsf{sk}'_i) = 1$.
      ◦ $\forall i \in [1, \ell]$, $\mathsf{Identify}_{\mathsf{S}}(\mathsf{gmpk}, \sigma, m, \mathsf{bsn}, \mathsf{sk}'_i) = 0$.
  • $(\sigma_0, m_0, \sigma_1, m_1, \mathsf{bsn}, \mathsf{sk}') \leftarrow \mathcal{A}_2(\mathsf{gmpk}, \mathsf{gmsk})$
  • If $\mathsf{bsn} = \perp$ then return 0.
  • If the following three conditions all hold then return 1,
      ◦ $\forall b \in \{0, 1\}$, $\mathsf{GVf}(\mathsf{gmpk}, \sigma_b, m_b, \mathsf{bsn}) = 1$.
      ◦ $\forall b \in \{0, 1\}$, $\mathsf{Identify}_{\mathsf{S}}(\mathsf{gmpk}, \sigma_b, m_b, \mathsf{bsn}, \mathsf{sk}') = 1$
      ◦ $\mathsf{Link}(\mathsf{gmpk}, \sigma_0, m_0, \sigma_1, m_1, \mathsf{bsn}) = 0$
  • Return 0.
```

```
*Experiment*: $\mathsf{Exp}^{\text{anon-}b}_{\mathcal{A}}(\lambda)$
  • $\mathsf{param} \leftarrow \mathsf{Setup}(1^\lambda)$.
  • $(\mathsf{gmpk}, \mathsf{gmsk}) \leftarrow \mathsf{GKg}(\mathsf{param})$.
  • $\mathsf{CU}, \mathsf{HU}, \mathsf{BU}, \mathsf{CL}, \mathsf{SL} := \emptyset$.
  • $d \leftarrow \mathcal{A}(\mathsf{gmpk}, \mathsf{gmsk} :$
    $\mathsf{AddU}, \mathsf{SndToU}, \mathsf{CrptU}, \mathsf{USK}, \mathsf{GSK}, \mathsf{Sign}, \mathsf{CH}_b)$.
  • If $\exists i, m, \sigma, \mathsf{bsn}$ s.t. $\mathsf{bsn} \neq \perp$, $(i, \mathsf{bsn}) \in \mathsf{CL}$
        and $(i, m, \mathsf{bsn}, \sigma) \in \mathsf{SL}$ then abort the game.
  • Return $d$.
```

```
*Experiment*: $\mathsf{Exp}^{\text{non-frame}}_{\mathcal{A}}(\lambda)$
  • $\mathsf{param} \leftarrow \mathsf{Setup}(1^\lambda)$.
  • $(\mathsf{gmpk}, \mathsf{gmsk}) \leftarrow \mathsf{GKg}(\mathsf{param})$.
  • $\mathsf{CU}, \mathsf{HU}, \mathsf{BU}, \mathsf{SL} := \emptyset$.
  • $(\sigma, i, m, \mathsf{bsn}) \leftarrow \mathcal{A}_1(\mathsf{gmpk}, \mathsf{gmsk} :$
        $\mathsf{AddU}, \mathsf{SndToU}, \mathsf{CrptU}, \mathsf{USK}, \mathsf{GSK}, \mathsf{Sign})$.
  • If the following four conditions all hold then return 1.
      ◦ $\mathsf{GVf}(\mathsf{gmpk}, \sigma, m, \mathsf{bsn}) = 1$.
      ◦ $i \in \mathsf{HU} \setminus \mathsf{BU}$.
      ◦ $\forall \sigma' : (i, m, \mathsf{bsn}, \sigma') \notin \mathsf{SL}$.
      ◦ $\mathsf{Identify}_{\mathsf{S}}(\mathsf{gmpk}, \sigma, m, \mathsf{bsn}, \mathsf{sk}_i) = 1$.
  • $(\sigma_0, m_0, \mathsf{bsn}_0, \sigma_1, m_1, \mathsf{bsn}_1, \mathsf{sk}) \leftarrow \mathcal{A}_2(\mathsf{gmpk}, \mathsf{gmsk})$.
  • If one of the following condition holds then return 0:
      ◦ $\exists b \in \{0, 1\} : \mathsf{GVf}(\mathsf{gmpk}, \sigma_b, m_b, \mathsf{bsn}_b) = 0$.
      ◦ $\forall b \in \{0, 1\}$ :
            $\mathsf{Link}(\mathsf{gmpk}, \sigma_0, m_0, \sigma_1, m_1, \mathsf{bsn}_b) = 0$.
  • If one of the following conditions holds then return 1:
      ◦ $\mathsf{Identify}_{\mathsf{S}}(\mathsf{gmpk}, \sigma_0, m_0, \mathsf{bsn}_0, \mathsf{sk}) = 1$ and
        $\mathsf{Identify}_{\mathsf{S}}(\mathsf{gmpk}, \sigma_1, m_1, \mathsf{bsn}_1, \mathsf{sk}) = 0$.
      ◦ $\mathsf{bsn}_0 \neq \mathsf{bsn}_1$ or $\mathsf{bsn}_0 = \perp$ or $\mathsf{bsn}_1 = \perp$.
  • Return 0.
```

Fig. 2. Security experiment for correctness, anonymity, traceability and non-frameability

For any adversary $\mathcal{A}$ and any $\lambda \in \mathbb{N}$, we define $\mathsf{Adv}^{\text{corr}}_{\mathcal{A}}(\lambda) := \Pr[\mathsf{Exp}^{\text{corr}}_{\mathcal{A}}(\lambda) = 1]$ and say the scheme is *correct* if $\mathsf{Adv}^{\text{corr}}_{\mathcal{A}}(\lambda) = 0$ for all adversaries $\mathcal{A}$.

- **Anonymity:** An adversary, who may control the group issuer, cannot distinguish which of two users of his choice signed a message as long as he cannot trivially decide this using the linking property. This is formulated by the game $\mathsf{Exp}^{\text{anon}}_{\mathcal{A}}(\lambda)$ in Fig. 2.
  For an adversary $\mathcal{A}$, $\lambda \in \mathbb{N}$, we define $\mathsf{Adv}^{\text{anon}}_{\mathcal{A}}(\lambda) := \big|\Pr[\mathsf{Exp}^{\text{anon-0}}_{\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\text{anon-1}}_{\mathcal{A}}(\lambda) = 1]\big|$ and say the scheme is *anonymous* if $\mathsf{Adv}^{\text{anon}}_{\mathcal{A}}(\lambda)$ is negligible in λ for all probabilistic polynomial-time (PPT) adversaries $\mathcal{A}$.
- **Traceability:** No group of users can create an untraceable signature as long as the issuer is honest. (A dishonest issuer could always join untraceable users to his group.) The game consists of two subgames which deal with untraceable signatures and signatures that do not link although they should. Since unlike in group signatures, users do have public keys corresponding to their secret keys, the group-join transcript is used to identify the user.

For an adversary $\mathcal{A}$ and $\lambda \in \mathbb{N}$ we define $\mathsf{Adv}_{\mathcal{A}}^{\text{trace}}(\lambda) := \Pr[\mathsf{Exp}_{\mathcal{A}}^{\text{trace}}(\lambda) = 1]$ and say the scheme is *traceable* if $\mathsf{Adv}_{\mathcal{A}}^{\text{trace}}(\lambda)$ is negligible in λ for all PPT adversaries $\mathcal{A}$.

- **Non-frameability:** No adversary, who may even control the group issuer, can frame an honest user by claiming that he signed a message if he never did. This game is again split into two subgames to deal with framing a user by creating a signature that traces to his key or one that links to a previous signature created by the user in question. For an adversary $\mathcal{A}$ and $\lambda \in \mathbb{N}$ we define $\mathsf{Adv}_{\mathcal{A}}^{\text{non-frame}}(\lambda) := \Pr[\mathsf{Exp}_{\mathcal{A}}^{\text{non-frame}}(\lambda) = 1]$ and say the scheme is *non-frameable* if $\mathsf{Adv}_{\mathcal{A}}^{\text{non-frame}}(\lambda)$ is negligble in λ for all PPT adversaries $\mathcal{A}$.

3 Efficient Signatures of Knowledge without Random Oracles

Let $\mathcal{L}$ be an *NP language*, defined by a polynomial-time computable relation R as $\mathcal{L} = \{x \mid \exists w : (x, w) \in R\}$. We call x a *statement* in $\mathcal{L}$ and w with $(x, w) \in R$ a *witness* for x. A *signature of knowledge* (SoK) for $\mathcal{L}$ consists of the following three algorithms: $\mathsf{SoKSetup}$ takes a security parameter 1^λ and outputs parameters par. If $(x, w) \in R$ then $\mathsf{SoKSign}(\mathsf{par}, R, x, w, m)$ outputs a signature σ on the message m w.r.t. statement x. The signature is verified by $\mathsf{SoKVerify}(\mathsf{par}, R, x, m, \sigma)$ which outputs 0 or 1. Signatures produced by $\mathsf{SoKSign}$ on inputs par output by $\mathsf{SoKSetup}$, and any (R, x, w, m) such that $(x, w) \in R$ should be accepted by $\mathsf{SoKVerify}$. The (game-based) security definition for SoK, called *SimExt security*, was introduced in [20] and requires the following:

- **Simulatability:** There exists a *simulator* which can simulate signatures *without* having a witness for the statement. The simulator consists of the algorithms $\mathsf{SoKSimSetup}$ and $\mathsf{SoKSimSign}$: the former outputs parameters together with a *trapdoor* tr and the latter outputs signatures on input $(\mathsf{par}, \mathsf{tr}, R, x, m)$. It is required that no adversary can distinguish the following two situations: (1) It is given par output by $\mathsf{SoKSetup}$ and access to a $\mathsf{SoKSign}$ oracle; or (2) it is given par output by $\mathsf{SoKSimSetup}$ and an oracle SoKSim that on input (R, x, w, m) outputs $\mathsf{SoKSimSign}(\mathsf{par}, \mathsf{tr}, R, x, m)$ if $(x, w) \in R$.
- **Extraction:** There exists an algorithm $\mathsf{SoKExtract}$ such that if an adversary, given $\mathsf{par} \leftarrow \mathsf{SoKSimSetup}$ and an oracle SoKSim as above, outputs a tuple (R, x, m, σ), we have the following: if $\mathsf{SoKVerify}(\mathsf{par}, R, x, m, \sigma) = 1$ and the adversary never queried (R, x, w', m), for any w', to its SoKSim oracle then $\mathsf{SoKExtract}$ extracts a witness for x from σ with overwhelming probability.

Chase and Lysyanskaya [20] offer a generic construction satisfying the above model, but it is inefficient due to the use of general Zero-Knowledge (ZK) proofs. The first efficient non-interactive ZK proofs without random oracles were introduced by Groth and Sahai [36]; however these proofs only apply to a restricted language. We will use their results to construct the first efficient signatures of knowledge in the standard model for the same language, namely *satisfiability of sets of Pairing-Product Equations (PPE)* (for which Groth-Sahai proofs are *proofs of knowledge*). Attempting to construct SoKs which satisfy SimExt security using GS proofs, we face the following dilemma: if we generate a regular Common Reference String (CRS) using $\mathsf{GSSetup}$ then this leads to sound proofs and we can use $\mathsf{GSExtract}$ to extract a valid witness from a correct proof. However, in order to simulate GS proofs, we need to set up the CRS via $\mathsf{GSSimSetup}$. As such proofs are information-theoretically independent of the witnesses, we cannot extract anymore.

In order to achieve simulatability and extractability at the same time, we revise the approach to simulatability using a well-known trick that has also been employed in the context of PPEs by Groth [35]. We include in the CRS a verification key for an appropriate signature scheme and then require the prover to prove the following modified statement: to know a witness for the original statement *or* to know a signature on the original statement and the message to be signed, under the key contained in the CRS.

To simulate SoKs, we can now use the signing key to sign the statement and the message, and use this signature as a witness for the modified statement. (Computational) witness indistinguishability of GS proofs guarantees that simulated signatures are indistinguishable from honestly created signatures, which use the witness of the original statement. Using a binding CRS, extractability follows since from an adversarially created SoK we can extract a witness for the *modified* statement. The witness is thus either a witness for the original statement, or it is a signature on a statement/message pair which was never signed by the SoKSim oracle, and thus is a forgery.

Choosing the Signature Scheme. We need to prove knowledge of a valid signature using GS proofs. We thus require a scheme whose signatures consist of group elements and whose validity is verified by evaluating PPEs.

In order to not rely on any additional assumptions other than those already required by Groth-Sahai proofs, an optimal candidate would be Waters' signatures [46], which are secure under the computational Diffie-Hellman (CDH) assumption. (Their main drawback is a long public key, which will result in long parameters for our SoK.) Since these signatures were defined over *symmetric* bilinear groups (where $\mathbb{G}_1 = \mathbb{G}_2$), they would limit the language to statements over such groups. Using the Groth-Sahai instantiation under the *decision-linear assumption* [11], which implies CDH, we would thus get signatures of knowledge for the same statements and under the same assumption as Groth-Sahai proofs.

To allow for a more general class of statements, we use the following generalization of Waters signatures to asymmetric groups, used by [21, 7]:

Parameter Generation. Given a bilinear group $\mathcal{P}$, to sign messages of the form $m = (m_1, \ldots, m_N) \in \{0,1\}^N$, choose $(Q, U_0, \ldots, U_N) \leftarrow \mathbb{G}_1^{N+2}$.

Key Generation. Choose a secret key $\mathsf{sk} \leftarrow \mathbb{Z}_p$ and define the public verification key as $\mathsf{vk} := [\mathsf{sk}]P_2$.

Signing. To sign $(m_1, \ldots, m_N)$ using key sk, choose a random $r \leftarrow \mathbb{Z}_p$ and output

$$\big(W_1 := [\mathsf{sk}]Q + [r](U_0 + \textstyle\sum_{i=1}^N [m_i]U_i),\ W_2 := [-r]P_1,\ W_3 := [-r]P_2\big) \ .$$

Verification. Check whether $e(W_1, P_2)\, e(U_0 + \sum_{i=1}^N [m_i]U_i, W_3) = e(Q, \mathsf{vk})$ and $e(W_2, P_2) = e(P_1, W_3)$.

This construction is unforgeable under chosen-message attack under the CDH$^+$ assumption. In order to sign arbitrary messages, we assume a collision-resistant hash function $\mathcal{H}\colon \{0,1\}^* \to \{0,1\}^N$ (for a suitable N).

Disjunctions of Pairing-Product Equations. Groth [35] shows how to express disjunctions of sets of PPEs as a new set of PPEs. The idea is the following: introduce a "selector equation" of the form $e(P_1, \underline{S} + \underline{T} - P_2) = 1$, which can only be satisfied if either S or T are different from 0. Setting one of them to 0 will enable us to simulate one clause of the disjunction: for each variable X_i, Groth introduces an auxiliary variable X_i' and adds the equation $e(\underline{X_i} - \underline{X_i}', \underline{S}) = 1$, which is satisfied if $S = 0$, meaning we can set X_i' to 0 in order to simulate. On the other hand, when simulating the other clause by setting $T = 0$, we must have $S = P_2$ to satisfy the selector equation, and furthermore $X_i = X_i'$. This guarantees soundness, as we need to have a satisfying witness for at least one clause.

We choose a more efficient approach inspired by that from [36]. In order to simulate equations of the form (1), it suffices to replace the constants G_ℓ by auxiliary variables G_ℓ', as then, setting all variables to 0 is a satisfying assignment for (1). Now it only remains to ensure that a signer without the trapdoor is forced to set G_ℓ' to G_ℓ, which is done by adding equations $e(G_\ell - \underline{G'}_\ell, \underline{S}) = 1$, where S can only be set to 0 when the prover knows a signature under the public key from the CRS.

With this intuition we now define our signature of knowledge of a satisfying witness of a set of pairing-product equations. Regarding the Chase-Lysyanskaya definition, we have fixed the relation R to be the set of all pairs $\big((E_k)_{k=1}^K, ((X_i)_{i=1}^m, (Y_j)_{j=1}^n)\big)$ such that $((X_i), (Y_j)) \in \mathbb{G}_1^m \times \mathbb{G}_2^n$ satisfies E_k for all $1 \le k \le K$.

3.1 A Concrete Construction of Signatures of Knowledge without Random Oracles

Setup. On input $\mathcal{P}$, run $(\mathsf{crs}, \mathsf{xk}) \leftarrow \mathsf{GSSetup}(\mathcal{P})$ and choose parameters $(Q, U_0, \ldots, U_N) \leftarrow \mathbb{G}_1^{N+2}$ and a key pair for Waters signatures: choose $t \leftarrow \mathbb{Z}_p$ and set $T := [t]P_2$. $\mathsf{SoKSetup}$ outputs $\mathsf{par} := (\mathsf{crs}, (Q, U_0, \ldots, U_N, T))$, whereas $\mathsf{SoKSimSetup}$ additionally outputs (xk, t) as an extraction/simulation trapdoor.

Signing. Let $E := (E_k)_{k=1}^K$ be the set of equations representing the statement w.r.t. which we sign, where E_k is

$$\prod_{j=1}^n e(A_{k,j}, \underline{Y_j}) \prod_{i=1}^m e(\underline{X_i}, B_{k,i}) \prod_{i=1}^m \prod_{j=1}^n e(\underline{X_i}, \underline{Y_j})^{\gamma_{k,i,j}} = \prod_{\ell=1}^{M_k} e(G_{k,\ell}, H_{k,\ell}) \ , \qquad (E_k)$$

and let $((X_i)_{i=1}^m, (Y_j)_{j=1}^n)$ be a witness of satisfiability of E. We first define a set of equations E':

(i) Modified equations. For all $1 \leq k \leq K$

$$\prod e(A_{k,j}, \underline{Y_j}) \prod e(\underline{X_i}, B_{k,i}) \prod e(\underline{G'_{k,\ell}}, H_{k,\ell}) \prod\prod e(\underline{X_i}, \underline{Y_j})^{\gamma_{k,i,j}} = 1 \ .$$

(ii) Selector equations. For all $1 \leq k \leq K, 1 \leq \ell \leq M_k : e(G_{k,\ell} - \underline{G'_{k,\ell}}, T - \underline{T'}) = 1$.

(iii) Signature-verification equations.

$$e(\underline{W_1}, P_2)\, e(U_0 + \textstyle\sum_{i=1}^{N} [h_i] U_i, \underline{W_3}) = e(Q, \underline{T'}) \qquad\qquad e(\underline{W_2}, P_2) = e(P_1, \underline{W_3}) \ .$$

To sign a message $m \in \{0,1\}^*$ under $\mathsf{par} := (\mathsf{crs}, (Q, U_0, \ldots, U_N, T))$ for the statement E using witness $((X_i), (Y_j))$ proceed as follows:

- Set $T' = W_1 = W_2 = W_3 := 0$ and $G'_{k,\ell} := G_{k,\ell}$, for all k and ℓ.
- Compute $h = (h_1, \ldots, h_N) := \mathcal{H}(E \| m) \in \{0,1\}^N$, where E is an encoding of the original equations.
- The SoK is a GS proof (of knowledge) Σ of satisfiability of the set of equations E', using as witness

$$(T', W_1, W_2, W_3, X_1, \ldots, X_m, Y_1, \ldots, Y_n, G'_{1,1}, \ldots, G'_{K,M_K}) \ . \qquad (2)$$

Verification. To verify a signature of knowledge Σ on m for the statement E under $\mathsf{par} := (\mathsf{crs}, (Q, U_0, \ldots, U_N, T))$, verify that Σ is valid under crs on the statement E' for the values $A_{k,j}, B_{k,i}, G_{k,\ell}, H_{k,\ell}$ and $\gamma_{k,i,j}$ from the description of E, values T and $(Q, U_0, \ldots, U_N)$ from par and h defined as $\mathcal{H}(E \| m)$.

Theorem 1. *The above is a signature-of-knowledge scheme satisfying SimExt security for the language of sets of (simulatable) pairing-product equations.*

Proof sketch. To simulate a signature without knowing a witness, one uses the trapdoor t to make a signature (W_1, W_2, W_3) on $(h_1, \ldots, h_N) := \mathcal{H}(E \| m)$, and sets $T' = T$ and all remaining witnesses components $X_i = Y_j = G'_{k,\ell} := 0$, which is easily verified to satisfy E'. Simulatability then follows from witness indistinguishability of GS proofs.

For "Extraction", consider an adversary that has never queried a signature for a pair (E, m), but outputs a SoK Σ for it. By soundness of GS proofs, using xk we can extract from Σ a witness for E' of the form (2). We must have $T' \neq T$, as otherwise (W_1, W_2, W_3) would be a forgery on $(E \| m)$ (which was never queried to the simulator) by equations (iii) of E'. Together with equations (ii) of E', $T' \neq T$ implies that $G'_{k,\ell} = G_{k,\ell}$ for all k, ℓ, and therefore, by (i), $((X_i), (Y_j))$ is a witness for the original equation E. We have thus extracted a witness for the statement E. $\square$

Note that to reduce the parameter size, but relying on stronger ("q-type") assumptions, we could use any of the structure-preserving signatures defined by Abe et al. [1] instead of Waters signatures.

4 New Building Blocks

4.1 Partially Weak Blind Signatures

Bernhard et al. [6] introduce Randomizable Weakly Blind Signatures (RwBS) as one of the building blocks for DAA. These are similar to blind signatures [22, 39, 44] except that blindness has to hold only against adversaries that never get to see the message they signed, that is, a signature should not be linkable to its issuing session. Moreover, randomizability means that given a signature, anyone can produce a fresh signature on the same message.

Definition 8. *A randomizable blind signature scheme* BS *(with a two-move signature request phase) consists of six probabilistic polynomial-time algorithms*

$$\mathsf{BS} := (\mathsf{BSSetup}, \mathsf{BSKeyGen}, \mathsf{BSRequest}, \mathsf{BSIssue}, \mathsf{BSVerify}, \mathsf{BSRandomize}) \ .$$

The syntax of these algorithms is defined as follows; all algorithms (bar $\mathsf{BSSetup}$) are assumed to take as implicit input a parameter set $\mathsf{param}_{\mathsf{BS}}$ as output by $\mathsf{BSSetup}$.

- $\mathsf{BSSetup}(1^\lambda)$ takes as input a security parameter λ and outputs a parameter set $\mathsf{param}_{\mathsf{BS}}$, assumed to contain a description of the key and message spaces for BS.

$\underline{\mathsf{BSKeyGen}(\mathsf{param}_{\mathsf{BS}})}$

- $G_R, F_U \leftarrow \mathbb{G}_2^\times,\ a, b \leftarrow \mathbb{Z}_p^\times$.
- For $i = 1, \ldots, k$:
  $c_i, d_i \leftarrow \mathbb{Z}_p^\times,\ G_i := [c_i]G_R, F_i := [d_i]F_U$.
- $c_Z, d_Z \leftarrow \mathbb{Z}_p^\times,\ G_Z := [c_Z]G_R, F_Z := [d_Z]F_U$.
- Pick $(A_0, A_1, \tilde{A}_0, \tilde{A}_1)$ randomly such that $e(A_0, \tilde{A}_0)e(A_1, \tilde{A}_1) = e([a]P_1, G_R)$.
- Pick $(B_0, B_1, \tilde{B}_0, \tilde{B}_1)$ randomly such that $e(B_0, \tilde{B}_0)e(B_1, \tilde{B}_1) = e([b]P_1, F_U)$.
- $\mathsf{sk} := (a, b, c_z, d_z, (c_i, d_i)_{i=1}^k)$.
- $\mathsf{pk} := (G_Z, F_Z, G_R, F_U, (G_i, F_i)_{i=1}^k, A_0, A_1, \tilde{A}_0, \tilde{A}_1, B_0, B_1, \tilde{B}_0, \tilde{B}_1)$.
- Return $(\mathsf{sk}, \mathsf{pk})$.

$\underline{\mathsf{BSSign}(\boldsymbol{M} = (M_i)_{i=1}^k \in \mathbb{G}_1^k)}$

- $z, r, t, u, w \leftarrow \mathbb{Z}_p^\times$.
- $Z := [z]P_1,\ R := [r - c_z z]P_1 \sum_{i:=1}^k [-c_i]M_i$,
  $S := [t]G_R,\ T := [(a - r)/t]P_1$,
  $U := [u - d_z z]P_1 \sum_{i:=1}^k [-d_i]M_i$,
  $V := [w]F_U,\ W := [(b - u)/w]P_1$.
- $\sigma := (Z, R, S, T, U, V, W)$.
- Return σ.

$\underline{\mathsf{BSVerify}(\mathsf{pk}, \sigma, \boldsymbol{M})}$

- Parse σ as (Z, R, S, T, U, V, W), $\boldsymbol{M}$ as $(M_i)_{i=1}^k$.
- Parse pk as $(G_Z, F_Z, G_R, F_U, (G_i, F_i)_{i=1}^k, A_0, A_1, \tilde{A}_0, \tilde{A}_1, B_0, B_1, \tilde{B}_0, \tilde{B}_1)$.
- Return 1 if the following equations hold or 0 otherwise:
  - $e(Z, G_Z)e(R, G_R)e(T, S) \prod_i e(M_i, G_i) = e(A_0, \tilde{A}_0)e(A_1, \tilde{A}_1)$
  - $e(Z, F_Z)e(U, F_U)e(W, V) \prod_i e(M_i, F_i) = e(B_0, \tilde{B}_0)e(B_1, \tilde{B}_1)$

Fig. 3. The structure-preserving AHO signature from [1].

- $\mathsf{BSKeyGen}(\mathsf{param}_{\mathsf{BS}})$ takes as input the system parameters and outputs a pair $(\mathsf{pk}_{\mathsf{BS}}, \mathsf{sk}_{\mathsf{BS}})$ of public/secret keys for the signer.
- $(\mathsf{BSRequest}^0, \mathsf{BSIssue}^1, \mathsf{BSRequest}^1)$ is an interactive protocol run between a user and a signer. The user goes first by calling $\mathsf{BSRequest}^0(m, \mathsf{pk}_{\mathsf{BS}})$ to obtain a value ρ_0 and some state information St_R^0 (which is assumed to contain m). Then the signer and user execute, respectively,

$$(\beta_1, \mathsf{St}_I^1) \leftarrow \mathsf{BSIssue}^1(\rho_0, \mathsf{sk}_{\mathsf{BS}}) \quad \text{and} \quad (\sigma, \mathsf{St}_R^1) \leftarrow \mathsf{BSRequest}^1(\beta_1, \mathsf{St}_R^0) \ ,$$

  where σ is a signature on the original message m (or the abort symbol $\perp$). We write

$$\sigma \leftarrow \big(\mathsf{BSRequest}(m, \mathsf{pk}_{\mathsf{BS}}) \Longleftrightarrow \mathsf{BSIssue}(\mathsf{sk}_{\mathsf{BS}})\big)$$

  for the output of correctly running this interactive protocol on the given inputs.
- $\mathsf{BSVerify}(m, \sigma, \mathsf{pk}_{\mathsf{BS}})$ is the public signature verification algorithm, which outputs 1 if σ is a valid signature on m and 0 otherwise.
- $\mathsf{BSRandomize}(\sigma)$ is given a signature σ on an unknown message m and produces another valid signature σ' on the same message.

The blind signature scheme is *correct* if signatures verify when both parties behave honestly, i.e. for all parameter sets output by $\mathsf{BSSetup}$ we have

$$\begin{aligned}\Pr [\ &(\mathsf{pk}_{\mathsf{BS}}, \mathsf{sk}_{\mathsf{BS}}) \leftarrow \mathsf{BSKeyGen}(\mathsf{param}_{\mathsf{BS}}),\ m \leftarrow \mathcal{M},\\ &\sigma \leftarrow (\mathsf{BSRequest}(m, \mathsf{pk}_{\mathsf{BS}}) \Longleftrightarrow \mathsf{BSIssue}(\mathsf{sk}_{\mathsf{BS}})) : \mathsf{BSVerify}(m, \sigma, \mathsf{pk}_{\mathsf{BS}}) = 1\ ] = 1 \ .\end{aligned}$$

```
Experiment: $\mathsf{Exp}^{\text{part-w-blind}}_{\mathsf{BS},\mathcal{A}}(\lambda)$
  • $\mathsf{param}_{\mathsf{BS}} \leftarrow \mathsf{BSSetup}(1^\lambda)$.
  • $(\mathsf{pk}_{\mathsf{BS}}, \mathsf{sk}_{\mathsf{BS}}) \leftarrow \mathsf{BSKeyGen}(\mathsf{param}_{\mathsf{BS}})$.
  • $m_0, m_1 \leftarrow \mathcal{M}$.
  • $(\rho_0, \mathsf{St}_R) \leftarrow \mathsf{BSRequest}^0(\mathsf{pk}_{\mathsf{BS}}, m_0)$.
  • $(\beta_1, \mathsf{St}_{\mathcal{A}}) \leftarrow \mathcal{A}(\mathsf{param}_{\mathsf{BS}}, \mathsf{pk}_{\mathsf{BS}}, \mathsf{sk}_{\mathsf{BS}}, \rho_0)$.
  • $\sigma_0 \leftarrow \mathsf{BSRequest}^1(\beta_1, \mathsf{St}_R)$.
  • If $\sigma_0 = \perp$ or $\mathsf{BSVerify}(\mathsf{pk}_{\mathsf{BS}}, m_0, \sigma_0) = 0$ then abort.
  • $b \leftarrow \{0,1\}$.
  • If $b = 0$ then $\sigma_1 \leftarrow \mathsf{BSRandomize}(\sigma_0)$.
  • Else $(\rho_0, \mathsf{St}_R) \leftarrow \mathsf{BSRequest}^0(\mathsf{pk}_{\mathsf{BS}}, m_1)$
        $\beta_1 \leftarrow \mathsf{BSIssue}(\mathsf{sk}_{\mathsf{BS}}, \rho_0)$
        $\sigma_0 \leftarrow \mathsf{BSRequest}^1(\beta_1, \mathsf{St}_R)$.
  • $b^* \leftarrow \mathcal{A}(\mathsf{St}_{\mathcal{A}}, \pi(\sigma_0), \pi(\sigma_1))$.
  • Return 1 if $b = b^*$ else return 0.
```

Fig. 4. The partial weak blindness game.

In addition, randomizing a signature should result in a valid signature, i.e. for all parameter sets output by $\mathsf{Setup}_{\mathsf{BS}}$ and key pairs $(\mathsf{pk}_{\mathsf{BS}}, \mathsf{sk}_{\mathsf{BS}})$ output by $\mathsf{BSKeyGen}$ we have for all m and σ

$$\mathsf{BSVerify}(m, \sigma, \mathsf{pk}_{\mathsf{BS}}) = 1 \quad \Longrightarrow \quad \mathsf{BSVerify}(m, \mathsf{BSRandomize}(\sigma), \mathsf{pk}_{\mathsf{BS}}) = 1 \ .$$

For our first construction of DAA we need a standard-model signature which is secure under non-interactive assumptions. In addition, to work with our signatures of knowledge, it is required to be *structure-preserving*: the signatures and the messages it signs must be group elements and the verification equations must be pairing-product equations.

We use a signature scheme by Abe et al. [1] signing vectors of group elements, which we call AHO after its authors, presented in Fig. 3. (We have transposed $\mathbb{G}_1$ and $\mathbb{G}_2$, since our messages are in $\mathbb{G}_1$ rather than in $\mathbb{G}_2$ as in [1].) Its security relies on the q-SFP assumption (see Sect. 2.1). Abe et al. show that the six elements (R, S, T, U, V, W) of an AHO signature can be randomized. (We are not aware of a fully randomizable structure-preserving scheme based on non-interactive assumptions.)

This randomizability is useful, since we show that if the signer is given parts of a (partial) randomization of a signature he issued earlier then this does not reveal anything about the message. This means that, when used as a certificate for DAA, we only need to hide part of the certificate in a DAA signature to guarantee anonymity. We therefore relax the notion of weak blindness from [6] to *partial weak blindness* defined w.r.t. a projection function π. In the security game the signer gets the projection of a (partial) randomization of a signature and should not be able to tell whether it is from a signature he issued or a signature on a random message.

Definition 9 (Partial Weak Blindness). *A blind signature scheme consisting of* $(\mathsf{BSSetup}, \mathsf{BSKeyGen}, \langle\mathsf{BSRequest}, \mathsf{BSIssue}\rangle, \mathsf{BSRandomize}, \mathsf{BSVerify})$ *has (perfect)* partial weak blindness *with respect to a projection function π if no efficient (unbounded) adversary can win the game* $\mathsf{Exp}^{\text{part-w-blind}}_{\mathsf{BS},\mathcal{A}}$ *in Fig. 4 with advantage non-negligibly greater than* $1/2$.

We show that for an appropriate projection function the AHO signature is actually a partially weak blind signature, where $\mathsf{BSRequest}$ simply sends the message in the clear and $\mathsf{BSIssue}$ is defined as Sign.

Lemma 1. *The AHO signature scheme as a blind-signature scheme has partial weak blindness under the projection function* $\pi(Z, R, S, T, U, V, W) = (S, T, V, W)$.

Abe et al. [1] show that after a partial randomization of a signature, the components (R, S, T, U, V, W) are uniformly random conditioned on the signature satisfying the verification equations. The adversary in the game knows

```
BSSetup(1^λ)
    $\mathcal{P} \leftarrow \mathsf{BGrpSetup}(1^\lambda)$.
    $(\mathsf{crs}_1, \mathsf{xk}_1) \leftarrow \mathsf{GSSetup}(\mathcal{P})$.
    $(\mathsf{crs}_2, \mathsf{xk}_2) \leftarrow \mathsf{GSSetup}(\mathcal{P})$.
    Output $\mathsf{param}_{\mathsf{BS}} := (\mathcal{P}, \mathsf{crs}_1, \mathsf{crs}_2)$.

BSKeyGen(param_BS)
    $x, y \leftarrow \mathbb{Z}_p; X := [x]P_2; Y := [y]P_2$.
    Output $(\mathsf{sk}_{\mathsf{BS}} := (x, y), \mathsf{pk}_{\mathsf{BS}} := (X, Y))$.

BSRequest^0(pk_BS, (M_1, M_2))
    $\Omega_1 = \mathsf{GSProve}_{\mathsf{ZK}}(\mathsf{crs}_1, \{M_2\} : M_1 \in \mathcal{L}_1)$.
    Output $(\rho_0 := (M_1, \Omega_1), \mathsf{St}_R^0 := (M_1, M_2))$.

BSIssue^1(sk_BS, ρ_0)
    Parse $\rho_0$ as $(M_1, \Omega_1)$.
    If $\mathsf{GSVerify}(\mathsf{crs}_1, M_1 \in \mathcal{L}_1, \Omega_1) = 0$ then return $\perp$.
    $a \leftarrow \mathbb{Z}_p^\times;\ A := [a]P_1;\ B := [ay]P_1$
    $C := [ay]M_1;\ D := [ax]P_1 + [axy]M_1$
    $\Omega_2 \leftarrow \mathsf{GSProve}_{\mathsf{ZK}}(\mathsf{crs}_2, \{[ay]P_2\} : (B, C, M_1) \in \mathcal{L}_2)$.
    Output $\beta_1 := ((A, B, C, D), \Omega_2)$.

BSRequest^1(pk_BS, β_1, St_R^0)
    Parse $\beta_1$ as $((A, B, C, D), \Omega_2)$.
    Parse $\mathsf{St}_R^0$ as $(M_1, M_2)$.
    Return $\perp$ if any of the following hold:
      ◦ $A = 0$.
      ◦ $e(A, Y) \neq e(B, P_2)$.
      ◦ $e(D, P_2) \neq e(A + C, X)$.
      ◦ $\mathsf{GSVerify}(\mathsf{crs}_2, (B, C, M_1) \in \mathcal{L}_2, \Omega_2) = 0$.
    Output $\sigma \leftarrow \mathsf{BSRandomize}((A, B, C, D))$.

BSVerify(pk_BS, (M_1, M_2), (A, B, C, D))
    If $A = 0$ or $e(A, Y) \neq e(B, P_2)$
    or $e(C, P_2) \neq e(B, M_2)$
    or $e(D, P_2) \neq e(A + C, X)$ then return 0.
    Return 1.

BSRandomize(σ)
    Parse $\sigma$ as $(A, B, C, D)$.
    $r \leftarrow \mathbb{Z}_p^\times;\ A' := [r]A;\ B' := [r]B;$
    $C' := [r]C;\ D' := [r]D$.
    Return $(A', B', C', D')$.
```

Fig. 5. A RwBS construction from the NCL signature scheme

the original signature and gets (S, T, V, W) of a randomization thereof. Since for every tuple (Z, S, T, V, W) and for every message vector $\boldsymbol{M}$ there exist exactly one pair (R, U) that satisfies the verification equations, the message is information-theoretically hidden from the adversary.

In our DAA construction we can thus give the part (S, T, V, W) of a (randomized) certificate in the clear (and prove knowledge of the remaining components) and still satisfy anonymity.

4.2 A Fully Randomizable Weakly Blind Signature Scheme for Group Elements

In order to provide a more efficient DAA scheme, we give a fully randomizable weakly blind signature satisfying the original definition of [6]. Ghadafi [32] presents a signature scheme called NCL, which is effectively a structure-preserving variant of CL-signatures [18], based on a variant of the LRSW assumption [42], given in Sect. 2.1.

Messages of the scheme are pairs of the form $([m]P_1, [m]P_2) \in \mathbb{G}_1 \times \mathbb{G}_2$ (cf. Footnote 1, on p. 3). The secret and verification keys are of the form $(x, y) \in \mathbb{Z}_p^2$ and $([x]P_2, [y]P_2)$, respectively, and a signature on a message (M_1, M_2) is a 4-tuple

$$\big(A := [a]P_1,\ B := [y]A,\ C := [ay]M_1,\ D := [x](A + C)\big)$$

for a random $a \leftarrow \mathbb{Z}_p^\times$. The verification equations are $A \neq 0$, $e(B, P_2) = e(A, Y)$, $e(C, P_2) = e(B, M_2)$, $e(D, P_2) = e(A, X)\, e(C, X)$ and $e(M_1, P_2) = e(P_1, M_2)$.

A signature is fully randomizable by choosing $a' \leftarrow \mathbb{Z}_p^\times$ and setting $A' := [a']A$, $B' := [a']B$, $C' := [a']C$, $D' := [a']D$. Moreover, observe that to compute a signature on a message (M_1, M_2), only M_1 is required, whereas verification of the signature could be done using M_2 only. A first idea, which turns out to work, to construct a weakly blind signature from NCL, would be to define $\mathsf{BSRequest}$ as only sending M_1.

However, in the reduction of weak blindness, the simulator (playing the user) will not have M_2 (otherwise it could break the notion itself) and can therefore not verify the correctness of the adversary's signature. The signer must therefore make a NIZK of correctness of the signature. Moreover, in the reduction of blind-signature unforgeability to unforgeability of NCL signatures, the simulator (playing the signer) needs the full message (M_1, M_2) to query its signing oracle. In $\mathsf{BSRequest}$ the user makes thus a NIZK of knowledge of M_2 corresponding to M_1. The NIZKs use different CRSs (as we use different properties in the reductions) and can be efficiently implemented using Groth-Sahai proofs. We show that the resulting blind-signature scheme satisfies weak blindness under DDH in $\mathbb{G}_1$.

The construction is provided in Fig. 5 using the following languages:

$$\begin{aligned}\mathcal{L}_1 &: \{(M_1, M_2) : e(P_1, \underline{M_2}) = e(M_1, P_2)\} \\ \mathcal{L}_2 &: \{((B, C, M_1), B_2) : e(P_1, \underline{B_2}) = e(B, P_2) \wedge e(M_1, \underline{B_2}) = e(C, P_2)\}\end{aligned}$$

Theorem 2. *Under the SXDH and the DH-LRSW assumption the scheme in Fig. 5 is an unforgeable weakly blind signature scheme as defined in [6], with message space* $\mathcal{M} := \{([m]P_1, [m]P_2) \mid m \in \mathbb{Z}_p\}$.

Proof. The following two lemmas prove Theorem 2.

Lemma 2. *If the DH-LRSW assumption holds, the* GS *proof system used in* $\mathsf{Request}^0_{\mathsf{BS}}$ *is sound and the* GS *proof system used in* $\mathsf{Issue}^1_{\mathsf{BS}}$ *is zero-knowledge then the above scheme is unforgeable (as defined in [6]). More formally, if* $\mathcal{A}$ *is an adversary against the unforgeability of the scheme, then there exist adversaries* $\mathcal{B}_1, \mathcal{B}_2$ *and* $\mathcal{B}_3$ *against the DH-LRSW assumption, the soundness of the* GS *proof system used in* $\mathsf{Request}^0_{\mathsf{BS}}$ *and the zero-knowledge property of the* GS *proof system used in* $\mathsf{Issue}^1_{\mathsf{BS}}$*, respectively such that*

$$\mathsf{Adv}^{\text{forge}}_{\mathsf{BS},\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathsf{DH\text{-}LRSW}}_{\mathcal{B}_1}(\lambda) + \mathsf{Adv}^{\text{Soundness}}_{\mathsf{GS},\mathcal{B}_2}(\lambda) + \mathsf{Adv}^{\mathsf{ZK}}_{\mathsf{GS},\mathcal{B}_3}(\lambda) + \mathsf{Adv}^{\mathsf{SXDH}}_{\mathcal{B}_4}(\lambda) \ .$$

Proof. The GS CRS crs_1 is instantiated in the binding setting yielding perfectly sound proofs, whereas the CRS crs_2 is instantiated in the hiding setting yielding perfectly witness-indistinguishable/zero-knowledge proofs. By the perfect soundness of GS proofs in the binding setting we have that $\mathsf{Adv}^{\text{Soundness}}_{\mathsf{GS},\mathcal{B}_2}(\lambda) = 0$. Also, by the perfect zero-knowledge of GS proofs in the hiding setting we have that $\mathsf{Adv}^{\mathsf{ZK}}_{\mathsf{GS},\mathcal{B}_3}(\lambda) = 0$. Also, note that by the hardness of the SXDH assumption, the adversary has a negligible advantage in telling apart a binding CRS from a hiding one and therefore this only negligibly changes $\mathcal{B}_3$'s success probability. The details of the reduction to the security of the SXDH assumption is given in [37] and hence we skip it.

Now we turn to show how adversary $\mathcal{A}$ can be used to construct an adversary $\mathcal{B}_1$ which breaks the DH-LRSW assumption. Let (X, Y) be $\mathcal{B}_1$'s input which it passes to $\mathcal{A}$ as the public key of the scheme. The CRS crs_1 is chosen to be a binding CRS while crs_2 is chosen to be a hiding CRS. Algorithm $\mathcal{A}$ proceeds to make a series of oracle calls to the blind signature issuer. To obtain a valid tuple (A, B, C, D) for a challenge (M_1, Ω_1), algorithm $\mathcal{B}_1$ extracts the witness M_2 from the GS proof Ω_1 and then it forwards (M_1, M_2) to its DH-LRSW oracle to obtain the tuple (A, B, C, D). Algorithm $\mathcal{B}_1$ also uses the CRS crs_2 to produce a simulated proof Ω_2. By the perfect zero-knowledge property of GS proofs we have that real proofs are indistinguishable from simulated proofs. Algorithm $\mathcal{B}_1$ now returns (A, B, C, D, Ω_2) to $\mathcal{A}$.

Eventually, $\mathcal{A}$ will terminate with a tuple

$$(((M_{1,1}, M_{1,2}), \sigma_1), \ldots, ((M_{k+1,1}, M_{k+1,2}), \sigma_{k+1}))$$

for the forgery game. If $\mathcal{A}$ is successful in winning its game then all entries verify, and therefore $\Omega_{2,i}$ for $i = 1, \ldots, k+1$ are all correct DH-LRSW-tuples for the entries $(M_{i,1}, M_{i,2})$. The oracle was called at most k times (or $\mathcal{A}$ would not have won) yet there are $k+1$ distinct messages with valid signatures in $\mathcal{A}$'s output. By looking at the oracle log, we can identify a valid NCL signature that was never queried to the DH-LRSW oracle and output it. Therefore $\mathcal{B}_1$ breaks the DH-LRSW assumption with the same advantage as $\mathcal{A}$ has of creating a forgery. □

Lemma 3. *If the DDH assumption holds in group* $\mathbb{G}_1$*, the* GS *proof system used in* $\mathsf{Request}^0_{\mathsf{BS}}$ *is zero-knowledge and the* GS *proof system used in* $\mathsf{Issue}^1_{\mathsf{BS}}$ *is sound then the above scheme satisfies the weak blindness property (defined in [6]). More formally, if* $\mathcal{A}$ *is an adversary against the weak blindness property of the scheme, then there exist adversaries* $\mathcal{B}_1, \mathcal{B}_2$ *and* $\mathcal{B}_3$ *against the DDH assumption in group* $\mathbb{G}_1$*, the zero-knowledge property of the* GS *proof system used in* $\mathsf{Request}^0_{\mathsf{BS}}$ *and the soundness of the* GS *proof system used in* $\mathsf{Issue}^1_{\mathsf{BS}}$*, respectively such that*

$$\mathsf{Adv}^{\text{weak-blind}}_{\mathsf{BS},\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathsf{DDH}}_{\mathcal{B}_1}(\lambda) + \mathsf{Adv}^{\mathsf{ZK}}_{\mathsf{GS},\mathcal{B}_2}(\lambda) + \mathsf{Adv}^{\text{Soundness}}_{\mathsf{GS},\mathcal{B}_3}(\lambda) + \mathsf{Adv}^{\mathsf{SXDH}}_{\mathcal{B}_4}(\lambda) \ .$$

Proof. In the game, we choose crs_1 as a hiding CRS while the CRS crs_2 is chosen to be binding. By the perfect zero-knowledge property of GS proofs in the hiding setting we have that $\mathsf{Adv}^{\mathsf{ZK}}_{\mathsf{GS},\mathcal{B}_2}(\lambda) = 0$. Also, by the perfect soundness of GS proofs in the binding setting we have that $\mathsf{Adv}^{\mathsf{Soundness}}_{\mathsf{GS},\mathcal{B}_3}(\lambda) = 0$.

Also, by the hardness of the SXDH assumption, the adversary has a negligible advantage in telling apart a binding CRS from a hiding one and therefore this only negligibly changes $\mathcal{B}_2$'s success probability. Again, the details of the reduction to the security of the SXDH assumption is given in [37] and hence we skip it.

We now turn to show how adversary $\mathcal{A}$ against the weak blindness property, whose aim is to distinguish a rerandomisation of a signature he has seen earlier from a fresh signature on a new message, can be used to construct an adversary $\mathcal{B}_1$ that breaks the DDH assumption in group $\mathbb{G}_1$. Let $(R = [\alpha]P_1, S = [\beta]P_1, T = [\gamma]P_1)$ be the input to adversary $\mathcal{B}_1$, where (α, β) are independent uniform random elements of $\mathbb{Z}_p$, and either $\gamma = \alpha\beta$, or γ is also an independent uniform random element of $\mathbb{Z}_p$. The goal of algorithm $\mathcal{B}_1$ is to determine which of the two possibilities has been given to it.

Algorithm $\mathcal{B}_1$ picks a secret key $\mathsf{sk}_{\mathsf{BS}}$ for the blind signature scheme by selecting $x, y \leftarrow \mathbb{Z}_p$ and passes it to algorithm $\mathcal{A}$. Then algorithm $\mathcal{B}_1$ requests a blind signature on the message $(M_1, M_2) = ([\alpha]P_1, [\alpha]P_2)$, by passing the element R along with a simulated proof Ω_1 (since it does not know the component $[\alpha]P_2$) to $\mathcal{A}$ as the message ρ_0 in the Request phase. Algorithm $\mathcal{A}$ will then respond with a tuple $((A, B, C, D), \Omega_2)$. By checking the proof Ω_2 algorithm $\mathcal{B}_1$ is guaranteed that the value (A, B, C, D) returned is a valid signature on (M_1, M_2).

To produce the challenge for $\mathcal{A}$, algorithm $\mathcal{B}_1$ now forms the challenge

$$(A^*, B^*, C^*, D^*) := (S, [y]S, [y]T, [x]S + [xy]T)$$

and passes back to $\mathcal{A}$ the pair of tuples (A, B, C, D) and (A^*, B^*, C^*, D^*). Since (A, B, C, D) is a valid signature on (M_1, M_2), we claim that if $T = [\alpha\beta]P_1$, the challenge will be identically distributed to the weak blindness game where $b = 0$; whilst if $T = [\gamma]P_1$, the challenge will be identically distributed to the weak blindness game with $b = 1$. In other words, Algorithm $\mathcal{B}_1$ will solve DDH with essentially the same advantage as that of $\mathcal{A}$ against the weak blindness game. This is done as follows:

- If $\gamma = \alpha\beta$: In the weak blindness game, $\mathcal{A}$ first sees $[m]P_1$ for some uniformly random message m. In our game, he sees $[\alpha]P_1$ where α is uniformly random, hence the distribution is identical. Since we have a proof that (A, B, C, D) are computed correctly, there is some value $a \in \mathbb{Z}_p^*$ such that $A = [a]P_1$, $B = [ay]P_1$, $C = [ay\alpha]P_1$ and $D = [ax(1 + y\alpha)]P_1$.
  A re-randomization of (A, B, C, D) has the form $([r]A, [r]B, [r]C, [r]D)$ for some $r \leftarrow \mathbb{Z}_p$, whereas the tuple we sent was of the form $([\beta]P_1, [y\beta]P_1, [y\alpha\beta]P_1, [x\beta + xy\alpha\beta]P_1)$. We substitute $r = \beta/a$ in and note that because β is independent of everything else and uniformly random, r is also uniform in $\mathbb{Z}_p$. Thus, the challenge is also a correct NCL signature on the message $([\alpha]P_1, [\alpha]P_2)$. This exactly corresponds to case $b = 0$ of the weak-blindness game.
- If γ is random: Here we argue identically up to the point where we send our tuple. This will be $([\beta]P_1, [y\beta]P_1, [y\gamma]P_1, [x\beta + xy\gamma]P_1)$ and we substitute $\delta = \gamma/\beta$ getting $([\beta]P_1, [y\beta]P_1, [\beta y\delta]P_1], [\beta x(1 + y\delta)]P_1)$. Because γ is independent and uniformly distributed, δ is also uniformly distributed. Therefore, we have a correct NCL signature on the message $([\delta]P_1, [\delta]P_2)$ which corresponds to case $b = 1$.

4.3 Linkable Indistinguishable Tags

The second building block introduced to construct DAA schemes generically in [6] is a *Linkable Indistinguishable Tag* (LIT). These tags are similar to Message Authentication Codes (MACs); however, they have additional and stronger security requirements. LIT schemes are defined w.r.t. a one-way function $\mathsf{PK}()$ such that a tag created with a secret key sk can be verified given $\mathsf{PK}(\mathsf{sk})$ rather than sk. Thus, $\mathsf{PK}(\mathsf{sk})$ can be thought of as a public key for the tag. A LIT scheme is defined by the following algorithms. We have made one small change to the definition of [6] in that we allow the key generation algorithm to depend on global parameters, such as a bilinear group, and we generate these parameters with a new algorithm $\mathsf{GlobalSetup}$. The LITs presented in the cited paper actually also depend on such parameters.

- $\mathsf{GlobalSetup}$ generates global parameters par.

```
*Experiment*: $\mathsf{Exp}^{\text{w-}f\text{-IND}}_{\mathsf{LIT},\mathcal{A}}(\lambda)$
  • $\mathsf{par} \leftarrow \mathsf{GlobalSetup}(1^\lambda)$.
  • $\mathsf{sk}_0, \mathsf{sk}_1 \leftarrow \mathsf{LITKeyGen}(\mathsf{par})$.
  • $b \leftarrow \{0,1\}$.
  • $(m_1, \ldots, m_q, m^*, \mathsf{St}) \leftarrow \mathcal{A}_1(1^\lambda)$.
  • For $i = 1$ to $q$ do
      ◦ If $m_i = m^*$ then $\tau_i := \perp$
      ◦ Else $\tau_i \leftarrow \mathsf{LITTag}(\mathsf{sk}_0, m_i)$.
  • $\tau^* := \mathsf{LITTag}(\mathsf{sk}_b, m^*)$.
  • $b^* \leftarrow \mathcal{A}_2(\mathsf{St}, \mathsf{par}, f(\mathsf{sk}_0), \tau_1, \ldots, \tau_q, \tau^*)$.
  • Return 1 if $b^* = b$, else 0.
```

```
*Experiment*: $\mathsf{Exp}^{\text{w-LINK}}_{\mathsf{LIT},\mathcal{A}}(\lambda)$
  • $(\mathsf{sk}_0, m_0, \mathsf{sk}_1, m_1, \tau) \leftarrow \mathcal{A}(\mathsf{par})$.
  • Return 1 if and only if :
      ◦ $\mathsf{LITTag}(\mathsf{sk}_0, m_0) = \tau$.
      ◦ $\mathsf{LITTag}(\mathsf{sk}_1, m_1) = \tau$.
      ◦ Either ($\mathsf{sk}_0 = \mathsf{sk}_1$ and $m_0 \neq m_1$)
            or ($\mathsf{sk}_0 \neq \mathsf{sk}_1$ and $m_0 = m_1$).
```

Fig. 6. Security games for indistinguishability (left) and linkability (right) of LIT.

- $\mathsf{LITKeyGen}(\mathsf{par})$ takes global parameters par (such as a bilinear group) and outputs a secret key sk. (From now on we implicitly assume par.)
- $\mathsf{LITTag}(\mathsf{sk}, m)$ is deterministic, takes as input a secret key sk and message m, and outputs a tag τ.
- $\mathsf{LITVerify}(\mathsf{PK}(\mathsf{sk}), m, \tau)$ is given the image of sk under PK, a message m and a tag τ and checks whether τ is a valid tag on the message m w.r.t. sk, outputting 1 or 0.

Security. Besides correctness, [6] define the notions linkability and indistinguishability, of which we will only require relaxations. A LIT is *weakly linkable* if the following holds: if two tags are identical then they are either w.r.t. the same key and the same message, or both keys *and* both messages are different. In particular, two tags under different keys on the same message (or under the same key on different messages) must be different. Weak linkability was used implicitly in [6]. We formalize it by experiment $\mathsf{Exp}^{\text{w-LINK}}_{\mathsf{LIT},\mathcal{A}}(\lambda)$ in Fig. 6 and say a LIT scheme is *weakly linkable* if the advantage $\mathsf{Adv}^{\text{w-LINK}}_{\mathsf{LIT},\mathcal{A}}(\lambda) := \Pr[\mathsf{Exp}^{\text{w-LINK}}_{\mathsf{LIT},\mathcal{A}}(\lambda) = 1]$ is a negligible function of λ for any PPT adversary $\mathcal{A}$.

For a LIT *f-indistinguishability* is defined w.r.t. a one-way function f and states that no adversary, having access to a $\mathsf{LITTag}(\mathsf{sk}, \cdot)$ oracle, can distinguish a tag on a message of his choice (for which he did not query the oracle) from a tag produced under a different random key. This should hold even if the adversary is given the image $f(\mathsf{sk})$ of the secret key in question. We weaken this property by requiring that the adversary submit all the oracle queries and announce the message to be challenged on *before* seeing the parameters and the one-way image. This is formalized by $\mathsf{Exp}^{\text{w-}f\text{-IND}}_{\mathsf{LIT},\mathcal{A}}(\lambda)$ in Fig. 6 and a LIT scheme is *weakly f-indistinguishable* if $\mathsf{Adv}^{\text{w-}f\text{-IND}}_{\mathsf{LIT},\mathcal{A}}(\lambda) := \big|2{\cdot}\Pr[\mathsf{Exp}^{\text{w-}f\text{-IND}}_{\mathsf{LIT},\mathcal{A}}(\lambda) = 1] - 1\big|$ is negligible in λ for any polynomial-time adversary $\mathcal{A}$.

Small message spaces. If the size of the message space is polynomial in the security parameter then f-indistinguishability defined by [6] is implied by its weak version: assuming an adversary $\mathcal{A}$ breaking the standard notion, we construct an adversary $\mathcal{B}$ breaking the weak notion as follows: Let $\{m_1, \ldots, m_\ell\}$ be the message space, with $\ell = poly(\lambda)$. Then $\mathcal{B}$ randomly picks $i \leftarrow \{1, \ldots, \ell\}$ and outputs its queries and the challenge as $(m_1, \ldots, m_{i-1}, m_{i+1}, \ldots, m_\ell, m^* := m_i)$. With non-negligible probability $\mathcal{A}$ will ask to be challenged on m_i, in which case $\mathcal{B}$ can simulate all Tag queries and use $\mathcal{A}$ to break weak f-indistinguishability.

A Weak LIT in the Standard Model. The weak Boneh-Boyen signature scheme [9] was used in [29, 2] to construct Verifiable Random Functions (VRF) [43] for small message spaces under two variants of the DDHI assumption. The proof of pseudorandomness uses a technique similar to that for the unforgeability of weak Boneh-Boyen signatures in [9]: if the messages $m_1, \ldots, m_n$ on which the VRF is to be evaluated and the challenge m^* (whose VRF value is to be distinguished from random) are known in advance then given a DDHI instance, we can set up the VRF parameters and the public key so that we can (1) construct the VRF values on $m_1, \ldots, m_n$ and (2) use the DDHI challenge to construct a challenge for pseudorandomness for m^*. (Note that this effectively corresponds to a weak notion of pseudorandomness for VRFs where the adversary needs to announce all queries and the challenge message in advance.) This weak notion is then used to simulate the actual pseudorandomness game for VRFs with small domains

```
$\underline{\mathsf{LITKeyGen}(\mathcal{P})}$
    Return $\mathsf{sk} \leftarrow \mathbb{Z}_p$.

$\underline{\mathsf{LITTag}(\mathsf{sk}, m)}$
    If $m = -\mathsf{sk}$ then return $\perp$.
    Return $\tau := [\frac{1}{\mathsf{sk}+m}]P_1$.

$\underline{\mathsf{LITVerify}(\mathsf{PK}(\mathsf{sk}), m, \tau)}$
    If $e(\tau, \mathsf{PK}(\mathsf{sk}) + [m]P_2) = e(P_1, P_2)$
        then return 1, else return 0.
```

Fig. 7. The WBB-based Linkable Indistinguishable Tag (WBB-LIT).

using the ideas discussed in the note on small message spaces above. We define our LIT as the VRF from [2] and use the first part of their proof of pseudorandomness of VRFs to prove weak f-indistinguishability w.r.t. $f(\mathsf{sk}) := [\mathsf{sk}]P_1$.

Theorem 3. *The* WBB-LIT *in Fig. 7 is a LIT for* $\mathsf{PK}(\mathsf{sk}) := [\mathsf{sk}]P_2$. *It is weakly linkable and satisfies weak f-indistinguishability for $f(\mathsf{sk}) := [\mathsf{sk}]P_1$ if the DDHI assumption holds in group $\mathbb{G}_1$.*

Since it is impossible to have $\tau = \mathsf{LITTag}(\mathsf{sk}_0, m_0) = \mathsf{LITTag}(\mathsf{sk}_1, m_1)$ with ($\mathsf{sk}_0 = \mathsf{sk}_1$ and $m_0 \neq m_1$) or ($\mathsf{sk}_0 \neq \mathsf{sk}_1$ and $m_0 = m_1$), weak linkability holds unconditionally. Weak f-indistinguishability is proved analogously to the pseudorandomness property of the VRF under the q-DDHI assumption (see [29, Theorem 2]).

LIT vs. VRF. To construct a LIT which is not only *weakly* indistinguishable *and* which supports large domains, a natural approach would be to consider large-domain VRFs. There have been several such schemes in the recent literature including the cascade construction of Boneh et al. [12] to increase the domain of the Dodis-Yampolskiy VRF [29], the BCKL construction [2] applied to the Dodis-Yampolskiy VRF [29] and the Hohenberger-Waters VRF [38].

Unfortunately, all of the abovementioned schemes violate the weak linkability requirement for LITs, as it is easy to find for example $\mathsf{sk}_0 \neq \mathsf{sk}_1$ and m such that $\mathsf{LITTag}(\mathsf{sk}_0, m) = \mathsf{LITTag}(\mathsf{sk}_1, m)$. While in the random-oracle model pseudorandomness is trivial to achieve, resulting in a simple ROM-LIT construction in [6], it is not clear how to construct fully indistinguishable LITs for large basename spaces without resorting to interactive assumptions (see Sect. 7). VRFs are already hard to construct, but due to its strong security requirements (an adversary must be unable to find tag collisions even if he can choose both the keys and the messages) LITs seem even harder.

5 A Generic Construction of pre-DAA in the Standard Model

Our first construction of pre-DAA uses AHO signatures (Sect. 4.1) as partially weakly blind signatures, the VRF from [2] as LIT given in Fig. 7 (which has the form of weak Boneh-Boyen signatures [9]) and Waters signatures [46] implicitly for the signatures of knowledge, which are themselves Groth-Sahai proofs [36]. The setup algorithm outputs a bilinear group and parameters for the SoK (consisting of a CRS for GS proofs and parameters and a verification key for Waters signatures).

To generate a group key pair, the issuer creates an AHO signature key pair (gmsk, gmpk). To join a group, a user creates a LIT key sk and runs the AHO signature request algorithm from the last section on the message $F_1 := [\mathsf{sk}]P_1$; the issuer replies using the AHO signing algorithm to produce a credential cred on F_1.

To make a DAA signature on a message m under a basename bsn, a user first (partially) rerandomizes his AHO signature cred and then splits it into a public part cred_P and a part cred_H which he will include in the witness for the SoK. Next, he creates a LIT tag $\tau := \mathsf{LITTag}(\mathsf{sk}, \mathsf{bsn})$ using his key on the basename. He then computes a signature of knowledge Σ on the message $\mathsf{bsn}\|m$ proving knowledge of (a function of) a LIT key sk and the hidden part of an AHO signature cred_H such that the tag and the AHO signature both verify with respect to this key. The DAA signature is $\sigma := (\mathsf{cred}_P, \tau, \Sigma)$.

We formalize the above. The language of the SoK needs to be a pairing-product equation as in (1) with witnesses in $\mathbb{G}_1$ and $\mathbb{G}_2$. Rather than proving knowledge of sk, the witness will be $F_1 := [\mathsf{sk}]P_1$ and $F_2 := [\mathsf{sk}]P_2$. The AHO signature is on F_1 rather than sk so F_1 is also sufficient to verify it. The signature is split into a public part $\mathsf{cred}_P = \pi(\mathsf{cred}) = (S, T, V, W)$, with π as in Lemma 1, and a hidden part $\mathsf{cred}_H = (Z, R, U)$. We let $\mathsf{BSVerify}'(\mathsf{gmpk}, F_1, (\mathsf{cred}_H, \mathsf{cred}_P))$ denote the AHO verification algorithm of a split signature on F_1. On the other hand, F_2 is the public key for the LIT from Sect. 4.3, so τ can be verified using $\mathsf{LITVerify}(F_2, \mathsf{bsn}, \tau)$. It remains to prove that (F_1, F_2) is a Diffie-Hellman pair, that is the embedding of the same key into both groups. The language of the signature of knowledge is thus

$$\mathcal{L} : \big\{\big((\mathsf{gmpk}, \mathsf{cred}_P, \mathsf{bsn}, \tau), (F_1, F_2, \mathsf{cred}_H)\big) \ : e(-P_1, \underline{F_2})\, e(\underline{F_1}, P_2) = 1 \\ \wedge\ \mathsf{BSVerify}'(\mathsf{gmpk}, \underline{F_1}, (\underline{\mathsf{cred}_H}, \mathsf{cred}_P)) = 1 \wedge\ \mathsf{LITVerify}(\underline{F_2}, \mathsf{bsn}, \tau) = 1\big\}\ .$$

If $\mathsf{bsn} = \bot$ then the DAA signature (which should not link to anything) is $(\mathsf{cred}_P, \Sigma)$ where Σ is a SoK for the language

$$\mathcal{L}' : \ \big\{((\mathsf{gmpk}, \mathsf{cred}_P), (F_1, \mathsf{cred}_H)) \ : \ \mathsf{BSVerify}'(\mathsf{gmpk}, \underline{F_1}, (\underline{\mathsf{cred}_H}, \mathsf{cred}_P)) = 1\big\}\ .$$

To verify a DAA signature, one verifies the SoK. To link two signatures under the same basename, one compares the contained tags τ and returns 1 if they are equal; to identify a transcript given sk, one checks if the first message by the user is the value $[\mathsf{sk}]P_1$; and to identify a signature, one checks the LIT tag τ using sk and bsn.

Our construction follows closely the blueprint from [6] (except for proving knowledge of a function of sk and hiding parts of the certificate) and is proven analogously.

6 A More Efficient pre-DAA Scheme in the Standard Model

Here our goal is to give a truly efficient standard-model pre-DAA scheme. We first replace the partial RwBS from Sect. 4.1 by the full RwBS from Sect. 4.2; this avoids having to include parts of the certificate in the signatures of knowledge (SoK).

Moreover, as SoKs, due to their strong security requirements, are less efficient than standard proofs of knowledge (PoK), we should aim for using a simple Groth-Sahai PoK instead of a SoK in a DAA signature. The main obstacle is that the user secret key sk is used both for the tag on bsn and (implicitly in the SoK) for the signature on the message m. Suppose we replaced the SoK of (a function of) sk by a PoK thereof and a regular signature on m. Non-frameability corresponds to a forgery of a signature on m, to which the notion must be reduced. In the reduction we thus have to extract a signature from the PoK, and therefore cannot simulate proofs, as Groth-Sahai proofs only allow extraction *or* simulation (while SimExt security of SoKs allows both at the same time.) However, if we do not simulate the PoK then when answering DAA-signing queries, we need to provide actual tags—for which we do not have the user secret key.

We overcome this by using a novel approach: we use a signature scheme which, in the reduction, allows us to simulate tags under the same secret key and a tag scheme which allows us to simulate signatures. We do so by choosing the schemes in a way that tags of one scheme and signatures of the other scheme have the same form—although the security requirements are different, and are based on different assumptions. In particular, we make use of the fact that the values of the VRF from [2] are essentially "weak" signatures from [9]. (These signatures are only secure against *weak* chosen-message attacks, where the adversary has to make all queries before seeing the public key.) Weak signatures can easily be turned into standard signatures using a hybrid construction, where one signs a verification key of a one-time signature and uses the corresponding secret key to sign the actual message. Unlike for the message space of the LIT, there is no restriction on the message spaces of the signature schemes (and thus the message space of our DAA is big enough to sign messages of arbitrary length by using a collision-resistant hash function first).

We separate the domains for the messages of the weak signatures (i.e., verification keys of the one-time signature scheme) and the messages of the tags (i.e., the basenames) by prepending a bit to the messages. In the reduction of non-frameability to weak signature unforgeability we can then use our (weak) signing oracle to obtain signatures *and* to simulate the tags: The basename space is polynomial in size and the verification keys of the one-time signatures can be produced beforehand. Thus, before starting the simulation of the non-frameability game (and before obtaining the public key for the weak signature) we can make our "weak" signature queries on all basenames and the one-time verification keys.

Moreover, in the proof of anonymity we use the trick the other way round and simulate signatures using the tag oracle. In the reduction to weak f-unlinkability of the tags, we can make all tag queries upfront, since the basename space is polynomial and the one-time verification keys can be chosen beforehand. Another advantage of this approach is that, since weak signatures have the form of LITs, they are unlinkable to the key that produced them, which means that we can include the weak signatures in the clear in the DAA.

```
$\underline{\mathsf{Setup}(1^\lambda)}$
  • $(\mathcal{P}, \mathsf{crs}_1, \mathsf{crs}_2) \leftarrow \mathsf{BSSetup}(1^\lambda)$.
  • $\mathsf{param} := (\mathcal{P}, \mathsf{crs}_1, \mathsf{crs}_2)$.
  • Return $\mathsf{param}$.

$\underline{\mathsf{GKg}(\mathsf{param})}$
  • $(\mathsf{gmpk}, \mathsf{gmsk}) \leftarrow \mathsf{BSKeyGen}(\mathsf{param})$.
  • Return $(\mathsf{gmpk}, \mathsf{gmsk})$.

$\underline{\mathsf{UKg}(\mathsf{param})}$
  • $\mathsf{sk}_i \leftarrow \mathsf{LITKeyGen}(\mathcal{P})$.
  • Return $\mathsf{sk}_i$.

$\underline{\langle \mathsf{Join}, \mathsf{Iss} \rangle}$
  • Run $(\mathsf{BSRequest}, \mathsf{BSIssue})$
        for message $(f_1(\mathsf{sk}_i), f_2(\mathsf{sk}_i)) \in \mathcal{M}_{\mathsf{BS}}$.
  • User has input $((f_1(\mathsf{sk}_i), f_2(\mathsf{sk}_i)), \mathsf{gmpk})$.
  • Issuer has input $\mathsf{gmsk}$.
  • User's output is $\mathsf{gsk}_i = \mathsf{cred}$.

$\underline{\mathsf{GSig}(\mathsf{gsk}_i, \mathsf{sk}_i, m, \mathsf{bsn})}$
  • $\mathsf{cred} \leftarrow \mathsf{BSRandomize}(\mathsf{gsk}_i)$.
  • $(\mathsf{vk_{ots}}, \mathsf{sk_{ots}}) \leftarrow \mathsf{OTSKeyGen}(1^\lambda)$.
  • $\sigma_w \leftarrow \mathsf{BBSign}(\mathsf{sk}_i, 1||\mathsf{vk_{ots}})$.
  • If $\mathsf{bsn} \neq \perp$
      ◦ $\tau \leftarrow \mathsf{LITTag}(\mathsf{sk}_i, 0||\mathsf{bsn})$.
      ◦ $\varphi := (\mathsf{gmpk}, \mathsf{cred}, \mathsf{bsn}, \tau, \mathsf{vk_{ots}}, \sigma_w)$ .
      ◦ $\Sigma \leftarrow \mathsf{GSProve}(\mathsf{crs}_1, \{(f_1(\mathsf{sk}_i), f_2(\mathsf{sk}_i))\} : \varphi \in \mathcal{L})$.
  • Else
      ◦ $\tau := \emptyset$.
      ◦ $\varphi := (\mathsf{gmpk}, \mathsf{cred}, \mathsf{vk_{ots}}, \sigma_w)$.
      ◦ $\Sigma \leftarrow \mathsf{GSProve}(\mathsf{crs}_1, \{(f_1(\mathsf{sk}_i), f_2(\mathsf{sk}_i))\} : \varphi \in \mathcal{L}')$.
  • $\sigma_{\mathsf{ots}} \leftarrow \mathsf{OTSSign}(\mathsf{sk_{ots}}, (m, \tau, \mathsf{bsn}))$.
  • $\sigma := (\mathsf{cred}, \tau, \sigma_w, \mathsf{vk_{ots}}, \Sigma, \sigma_{\mathsf{ots}})$.

$\underline{\mathsf{GVf}(\mathsf{gmpk}, m, \mathsf{bsn}, \sigma)}$
  • Parse $\sigma$ as $(\mathsf{cred}, \tau, \sigma_w, \mathsf{vk_{ots}}, \Sigma, \sigma_{\mathsf{ots}})$.
  • If $\mathsf{OTSVerify}(\mathsf{vk_{ots}}, (m, \tau, \mathsf{bsn}), \sigma_{\mathsf{ots}}) = 0$, return 0.
  • If $\mathsf{bsn} \neq \perp$ then
      ◦ $\varphi := (\mathsf{gmpk}, \mathsf{cred}, \mathsf{bsn}, \tau, \mathsf{vk_{ots}}, \sigma_w)$.
      ◦ Return $\mathsf{GSVerify}(\mathsf{crs}_1, \varphi \in \mathcal{L}, \Sigma)$.
  • If $\tau = \emptyset$ then
      ◦ $\varphi := (\mathsf{gmpk}, \mathsf{cred}, \mathsf{vk_{ots}}, \sigma_w)$.
      ◦ Return $\mathsf{GSVerify}(\mathsf{crs}_1, \varphi \in \mathcal{L}', \Sigma)$.
  • Return 0.

$\underline{\mathsf{Identify_T}(\mathsf{gmpk}, \mathsf{sk}_i, \mathcal{T})}$
  • If $\mathcal{T}$ is a valid transcript then check if the user message
    in $\mathsf{Join}^0 = \mathsf{BSRequest}^0$ is $(f_1(\mathsf{sk}_i), \Omega)$, for some $\Omega$.
  • If so return 1, otherwise return 0.

$\underline{\mathsf{Identify_S}(\mathsf{gmpk}, \mathsf{sk}_i, m, \mathsf{bsn}, \sigma)}$
  • Parse $\sigma$ as $(\mathsf{cred}, \tau, \sigma_w, \mathsf{vk_{ots}}, \Sigma, \sigma_{\mathsf{ots}})$.
  • If $\mathsf{BSVerify}(\mathsf{gmpk}, (f_1(\mathsf{sk}_i), f_2(\mathsf{sk}_i)), \mathsf{cred}) = 0$
        then return 0.
  • If $\mathsf{OTSVerify}(\mathsf{vk_{ots}}, (m, \tau, \mathsf{bsn}), \sigma_{\mathsf{ots}}) = 0$
        then return 0.
  • Return 1 iff one of the following hold
      ◦ $\mathsf{bsn} = \perp, \tau = \emptyset$ and
        $\mathsf{BBVerify}(f_2(\mathsf{sk}_i), 1||\mathsf{vk_{ots}}, \sigma_w) = 1$.
      ◦ $\mathsf{bsn} \neq \perp$, $\mathsf{LITVerify}(f_2(\mathsf{sk}_i), 0||\mathsf{bsn}, \tau) = 1$
        and $\mathsf{BBVerify}(f_2(\mathsf{sk}_i), 1||\mathsf{vk_{ots}}, \sigma_w) = 1$.

$\underline{\mathsf{Link}(\mathsf{gmpk}, \sigma_0, m_0, \sigma_1, m_1, \mathsf{bsn})}$
  • If $\mathsf{GVf}(\mathsf{gmpk}, m_0, \mathsf{bsn}, \sigma_0) = 0$ return $\perp$.
  • If $\mathsf{GVf}(\mathsf{gmpk}, m_1, \mathsf{bsn}, \sigma_1) = 0$ return $\perp$.
  • If $\mathsf{bsn} = \perp$ return 0.
  • Parse $\sigma_0$ as $(\mathsf{cred}_0, \tau_0, \sigma_{w_0}, \mathsf{vk_{ots_0}}, \Sigma_0, \sigma_{\mathsf{ots}_0})$.
  • Parse $\sigma_1$ as $(\mathsf{cred}_1, \tau_1, \sigma_{w_1}, \mathsf{vk_{ots_1}}, \Sigma_1, \sigma_{\mathsf{ots}_1})$.
  • Return 1 if and only if $\tau_0 = \tau_1$.
```

Fig. 8. An efficient pre-DAA scheme construction in the standard model

Our construction (shown in Fig. 8) uses the LIT scheme from Fig. 7 and the RwBS scheme from Sect. 4.2. Unlike the construction in Sect. 5, it replaces the signatures of knowledge by a simple proof of knowledge and a (hybrid) signature in the clear. As in the generic scheme, the user obtains a credential gsk from the issuer. To make a DAA signature, the user randomizes gsk to cred, chooses a one-time signature key pair $(\mathsf{sk_{ots}}, \mathsf{vk_{ots}})$ and uses his secret key sk to produce a LIT tag τ on $0||\mathsf{bsn}$ (if $\mathsf{bsn} = \perp$ then $\tau := \emptyset$), and a weak signature σ_w on $1||\mathsf{vk_{ots}}$. He then produces a GS PoK Σ of $(f_1(\mathsf{sk}) := [\mathsf{sk}]P_1, f_2(\mathsf{sk}) := [\mathsf{sk}]P_2)$ showing it is well-formed, that cred is valid on it and that τ and σ_w verify under $f_2(\mathsf{sk})$. The DAA signature is then defined as $\sigma := (\mathsf{cred}, \tau, \sigma_w, \mathsf{vk_{ots}}, \Sigma, \sigma_{\mathsf{ots}})$, where σ_{ots} is a one-time signature produced with $\mathsf{sk_{ots}}$ on the tuple (m, τ, bsn). Note that if $\mathsf{vk_{ots}}$ does not lie in the message space of the LIT, we can use a collision-resistant hash function to hash it into that space. Moreover, note that the one-time signature also only needs to be weakly unforgeable, as the message (m, τ, bsn) is known before $\mathsf{vk_{ots}}$ is chosen.

The languages for the GS proofs in our construction are defined as follows, where $\mathcal{L}'$ is used when $\mathsf{bsn} = \perp$ and $\mathcal{L}$ otherwise.

$$\begin{aligned}\mathcal{L}:\ \big\{\big((\mathsf{gmpk}, \mathsf{cred}, \mathsf{bsn}, \tau, \mathsf{vk}_{\mathsf{ots}}, \sigma_w), (F_1, F_2)\big)\ &: e(-P_1, \underline{F_2}) \cdot e(\underline{F_1}, P_2) = 1\\ &\wedge\ \mathsf{BSVerify}(\mathsf{gmpk}, (\underline{F_1}, \underline{F_2}), \mathsf{cred}) = 1 \wedge\ \mathsf{LITVerify}(\underline{F_2}, 0||\mathsf{bsn}, \tau) = 1\\ &\wedge\ \mathsf{BBVerify}(\underline{F_2}, 1||\mathsf{vk}_{\mathsf{ots}}, \sigma_w) = 1\big\}\\ \mathcal{L}':\ \big\{\big((\mathsf{gmpk}, \mathsf{cred}, \mathsf{vk}_{\mathsf{ots}}, \sigma_w), (F_1, F_2)\big)\ &: e(-P_1, \underline{F_2}) \cdot e(\underline{F_1}, P_2) = 1\\ &\wedge\ \mathsf{BSVerify}(\mathsf{gmpk}, (\underline{F_1}, \underline{F_2}), \mathsf{cred}) = 1 \wedge\ \mathsf{BBVerify}(\underline{F_2}, 1||\mathsf{vk}_{\mathsf{ots}}, \sigma_w) = 1\big\}\end{aligned}$$

6.1 Security of the Construction

Theorem 4. *If the RwBS scheme is unforgeable and weakly blind, the LIT scheme is weakly linkable and weakly f-indistinguishable, the Groth-Sahai proof system is sound and zero-knowledge, and the one-time signature scheme is weakly unforgeable then the construction in Fig. 8 is a secure pre-DAA scheme.*

Proof. We first show correctness and that the scheme has uniquely identifiable transcripts.

Correctness. This follows from the correctness of the RwBS, the LIT and the one-time signature schemes and the perfect correctness of the GS proof system.

Uniquely Identifiable Transcripts. Given a secret key sk, we define Check_T to check whether the user's first message is $(f_1(\mathsf{sk}), \Omega)$, for some Ω. The injectivity of f_1 ensures uniqueness.

Anonymity. We prove the following lemma.

Lemma 4. *If the blind signature scheme (used in the* Join/Issue *protocol) is weakly blind, the GS proof system is zero-knowledge, and the LIT is weakly f-indistinguishable then our pre-DAA scheme has the anonymity property.*

Proof. We show that any efficient adversary with a non-negligible advantage in the anonymity game can be used to break the zero-knowledge property of GS proofs, the weak-blindness of the blind signature scheme, or the indistinguishability of the LIT. For $b = 0$ and $b = 1$, we define a sequence of six games, where Game 1 is $\mathsf{Exp}_{\mathcal{A}}^{\text{anon-}b}(\lambda)$ when the experiment guesses correctly the challenge user, and Game 6 is independent of b. If we then prove that $\mathcal{A}$ behaves differently in two consecutive games only with negligible probability, we have that

$$\left|\Pr[\mathsf{Exp}_{\mathcal{A}}^{\text{anon-1}}(\lambda) = 1] - \Pr[\mathsf{Exp}_{\mathcal{A}}^{\text{anon-0}}(\lambda) = 1]\right|$$

is negligible, and thus the scheme satisfies anonymity.

We start by showing that if for an adversary $\mathcal{A}$ the winning advantage is non-negligible, then this is still the case if the game aborts if the challenge user chosen by $\mathcal{A}$ is different from a random user which the game has chosen beforehand. Let $q_{\mathcal{A}}$ be the maximum number of users adversary $\mathcal{A}$ can create in the game. We randomly pick $i \in \{1, \ldots, q_{\mathcal{A}}\}$. Since i is independently chosen, the probability that i equals a particular user is $\frac{1}{q_{\mathcal{A}}}$. Thus, we have

$$\begin{aligned}\mathsf{Adv}_{\mathcal{A}}^{\text{anon}}(\lambda) &= \left|\Pr[\mathsf{Exp}_{\mathcal{A}}^{\text{anon-1}}(\lambda) = 1] - \Pr[\mathsf{Exp}_{\mathcal{A}}^{\text{anon-0}}(\lambda) = 1]\right|\\ &= \left|\Pr[\mathsf{Exp}_{\mathcal{A}}^{\text{anon-1}}(\lambda) = 1 \,|\, i_1 = i] \cdot \Pr[i_1 = i] - \Pr[\mathsf{Exp}_{\mathcal{A}}^{\text{anon-0}}(\lambda) = 1 \,|\, i_0 = i] \cdot \Pr[i_0 = i]\right|\\ &= \tfrac{1}{q_{\mathcal{A}}} \cdot \left|\Pr[\mathsf{Exp}_{\mathcal{A}}^{\text{anon-1}}(\lambda) = 1 \,|\, i_1 = i] - \Pr[\mathsf{Exp}_{\mathcal{A}}^{\text{anon-0}}(\lambda) = 1 \,|\, i_0 = i]\right|\ ,\end{aligned}$$

which proves the following:

Claim 1. *If $\mathcal{A}$ has non-negligible advantage in winning the anonymity game then the probability that $\mathcal{A}$ wins the game and that the user i_b in the call to CH_b is the randomly drawn user i is non-negligible.*

By contraposition, to prove anonymity of the scheme, it suffices to show that the above conditional probabilities are close. To do so, we fix $b = 0$ or $b = 1$ and define a sequence of games arguing that $\mathcal{A}$'s behaviour changes only negligibly from one game to the next one. The last game will be independent of b; so overall we will have shown that $\mathcal{A}$'s advantage in winning the anonymity game is negligible.

Note that when the adversary calls the challenge oracle, it gets a signature $\sigma = (\mathsf{cred}, \tau, \sigma_w, \mathsf{vk}_{\mathsf{ots}}, \Sigma, \sigma_{\mathsf{ots}})$, where τ and σ_w are a LIT and a weak signature, both under key sk_{i_b}, and cred is a blind signature on $(f_1(\mathsf{sk}_{i_b}), f_2(\mathsf{sk}_{i_b}))$. In our sequence of games, we could thus first replace Σ by a simulated GS proof, and simulate the join protocol for our target user i. Using now the fact that the weak signature σ_w has the same form as a tag, we could then replace τ and σ_w (interpreted as tag) by tags under random keys (by indistinguishability of the LIT), and finally replace cred by a signature on a random message (by the weak blindness of the RwBS). This last game would then not involve sk_{i_b} anymore and would thus be independent of the bit b.

It is in the last step that the intuition fails unfortunately, due to the fact that the blind signature is only *weakly* blind. The messages to be signed in the weak-blindness game are randomly chosen by the experiment (rather than by the adversary as for the standard blindness notion). This means that in the last step of the sequence of games, the simulator does not have the secret key of user i (sk_i is implicitly chosen by the weak-blindness challenger). As the simulator must also answer signing queries on behalf of user i, these must be simulated without knowledge of the secret key as well. We thus first have to replace *all* the LITs τ (and weak signatures σ_w) produced by user i (and not only those contained in the challenge signature) by LITs (and weak signatures) under random keys in the previous game. We make the hybrid steps precise in the following:

Game 1. Before running the game, we pick i uniformly from $\{1, \ldots, q_{\mathcal{A}}\}$ and abort if in $\mathcal{A}$'s challenge-oracle call we have $i_b \neq i$.

Game 2. We act as in Game 1 but we replace crs_1 with a hiding CRS as output by $\mathsf{GSSimSetup}$.

By the security of GS proofs (i.e. the SXDH assumption) the difference between Game 1 and 2 is negligible. Note that crs_1 is the CRS used for the proofs constructed in GSig. Moreover, setting crs_1 to enable simulation is what is needed in the proof of weak blindness of our blind signature scheme.

Game 3. In this game, when the adversary requests user i to join, we use $f_1(\mathsf{sk}_i)$ and simulate Ω_2, that is the proof of knowledge of $f_2(\mathsf{sk}_i)$. Moreover, whenever a signature is requested from user i (via the Sign or the CH_b oracles), we simulate the proof Σ contained in it.

Both proofs Ω_2 and Σ are under crs_1, which allows simulation from Game 2 on, by using the trapdoor output when running $(\mathsf{crs}_1, \mathsf{tr}) \leftarrow \mathsf{GSSimSetup}$. Under simulated CRSs, simulated proofs and real proofs are distributed identically. Game 2 is thus information-theoretically indistinguishable from Game 3.

Game 4. Same as Game 3, except that whenever the experiment creates a DAA signature on behalf of user i (i.e. when $\mathcal{A}$ calls CH_b or Sign for user i), we do the following: if the basename bsn has already been queried, we use the same tag τ as in the previous query; if not, we pick independent uniformly random keys and produce the tags τ using these random keys.

It is the hop from Game 3 to Game 4 that is the main difference to the proof for the random-oracle scheme in [6]: Interpreting σ_w as a second LIT, the DAA signatures now contain *two* tags, and these tags are only *weakly* f-indistinguishable (WfI), meaning that the messages to be queried and the challenge have to be announced beforehand. When replacing a tag τ we also have to simulate the tags σ_w (as the simulator in the reduction to WfI does not have the secret key to produce σ_w), and we have to output all messages to be tagged and the challenge message in advance.

Games 3 and 4 are shown to be negligibly close by a hybrid argument and reduction to the WfI of the LIT. Let $s_{\mathcal{A}}$ be a (polynomial) upper bound on the number of Sign and CH_b queries that $\mathcal{A}$ makes. We define a sequence of games $(G_j)_{j=0}^{s_{\mathcal{A}}}$ such that in G_j, we answer the first j queries for DAA signatures (Sign or CH_b) on *different* basenames for user i by using the secret key sk_i (that was used in the Join protocol) for the LIT tag τ; from query $j + 1$ on we use independent random keys for τ. The tags σ_w are constructed correctly. This construction gives us $G_{s_{\mathcal{A}}} =$ Game 3 and $G_0 =$ Game 4, and the difference between any two consecutive games lies in a single construction of a LIT tag.

We now construct an adversary $\mathcal{B}$ that breaks weak f-indistinguishability (defined in Fig. 6) if there is a non-negligible difference between G_{j-1} and G_j. $\mathcal{B}$ has to output all messages it wants tags on and the one it wants to be challenged on in advance. $\mathcal{B}$ first computes $s_{\mathcal{A}}$ key pairs $((\mathsf{vk_{ots}})_i, (\mathsf{sk_{ots}})_i)$ for the one-time signature scheme. Let the set $\{\mathsf{bsn}_1, \ldots, \mathsf{bsn}_n\}$ be the (polynomial-sized) set of possible basenames. $\mathcal{B}$ picks a random $k \leftarrow \{1, \ldots, n\}$ and outputs the following messages: $0\|\mathsf{bsn}_1, \ldots, 0\|\mathsf{bsn}_{k-1}, 0\|\mathsf{bsn}_{k+1}, \ldots, 0\|\mathsf{bsn}_n, 1\|(\mathsf{vk_{ots}})_1, \ldots, 1\|(\mathsf{vk_{ots}})_{s_{\mathcal{A}}}$; and as challenge message: $0\|\mathsf{bsn}_k$. The challenger then sends $\mathcal{B}$ the following: the parameters $\mathcal{P}$, $f_1(\mathsf{sk}_0) := [\mathsf{sk}_0]P_1$ and the tags $\tau_1, \ldots, \tau_n$ on the basenames and tags $\sigma_{w1}, \ldots, \sigma_{ws_{\mathcal{A}}}$ on the verification keys, all under sk_0, as well as the challenge tag τ^* on $0\|\mathsf{bsn}_k$.

Adversary $\mathcal{B}$ then simulates the anonymity game for $\mathcal{A}$ using $f_1(\mathsf{sk}_0)$ as the blinded secret key for user i in the Join protocol. Moreover, $\mathcal{B}$ uses the tags σ_w to simulate the weak signature in all queries and the tags τ_i from the first query up to query number $j-1$. In the j'th query it does the following: If $\mathcal{A}$ does not query a DAA signature for basename bsn_k then $\mathcal{B}$ aborts. (Note that $\mathcal{B}$ continues with non-negligible probability.) Otherwise, it uses τ^* from the challenger. For all remaining queries (for $i = j+1$ to $s_{\mathcal{A}}$), $\mathcal{B}$ produces tags τ_i' under randomly chosen keys, but uses σ_{wi} given by the challenger. It completes the construction of the DAA signature by simulating the proof of knowledge Σ and using the secret key $\mathsf{sk_{ots}}$ to compute a one-time signature σ_{ots}.

If the challenger's bit $b = 0$ then τ^* was produced under the same key sk_0 as tags $\tau_{1,i}, \ldots, \tau_{1,j-1}$; $\mathcal{A}$ is thus playing game G_j. Otherwise, τ^* was for a random key and $\mathcal{A}$ is playing Game G_{j-1}. Thus, by weak f-indistinguishability of the LIT scheme, the difference between two games is negligible and therefore so is the difference between Games 3 and 4.

Game 5. Same as Game 4, except that in all queries to CH_b or Sign for user i we also replace the signatures/tags σ_w by tags under independently uniformly random keys.

The fact that Games 4 and 5 are indistinguishable is shown analogously to the hop from Game 3 to Game 4, that is, by a hybrid reduction to weak f-indistinguishability of the LIT. Note that this time, we do not have to guess on which message the challenge LIT will be, since it is the experiment that chooses the keys $\mathsf{vk_{ots}}$ (which are the messages for σ_w) and the order in which they are used.

Game 6. The difference between this game and the previous one is that after running Join for user i, we discard the obtained gsk_i and replace it with a blind signature on a random secret key. Observe that the game is now independent of the bit b, as the challenge signature is independent from gsk_i and sk_i.

If the difference between Games 5 and 6 was non-negligible, we could build an adversary $\mathcal{B}$ that breaks weak blindness of the RwBS scheme as follows. Adversary $\mathcal{B}$ receives param, $\mathsf{pk_{BS}}$ and $\mathsf{sk_{BS}}$ from its challenger and hands it to $\mathcal{A}$ as $\mathsf{gmpk} := \mathsf{pk_{BS}}$ and $\mathsf{gmsk} := \mathsf{sk_{BS}}$. It simulates Game 5 for $\mathcal{A}$, except when joining user i, it relays $\mathcal{A}$'s messages impersonating the issuer to its challenger. After obtaining a signature cred from the challenger, $\mathcal{B}$ sets $\mathsf{gsk}_i := \mathsf{cred}$ and continues the simulation until $\mathcal{A}$ outputs d, which $\mathcal{B}$ returns to its challenger.

If the challenger's bit in the weak-blindness game was 0 then gsk_i is a randomization of the signature that $\mathcal{A}$ issued; $\mathcal{A}$ plays thus Game 5. If the bit was 1 then gsk_i is set to a signature on a random message, i.e. a random user key, which means that $\mathcal{A}$ is playing Game 6.

Note that we could not have replaced the blind signature in an earlier game, since it is only weakly blind: the blindness adversary does not get to see the message, so it could not simulate the anonymity game if the user i's key was used elsewhere in the experiment.

We will now prove that our scheme satisfies traceability. We deal with the two ways an adversary could brake this notion separately.

Traceability Game 1. To win this game, the adversary must output a message, a basename and a signature that verifies on the latter, as well as a collection of secret keys such that all transcripts accepted by the honest issuer identify to one of these keys; however the signature does not identify with any of them (and is thus untraceable). Intuitively, the adversary can win this game is by either forging the blind signature used in the $\mathsf{Join}/\mathsf{Issue}$ protocol or by breaking the soundness of the GS proof system by producing a proof for a fake statement. Since GS proofs are perfectly sound in the binding setting (which is used in Setup), it must be done by a forgery. We will use adversary $\mathcal{A}$ against the first traceability notion to construct an adversary $\mathcal{B}_1$ that wins the unforgeability game of the weak blind signature as follows:

Adversary $\mathcal{B}_1$ receives the public key $\mathsf{pk}_{\mathsf{BS}}$ from the challenger in the blind-signature unforgeability game and passes it to $\mathcal{A}$ as gmpk. The common reference string crs_1 for the GS proof system (used to produce Σ) is chosen to be a binding one, so we can extract the witness from Σ. In order to win the unforgeability game, we need to output $q+1$ signatures on different messages after querying our oracle only q times. The traceability adversary however does not have a restriction on oracle calls, in particular, he could (according to the instantiation) register one user multiple times. Now due to randomizability, we can derive many signatures for the same message from one signature, which will allow us not to "waste" oracle queries.

$\mathcal{B}_1$ receives from its challenger the parameters and a public key, which it forwards to $\mathcal{A}$ as gmpk. $\mathcal{B}_1$ simulates issuing queries as follows: if $\mathcal{A}$'s message (F_1, Ω_1), is such that F_1 was not contained in any previous query, it forwards (F_1, Ω_1) to its $\mathsf{BSIssue}$ oracle and forwards the response (A, B, C, D, Ω_2) to $\mathcal{A}$. If F_1 has already been queried, it randomizes the oracle's response it got when it queried it. Note that it cannot adapt Ω_2 to a randomization, which is why this proof has to be zero-knowledge: In the simulation $\mathcal{B}_1$ sets up the parameters such that crs_2 allows simulating Ω_2. (This was formalized in [6] as requiring "*issuer simulatability*" from the blind signature scheme.)

Let $(\sigma, m, \mathsf{bsn}, \mathsf{sk}'_1, \ldots, \mathsf{sk}'_\ell)$ be $\mathcal{A}$'s output where $\sigma = (\mathsf{cred}, \tau, \sigma_w, \mathsf{vk}_{\mathsf{ots}}, \Sigma, \sigma_{\mathsf{ots}})$. Since σ is valid on m and bsn, by the soundness of Σ, we can extract $f_1(\mathsf{sk}^*)$ and $f_2(\mathsf{sk}^*)$ on which cred is a valid blind signature and for which τ and σ_w are valid on bsn (if $\mathsf{bsn} \neq \perp$) and $\mathsf{vk}_{\mathsf{ots}}$, respectively. We have thus $\mathsf{Identify}_{\mathsf{S}}(\sigma, m, \mathsf{bsn}, \mathsf{sk}^*) = 1$. Since $\mathsf{Identify}_{\mathsf{S}}$ outputs 0 for all sk'_i, we have $f_1(\mathsf{sk}^*) \neq f_1(\mathsf{sk}'_i)$ for all $1 \leq i \leq \ell$. Adversary $\mathcal{B}_1$ returns $(f_1(\mathsf{sk}^*), f_2(\mathsf{sk}^*))$ and cred along with the $\mathcal{A}$'s ℓ different issue queries $(f_1(\mathsf{sk}'_i), f_2(sk'_i))$ (where $f_2(sk'_i)$ is extracted from Ω_1, which is perfectly sound) together with the responses of its $\mathsf{BSIssue}$ oracle. These are thus $\ell + 1$ different messages and signatures on it, whereas $\mathcal{B}_1$ only made ℓ oracle queries.

By the unforgeability of the weak blind signature scheme, we have that the probability of this happening is negligible and therefore the pre-DAA scheme satisfies the definition of Traceability 1.

Traceability Game 2. In this game a successful output $(\sigma_0, m_0, \sigma_1, m_1, \mathsf{bsn}, \mathsf{sk}')$ must satisfy the following: σ_0 is valid on m_0 and bsn, σ_1 is valid on m_1 and bsn, both signatures are identified with sk', but Link on them outputs 0. For $b = 0, 1$, let τ_b be the tag (on bsn) contained in σ_b. Since winning the game implies $\mathsf{bsn} \neq \perp$ and both signatures identify with sk', we have $\mathsf{LITTag}(\mathsf{sk}', 0\|\mathsf{bsn}) = \tau_0$ and $\mathsf{LITTag}(\mathsf{sk}', 0\|\mathsf{bsn}) = \tau_1$ (by definition of $\mathsf{Identify}_{\mathsf{S}}$), thus $\tau_0 = \tau_1$. On the other hand, since σ_0 and σ_1 do not link and the first three conditions in the definition of Link (see Fig. 8) are not satisfied we must have $\tau_0 \neq \tau_1$. Together, this is a contradiction.

Non-Frameability Game 1. This is the notion whose proof substantially deviates from that of the ROM pre-DAA scheme in [6] and our first scheme (Sect. 5), which both made use of the fact that we can simulate and extract from a signature of knowledge at the same time. Using Groth-Sahai proofs directly, we cannot simulate anymore, since we need to extract a forgery from a successful adversary's output. We thus deviate from the black-box proof in [6] and directly use the fact that our tags are weak Boneh-Boyen signatures, and can thus be obtained from a signing oracle. We give the details:

To win this game, the adversary must output a tuple $(\sigma, i, m, \mathsf{bsn})$ such that $\sigma = (\mathsf{cred}, \tau, \sigma_w, \mathsf{vk}_{\mathsf{ots}}, \Sigma, \sigma_{\mathsf{ots}})$ is valid on m and bsn, and it identifies with honest user i's key, although that user never produced a signature on m and bsn. The adversary impersonates the issuer and has at his disposal oracles to join honest users, query signatures from them, and obtain their secret keys making them dishonest.

By perfect soundness of GS proofs (since crs_1 is set up as a binding CRS), from an adversary's output we can extract the witness used for Σ contained in σ. Let $u(\lambda)$ be the maximum number of honest users adversary $\mathcal{A}$ creates in the game (for some polynomial u). We randomly pick a user $i \leftarrow \{1, \ldots, u(\lambda)\}$ and abort if the "framed" user is not this user i. We have a probability of $\frac{1}{u(\lambda)}$ of guessing the correct user.

We first show that by unforgeability of the one-time signature scheme, with overwhelming probability $\mathcal{A}$ does not reuse a value $\mathsf{vk}_{\mathsf{ots}}$ contained in a reply from the Sign oracle: if it does then it must forge a signature σ_{ots} under $\mathsf{vk}_{\mathsf{ots}}$, since it cannot reuse the signature given to it in the response containing $\mathsf{vk}_{\mathsf{ots}}$. Indeed, let the given one-time signature's message be $M' := (m', \tau', \mathsf{bsn}')$, and let the message of the one contained in the adversary's output is $M := (m, \tau, \mathsf{bsn})$. We show that $M' \neq M$. Suppose $M = M'$, thus $m = m'$. Moreover, $\mathsf{bsn} = \mathsf{bsn}'$ and $\tau = \tau'$, which (by linkability of the LIT) is only possible if the used secret keys to produce the tags are equal. This now means that $\mathsf{sk} = \mathsf{sk}'$, meaning the framed user is the same as user i for which the signature was queried. $\mathcal{A}$ has thus made a query (i, m, bsn) to its Sign oracle, and can thus not have won the non-frameability game 1. Moreover, the one-time signature only needs to be weakly secure, since in the reduction M' (which the simulator must query to its signing oracle) is known before $\mathsf{vk_{ots}}$ is selected, meaning the simulator can make a weak chosen-message query.

It remains to show that $\mathcal{A}$ also has a negligible advantage in winning the game by using a new key $\mathsf{vk_{ots}}$. Recall that the space of basenames is a polynomial-size set $\{\mathsf{bsn}_1, \ldots, \mathsf{bsn}_n\}$. We construct an adversary $\mathcal{B}$ against weak unforgeability of Boneh-Boyen signatures. Let q be the maximum number of Sign queries for user i that $\mathcal{A}$ makes in the game. Adversary $\mathcal{B}$ chooses q different one-time-signature key pairs $((\mathsf{sk_{ots}})_j, (\mathsf{vk_{ots}})_j)$ for $j = 1, \ldots, q$ and sends the following messages to its challenger to be signed: $0\|\mathsf{bsn}_1, \ldots, 0\|\mathsf{bsn}_n, 1\|(\mathsf{vk_{ots}})_1, \ldots, 1\|(\mathsf{vk_{ots}})_q$. $\mathcal{B}$ receives the public key $(f_1(\mathsf{sk}), f_2(\mathsf{sk}))$ and the signatures on the requested messages. $\mathcal{B}$ can now use these signatures to form the tags τ and the weak signatures σ_w in the DAA signatures it must simulate for user i. Moreover, it uses $(f_1(\mathsf{sk}), f_2(\mathsf{sk}))$ as witness to produce an honest proof Σ and $\mathsf{sk_{ots}}$ to make a one-time signature, which completes the simulation of a DAA signature. $\mathcal{B}$ can thus answer any Sign queries on behalf of user i.

If $\mathcal{A}$ is successful and has used a $\mathsf{vk_{ots}}$ that was not used before then σ_w is a forgery: it is valid on $1\|\mathsf{vk_{ots}}$ and $1\|\mathsf{vk_{ots}}$ was not queried to the oracle: all keys are different by assumption, and all messages regarding the basenames start with 0. Together we have that our pre-DAA scheme satisfies the definition of Non-frameability 1.

Non-Frameability Game 2. We argue that it even if the adversary is computationally unbounded, it is impossible for it to win the second non-frameability game against our pre-DAA scheme construction using the WBB-LIT scheme and Groth-Sahai proofs.

The adversary wins the game if it can produce two signatures that link with respect to one of two given basenames and he can demonstrate that they should not have linked due to one of three conditions: a signature verifies to an empty basename, the signatures verify with respect to different basenames or the signatures do not both trace to the same key.

From the fact that the two provided signatures link with respect to some basename we can deduce that they also verify with respect to this basename as our linking algorithm outputs $\perp$ if they do not.

Let $(\sigma_0, m_0, \mathsf{bsn}_0, \sigma_1, m_1, \mathsf{bsn}_1, \mathsf{sk})$ be a winning adversary's output. We have that $\mathsf{GVf}(\mathsf{gmpk}, \sigma_0, m_0, \mathsf{bsn}_0) = 1$, $\mathsf{GVf}(\mathsf{gmpk}, \sigma_1, m_1, \mathsf{bsn}_1) = 1$ and that $\mathsf{Link}(\mathsf{gmpk}, \sigma_0, m_0, \sigma_1, m_1, \mathsf{bsn}) = 1$ for $\mathsf{bsn} = \mathsf{bsn}_0$ or $\mathsf{bsn} = \mathsf{bsn}_1$. We investigate all possible winning conditions:

- Case $\mathsf{bsn}_0 = \perp$: We have that $\mathsf{GVf}(\mathsf{gmpk}, \sigma_0, m_0, \mathsf{bsn}_0) = 1$, so the tag τ in σ_0 must be $\emptyset$. But then the Link algorithm will reject if $\mathsf{bsn} = \perp$ so it means that we must have $\mathsf{bsn} = \mathsf{bsn}_1 \neq \perp$ but σ_0 will not verify to a non-empty basename when its LIT tag is $\tau = \emptyset$ (by the definition of the GVf algorithm). Therefore, this is a contradiction. The same holds for $\mathsf{bsn}_1 = \perp$.
- Case $\mathsf{bsn}_0 \neq \mathsf{bsn}_1$, $\mathsf{bsn}_0 \neq \perp$ and $\mathsf{bsn}_1 \neq \perp$: Here we have that both signatures verify with respect to their distinct basenames and link with respect to one basename $\mathsf{bsn} \in \{\mathsf{bsn}_0, \mathsf{bsn}_1\}$. This means that one of the two signatures verifies with respect to both basenames.
  Let τ_i for $i \in \{0, 1\}$ be the tag in this signature. Since the signature verifies correctly and the Groth-Sahai proofs in this setting are perfectly sound, we must have that $\tau_i = \left[\frac{1}{\mathsf{sk}+(0\|\mathsf{bsn}_0)}\right] P_1 = \left[\frac{1}{\mathsf{sk}+(0\|\mathsf{bsn}_1)}\right] P_1$. By the deterministic nature of the WBB-LIT scheme, this implies $\mathsf{bsn}_0 = \mathsf{bsn}_1$, which again is a contradiction.
- Case only one of the signatures identifies to sk: We assume w.l.o.g. that $\mathsf{bsn}_0 = \mathsf{bsn}_1 \neq \perp$ (otherwise, we are in one of the above cases). The two signatures link with respect to some basename $\mathsf{bsn} \in \{\mathsf{bsn}_0, \mathsf{bsn}_1\}$ so they must have identical tags $\tau_0 = \tau_1$. Using the same argument as in the previous case, we have that $\left[\frac{1}{\mathsf{sk}+(0\|\mathsf{bsn})}\right] P_1 = \left[\frac{1}{\mathsf{sk}'+(0\|\mathsf{bsn})}\right] P_1$ which, again by the deterministic nature of the WBB-LIT scheme, implies that $\mathsf{sk} = \mathsf{sk}'$ which contradicts the result of applying $\mathsf{Identify_S}$ to one of the signatures (i.e. the one that does not identify w.r.t. sk).

6.2 Efficiency of the Construction

Details of the Verification Equations. We give the details of the equations for the NIZK Groth-Sahai proof contained in a DAA signature.

1. Verification Equations for $\mathsf{cred} = (A, B, C, D)$:

$$A \neq 0$$
$$e(A, Y) = e(B, P_2) \qquad e(C, \underline{P_2'})\, e(-B, \underline{f_2(\mathsf{sk})}) = 1$$
$$e(D, P_2) = e(A + C, X)$$

2. Validity of keys:

$$e(\underline{f_1(\mathsf{sk})}, P_2)\, e(-P_1, \underline{f_2(\mathsf{sk})}) = 1$$

3. Tag and signature validity:

$$e(\tau, \underline{f_2(\mathsf{sk})})\, e([(0\|\mathsf{bsn})]\tau - P_1, \underline{P_2'}) = 1$$
$$e(\sigma_w, \underline{f_2(\mathsf{sk})})\, e([(1\|\mathsf{vk}_{\mathsf{ots}})]\sigma_w - P_1, \underline{P_2'}) = 1$$

4. Simulation-enabling equation (SEE):

$$[\underline{d}]P_2 - [\underline{d}]\underline{P_2'} = 0$$

Following [37], we made the equations simulatable (and thus the proof zero-knowledge) by replacing P_2 by a variable P_2', which makes the assignment of 0 to all variables a satisfying witness. Note that we rewrote the equations in 3. above to increase efficiency. SEE in 4. is a *multi-scalar multiplication* equation in $\mathbb{G}_2$ over variables $P_2' \in \mathbb{G}_2$ and $d \in \mathbb{Z}_p$, for which Groth and Sahai [37] also show how to construct proofs. When making a proof, the prover must make a *trivial* commitment (which is done using randomness 0 and which does thus not hide the value) to $d = 1$. In the soundness setting this gives us $P_2 = P_2'$ and thus soundness of the set of equations. If the CRS is set up for simulation, GS commitments to $\mathbb{Z}_p$ elements can be *trapdoor-opened* to any value. In order to simulate, we set $d = 0$ and $P_2' = 0$, which satisfies the SEE, and allows us to set all other variables to 0 as well.

Efficieny. The group public key contains 2 group elements from $\mathbb{G}_2$. Using the weak Boneh-Boyen signatures [9] as one-time signatures (as done by Groth in a similar construction [34]), a pre-DAA signature is in $\mathbb{G}_1^{25} \times \mathbb{G}_2^{11}$. To evaluate the performance of our construction, we compare our scheme to standard-model instantiations of the most related primitive: group signatures satisfying the model from [5]. Groth's scheme [34] is currently the state of the art and is the most efficient scheme in the standard model. A group signature consists of 50 group elements of a *symmetric* bilinear group.

Transpositions of Groth's scheme to asymmetric bilinear groups (which are more efficient than their symmetric counterparts) yield signatures of size $\mathbb{G}_1^{25} \times \mathbb{G}_2^{19}$ and $\mathbb{G}_1^{24} \times \mathbb{G}_2^{15}$ in Type-2 and Type-3 settings, respectively [40]. (In the notation of [31], an elliptic curve is of Type 2 if $\mathbb{G}_1 \neq \mathbb{G}_2$ and there is an efficiently computable isomorphism from $\mathbb{G}_2$ to $\mathbb{G}_1$, whereas there is none for Type-3 curves.) We emphasize that linkability, which is inherent to DAA, is a complex requirement that is not necessitated in group signatures. Moreover, in contrast to the joining protocol in Groth's scheme which requires many rounds of interaction, our DAA joining protocol is round-optimal and hence admits concurrent joining of members.

7 Large Basename Spaces

We note here that by relying on an *interactive* variant of the q-DDHI assumption used in [41] to construct a pseudo-random function with a large domain, we would obtain a pre-DAA scheme without any restrictions, i.e., with an arbitrarily large basename space. The interactive variant of q-DDHI provides the adversary with an oracle for weak Boneh-Boyen signatures (i.e., VRF or LIT values): the adversary gets $P_i, [x]P_i$ for some random integer $x \leftarrow \mathbb{Z}_p$ and has adaptive access to an oracle which on m replies with $[\frac{1}{x+m}]P_i$. The adversary is allowed to call the oracle q times and wins if it can distinguish a BB VRF value on a challenge message of his choice with the condition that such a message was not queried to the oracle from a random group element from $\mathbb{G}_i$. As the requirement of submitting all queries upfront is overcome, this immediately yields a fully f-indistinguishable LIT.

Making this interactive assumption, we can prove the scheme in Sect. 5 anonymous in the same way as the ROM scheme in [6], now that all components satisfy the original notions.

Unfortunately, for the more efficient scheme given in Fig. 8, the polynomial basename space was also needed in the proof of non-frameability, where we had to submit all tag queries to our weak signing oracle in advance. (Note that this problem does not arise in our first scheme, where we rely on the simulation-extractability of the signature of knowledge, following the proof strategy of [6].)

8 Conclusion and Open Problems

We have constructed the first DAA schemes in the standard model. For our first scheme, we constructed the first efficient signatures of knowledge which do not require random oracles. In addition, we have provided the first instantiations of randomizable weakly blind signatures in the standard model. As another building block we presented a linkable indistinguishable tag for a polynomial domain space in the standard model.

We leave as an open problem the construction of a LIT scheme for large domain spaces whose security is based on standard non-interactive intractability assumptions.

It would also be interesting to investigate under what assumptions our second scheme would support super-polynomial basename spaces; or what modifications would be necessary. We leave the construction of a scheme with an efficiency comparable to that of the scheme in Fig. 8 that satisfies the pre-DAA model of [6] without restricting the number of basenames as another open problem.

Acknowledgements

This work has been supported in part by ERC Advanced Grant ERC-2010-AdG-267188-CRIPTO, EPSRC via grant EP/H043454/1 and the European Commission through the ICT Programme under Contract ICT2007216676 ECRYPT II. The authors would like to thank anonymous reviewers whose comments helped to improve the paper.

References

1. M. Abe, G. Fuchsbauer, J. Groth, K. Haralambiev and M. Ohkubo. Structure-preserving signatures and commitments to group elements In *Advances in Cryptology – CRYPTO 2010*, Springer LNCS 6223, 209–236, 2010.
2. M. Belenkiy, M. Chase, M. Kohlweiss and A. Lysyanskaya. Compact e-cash and simulatable VRFs revisited. *Pairing-Based Cryptography – Pairing 2009*, Springer LNCS 5671, 114–131, 2009.
3. M. Bellare, R. Canetti and H. Krawczyk. Pseudorandom functions revisited: the cascade construction and its concrete security. In *FOCS '96*, 1996.
4. M. Bellare and P. Rogaway. Random oracles are practical: A Paradigm for Designing Efficient Protocols. In: *ACM Conference on Computer and Communications Security 1993*, ACM, pp. 62–73.
5. M. Bellare, H. Shi and C. Zhang. Foundations of group signatures: The case of dynamic groups. *Topics in Cryptology – CT-RSA 2005*, Springer LNCS 3376, 136–153, 2005.
6. D. Bernhard, G. Fuchsbauer, E. Ghadafi, N.P. Smart and B. Warinschi. Anonymous attestation with user-controlled linkability. *Cryptology ePrint Archive*. Report 2011/658, available at `http://eprint.iacr.org/2011/658`.
7. O. Blazy, G. Fuchsbauer, D. Pointcheval and D. Vergnaud Signatures on Randomizable Ciphertexts In *Public Key Cryptography – PKC 2011*, Springer LNCS 6571, 403–422, 2011.
8. D. Boneh and X. Boyen. Efficient selective-ID secure identity based encryption without random oracles. In *Advances in Cryptology – Eurocrypt 2004*, Springer LNCS 3027, 223–238, 2004.
9. D. Boneh and X. Boyen. Short signatures without random oracles. *Advances in Cryptology – Eurocrypt 2004*, Springer LNCS 3027, 56–73, 2004.
10. D. Boneh and X. Boyen. Short Signatures Without Random Oracles and the SDH Assumption in Bilinear Groups. *Journal of Cryptology*, **21(2)**, 149–177, 2008.
11. D. Boneh, X. Boyen and H. Shacham. Short Group Signatures. In *Advances in Cryptology – CRYPTO 2004*, Springer LNCS 3152, 41–55, 2004.
12. Dan Boneh, H.W. Montgomery and A. Raghunathan. Algebraic pseudorandom functions with improved efficiency from the augmented cascade. *Computer and Communications Security – CCS 2010*, ACM Press, 131–140, 2010.

13. E. Brickell, J. Camenisch and L. Chen. Direct anonymous attestation. *Computer and Communications Security – CCS 2004*, ACM Press, 132–145, 2004.
14. E. Brickell, L. Chen and J. Li. Simplified security notions for direct anonymous attestation and a concrete scheme from pairings. *Int. Journal of Information Security*, **8**, 315–330, 2009.
15. E. Brickell, L. Chen and J. Li. A new direct anonymous attestation scheme from bilinear maps. *Trusted Computing - Challenges and Applications – TRUST 2008*, Springer LNCS 4968, 166–178, 2008.
16. E. Brickell and J. Li. Enhanced privacy ID: A direct anonymous attestation scheme with enhanced revocation capabilities. *Privacy in the Electronic Society – WPES 2007*, ACM Press, 21–30, 2007.
17. E. Brickell and J. Li. Enhanced privacy ID from bilinear pairing. *Cryptology ePrint Archive*. Report 2009/095, available at `http://eprint.iacr.org/2009/095`.
18. J. Camenisch and A. Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *Advances in Cryptology – CRYPTO 2004*, Springer LNCS 3152, 56–72, 2004.
19. R. Canetti, O. Goldreich and S. Halevi. The random oracle methodology, revisited (preliminary version). In *30th ACM STOC* ACM Press, 209–218, 1998.
20. M. Chase and A. Lysyanskaya. On signatures of knowledge. In *Advances in Cryptology – CRYPTO 2006*, Springer LNCS 4117, 78–96, 2006.
21. S. Chatterjee, D. Hankerson, E. Knapp and A. Menezes. Comparing two pairing-based aggregate signature schemes. *Des. Codes Cryptography*, **55**, 141–167. May 2010.
22. D. Chaum. Blind signatures for untraceable payments. In *Advances in Cryptology – CRYPTO 1982*, Plenum Press, 199–203, 1983.
23. L. Chen. A DAA scheme requiring less TPM resources. *Int. Conference on Information Security and Cryptology - Inscrypt 2009*.
24. L. Chen, P. Morrissey and N. P. Smart. Pairings in trusted computing. *Pairings in Cryptography – Pairing 2008*, Springer LNCS 5209, 1–17, 2008.
25. L. Chen, P. Morrissey and N. P. Smart. On proofs of security of DAA schemes. *Provable Security – ProvSec 2008*, Springer LNCS 5324, 167–175, 2008.
26. L. Chen, P. Morrissey and N. P. Smart. DAA: Fixing the pairing based protocols. *Cryptology ePrint Archive*. Report 2009/198, available at `http://eprint.iacr.org/2009/198`.
27. L. Chen, D. Page and N.P. Smart. On the design and implementation of an efficient DAA scheme. *Smart Card Research and Advanced Application – CARDIS 2010*, Springer LNCS 6035, 223–237, 2010.
28. X. Chen and D. Feng. Direct anonymous attestation for next generation TPM. *Journal of Computers*, **3**, 43–50. 2008.
29. Y. Dodis and A. Yampolskiy. A verifiable random function with short proofs and keys. In *Public Key Cryptography – PKC 2005*, Springer LNCS 3386, 416–431, 2005.
30. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology – CRYPTO 1986*, Springer LNCS 263, 186–194, 1986.
31. S. Galbraith, K. Paterson and N.P. Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, **156**, 3113–3121, 2008
32. E. Ghadafi. Formalizing group blind signatures and practical constructions without random oracles. In *Cryptology ePrint Archive, Report 2011/402*, `http://eprint.iacr.org/2011/402.pdf`.
33. E. Ghadafi, N.P. Smart and B. Warinschi. Groth-Sahai proofs revisited. In *PKC 2010*, Springer LNCS 6056, 177–192, 2010.
34. J. Groth. Fully anonymous group signatures without random oracles. In *Advances in Cryptology – ASIACRYPT 2007*, Springer LNCS 4833, 164–180, 2007.
35. J. Groth. Simulation-sound NIZK proofs for a practical language and constant size group signatures. In *Advances in Cryptology – ASIACRYPT 2006*, Springer LNCS 4284, 444–459, 2006
36. J. Groth and A. Sahai. Efficient non-interactive proof systems for bilinear groups. In *Advances in Cryptology – Eurocrypt 2008*, Springer LNCS 4965, 415–432, 2008.
37. J. Groth and A. Sahai. Efficient non-interactive proof systems for bilinear groups (full version). `http://www.brics.dk/~jg/WImoduleFull.pdf`
38. S. Hohenberger and B. Waters. Constructing verifiable random functions with large input spaces. *Advances in Cryptology – Eurocrypt 2010*, Springer LNCS 6110, 656–672, 2010.
39. A. Juels, M. Luby and R. Ostrovsky. Security of blind digital signatures. In *Advances in Cryptology – CRYPTO '97*, Springer LNCS 1294, 150–164, 1997.
40. S.A. Kakvi. Efficient fully anonymous group signatures based on the Groth group signature scheme. Masters thesis, University College London, 2010. `http://www.scribd.com/doc/36974958/Dissertation`.
41. M. Kohlweiss. Cryptographic protocols for privacy enhancing identity management. PhD thesis, Katholieke Universiteit Leuven, March 2010. `http://www.cosic.esat.kuleuven.be/publications/thesis-181.pdf`.
42. A. Lysyanskaya, R. Rivest, A. Sahai and S. Wolf. Pseudonym systems. In *Selected Areas in Cryptography – SAC '99*, Springer LNCS 1758, 184–199, 1999.

43. S. Micali, M.O. Rabin and S.P. Vadhan. Verifiable random functions. In *Symposium on Foundations of Computer Science – FOCS '99*, IEEE Computer Society, 120–130, 1999.
44. D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, **13(3)**, 361–396, 2000.
45. Trusted Computing Group. TCG TPM specification 1.2. Available at `http://www.trustedcomputinggroup.org`, 2003.
46. B. Waters. Efficient identity-based encryption without random oracles. *Advances in Cryptology – Eurocrypt 2005*, Springer LNCS 3494, 114–127, 2005.