Anonymous attestation with user-controlled linkability

D. Bernhard G. Fuchsbauer E. Ghadafi N.P. Smart B. Warinschi

Dept. Computer Science,
University of Bristol,
Merchant Venturers Building,
Woodland Road,
Bristol, BS8 1UB,
United Kingdom.
{bernhard,georg,ghadafi,nigel,bogdan}@cs.bris.ac.uk

Abstract

This paper is motivated by the observation that existing security models for Direct Anonymous Attestation (DAA) have problems to the extent that insecure protocols may be deemed secure when analysed under these models. This is particularly disturbing as DAA is one of the few complex cryptographic protocols resulting from recent theoretical advances actually deployed in real life. Moreover, standardisation bodies are currently looking into designing the next generation of such protocols.

Our first contribution is to identify issues in existing models for DAA and explain how these errors allow for proving security of insecure protocols. These issues are exhibited in all deployed and proposed DAA protocols (although they can often be easily fixed).

Our second contribution is a new security model for a class of "pre-DAA scheme", i.e., DAA schemes where the computation on the user side takes place entirely on the trusted platform. Our model captures more accurately than any previous model the security properties demanded from DAA by the Trusted Computing Group (TCG), the group that maintains the DAA standard. Extending the model from pre-DAA to full DAA is only a matter of refining the trust models on the parties involved.

Finally, we present a generic construction of a DAA protocol from new building blocks tailored for anonymous attestation. Some of them are new variations on established ideas, and may be of independent interest. We give instantiations for these building blocks that yield a DAA scheme more efficient than the one currently deployed, and as efficient as the one about to be standardised by the TCG which has no valid security proof.

Keywords. DAA protocol, group signatures, security models.

Contents

1 Introduction

Direct Anonymous Attestation (DAA) [4] is one of the most complex cryptographic protocols deployed in the real world. This protocol, standardised by the Trusted Computing Group (TCG), allows a small embedded processor on a PC motherboard, called a Trusted Platform Module (TPM), to attest to certain statements about the configuration of the machine to third parties. One can think of this attestation as a signature on the current configuration of the embedding machine. The key requirement behind DAA is that this attestation is done in a way that maintains the privacy (i.e. anonymity) of the machine. The large scale of TPM distribution (there are about 200 million TPMs embedded in various platforms), and the potential for interesting applications that rely on trusted computation, triggered a significant research effort on DAA security [4, 6, 5, 7, 8, 16, 21, 20, 22, 23, 15].

This paper is motivated by the observation that all existing security models for DAA are deficient: they are either unrealisable in the standard model, do not capture some of the required functionality of the scheme or, worse, do not cover all realistic attack scenarios. In fact, even the existing deployed protocol [4] does not possess security properties one would expect given the informal description of what a DAA scheme enables. The reason for this is that the underlying security model in [4] does not capture certain desired properties. The main contribution of this paper is a security model for DAA that improves on all of these points. In addition, we give a construction which we prove secure with respect to our model. Our construction is in terms of abstract building blocks that we identify in this paper and which, for efficiency, we instantiate in the random-oracle model. Below we put our work in context and detail our results.

Issues in existing security models for DAA. Existing models for DAA are informed by the TPM standard put forth by TCG [36]. This standard reflects some intuitively appealing security guarantees but, like many other industrial standards, the specification is fuzzy in important respects. Some of the aspects that are left open to interpretation have unfortunately been imported by the more rigorous formal security models for DAA. Our first contribution is identifying significant shortcomings in all of the existing models. In brief, we argue that current models may allow security proofs for schemes against which attacks considered by TCG as complete breaks may still exist; indeed, the deployed scheme from [4] is such an example.[1] Our findings in relation to security models apply to both the original simulation-based model [4], including later attempts to enhance security [22], as well as the more recently proposed game-based models [6, 16]. In Section 2 we detail some of the problems with the model of [4], still considered to be the quintessential model for DAA.

We note that our findings regarding the models do not imply that schemes analysed with respect to them are necessarily insecure. Nevertheless, we show that the underspecification of the execution setting in [4] allows for situations where attacks against the scheme are possible.

New model. In light of the above discussion, it is fair to say that all of the existing models proposed so far for DAA security raise various issues as to applicability, sometimes in several respects. The absence of a good model is however critical for a rigorous analysis of any new anonymous attestation protocols: currently, TCG is in the process of specification of the next generation of TPMs. Without a complete formal model against which their goals can be compared the mistakes of the past are likely to be repeated. The main contribution of this paper is a security model for direct anonymous attestation. We leave most of the discussion of our model and the design decisions that we took for Section 3 and here we only highlight some of its more important aspects.

We chose to formalise our notion using game-based definitions rather than simulation. Our choice was motivated not only by some of the criticism generally applicable to simulation-based security (sensitivity to adaptive corruption; sometimes too strong to be realisable). We also felt that specifying the different security properties separately leads to a better understanding of what DAA should achieve. Despite occasional claims to the contrary, for complex interactions and requirements, specifying security through a single ideal specification is not always clearer, or cleaner, than through cryptographic games. The problems that we have uncovered in the simulation-based definition of [4] support our claim. They show that it is in fact quite difficult to assess whether a given functionality does capture the desired security properties despite years of scrutiny.

To simplify the understanding of how we model the security properties of DAA schemes, we proceed in two steps. First, we eliminate the need for complex trust scenarios involving three parties (the Host, the TPM, and the Issuer) and model the TPM and the Host as a single party in the system (as opposed to separate entities). On the one hand, this reduces the complexity of the model (avoiding three-party protocols and the associated complex trust). On the other hand, the resulting model directly captures the security of DAA protocols where the

[1]Note that repairing [4] to avoid our problem is trivial, and whether one considers our observation to be an "attack" depends on one's view as to what a DAA protocol is meant to achieve. The motivation of the work in this paper is to clarify misunderstandings as to what the goals are.

computation is performed entirely by the TPM (but whose input comes from the Host). For example, the DAA protocol for Mobile Trusted Modules (MTMs) falls in this class. To clearly reflect that our models do not directly deal with three-party protocols, we call this primitive pre-DAA. In a second step we explain how to turn pre-DAA models into models for full DAA by considering slightly more refined trust settings where it is not the case that the Host and the TPM are both either simultaneously honest or simultaneously corrupt. An additional benefit of our simplification is that it allows for simpler design of DAA schemes: start with the design of a pre-DAA scheme and then, if needed, "outsource" the non-sensitive data storage and computation to the Host.

Past DAA models (and those that we develop here) are inspired by models for group signatures [10, 12]. The trickiest issue to deal with (and one where past models are lacking) is the concept of an "identity". Unlike in group signatures, where parties are assumed to possess certified public/secret keys, the identity of a TPM is more difficult to define, as it does not possess any public key for the underlying group signature. Parties do however possess public authentication keys (called endorsement keys) which, as a security requirement, are not allowed to be linked to any public data used in the group-signature-like functionality. Yet, specifying an identity is crucial in defining security notions like anonymity, non-frameability etc. for group signatures. In previous models this issue was treated rather superficially and led to ambiguities in definitions. In contrast, we avoid similar problems by making the identity of a TPM a well-defined object (albeit information-theoretically).[2]

Our model for pre-DAA schemes does not explicitly capture how an issuer authenticates a TPM, as this question is somehow orthogonal to the main functionality of a DAA protocol. As this is nevertheless an important issue for the use of TPMs in practice, we discuss various ways of authenticating this channel, paying particular attention to the types of authentication which have been opted for by the TCG group in relation to DAA.

Construction. Our final contribution is a construction of a pre-DAA scheme proven to satisfy our security definitions in the random-oracle model. Our construction is built generically from two building blocks: a weak blind-signature scheme and a tagging scheme with special properties. We introduce the syntax and the security requirements we demand from these building blocks in Section 6 and give details of their security models as well as efficient constructions in Sections 8.4 and 8.5. The generic construction of our pre-DAA scheme is given in Section 7 and a concrete instantiation obtained by instantiating the building blocks is spelled out in Section 9. Using our methodology, we show how to turn our scheme into a fully fledged DAA scheme in Section 4.

Our protocol is highly efficient and fully practical. In terms of efficiency our scheme is virtually identical to that presented in [22]. Implementation results in [23] show that the scheme in [22] is significantly faster than the RSA-based scheme from [4], and hence these results will carry over to our own proposal. Our scheme has thus all the computational benefits of the one discussed in [22, 23], yet it comes with a fully developed security model and a proof that it satisfies the model. We note that we could not prove the security of the scheme of [22] (which has only been proved secure with respect to a flawed model) within the model of the current paper.

Our construction, following closely that of [22] (but being secure with respect to a well defined model), inherits the design heritage of that scheme and indeed others. Our use of a tag to obtain the linking functionality between signatures, group signatures and credential systems appears in various prior work [2, 17, 33, 35]. Indeed, all existing pairing-based DAA schemes [6, 5, 7, 8, 16, 21, 20, 22, 23, 15] use exactly the same tag derived from BLS signatures [9]; our abstraction of the required functionality may however lead to new constructions.

Our basic group-signature-like construction again closely follows the prior work on pairing-based DAA, and is itself closely related to the group-signature construction by Groth [29]. However, by identifying the joining protocol as a variant of a blind-signature issuing, we bring to the fore the need for the issuer to provide a proof of knowledge, rather than the user, as in [22] etc. The fact that the proof of knowledge is the wrong way round is the key reason we are unable to show the protocols in [22] etc are secure.

The paper is organized into essentially two parts; the first part deals with definitional issues related to DAA, whilst the second relates to a practical instantiation.

In more detail the first part is structured as follows: In Section 2 we first discuss issues and problems in existing security models for DAA protocols, a present an overview of why our security model corrects, simplifies and expands on previous models. We then in Section 3 we describe our new security model for a pre-DAA scheme. Since an pre-DAA scheme is not a full DAA scheme we then turn, in Section 4, to show how a pre-DAA scheme scheme can be turned into a DAA scheme via considering the Host as a mechanism for outsourcing storage and computation for the TPM. A key issue which still needs to be addressed is how the TPM authenticates itself to the host, to ensure a complete treatment in Section 5 we briefly turn to this issue and show how existing solutions for

[2]Interestingly, most of the added complication is due to the TCG's requirement that holders of secret keys should be able to revoke their key by publishing it on a list. Despite this being a requirement of the TCG, we are unsure how in practice a user would obtain the key (embedded in the TPM) so as to be able to revoke it.

this fit into our framework.

In the second half of the paper we turn to showing that our definition of a pre-DAA scheme can be realised efficiently in practice. As remarked above our construction of an pre-DAA scheme scheme is "generic", in that we base it on sub-components which we combine together via a general theorem. In Section 6 we present an overview of the three components; namely a form of blind-signature, a special tagging algorithm and signature proofs of knowledge. We then in Section 7 present our generic construction of a pre-DAA scheme, with a proof of security with respect to our prior definitions. Having presented a generic construction, all that remains is to instantiate our components which is done in Section 8. Finally in Section 9 we present the precise instantiation of our pre-DAA scheme and DAA schemes using these components.

2 Issues in existing security models for DAA

We first informally describe the goals of a DAA scheme. This discussion is necessary both to understand our criticism of existing models and to motivate the security model that we develop in the next section. In a DAA scheme a user, typically consisting of a Trusted Platform Module (TPM) and a Host, is allowed to join a group, maintained by an issuer, by executing a join protocol. We assume that the execution of the protocol takes place over an authentic channel; in particular, there is no notion of a user public key. However, each user has a secret key and the result of the joining protocol is some sort of credential associated to this secret key, to be used later as a signing key. In practice, one expects that a user would generate a distinct secret key for each group he joins. How the distinct secret keys and authentic channels are provided in practice is dealt with in Section 5.

Once the user has joined, he can produce signatures on behalf of the group, much like in group signatures. These signatures should generally be unlinkable, so as to guarantee anonymity. However, a form of user-controlled linkability is provided. In particular, there is a parameter bsn (for *basename*) passed to the sign and verify algorithms, which controls linking of signatures. If $\mathsf{bsn} = \perp$ then the resulting signature should be unlinkable to any other; but if $\mathsf{bsn} \neq \perp$ then signatures from the *same* signer with the *same* bsn should be linkable. Unlike for group signatures, there is no group opener who can revoke the anonymity of a signer. However, the current TCG specification requires that it be possible to locally detect if a signature has been produced by a user whose secret key has been compromised. We interpret this as essentially requiring that it be possible to identify if signatures were produced by a TPM with a given secret key. This mirrors the use of a so-called $\mathsf{RogueList}$ in DAA schemes and the variant of Verifier-Local Revocation (VLR) [11] in group signature schemes. However, the data entries used to determine a compromised user are the long-term user private keys, and not some information which can be linked back to the user's identity as for VLR. Note that the issuer has no control over who is placed on the $\mathsf{RogueList}$, as the issuer does not have access to the underlying keys, and hence a pre-DAA scheme is simpler than a standard VLR group signature in this regard. In some sense a user is the only person able to revoke his own secret key.

In the following we argue that all of the existing models for DAA fail to capture one or more of the security properties desired by the TCG. The focus of this paper is on new models and constructions, so we only devote space to [4] for which we describe in detail one problem.

2.1 Simulation-based models

The original security model for DAA [4] is based on simulation (in the sense of universal composability (UC)). In line with the TPM standard, the ideal functionality designed to capture the security of the protocol allows (and in fact demands) for the signing/verification process to be interactive. As explained above, transactions of a TPM with the same basename and secret key should be linkable via a "linking" algorithm. The ideal functionality captures this requirement only indirectly: when a transaction occurs, the ideal adversary is provided with a pseudonym for the TPM involved in that transaction and this pseudonym can later be used to link with other transactions of the same TPM.

A crucial observation is that granting the *simulator* the capability of linking transactions (via an extra operation on his interface to the ideal functionality) has no implications on the linkability of an actual implementation of the protocol! Indeed, nothing prevents a simulator from enjoying capabilities not present in the real protocol; granting the simulator (but not the environment, via honest parties) extra capabilities can only make an actual realisation easier to achieve.

The problem stems from the fact that the interface of the ideal functionality does not allow the environment explicit access to a linking algorithm, and thus the ideal functionality does not capture any security requirements on such an algorithm. As further evidence for this assertion, consider some protocol that realises the ideal functionality of [4]. Then the same proof of security still applies if one adds to the protocol specification an arbitrary linking algorithm, even one that links all transactions or one that links none. The obvious conclusion is therefore that the way in which the functionality of [4] captures controlled linkability (as demanded by the standard) is unfortunately flawed. Later attempts to rectify this problem [21, 20, 22] failed. For the particular ideal functionality defined in [22] it is trivial to distinguish between the ideal and the real world. This succession of failures led authors to consider game-based models.

This problem is not just of academic interest: the currently deployed DAA protocol from [4] is based on this flawed model of linking. Security engineers often refer to DAA as providing a signature functionality, but when interpreted in this way, the scheme from [4] suffers from an attack which we describe below. We also explain that the attack may not exist in other execution scenarios where DAA is interpreted as an authentication process. We show how to fix the protocol to completely avoid the attack. However, the attack is due to the underspecification of the execution model on which the security definition of [4] relies, so clearly a precise security model for DAA protocols is needed.

At their heart all simulation-based models assume an interactive signature/verification protocol, for reasons we will come to in a moment. Whilst in [22] an attempt was made to address this, the result is a model in which it is trivial to distinguish between the real and ideal world. We therefore return to the original model of [4].

In essence the simulation-based model in [4] is a model of an *authentication* protocol, not of a *signature* protocol. Indeed, if the verifier maintains sessions, uses nonces as session identifiers and fixes a single basename at the start of each session that he expects a signature for, we will never be able to "replay" a signature. However, if the signatures are generated and verified interactively what does it mean for signatures to be linkable? Since interaction implies that linkability is relative to a given verifier at a given point in time. Yet one can imagine many situations in which a signer may want to link signatures to a number of verifiers, but if signatures are not long lived it is hard to see what this means.

Indeed, if the resulting scheme from [4] is used in a situation where the signatures are not verified interactively then there is an attack against the linkability: A signature for a non-empty basename will still verify if submitted for verification with the empty basename. This means we can produce a valid signature on a message/basename pair without a user's secret key even though the user never signed this pair. The scheme could very easily be modified to defend against this attack: The basename could be added to the input of the hash used in the signature proof of knowledge. It would even suffice to add a bit that is 0 for an empty basename and 1 otherwise. Interestingly, the basename is hashed in this way in later schemes such as that of [6].

We pause at this point to stress this point: The existing DAA scheme deployed in millions of computers around the world does not meet the intuitive security guarantees one would expect of a DAA scheme. This point was not picked up in the original paper because the security model was not able to sufficiently capture the linkability requirements. This is partly due to the ambiguity over whether a DAA scheme is an authentication protocol or a signature scheme. Whilst this point may be easy for cryptographers to grasp, we do not feel the difference is sufficient for security engineers using DAA. After all if a bit-string can be intuitively used as a signature, then engineers will use it as such. This is the *major* motivation for the work in this paper; to both define the security requirements correctly and simply; to ensure the outputs can be used as signatures with controlled linkability, and to also present a scheme which provably meets our formal requirements.

Although the simulation-based model of [4] is not universally composable (UC) [14], it is instructive to look at signatures in a UC setting. Following the paper of Canetti [13] on the subject, we note that in a first attempt, a signature functionality could be viewed simply as a registration functionality: The honest signer can register messages as "signed" and verifiers can query if a message was registered. Such a model is too simplistic and does not cover all applications of digital signatures; indeed in any implementation of a signature scheme, signatures can be processed in many ways: Transmitted, encrypted, even signed. It is necessary to model the signature itself as an object of some kind.

For a signature protocol to UC-securely implement a signature functionality, the outputs of the two must be indistinguishable. In other words, the signatures from the functionality must have the same distribution as those in the protocol, which at first glance looks impossible as the functionality cannot depend on an implementation of itself. This problem is overcome by letting the functionality ask the *adversary* to produce either the signatures [13] or a signature algorithm [14]. While this works fine for standard signature schemes, it poses a new problem for pre-DAA as the signature must bind to a user identity (more precisely: to a secret key) yet still be anonymous.

Can we give a simulation-based proof following the current UC framework? The answer is no in the plain model, for the following reason. In [25] a proof is given that UC-secure bit commitment is impossible, more specifically that given any UC functionality for bit commitment, no protocol can UC-securely implement it without further setup assumptions. Such a protocol would have to be both information-theoretically hiding and binding which is known to be impossible. This impossibility result extends to any functionality from which commitment could be derived; one of the examples given in the paper is group signatures.

A pre-DAA scheme produces signatures that are anonymous (hiding the signer) yet revocable or openable (binding to the signer). Therefore, if bit commitment could be built generically from such pre-DAA schemes then it is impossible to construct, in the plain model, a UC-secure pre-DAA functionality. Now it is easy to see that, given a pre-DAA scheme we can implement bit commitment: Let the committer pick two keys $\mathfrak{sk}_0$ and $\mathfrak{sk}_1$ and play the role of these users. Let the verifier play the role of the issuer. The committer runs the Join protocol twice, first with $\mathfrak{sk}_0$ then with $\mathfrak{sk}_1$ in that order. The verifier saves the transcripts. To commit to a bit b, the committer signs any message and basename with $\mathfrak{sk}_b$ using the blind signature obtained while joining and gives the verifier this signature, who checks that it verifies correctly. To reveal b, the committer publishes both secret keys. The verifier identifies both transcripts, using the order they were created in to determine which key is $\mathfrak{sk}_0$ and which is $\mathfrak{sk}_1$. Then he checks which of the two keys the signature identifies to, obtaining b.

Thus if eventually one wishes to construct DAA protocols in the plain model (i.e. with no random oracles or CRSs) then one will need to restrict to game based definitions (or at least non-UC simulation based definitions).

But even in the random oracle (within which we work) a simulation based definition we feel is not the way to proceed. Simulation based definitions are very good at capturing secrecy gaurantees; they are less good at capturing the security gaurantees needed in our work. For example simulation based signature functionalities are known to be complex. In addition we need to capture complex linkability requirements for such signature functionalities. Thus the complexity of defining a simulation based security notion for DAA schemes is likely to be overly complex, to produce proofs which are hard to verify, and for which verifying whether the ideal functionality actually captures the intuitive security notions may be non-trivial. We are thus moved to example game based models.

2.2 Game-based models

More recent attempts at security models for DAA resort to cryptographic games [6, 16]. As usual, such games attempt to capture (typically, one-by-one) the different security properties required by the TPM specification. These attempts also failed. Our model provides a number of advantages over the previous game-based models. Indeed, our model captures a number of attack modes and security properties which are *not covered* by the previous game-based models. We outline these below:

In the equivalent game-based DAA models of [6, 16] the issue of identification of dishonest TPMs within the model is skirted around by assuming that all adversarially controlled users have their secret keys already exposed via the $\mathsf{RogueList}$. The identity of the adversarial user is then assumed to be uniquely associated with the value exposed in $\mathsf{RogueList}$. However, this does not capture an attack in which an adversary can engage in a number of $(\mathsf{Join}, \mathsf{Iss})$ protocols with an honest issuer, and then produce another dishonest user for which signatures verify. In particular, the model makes no mention of how such a dishonest user could ever be traced, even if its identity, i.e. its secret key, is eventually disclosed. Hence, the previous models assume a very strong form of static corruption in that not only the dishonest users are statically corrupted at the start, but also no new dishonest users can be created. This last point is a problem as there is no overarching PKI which is used to authenticate users as in group signatures. It is in part to deal with this last issue that we introduce our notion of uniquely identifiable transcript, so as to be able to define the identity of a user unambiguously.

In [6, 16] the game for correctness does not assume that a valid signature can be correctly identified. Hence, the models in [6, 16] are not able to argue about the correctness of the $\mathsf{RogueTag}$ process. In contrast, we will require for correctness that a valid transcript can always be validly identified. Bar these changes, the correctness definition in [6, 16] and our own one are essentially the same.

In [6, 16] there is only one game for traceability/non-frameability: The adversary wins the game if it can output a signature for an honest user which has not been the output of a signature query (a property captured by our non-frameability game), or if the adversary can come up with two signatures which should be linked but which are not (a property captured by our traceability game). The games in [6, 16] do not capture attacks in which the adversary produces two signatures which are linked, but should not be (e.g. a linking algorithm which always outputs one is correct in the model of [6, 16]). In addition, it does not capture an attack in which an adversary outputs a signature which cannot be traced when the value $\mathfrak{sk}_i$ is revealed, a feature due to the corruption model mentioned above. Finally, in [6] the game for user-controlled traceability requires that a test is made to determine whether a signature is "associated with the same identity and basename" without defining formally what this means or how it is done.

In summary, our game-based model improves over the previous one by capturing the following notions: signatures should be correctly identified by the $\mathsf{RogueTag}$ process, signatures which are not linked should not be linkable, and signatures must be traceable to a specific instance of a $(\mathsf{Join}, \mathsf{Iss})$ protocol.

3 Security models for pre-DAA

We first discuss syntax and then go on to defining the security games. We present game-based security notions for pre-DAA schemes which combine notions from the game-based security models for group signatures [12] and the game-based definitions for DAA [6, 16].

3.1 Syntax

A pre-DAA scheme is given by the tuple of algorithms

$$\text{pre-DAA} = (\mathsf{Setup}, \mathsf{GKg}, \mathsf{UKg}, \mathsf{Join}, \mathsf{Iss}, \mathsf{GSig}, \mathsf{GVf}, \mathsf{Identify}_\mathsf{T}, \mathsf{Identify}_\mathsf{S}, \mathsf{Link}) \ .$$

The functionality of these algorithms is as follows.

- $\mathsf{Setup}(1^\lambda)$: This probabilistic setup algorithm takes a security parameter 1^λ and outputs a description param of any system parameters (e.g. underlying abelian groups etc). It also sets up a public list $\mathsf{RogueList}$ which is initially set to be empty.
- $\mathsf{GKg}(\mathsf{param})$: This outputs a public/secret key pair $(\mathfrak{gmpk}, \mathfrak{gmsk})$ for the issuer $\mathcal{M}$.
- $\mathsf{UKg}(\mathsf{param})$: This is a probabilistic algorithm to generate user private keys. When run by user i it outputs the user's secret key $\mathfrak{sk}_i$. Unlike for group signatures, there is no notion of a corresponding user public key.
- $(\mathsf{Join}, \mathsf{Iss})$: This is an interactive protocol between a new group member i and the issuer $\mathcal{M}$. Each of the algorithms takes as input a state and a message and produces a new state and a message plus a decision in $\{\mathsf{accept}, \mathsf{reject}, \mathsf{cont}\}$. The initial state of Join is $\mathfrak{gmpk}$ and the private key of the user $\mathfrak{sk}_i$, whilst that of Iss is $(\mathfrak{gmpk}, \mathfrak{gmsk})$. The final state of Join is assigned to $\mathfrak{gsk}_i$. The issuer outputs accept or reject. We assume that the protocol starts with a call to Join.
- $\mathsf{GSig}(\mathfrak{gsk}_i, \mathfrak{sk}_i, m, \mathsf{bsn})$: This is a probabilistic signing algorithm that takes as input a group signing key $\mathfrak{gsk}_i$, a user secret key $\mathfrak{sk}_i$, a message m and a basename bsn, and returns a signature σ.
- $\mathsf{GVf}(\mathfrak{gmpk}, \sigma, m, \mathsf{bsn})$: This deterministic verification algorithm takes as input the group public key $\mathfrak{gmpk}$, a signature σ, a message m, and a basename bsn. It returns 1 or 0 indicating acceptance or rejection.
- $\mathsf{Identify}_\mathsf{T}(\mathcal{T}, \mathfrak{sk}_i)$: This outputs 1 if the transcript $\mathcal{T}$ corresponding to an execution of the $(\mathsf{Join}, \mathsf{Iss})$ protocol corresponds to a valid run with the secret key $\mathfrak{sk}_i$. (Further requirements that we impose on a protocol ensure that the result of this procedure is well-defined).
- $\mathsf{Identify}_\mathsf{S}(\sigma, m, \mathsf{bsn}, \mathfrak{sk}_i)$: This outputs 1 if the signature σ could have been produced with the key $\mathfrak{sk}_i$.
- $\mathsf{Link}(\mathfrak{gmpk}, \sigma, m, \sigma', m', \mathsf{bsn})$: This returns 1 if and only if the two signatures verify with respect to the basename bsn, which must be different from $\perp$, and σ and σ' were produced by the same user.

In our security model for non-frameability a dishonest issuer will be able to access $\mathfrak{gsk}_i$ via an oracle query, thus user security rests solely on secrecy of $\mathfrak{sk}_i$. This creates the knock-on effect of the GSig algorithm to require both $\mathfrak{gsk}_i$ and $\mathfrak{sk}_i$ in the above syntax. This change from the standard syntax and security of group signatures is to enable our later division of this algorithm between a TPM and a host computer in Section 4; looking ahead, the TPM will control $\mathfrak{sk}_i$ and the Host will control $\mathfrak{gsk}_i$.

Identities and pre-DAA schemes. In our security model for pre-DAA schemes we would like the users to be anonymous even in the presence of an adversarially controlled issuer, just as in the case of group signatures. However, the user identity must be linkable to signatures when passed to the $\mathsf{Identify}_\mathsf{S}$ algorithm. Yet, users have no public keys which are bound to their identities. In standard group signatures the user private key is associated with a (certified) public key, and hence identities are a well defined notion. The problem arises in that there could be a scheme which enables a user to engage in a Join protocol using one key, but to use the obtained credential to sign with a different one, making the credential a credential on both keys in some sense. In defining security for adversarially controlled issuers this is not a problem, the problem arises when dealing with dishonest users, and trying to define security notions for revocation.

To deal with this problem we need to be able to associate a unique identity/secret key to each execution (even if the user is malicious). In brief, we ask that the joining protocol is such that if the issuer is honest and accepts after a given run of the protocol then there exists a unique secret key for the user which could have led to the given transcript, if the user had followed the protocol. We decree that key to be the key associated to the particular transcript (even if the user may not have followed the protocol). We define a notion of *uniquely identifiable transcripts* to formally capture this notion.

We then require that the $(\mathsf{Join}, \mathsf{Iss})$ protocol of a (pre-)DAA scheme has uniquely identifying transcripts, so that we can associate a unique identity to each valid run, namely $\mathfrak{sk}_i$. Without such a requirement, it is hard to envision a way to define rigorously, let alone enforce, the property (specified by the standards) that if an identity is exposed via leaking of $\mathfrak{sk}_i$ of a TPM then one can revoke signatures of that TPM. Indeed, in this situation the secret key of a malicious TPM is not a well-defined notion. From this perspective, the transcript of the $(\mathsf{Join}, \mathsf{Iss})$ protocol acts as a public key for the user.

3.2 Security Definitions

In this subsection we detail our security games. 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.

To define formally our notion of identifying an identity with a transcript we use the following notation. We write $\mathcal{T} = \mathcal{T}(\mathfrak{sk}_i, r_U, \mathfrak{gmsk}, r_I)$ for the transcript of an honest execution of the $(\mathsf{Join}, \mathsf{Iss})$ protocol by a user with secret key $\mathfrak{sk}_i$ and random coins r_U with an issuer with secret key $\mathfrak{gmsk}$ and random coins r_I. We let G_{sk} be the set of all possible issuer secret keys, U_{sk} the set of all possible user secret keys, R_U the space of randomness used by the user in the $(\mathsf{Join}, \mathsf{Iss})$ protocol, and R_I the space of randomness used by the issuer in the $(\mathsf{Join}, \mathsf{Iss})$ protocol.

Definition 1. *We say that* $(\mathsf{Join}, \mathsf{Iss})$ *has* uniquely identifying transcripts *if there exists a predicate*

$$\mathsf{Check}_T : \{0,1\}^* \times \mathsf{G}_{\mathsf{sk}} \times \mathsf{U}_{\mathsf{sk}} \times \mathsf{R}_I \times \mathsf{R}_U \to \{0,1\} \qquad \textit{such that}$$

- *if both parties are honest and run* $(\mathsf{Join}, \mathsf{Iss})$*, with input* $(\mathfrak{sk}, r_U)$ *and* $(\mathfrak{gmsk}, r_I)$ *respectively, to produce transcript* $\mathcal{T}$ *then* $\mathsf{Check}_T(\mathcal{T}, \mathfrak{gmsk}, \mathfrak{sk}, r_I, r_U) = 1$*;*
- *for all protocols* Join' *interacting with an honest issuer protocol* Iss*, which has input* $(\mathfrak{gmsk}, r_I)$*, producing transcript* $\mathcal{T}$*, if at the end of the protocol the issuer accepts then there is at most one value* $\mathfrak{sk} \in \mathsf{U}_{\mathsf{sk}}$ *(but possibly many values of* r_U*) such that* $\mathsf{Check}_T(\mathcal{T}, \mathfrak{gmsk}, \mathfrak{sk}, r_I, r_U) = 1$.

Notice that the above definition does not imply that $\mathfrak{sk}_i$ can be efficiently extracted from the protocol (e.g. via some knowledge extractor), but only that there is at most one solution. Also note that we do not preclude that a different value of $\mathfrak{sk}_i$ is associated with each different transcript, i.e. it is not that the $\mathfrak{sk}_i$ is unique globally, only that each transcript has a unique $\mathfrak{sk}_i$ associated with it.

The abilities of an adversary are modeled by a series of oracles as follows:

- $\mathsf{AddU}(i)$: The adversary can use this oracle to create a new honest user i.
- $\mathsf{CrptU}(i)$: The adversary can use this oracle to create a new corrupt user i.
- $\mathsf{InitU}(i)$: The adversary can use this oracle to create a group signing key for honest user i.
- $\mathsf{SndToI}(i, M)$: The adversary can use this oracle to impersonate user i and engage in the group-join protocol with the honest issuer that executes Iss.
- $\mathsf{SndToU}(i, M)$: This oracle models the situation that the adversary has corrupted the issuer. The adversary can use this oracle to engage in the group-join protocol with the honest user that executes Join.

- $\mathsf{GSK}(i)$: Calling this oracle enables the adversary to obtain the group signing key $\mathfrak{gsk}_i$ of user i. The user remains honest.
- $\mathsf{USK}(i)$: The adversary can call this oracle to obtain the secret keys of user i. Here, the adversary obtains the long-term private key in addition to the group signing key. This corresponds to the Corrupt query in the model of [6, 16]. After calling this oracle, control of party i passes to the adversary.
- $\mathsf{Sign}(i, \mathfrak{gsk}, m, \mathsf{bsn})$: This oracle allows the adversary to obtain signatures from an honest group member, using a possibly adversarially chosen $\mathfrak{gsk}$. It takes as input the identity of the group member i, the group signing key $\mathfrak{gsk}$, a message m and a basename bsn. It outputs a signature of member i on this data.
- $\mathsf{CH}_b(i_0, i_1, \mathsf{bsn}, m)$: This oracle can only be called once (namely to get a challenge in the anonymity game). The adversary sends a pair of honest identities (i_0, i_1), a message m and a basename bsn to the oracle and gets back a signature σ by the signer i_b.

Note that apart from the primitive-specific changes to the security model from [12] already mentioned, we have split the AddU oracle from [12] into two oracles AddU and InitU. This is purely for ease of exposition.

We now proceed to define our security notions for pre-DAA schemes. We contrast our notions with those for group signatures [12] and existing ones for DAA [6, 16]. We define security and correctness by means of four games: correctness, anonymity, traceability and non-frameability. In [6, 16] these are called correctness, user-controlled anonymity and user-controlled traceability, with a rather complicated game for the latter property. We simplify this into four games, which is more consistent with the security models for group signatures. The main difference between our model and those of [6, 16] is that we assume a user is a single entity and is not split into a Host and a TPM. This assumption simplifies the exposition and descriptions.

Using the above oracles the security games are formalised in Figure 2. The experiments manage lists $\mathsf{St}_U, \mathsf{St}_I$, as well as dec_I and dec_U, the entries of the two latter being initially set to cont. The underlying "code" of the various oracles available to the adversary are given in Figure 1.

Correctness. We require that signatures produced by honest users are accepted by verifiers and that a user who produces a valid signature can be traced correctly. In addition, we require that two signatures produced by the same user with the same basename are linked. To formalise this we associate to the pre-DAA scheme, any adversary $\mathcal{A}$ and any $\lambda \in \mathbb{N}$ the experiment $\mathsf{Exp}_{\mathcal{A}}^{corr}(\lambda)$ defined in Figure 2. We define $\mathsf{Adv}_{\mathcal{A}}^{corr}(\lambda) = \Pr\left[\mathsf{Exp}_{\mathcal{A}}^{corr}(\lambda) = 1\right]$ and we say that the scheme is *correct* if $\mathsf{Adv}_{\mathcal{A}}^{corr}(\lambda) = 0$ for all adversaries $\mathcal{A}$ and all $\lambda \in \mathbb{N}$. We reiterate that unlike in the case of group signatures [12], we require that two signatures are linked if they are produced with the same bsn and $\mathsf{bsn} \neq \perp$.

Anonymity. Intuitively, the goal of the adversary is either to determine which of two identities has produced a given signature, or to link two supposedly unlinkable signatures. This is formalised by requiring the adversary to guess the bit b used by the oracle CH_b, which returns a signature by the user i_b. As in the case of group signatures, the adversary has access to the issuer's secret key; thus, not even a dishonest issuer should be able to break the anonymity of the scheme. However, as opposed to the models in [10, 12], in (pre-) DAA schemes users can trivially identify signatures produced under their own key due to the functionality $\mathsf{Identify}_\mathsf{S}$. The adversary can thus only query a challenge signature for users it has neither corrupted nor queried the USK oracle for. Moreover, the adversary is not allowed to query the signing and challenge oracle for the same user i and basename $\mathsf{bsn} \neq \perp$, as it could then link the two signatures using Link.

In the anonymity experiment, the adversary can access the oracles AddU, SndToU, CrptU, USK, GSK and Sign to add honest users, to run the Join protocol with an honest user, create corrupt users, obtain the state information of previously honest users, and to obtain signatures from honest users, respectively. The adversary can query the CH_b oracle at one point in the game, and his goal is to guess the bit b. With $\mathsf{Exp}_{\mathcal{A}}^{anon\text{-}b}$ for an adversary $\mathcal{A}$ and $b \in \{0, 1\}$ as detailed in Figure 2, we define

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

and we say that the scheme has *anonymity* if $\mathsf{Adv}_{\mathcal{A}}^{anon}(\lambda)$ is negligible in λ for any polynomial-time adversary $\mathcal{A}$.

Traceability. The traceability game consists of two subgames, neither of which the adversary should be able to win. The first one formalises the requirement that no adversary should be able to produce a signature which cannot be traced to a secret key stemming from a run of the group-join protocol. The second subgame guarantees that no adversary can produce two signatures under the same secret key and for the same basename that do not link.

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

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

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

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

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

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

$\mathsf{SndToI}(i, M)$:
    • 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 $\mathsf{St}_I^i \leftarrow (\mathfrak{gmpk}, \mathfrak{gmsk})$.
    • $(\mathsf{St}_I^i, M', \mathsf{dec}_I^i) \leftarrow \mathsf{Iss}(\mathsf{St}_I^i, M)$.
    • Return $(M', \mathsf{dec}_I^i)$.

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

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

Figure 1: Oracles defining user registration in the security games for a pre-DAA scheme

In the first subgame we assume an honest issuer, just as in the case for traceability of group signatures. (This is necessary, as a dishonest issuer could always register dummy users that would be untraceable.) The adversary is given access to the oracles SndToI and CrptU (oracles simulating honest users would be redundant, as the adversary can simulate them on his own). The SndToI oracle allows the adversary to interact with the honest issuer and the CrptU oracle is required to "register" corrupted users.

It is in this game that our notion of identifying users by their transcripts comes to the fore. After interacting with the issuer, the adversary must output all the identities (i.e. secret keys) associated to the runs of the protocol (Join, Iss) which the issuer accepted. His goal is then to produce a signature that verifies but is not identifiable to any of the secret keys. This implies that the adversary cannot combine the information obtained from many (Join, Iss) runs to produce a group member who has not run the issuing protocol.

In the second game the adversary impersonates the issuer as well as all users. No oracles are required, as there are no honest parties. The adversary's goal is to produce two valid signatures for the same basename for one user which do not link. (That is, both signatures should be traced to the same secret key via $\mathsf{Identify}_\mathsf{S}$, but Link outputs 0 on input these signatures.) Hence, the two games capture the two notions of traceability: users can be traced via their secret keys or via linkable signatures. Traceability thus establishes completeness of $\mathsf{Identify}_\mathsf{S}$ and Link (i.e. they output 1 when they should).

Let $\mathcal{A}$ be an adversary performing the traceability experiment given in Figure 2. We define

$$\mathsf{Adv}_{\mathcal{A}}^{trace}(\lambda) = \Pr[\mathsf{Exp}_{\mathcal{A}}^{trace}(\lambda) = 1],$$

and we say that the scheme has *traceability* if $\mathsf{Adv}_{\mathcal{A}}^{trace}(\lambda)$ is a negligible function of λ for any polynomial-time adversary $\mathcal{A}$.

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

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

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

Figure 2: Security experiment for correctness, anonymity, traceability and non-frameability

Non-Frameability. As for traceability, there are two types of non-frameability, since users can be framed via their secret key or via the basename. Again, we define two subgames. In the first one the adversary's goal is to output a signature which can be traced to a specific user i, but which is for a message/basename pair that user i has never signed. In this experiment the adversary has access to the secret key of the issuer and it can access the oracles $\mathsf{AddU}, \mathsf{SndToU}, \mathsf{CrptU}, \mathsf{USK}, \mathsf{GSK}$ and Sign to interact with or corrupt honest users.

While in the first subgame the adversary tries to frame honest users, in the second subgame we give the adversary control over all users (and the issuer). His goal is to output signatures that link although they should not: they are from different users, the basenames are different, or one the basenames is $\perp$. Note that by granting the adversary full control over the issuer and all users, this notion is stronger than requiring only that the adversary cannot frame an *honest* user via Link.

While the first subgame guarantees soundness of $\mathsf{Identify}_\mathsf{S}$ (it only outputs 1 for signatures that were indeed produced with the respective key), the second subgame guarantees soundness of Link: it only outputs 1 if the signatures stem from the same signer, the basenames are identical and different from $\perp$.

Let $\mathcal{A}$ be an adversary performing the non-frameability experiment given in Figure 2. We define

$$\mathsf{Adv}_{\mathcal{A}}^{non\text{-}frame}(\lambda) = \Pr[\mathsf{Exp}_{\mathcal{A}}^{non\text{-}frame}(\lambda) = 1] \ ,$$

and we say that the scheme has *non-frameability* if the advantage $\mathsf{Adv}_{\mathcal{A}}^{non\text{-}frame}(\lambda)$ is a negligible function of λ for any polynomial-time adversary $\mathcal{A}$.

Note that the first subgame for both traceability and non-frameability mirrors the standard notions for traceability and non-frameability from group signatures (defined in [12]). The second subgames for the two notions fully capture the security notions we require for linkability. We see the definitional clarity in these two respects as an important contribution of our proposed model.

4 From pre-DAA to full DAA schemes

The major difference between a DAA scheme and our pre-DAA scheme is that in a DAA scheme the user is split between a trusted device which has a small amount of memory and limited computing power (namely the TPM), and a more powerful, but untrusted, machine called the Host. In addition, the user can register with a number of Issuers, and each time he registers he uses a *different* underlying secret key $\mathfrak{sk}_i$. He may also register with the same issuer a number of times, and obtain a number of distinct group signing keys $\mathfrak{gsk}_i$ on different underlying keys $\mathfrak{sk}_i$. However, the TPM has very little memory which means that it cannot hold a large number of secret keys $\mathfrak{sk}_i$, nor can it store a large number of group signing keys $\mathfrak{gsk}_i$. Moreover, it cannot store both of these items on the Host as the Host is untrusted.

To get around these problems, the TPM recomputes the values of $\mathfrak{sk}_i$ as and when needed, via the use of a pseudo-random function (PRF) applied to a fixed secret (usually called $\mathsf{DAASeed}$), the issuer identifier, and a counter value. The storage of $\mathfrak{gsk}_i$ is then provided by the Host. The signing operation $\mathsf{GSig}(\mathfrak{gsk}_i, \mathfrak{sk}_i, m, \mathsf{bsn})$ becomes an interactive protocol between the TPM and the Host. We denote the pair of interactive protocols by $(\mathsf{GSig}^{\mathsf{TPM}}, \mathsf{GSig}^{\mathsf{Host}})$. This signing protocol between the TPM and Host should correspond to a secure function evaluation by the Host of the signature value. The TPM is essentially delegating the computation of (some of) the signature to the Host, without revealing $\mathfrak{sk}_i$, nor allowing the Host to compute additional signatures. The fact that the value of $\mathfrak{gsk}_i$ is stored on the possibly untrusted Host is the rationale for requiring this value to be available to the adversary in our non-frameability security game for pre-DAA schemes.

Finally, we note that the signing operation of a DAA protocol is often an interactive operation between the user (TPM and Host) and the verifier, in that the verifier introduces some random nonce into the signing process at the start of the computation. However this situation is easily handled by adding this nonce to the message to be signed.

Following this discussion it is clear how to define a DAA protocol from a pre-DAA scheme.

- $\mathsf{DAA\text{-}Setup}(1^\lambda)$: This runs the setup algorithm $\mathsf{Setup}(1^\lambda)$ of the pre-DAA scheme.
- $\mathsf{Issuer\text{-}Kg}(\mathsf{param})$: It takes as input param and outputs secret-public key pair $(\mathfrak{gmsk}, \mathfrak{gmpk})$ for the issuer obtained by calling $\mathsf{GKg}(\mathsf{param})$. Each issuer is assumed to have a unique identifying string ID_i.
- $\mathsf{TPM\text{-}Kg}(\mathsf{param})$: This algorithm generates a secret key $\mathsf{DAASeed} \in \{0,1\}^\lambda$, which is stored in the TPM.
- $\mathsf{Host\text{-}Setup}(\mathsf{param})$: The Host maintains a list of group signing keys obtained from the issuer, initially set to the empty list. Each group signing key will be stored as a tuple $(\mathsf{ID}, \mathsf{cnt}, \mathsf{cred})$ which says that cred is the cnt'th group signing key obtained from the issuer identified by ID.
- $(\mathsf{DAA\text{-}Join}, \mathsf{DAA\text{-}Iss})$: This is an interactive protocol between the TPM, the Host and the Issuer. See Figure 3 for a description of this protocol, which uses the $(\mathsf{Join}, \mathsf{Iss})$ protocol of the pre-DAA scheme run between the TPM and the Issuer, with the host acting mainly as a router. Note that the Host needs to inform the TPM of the name of the issuer as well as the counter value it has got to for this issuer, since the TPM has restricted long term memory. At the end of the protocol the Host should learn the value of the group signing key, which should become the value $(\mathsf{ID}, \mathsf{cnt}, \mathfrak{gsk}_{\mathsf{ID},\mathsf{cnt}})$ held in its table. Whether this value is sent to the Host by the TPM or the Issuer is immaterial.

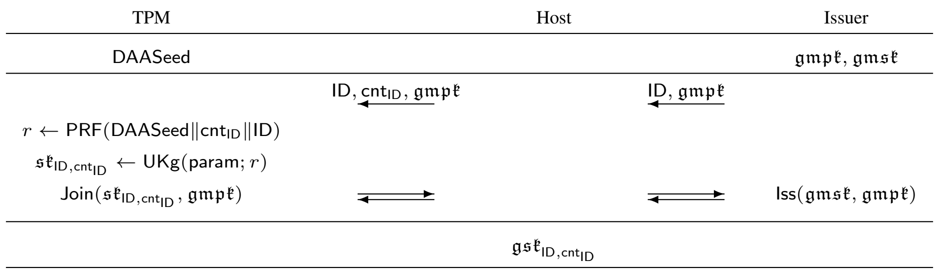
TPM
Host
Issuer
DAASeed
$\mathfrak{gmpk}, \mathfrak{gmsk}$
$\mathsf{ID}, \mathsf{cnt}_{\mathsf{ID}}, \mathfrak{gmpk}$
$\mathsf{ID}, \mathfrak{gmpk}$
$r \leftarrow \mathsf{PRF}(\mathsf{DAASeed}\|\mathsf{cnt}_{\mathsf{ID}}\|\mathsf{ID})$
$\mathfrak{sk}_{\mathsf{ID},\mathsf{cnt}_{\mathsf{ID}}} \leftarrow \mathsf{UKg}(\mathsf{param}; r)$
$\mathsf{Join}(\mathfrak{sk}_{\mathsf{ID},\mathsf{cnt}_{\mathsf{ID}}}, \mathfrak{gmpk})$
$\mathsf{Iss}(\mathfrak{gmsk}, \mathfrak{gmpk})$
$\mathfrak{gsk}_{\mathsf{ID},\mathsf{cnt}_{\mathsf{ID}}}$

Figure 3: The DAA Join Protocol

- $(\mathsf{DAA\text{-}Sig}^{\mathsf{TPM}}, \mathsf{DAA\text{-}Sig}^{\mathsf{Host}}, \mathsf{DAA\text{-}Vf})$: This is a protocol between the TPM, the Host and a possibly interactive verifier.
  - An online verifier produces a nonce which is appended to the message m being signed.
  - The Host informs the TPM of the counter value cnt and issuer ID which it wants to be used for the signature. It also (depending on the nature of the signing protocol) informs the TPM of the basename bsn and the message m being signed.
  - The TPM recovers the random coins r for the UKg algorithm by computing, for some Pseudo-Random Function PRF, the value $r = \mathsf{PRF}(\mathsf{DAASeed}\|\mathsf{cnt}\|\mathsf{ID})$.
  - The TPM calls $\mathsf{UKg}(\mathsf{param})$ with randomness r to recover the key $\mathfrak{sk}_{\mathsf{ID},\mathsf{cnt}}$.
  - The TPM and the Host interact using the protocols $(\mathsf{GSig}^{\mathsf{TPM}}, \mathsf{GSig}^{\mathsf{Host}})$ to obtain a signature on the message.
  - The verifier checks the signature by using the function $\mathsf{GVf}(\mathfrak{gmpk}, \sigma, m, \mathsf{bsn})$.
- $\mathsf{Identify}_{\mathsf{S}}(\sigma, m, \mathsf{bsn}, \mathfrak{sk}_i)$: This outputs 1 if the signature σ could have been produced with the key $\mathfrak{sk}_i$.
- $\mathsf{Link}(\mathfrak{gmpk}, \sigma, m, \sigma', m', \mathsf{bsn})$: returns 1 if and only if σ and σ' verify with respect to the basename bsn and when $\mathsf{bsn} \neq \perp$ we also have that σ and σ' were produced by the same user.

Note that using our $\mathsf{Identify}_{\mathsf{S}}$ and GVf algorithms we can create the functionality of using the $\mathsf{RogueList}$ in a DAA protocol: $\mathsf{RogueTag}$ adds a value of $\mathfrak{sk}_i$ to $\mathsf{RogueList}$ and if the verifier passes a $\mathsf{RogueList}$ to $\mathsf{DAA\text{-}Vf}$ then we modify $\mathsf{DAA\text{-}Vf}$ to additionally call $\mathsf{Identify}_{\mathsf{S}}$ for all $\mathfrak{sk}_i \in \mathsf{RogueList}$; rejecting the signature if any call to $\mathsf{Identify}_{\mathsf{S}}$ returns 1.

The security games for our DAA protocol then follow immediately from the equivalent security games of the pre-DAA scheme as soon as one deals with the corruption model for the Host. First we assume that an honest (resp. corrupt or broken) user in the pre-DAA model corresponds to an honest (resp. corrupt or broken) TPM in the security DAA model, and an honest (resp. dishonest) issuer in the pre-DAA model corresponds to an honest (resp. dishonest) issuer in the DAA model. For the anonymity game, a dishonest Host can always determine whether or not its embedded TPM was involved in some signature production protocol since the Host controls all communication with the TPM. Thus for the anonymity game for DAA we translate the equivalent pre-DAA anonymity game and assume that an honest TPM is always embedded in an honest Host. For the traceability game, there are no honest users, and hence no honest TPMs and Hosts and the issue does not arise. For non-frameability, we make no assumptions as to the honesty of the Hosts. However, our pre-DAA model translates directly, since we have assumed that the group signing key $\mathfrak{gsk}_i$ in the pre-DAA scheme is available to a dishonest issuer.

5 Adding authentication to a DAA scheme

The final part of the jigsaw in deriving a fully fledged DAA scheme is to determine how the TPM authenticates itself to the issuer in the Join protocol, or equivalently how the user authenticates itself to the issuer in our DAA scheme above. The standard way for this to be done in group signature schemes is for the users initial secret key to be associated to a public key. The public key is then authenticated by some PKI, and the communication from the user to the issuer is then authenticated, via digital signatures say, using the public key. For various reasons, which we discuss below, this is *not* the preferred option of the TCG group.

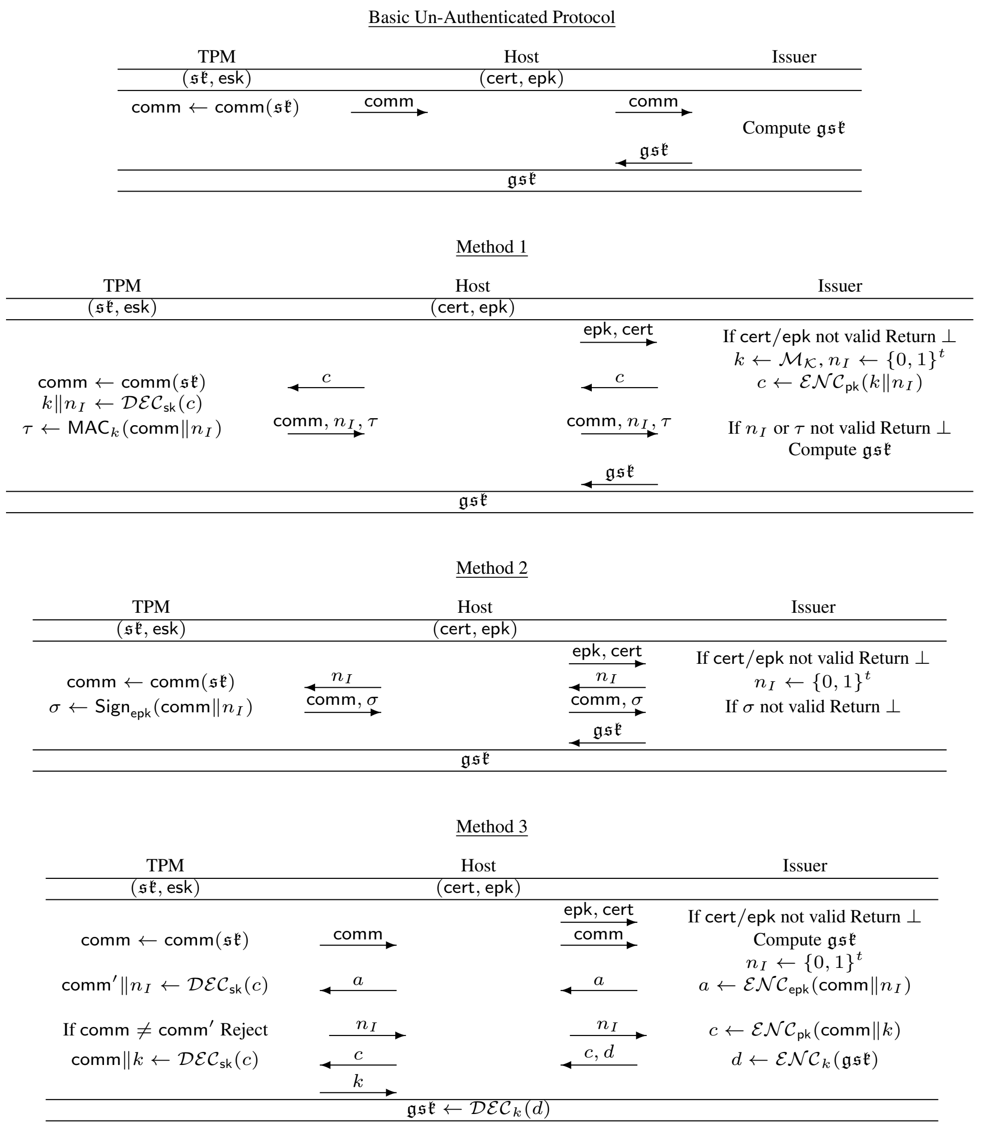
Basic Un-Authenticated Protocol
TPM
Host
Issuer
$(\mathfrak{sk}, \mathsf{esk})$
$(\mathsf{cert}, \mathsf{epk})$
$\mathsf{comm} \leftarrow \mathsf{comm}(\mathfrak{sk})$
comm
Compute $\mathfrak{gsk}$
$\mathfrak{gsk}$
Method 1
If cert/epk not valid Return $\perp$
$k \leftarrow \mathcal{M}_{\mathcal{K}}, n_I \leftarrow \{0,1\}^t$
$c \leftarrow \mathcal{ENC}_{\mathsf{pk}}(k \| n_I)$
$k \| n_I \leftarrow \mathcal{DEC}_{\mathsf{sk}}(c)$
$\tau \leftarrow \mathsf{MAC}_k(\mathsf{comm} \| n_I)$
comm, n_I, τ
If n_I or τ not valid Return $\perp$
Method 2
$\mathsf{epk}, \mathsf{cert}$
n_I
$\sigma \leftarrow \mathsf{Sign}_{\mathsf{epk}}(\mathsf{comm} \| n_I)$
comm, σ
If σ not valid Return $\perp$
Method 3
$\mathsf{comm}' \| n_I \leftarrow \mathcal{DEC}_{\mathsf{sk}}(c)$
a
$a \leftarrow \mathcal{ENC}_{\mathsf{epk}}(\mathsf{comm} \| n_I)$
If $\mathsf{comm} \neq \mathsf{comm}'$ Reject
$c \leftarrow \mathcal{ENC}_{\mathsf{pk}}(\mathsf{comm} \| k)$
$\mathsf{comm} \| k \leftarrow \mathcal{DEC}_{\mathsf{sk}}(c)$
c, d
$d \leftarrow \mathcal{ENC}_k(\mathfrak{gsk})$
k
$\mathfrak{gsk} \leftarrow \mathcal{DEC}_k(d)$

Figure 4: Three methods for authenticating the TPM

For DAA protocols, there are a number of methods in the literature to authenticate the user, all of which make use of so called *endorsement key*. It is for this reason that we examine the authentication of users as a separate operation in our presentation, so we can mix-and-match different authentication mechanisms. We assume the TPM upon manufacture is embedded with the private key esk of some public key algorithm, the associated public key epk being certified by some authority, and the resulting certificate $(\mathsf{cert}, \mathsf{epk})$ pair being stored by the Host.

There are a number of proposals in the literature for the use of the endorsement key. We highlight three proposals, all of which provide the necessary authentication, but all of which have different drawbacks and advantages. The three methods are summarised in Figure 4, where we assume a simple one-round issuing protocol (as for example in Figure 15). The generalisation of all four methods to more complex Join protocols is immediate. In the first two protocols we protect against replays by the issuer requiring the TPM to authenticate a specific nonce n_I. Most notation that we use in the figure is self-explanatory. The notation comm stands for a commitment to the secret key; notice that these are not necessarily cryptographic commitments but only some one-way function of $\mathfrak{sk}$.

Method 1. In [4] the endorsement key is a public key encryption key, with which the issuer encrypts a one-time authentication key (i.e. a MAC key) to the user. The user then authenticates his part of the issuing protocol by means of this authentication key. In [4], and in the deployed RSA based DAA protocol, this is done by computing a hash over the data and the authentication key, clearly a better solution would be to use a specially designed message authentication code, as in [23].

Method 2. In [22] the endorsement key is used in a different way; in particular, the endorsement key is the key for a public key signature algorithm. In this proposal the TPM signs the transcript using the signing key.

Method 3. In [24] the endorsement key is the key for a public encryption scheme. The idea is that before the issuer produces a certificate on the public key it runs a challenge-response protocol with the user to check that it is interacting with a valid TPM. If this part of the protocol terminates successfully, the issuer sends a hybrid encryption of the resulting group signing key under the endorsement key. The KEM part is forwarded by the Host to the TPM which decrypts it to reveal the symmetric encryption key for the DEM part, which he then sends to the Host. The Host obtains the group signing key in the obvious way.

All three of these proposals obtain the same effect, but with distinct side effects which we now discuss: The industrial group, TCG, behind the deployment of the DAA protocol prefer the encrypt followed by MAC solution as they are worried about the publicly verifiability of the signature variant enabling third parties to link different issuing protocols. Essentially, the Method 1 forms a *deniable authentic channel* from the TPM to the Issuer. Method 2 replaces the authentication via a MAC with authentication via a digital signature scheme, but unfortunately this clearly destroys deniability. Finally, Method 3 is close to Method 1 (with which it shares the overall structure) with the added advantage that its implementation is extremely simple (using the current set of TPM commands): it only requires two calls to the same TPM instruction. The protocol is also deniable: an execution of the protocol can be simulated by the issuer itself. The simplicity comes at the expense of a loss of anonymity: a curious issuer, or collusion of curious issuers, can still violate the anonymity in the issuing protocol using the encryption variant by maintaining information as to which authentication key was sent to which user. Nevertheless, this last method seems to be favoured by the TCG group for its TPM.Next specification.

6 Building blocks

In this section we present two new primitives which are variations of two classical primitives: blind signatures and message-authentication codes (MACs). We also recap on signature proofs of knowledge, which we will require in our construction.

6.1 Randomizable Weakly Blind Signatures

We start by giving a variant of a blind signature scheme in which a signer outputs a signature on a blinded message, but never gets to see the message he signed. Such a scheme will be the basis of our registration protocols, as the issuer will never see the user's secret key she signs. We also require that signatures can be *randomized*. Two example instantiations of this primitive can be found in Section 8.4.

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

$$\mathsf{BS} = (\mathsf{Setup}_{\mathsf{BS}}, \mathsf{KeyGen}_{\mathsf{BS}}, \mathsf{Request}_{\mathsf{BS}}, \mathsf{Issue}_{\mathsf{BS}}, \mathsf{Verify}_{\mathsf{BS}}, \mathsf{Randomize}_{\mathsf{BS}}) \ .$$

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

- $\mathsf{Setup}_{\mathsf{BS}}(1^\lambda)$ takes as input a security parameter λ and outputs a parameter set param, assumed to contain a description of the key and message spaces for BS.
- $\mathsf{KeyGen}_{\mathsf{BS}}(\mathsf{param})$ takes as input the system parameters and outputs a pair $(\mathsf{pk}_{\mathsf{BS}}, \mathsf{sk}_{\mathsf{BS}})$ of public/private keys for the signer.
- $(\mathsf{Request}^0_{\mathsf{BS}}, \mathsf{Issue}^1_{\mathsf{BS}}, \mathsf{Request}^1_{\mathsf{BS}})$ is an interactive protocol run between a user and a signer. The user goes first by calling $\mathsf{Request}^0_{\mathsf{BS}}(m, \mathsf{pk}_{\mathsf{BS}})$ to obtain a value ρ_0 and some state information St^0_R (which is assumed to contain m). Then the signer and user execute, respectively,

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

Experiment: $\mathsf{Exp}^{blind}_{\mathsf{BS},\mathcal{A}}(\lambda)$

```
• $\mathsf{param} \leftarrow \mathsf{Setup}_{\mathsf{BS}}(1^\lambda)$.
• $(\mathsf{pk}_{\mathsf{BS}}, \mathsf{sk}_{\mathsf{BS}}) \leftarrow \mathsf{KeyGen}_{\mathsf{BS}}(1^\lambda)$.
• $(m_0, m_1, \mathsf{St}^0_{\mathsf{issue}})$
          $\leftarrow \mathcal{A}(\mathsf{find}, \mathsf{pk}_{\mathsf{BS}}, \mathsf{sk}_{\mathsf{BS}}, \mathsf{param})$.
• $b \leftarrow \{0,1\}$.
• $(\rho_0, \mathsf{St}^0_R) \leftarrow \mathsf{Request}^0_{\mathsf{BS}}(m_b, \mathsf{pk}_{\mathsf{BS}})$.
• For $i = 1$ to $r$ do
      $(\beta_i, \mathsf{St}^i_{\mathsf{issue}}) \leftarrow \mathcal{A}(\mathsf{issue}, i, \rho_{i-1}, \mathsf{St}^{i-1}_{\mathsf{issue}})$.
      $(\rho_i, \mathsf{St}^i_R) \leftarrow \mathsf{Request}^i_{\mathsf{BS}}(\beta_i, \mathsf{St}^{i-1}_R)$.
• $\sigma_b \leftarrow \rho_r$.
• $(\rho_0, \mathsf{St}^0_R) \leftarrow \mathsf{Request}^0_{\mathsf{BS}}(m_{1-b}, \mathsf{pk}_{\mathsf{BS}})$.
• For $i = 1$ to $r$ do
      $(\beta_i, \mathsf{St}^i_{\mathsf{issue}}) \leftarrow \mathcal{A}(\mathsf{issue}, i, \rho_{i-1}, \mathsf{St}^{i-1}_{\mathsf{issue}})$.
      $(\rho_i, \mathsf{St}^i_R) \leftarrow \mathsf{Request}^i_{\mathsf{BS}}(\beta_i, \mathsf{St}^{i-1}_R)$.
• $\sigma_{1-b} \leftarrow \rho_r$.
• If $\sigma_0 = \perp$ or $\sigma_1 = \perp$ then abort.
• $b^* \leftarrow \mathcal{A}(\mathsf{guess}, \sigma_0, \sigma_1, \mathsf{St}^r_{\mathsf{issue}})$.
• Return 1 if $b = b^*$ else return 0.
```

Experiment: $\mathsf{Exp}^{weak\text{-}blind}_{\mathsf{BS},\mathcal{A}}(\lambda)$

```
• $\mathsf{param} \leftarrow \mathsf{Setup}_{\mathsf{BS}}(1^\lambda)$.
• $(\mathsf{pk}_{\mathsf{BS}}, \mathsf{sk}_{\mathsf{BS}}) \leftarrow \mathsf{KeyGen}_{\mathsf{BS}}(1^\lambda)$.
• $m_0, m_1 \leftarrow \mathcal{M}$.
• $\mathsf{St}^0_{\mathsf{issue}} \leftarrow \mathcal{A}(\mathsf{issue}, \mathsf{pk}_{\mathsf{BS}}, \mathsf{sk}_{\mathsf{BS}}, \mathsf{param})$.
• $(\rho_0, \mathsf{St}^0_R) \leftarrow \mathsf{Request}^0_{\mathsf{BS}}(m_0, \mathsf{pk}_{\mathsf{BS}})$.
• For $i = 1$ to $r$ do
      $(\beta_i, \mathsf{St}^i_{\mathsf{issue}}) \leftarrow \mathcal{A}(\mathsf{issue}, i, \rho_{i-1}, \mathsf{St}^{i-1}_{\mathsf{issue}})$.
      $(\rho_i, \mathsf{St}^i_R) \leftarrow \mathsf{Request}^i_{\mathsf{BS}}(\beta_i, \mathsf{St}^{i-1}_R)$.
• $\sigma_0 \leftarrow \rho_r$.
• If $\sigma_0 = \perp$ then abort.
• $b \leftarrow \{0,1\}$.
• If $b = 0$ then
      $\sigma_1 \leftarrow \mathsf{Randomize}_{\mathsf{BS}}(\sigma_0)$.
• Else
      $(\rho_0, \mathsf{St}^0_R) \leftarrow \mathsf{Request}^0_{\mathsf{BS}}(m_1, \mathsf{pk}_{\mathsf{BS}})$.
      $\mathsf{St}^0_I \leftarrow \mathsf{sk}_{\mathsf{BS}}$.
      For $i = 1$ to $r$ do
          $(\beta_i, \mathsf{St}^i_I) \leftarrow \mathsf{Issue}^i_{\mathsf{BS}}(\rho_{i-1}, \mathsf{St}^{i-1}_I)$.
          $(\rho_i, \mathsf{St}^i_R) \leftarrow \mathsf{Request}^i_{\mathsf{BS}}(\beta_i, \mathsf{St}^{i-1}_R)$.
      $\sigma_1 \leftarrow \rho_r$.
• $b^* \leftarrow \mathcal{A}(\mathsf{guess}, \sigma_0, \sigma_1, \mathsf{St}^r_{\mathsf{issue}})$.
• Return 1 if $b = b^*$ else return 0.
```

Figure 5: Two notions of blindness for a blind signature scheme

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

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

for the output of correctly running this interactive protocol on the given inputs.

- $\mathsf{Verify}_{\mathsf{BS}}(m, \sigma, \mathsf{pk}_{\mathsf{BS}})$ is the public signature verification algorithm, which outputs 1 if σ is a valid signature on m and 0 otherwise.
- $\mathsf{Randomize}_{\mathsf{BS}}(\sigma)$ 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 $\mathsf{Setup}_{\mathsf{BS}}$ we have

$$\Pr\big[\, (\mathsf{pk}_{\mathsf{BS}}, \mathsf{sk}_{\mathsf{BS}}) \leftarrow \mathsf{KeyGen}_{\mathsf{BS}}(1^\lambda),\ m \leftarrow \mathcal{M}, \\ \sigma \leftarrow (\mathsf{Request}_{\mathsf{BS}}(m, \mathsf{pk}_{\mathsf{BS}}) \Longleftrightarrow \mathsf{Issue}_{\mathsf{BS}}(\mathsf{sk}_{\mathsf{BS}})) : \mathsf{Verify}_{\mathsf{BS}}(m, \sigma, \mathsf{pk}_{\mathsf{BS}}) = 1 \,\big] = 1 \ .$$

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 $\mathsf{KeyGen}_{\mathsf{BS}}$ we have for all m and σ

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

Security. The standard security model for blind signatures [31, 34] consists of two properties: blindness and unforgeability. In the traditional security model blindness states that an adversarial signer, who can choose two messages m_0 and m_1, cannot tell in which order the messages were asked to be signed, when presented with the

Experiment: $\mathsf{Exp}^{forge}_{\mathsf{BS},\mathcal{A}}(\lambda)$

- $\mathsf{param} \leftarrow \mathsf{Setup}_{\mathsf{BS}}(1^\lambda)$.
- $(\mathsf{pk}_{\mathsf{BS}}, \mathsf{sk}_{\mathsf{BS}}) \leftarrow \mathsf{KeyGen}_{\mathsf{BS}}(1^\lambda)$.
- $((m_1, \sigma_1), \ldots, (m_{k+1}, \sigma_{k+1}))$
  $\leftarrow \mathcal{A}^{\mathsf{Issue}^i_{\mathsf{BS}}(\cdot,\cdot)}(\mathsf{pk}_{\mathsf{BS}}, \mathsf{param})$.
- Return 0 if one of the following holds:
  - $\mathcal{A}$ called its oracle more than k times.
  - $\exists i, j \in \{1, \ldots, k+1\}$, with $i \neq j$, such that $m_i = m_j$.
  - $\exists i \in \{1, \ldots, k+1\}$ such that $\mathsf{Verify}_{\mathsf{BS}}(m_i, \sigma_i, \mathsf{pk}_{\mathsf{BS}}) = 0$.
- Return 1.

Figure 6: Forgery security game for a blind signature scheme

final signatures. More formally, we consider an adversary $\mathcal{A}$ which has three modes find, issue and guess, running in the experiment $\mathsf{Exp}^{blind}_{\mathsf{BS},\mathcal{A}}(\lambda)$ of Figure 5.

This traditional model is unnecessarily strong for us, since we are never going to output the messages for the adversary to see. Instead, what we require is that an adversary impersonating a possibly dishonest issuer that issues a blind signature on a message unknown to him cannot distinguish a randomization of the resulting signature from a blind signature on a different message. This is captured in experiment $\mathsf{Exp}^{weak\text{-}blind}_{\mathsf{BS},\mathcal{A}}(\lambda)$ of Figure 5, with an adversary running in two modes issue and guess. We define

$$\mathsf{Adv}^{weak\text{-}blind}_{\mathsf{BS},\mathcal{A}}(\lambda) = 2 \cdot \left| \Pr[\mathsf{Exp}^{weak\text{-}blind}_{\mathsf{BS},\mathcal{A}}(\lambda) = 1] - 1/2 \right|$$

and say that the scheme is *weakly blind* if $\mathsf{Adv}^{weak\text{-}blind}_{\mathsf{BS},\mathcal{A}}(\lambda)$ is a negligible function of λ for any polynomial-time adversary $\mathcal{A}$.

On the other hand, unforgeability deals with an adversarial user whose goal is to obtain signatures on $k+1$ different messages given only k interactions with the honest signer. Formally, we consider an adversary $\mathcal{A}$, having oracle access to the function $\mathsf{Issue}^i_{\mathsf{BS}}(\rho, \mathsf{sk}_{\mathsf{BS}})$, running in the forge experiment in Figure 6. We define

$$\mathsf{Adv}^{forge}_{\mathsf{BS},\mathcal{A}}(\lambda) = \Pr[\mathsf{Exp}^{forge}_{\mathsf{BS},\mathcal{A}}(\lambda) = 1]$$

and say that the scheme is *unforgeable* if $\mathsf{Adv}^{forge}_{\mathsf{BS},\mathcal{A}}(\lambda)$ is a negligible function of λ for any polynomial-time adversary $\mathcal{A}$.

Simulatable blind signatures. In order to instantiate our pre-DAA scheme, we require an additional property of a blind signature scheme. We call it *issuer-simulatable* if there exists $\mathsf{SimIssue}_{\mathsf{BS}}$ simulating $\mathsf{Issue}_{\mathsf{BS}}$ as follows: An adversary that is allowed to interact with an $\mathsf{Issue}_{\mathsf{BS}}$ oracle an arbitrary number of times, but requesting a signature on the same message each time. We require that such adversary cannot detect if the oracle is replaced by $\mathsf{SimIssue}_{\mathsf{BS}}$, which instead of getting the signing key is given one-time access to an $\mathsf{Issue}_{\mathsf{BS}}$ oracle.

6.2 Linkable Indistinguishable Tags

Our second primitive is called *Linkable Indistinguishable Tag* (LIT). Unlike message authentication codes (MACs), tags need not be unforgeable for our construction. We note that our example instantiation in Section 8.5, which is essentially a deterministic digital signature scheme "in disguise", can however be proved to be UF-CMA secure as a standard MAC algorithm.

Syntax. A LIT is given by a pair of algorithms $(\mathsf{KeyGen}_{\mathsf{LIT}}, \mathsf{Tag}_{\mathsf{LIT}})$.

- $\mathsf{KeyGen}_{\mathsf{LIT}}(1^\lambda)$: This outputs a key $\mathfrak{sk}$, pulled from some space $\mathcal{K}_{\mathsf{LIT}}$ of size 2^λ. This algorithm also implicitly sets the underlying message space $\mathcal{M}_{\mathsf{LIT}}$.

Experiment: $\mathsf{Exp}^{f\text{-}\mathsf{IND}}_{\mathsf{LIT},\mathcal{A}}(\lambda)$

- $\mathfrak{sk}_0, \mathfrak{sk}_1 \leftarrow \mathsf{KeyGen}_{\mathsf{LIT}}(1^\lambda)$.
- $c \leftarrow f(\mathfrak{sk}_0)$.
- $b \leftarrow \{0,1\}$.
- $(m^*, \mathsf{St}) \leftarrow \mathcal{A}_1^{\mathsf{Tag}_{\mathsf{LIT}}(\cdot,\mathfrak{sk}_0)}(c)$.
- $\tau^* \leftarrow \mathsf{Tag}_{\mathsf{LIT}}(m^*, \mathfrak{sk}_b)$.
- $b^* \leftarrow \mathcal{A}_2^{\mathsf{Tag}_{\mathsf{LIT}}(\cdot,\mathfrak{sk}_0)}(\tau^*, \mathsf{St})$.
- If m^* was asked of $\mathsf{Tag}_{\mathsf{LIT}}$ return 0.
- Return 1 if $b^* = b$, else 0.

Experiment: $\mathsf{Exp}^{\mathsf{LINK}}_{\mathsf{LIT},\mathcal{A}}(\lambda)$

- $(m_0, \tau_0, \mathfrak{sk}_0, m_1, \tau_1, \mathfrak{sk}_1) \leftarrow \mathcal{A}(1^\lambda)$.
- Return 1 if and only if :
  - $\mathsf{Tag}_{\mathsf{LIT}}(m_0, \mathfrak{sk}_0) = \tau_0$
  - $\mathsf{Tag}_{\mathsf{LIT}}(m_1, \mathfrak{sk}_1) = \tau_1$.
  - $\tau_0 = \tau_1$.
  - Either $\mathfrak{sk}_0 \neq \mathfrak{sk}_1$ or $m_0 \neq m_1$.

Figure 7: The IND and LINK experiments for a LIT

- $\mathsf{Tag}_{\mathsf{LIT}}(m, \mathfrak{sk})$: Given a message $m \in \mathcal{M}_{\mathsf{LIT}}$ and a key, this deterministic algorithm produces the authentication tag $\tau \in \mathcal{T}_{\mathsf{LIT}}$.

Since we restrict ourselves to *deterministic* tag algorithms, $\mathsf{Tag}_{\mathsf{LIT}}$ is a function. This makes verification trivial: to verify a tuple $(m, \mathfrak{sk}, \tau)$, check whether $\mathsf{Tag}_{\mathsf{LIT}}(m, \mathfrak{sk}) = \tau$.

Security. An adversary can break a LIT in one of two ways: by breaking an indistinguishability property, or by breaking a linkability property. In the first case we give the adversary the image of the secret key under a one-way function f; security is therefore also relative to this function. The one-way function acts like a public key corresponding to the secret key. However, unlike in a public-key signature scheme, the one-way function does not allow one to publicly verify a given message/tag pair. This function f allows us to tie up the LIT with the blind signature schemes presented earlier.

Indistinguishability, with respect to f, is defined as being unable, given access to a tag oracle for one key, to tell whether a new tag on an adversarily chosen message is for the same key or not. Formally this is described in Figure 7, where the adversary is not allowed to query its $\mathsf{Tag}_{\mathsf{LIT}}$ oracle with the message m^*. We define $\mathsf{Adv}^{f\text{-}\mathsf{IND}}_{\mathsf{LIT},\mathcal{A}}(\lambda)$ to be the probability $2 \cdot |\Pr[\mathsf{Exp}^{f\text{-}\mathsf{IND}}_{\mathsf{LIT},\mathcal{A}}(\lambda) = 1] - 1/2|$.

We define the linkability game as in Figure 7. Linkability does not depend on any secret key; it must hold even for adversarially chosen keys. Intuitively, linkability should guarantee that an adversary cannot produce two valid tags which are equal, unless they are tags on the same message/key pair. We define

$$\mathsf{Adv}^{\mathsf{LINK}}_{\mathsf{LIT},\mathcal{A}}(\lambda) := \Pr[\mathsf{Exp}^{\mathsf{LINK}}_{\mathsf{LIT},\mathcal{A}}(\lambda) = 1] \ .$$

6.3 Signature proofs of knowledge

An NP statement is a statement whose validity can be efficiently checked given a *witness* for that statement. A signature proof of knowledge (SPK) [19] is a non-interactive algorithm which takes a statement, some witness for its validity, and a message m, and outputs a string σ:

$$\sigma \leftarrow \mathsf{SPoK}(\{\mathsf{witness}\} : \mathsf{statement})(m) \ .$$

In the random oracle model (ROM) a SPK can be constructed efficiently, via the Fiat–Shamir [26] heuristic, if the NIZK proof associated with $\mathsf{NIZK}(\{\mathsf{witness}\} : \mathsf{statement})$ can be derived from a Sigma protocol.

We write $\mathsf{Verify}_{\mathsf{SPoK}}(\sigma, \mathsf{statement}, m)$ for the procedure of verifying the signature proof of knowledge, and say a signature proof of knowledge is valid if the verification procedure outputs 1. Such a signature proof of knowledge derived from a Sigma protocol has three properties, the second of which follows from the Forking Lemma [34].

- Perfect Correctness: If the witness is indeed a witness for the truth of the statement and the SPoK algorithm is executed correctly then $\mathsf{Verify}_{\mathsf{SPoK}}(\sigma, \mathsf{statement}, m)$ will always output 1.
- Soundness: If an algorithm $\mathcal{A}$, working in the ROM, produces a valid proof with probability ϵ then there exists an algorithm $\mathcal{B}$ which can program the underlying random oracle and which will output the witness with probability ϵ^2/q, where q is the number of queries to the random oracle.
- Zero-Knowledge: By programming the random oracle, valid SPKs of (even false) statements can be produced without a witness. We call the algorithm which produces such fake proofs the *simulator*.

```
$\mathsf{Setup}(1^\lambda)$:
  • $\mathsf{param} \leftarrow \mathsf{Setup}_{\mathsf{BS}}(1^\lambda)$. Return $\mathsf{param}$.

$\mathsf{GKg}(\mathsf{param})$:
  • $(\mathfrak{gmpk}, \mathfrak{gmsk}) \leftarrow \mathsf{KeyGen}_{\mathsf{BS}}(1^\lambda)$.
  • Return $(\mathfrak{gmpk}, \mathfrak{gmsk})$.

$\mathsf{UKg}(\mathsf{param})$:
  • $\mathfrak{sk}_i \leftarrow \mathsf{KeyGen}_{\mathsf{LIT}}(1^\lambda)$. Return $\mathfrak{sk}_i$.

$\mathsf{GSig}(\mathfrak{gsk}_i, \mathfrak{sk}_i, m, \mathsf{bsn})$:
  • $\sigma_0 \leftarrow \mathsf{Randomize}_{\mathsf{BS}}(\mathfrak{gsk}_i)$.
  • If $\mathsf{bsn} \neq \perp$
      • $\tau \leftarrow \mathsf{Tag}_{\mathsf{LIT}}(\mathsf{bsn}, \mathfrak{sk}_i)$.
      • $\Sigma \leftarrow \mathsf{SPoK}\big(\{\mathfrak{sk}_i\} : \mathcal{L}(\cdot, \sigma_0, \mathsf{bsn}, \tau, \mathfrak{gmpk})\big)(\mathsf{bsn}\|m)$.
  • Else
      • $\tau \leftarrow \emptyset$.
      • $\Sigma \leftarrow \mathsf{SPoK}\big(\{\mathfrak{sk}_i\} : \mathcal{L}'(\cdot, \sigma_0, \mathfrak{gmpk})\big)(\mathsf{bsn}\|m)$.
  • $\sigma \leftarrow (\tau, \sigma_0, \Sigma)$.

$\mathsf{Identify}_{\mathsf{S}}(\sigma, m, \mathsf{bsn}, \mathfrak{sk}_i)$:
  • Parse $\sigma$ as $(\tau, \sigma_0, \Sigma)$.
  • If $\mathsf{Verify}_{\mathsf{BS}}(\mathfrak{sk}_i, \sigma_0, \mathfrak{gmpk}) = 0$
      • Return 0
  • Return 1 iff one of the following hold
      • $\mathsf{bsn} = \perp$ and $\tau = \emptyset$.
      • $\mathsf{Tag}_{\mathsf{LIT}}(\mathsf{bsn}, \mathfrak{sk}_i) = \tau$.

$(\mathsf{Join}, \mathsf{Iss})$:
  • Execute $(\mathsf{Request}_{\mathsf{BS}}, \mathsf{Issue}_{\mathsf{BS}})$ for message $\mathfrak{sk}_i$.
  • User has input $(\mathfrak{sk}_i, \mathfrak{gmpk})$.
  • Issuer has input $\mathsf{sk}_{\mathsf{BS}}$.
  • The issuer's and user's output is $\mathfrak{gsk}_i = \sigma$.
  • Note that the first (user-sent) message will be $f(\mathfrak{sk}_i)$ by assumption.

$\mathsf{GVf}(\mathfrak{gmpk}, \sigma, m, \mathsf{bsn})$:
  • Parse $\sigma$ as $(\tau, \sigma_0, \Sigma)$.
  • If $\mathsf{bsn} \neq \perp$ return
        $\mathsf{Verify}_{\mathsf{SPoK}}\big(\Sigma, \mathcal{L}(\cdot, \sigma_0, \mathsf{bsn}, \tau, \mathfrak{gmpk}), (\mathsf{bsn}\|m)\big)$.
  • If $\tau = \emptyset$ return
        $\mathsf{Verify}_{\mathsf{SPoK}}\big(\Sigma, \mathcal{L}'(\cdot, \sigma_0, \mathfrak{gmpk}), (\perp\|m)\big)$.
  • Return 0

$\mathsf{Identify}_{\mathsf{T}}(\mathcal{T}, \mathfrak{sk}_i)$:
  • Check if transcript is valid from the point of view of a user who sent $f(\mathfrak{sk}_i)$ as the first message.
  • If so return 1, otherwise return 0.

$\mathsf{Link}(\mathfrak{gmpk}, \sigma_0, m_0, \sigma_1, m_1, \mathsf{bsn})$:
  • If $\mathsf{GVf}(\mathfrak{gmpk}, \sigma_0, m_0, \mathsf{bsn}) = 0$ return $\perp$.
  • If $\mathsf{GVf}(\mathfrak{gmpk}, \sigma_1, m_1, \mathsf{bsn}) = 0$ return $\perp$.
  • If $\mathsf{bsn} = \perp$ return 0.
  • Parse $\sigma_0$ as $(\tau_0, \sigma_0', \Sigma_0)$ and $\sigma_1$ as $(\tau_1, \sigma_1', \Sigma_1)$.
  • Return 1 if and only if $\tau_0 = \tau_1$.
```

Figure 8: General pre-DAA scheme construction in the ROM

7 A pre-DAA scheme in the random oracle model

Our construction of a pre-DAA scheme combines a randomizable weakly blind signature scheme and a Linkable Indistinguishable Tag (LIT), as introduced in Section 6, which are *compatible* in a sense to be defined below. The basic idea of our construction is that every user holds a secret key for a LIT scheme, and when they join the group, they are issued a blind signature on this secret key. To issue a signature on a basename/message pair, a user randomizes the blind signature and computes a LIT on the basename, and then provides a zero-knowledge signature proof of knowledge that they know the secret key which verifies the LIT and which is signed by the blind signature.

To enable this to work we require the two component schemes to be compatible. It is readily verified that our example constructions of both primitives, given in Sections 8.4 and 8.5, all satisfy the following requirements.

Definition 2. *A randomizable weakly blind signature scheme and a LIT are* compatible*, if the following four conditions hold, for the same injective one-way function f;*

- *The key space of the LIT is equal to the message space of the blind signature scheme.*
- *The LIT is indistinguishable w.r.t. f and linkable as defined in Section 6.*

Table 1: Security Properties

Security property of pre-DAA	Underlying primitive	Security property of primitive
Anonymity	SPK LIT Blind Signature	Zero-knowledge Indistinguishability Weak blindness
Traceability 1	SPK Blind Signature	Soundness Unforgeability, issuer-simulatability
Traceability 2	–	–
Non-frameability 1	SPK LIT f	Zero-knowledge, soundness Indistinguishability One-wayness
Non-frameability 2	SPK LIT	Soundness Linkability

- *The (one-round) blind signature scheme is weakly blind, unforgeable and issuer-simulatable as defined in Section 6.*
- *In the blind signature issuing protocol the user's first message is f of the message to be signed. Moreover, from the output of $\mathsf{Issue}_{\mathsf{BS}}$ one can derive a blind signature, whose validity can be checked given $f(m)$.*

General construction. We present our construction of a pre-DAA scheme from a randomizable weakly blind signature scheme and a LIT, which are compatible w.r.t. an injective one-way function f. Denote the two schemes by

$$\begin{aligned} \mathsf{BS} &= (\mathsf{Setup}_{\mathsf{BS}}, \mathsf{KeyGen}_{\mathsf{BS}}, \mathsf{Request}_{\mathsf{BS}}, \mathsf{Issue}_{\mathsf{BS}}, \mathsf{Verify}_{\mathsf{BS}}, \mathsf{Randomize}_{\mathsf{BS}}) \ , \\ \mathsf{LIT} &= (\mathsf{KeyGen}_{\mathsf{LIT}}, \mathsf{Tag}_{\mathsf{LIT}}) \ . \end{aligned}$$

In addition, we assume the existence of Sigma protocols for the following two languages, written as NP relations, whose two components correspond to the statement and its witness, respectively:

$$\begin{aligned} \mathcal{L}: &\quad \big\{\big((\mathsf{pk}_{\mathsf{BS}}, \sigma, m, \tau), \mathfrak{sk}\big) \ : \ \mathsf{Verify}_{\mathsf{BS}}(\mathfrak{sk}, \sigma, \mathsf{pk}_{\mathsf{BS}}) = 1 \ \wedge \ \mathsf{Verify}_{\mathsf{MAC}}(m, \tau, \mathfrak{sk}) = 1\big\} \\ \mathcal{L}': &\quad \big\{\big((\mathsf{pk}_{\mathsf{BS}}, \sigma), \mathfrak{sk}\big) \ : \ \mathsf{Verify}_{\mathsf{BS}}(\mathfrak{sk}, \sigma, \mathsf{pk}_{\mathsf{BS}}) = 1\big\} \end{aligned}$$

From these Sigma protocols we derive a signature proof of knowledge for the corresponding language described above. In what follows, we let $\mathcal{L}(\mathfrak{sk}, \sigma, m, \tau, \mathsf{pk}_{\mathsf{BS}})$ denote the statement $\mathsf{Verify}_{\mathsf{BS}}(\mathfrak{sk}, \sigma, \mathsf{pk}_{\mathsf{BS}}) = 1 \wedge \mathsf{Tag}_{\mathsf{LIT}}(m, \mathfrak{sk}) = \tau$, and let $\mathcal{L}'(\mathfrak{sk}, \sigma, \mathsf{pk}_{\mathsf{BS}})$ denote the statement $\mathsf{Verify}_{\mathsf{BS}}(\mathfrak{sk}, \sigma, \mathsf{pk}_{\mathsf{BS}}) = 1$. The algorithms for our pre-DAA scheme are presented in Figure 8. Note that we do not require the user to prove knowledge of his key for the LIT in the Join stage, unlike various previous DAA scheme proposals. This is because if the user does not know the key then they will not be able to sign messages, and we do not need to rewind a user during the Join protocol in any of our security proofs.

When instantiated with our example weakly blind signature schemes and Linkable Indistinguishable Tags, we obtain a highly efficient pre-DAA scheme, details of which are given in Section 9. This enables us, via the discussion in Section 4, to obtain a very efficient full DAA scheme based on pairings. The efficiency of our resulting scheme is better than the existing deployed one based on RSA, and as efficient as all prior ones based on pairings; with the benefit that our scheme comes with a fully expressed security model and proof.

Theorem 1. *In the random-oracle model there are efficient reductions of each of the security properties of our pre-DAA construction to properties of the underlying signature proof of knowledge, the function f, the Linkable Indistinguishable Tag or the weakly blind signature, as summarised in Table 1.*

7.1 Proof of Theorem 1

Our construction builds a pre-DAA scheme from an injective one-way function f, a Linkable Indistinguishable Tag and a randomizable weakly blind signature scheme. Throughout the proof it is worth keeping in mind that a signature of a pre-DAA scheme consists of three parts:

- A LIT tag on the basename (which is empty if $\mathsf{bsn} = \perp$).

- A (randomized) blind signature.
- A signature proof of knowledge (SPK) on the message and basename proving that the secret key for the LIT and the message of the blind signature are the same and known to the signer.

Note that tags and blind signatures that are part of an honest signer's signature could be reused by an adversary. The unforgeability notions of the scheme rely thus crucially on the signature of knowledge of the user secret key.

We will now prove that the scheme defined in Figure 8 satisfies the definitions from Section 3.

Correctness. This can be checked by just working through the protocol.

Uniquely identifiable transcripts. Given a secret key $\mathfrak{sk}$, we define Check_T to check whether the user's first message is $f(\mathfrak{sk})$ (which is the case by Definition 2). Uniqueness holds thus by injectivity of f.

Anonymity. We will show that any efficient adversary with non-negligible advantage in the anonymity game can be used to break the underlying signature proof of knowledge, the blind signature scheme, or the LIT. For $b = 0$ and $b = 1$, we will define a sequence of five games, where Game 1 is $\mathsf{Exp}_{\mathcal{A}}^{anon\text{-}b}(\lambda)$ when the experiment guesses correctly the challenge user, and Game 5 is independent of b. If we then prove that $\mathcal{A}$ behaves differently in two consecutive games only with negligible probability, we have

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

is negligible, and the scheme satisfies thus 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 when $\mathcal{A}$ does not pick as a challenge a user which was preselected beforehand. Let $q_{\mathcal{A}}$ be a (polynomial) bound on the number of users $\mathcal{A}$ can create. Pick $i \in_R \{1, \ldots, q_{\mathcal{A}}\}$ uniformly at random. Since i is independently chosen, the probability that i equals a particular user in the experiment run is $\frac{1}{q_{\mathcal{A}}}$. Thus, we have

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

Lemma 1. *If $\mathcal{A}$ has non-negligible advantage in winning the anonymity game then the probability that $\mathcal{A}$ wins the game and 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.

We start with a first intuition of how to convert the game into one that is independent of b. When the adversary calls the challenge oracle it gets a signature $\sigma = (\tau, \sigma_0, \Sigma)$, where τ is a LIT under key $\mathfrak{sk}_{i_b}$ and σ_0 is a blind signature on $\mathfrak{sk}_{i_b}$. In our sequence of games we could thus first replace the SPK Σ by a simulated proof, and simulate the join protocol for our target user i. We could then replace τ by a tag under a random key (by indistinguishability of the LIT), and finally replace σ_0 by a signature on a random message. This last game would then not involve $\mathfrak{sk}_{i_b}$ anymore and would thus be independent of the bit b.

It is in the last step that the intuition fails however: if we wanted to reduce a non-negligibly different behaviour of $\mathcal{A}$ to breaking weak blindness, we would have to simulate the anonymity game of our pre-DAA scheme. This includes answering signing queries on behalf of user i, which necessitates $\mathfrak{sk}_i$, i.e., the message of the blind signature $\mathfrak{gsk}_i$, which the adversary in the weak blindness game does not obtain. We thus have to replace *all* the LITs produced by user i by LITs under random keys in the previous game.

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 for the SPK contained in the challenge signature, we use the simulator for the underlying ZK protocol. If this fails, we abort the game.

It follows from the zero-knowledge property of the signature proof of knowledge that the difference between Game 1 and 2 is negligible.

Game 3. In this game, when the adversary requests user i to join, we send $f(\mathfrak{sk}_i)$ to the adversary and then derive the blind signature from its response, using $f(\mathfrak{sk}_i)$ to check its validity.

By the last property of Definition 2, satisfied by our blind signature scheme, $f(\mathfrak{sk}_i)$ suffices to simulate $\mathsf{Join}_{\mathsf{BS}}$.

Game 4. Game 4 is defined as Game 3, except that whenever the experiment creates a signature on behalf of user i (i.e. when $\mathcal{A}$ calls CH_b or Sign for 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 an independently uniformly random key $\mathfrak{sk}$ and set $\tau \leftarrow \mathsf{Tag}_{\mathsf{LIT}}(\mathsf{bsn}, \mathfrak{sk})$ rather than using $\mathfrak{sk}_i$.

First note that reusing the tag for a previously queried basename does not alter the experiment, since we assumed our LITs to be deterministic. Secondly, since the SPK is simulated, we do not require a correct witness.

Games 3 and 4 are shown to be negligibly close by a hybrid argument and reductions to LIT indistinguishability. Let $s_{\mathcal{A}}$ be a (polynomial) upper bound on the number of Sign queries $\mathcal{A}$ can make. We define a sequence of games $(G_j)_{j=0}^{s_{\mathcal{A}}+1}$ such that in G_j, we answer the first j queries to Sign for user i or to CH_b for *distinct* basenames by using the secret key $\mathfrak{sk}_i$ (that was used in the Join protocol) for the LIT; from query $j+1$ on we use independent random keys for the LITs. This construction gives us G_0 = Game 4 and $G_{s_{\mathcal{A}}+1}$ = Game 3; 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 LIT indistinguishability (see Figure 7) if there is a non-negligible difference between G_j and G_{j+1}. Adversary $\mathcal{B}$ is given $c = f(\mathfrak{sk}_0)$ by its challenger and simulates the anonymity game for $\mathcal{A}$ using c as the blinded secret key for user i in the Join protocol. For the first j distinct tags in $\mathcal{A}$'s queries for user i, $\mathcal{B}$ uses its $\mathsf{Tag}_{\mathsf{LIT}}$ oracle, for the next query, $\mathcal{B}$ forwards the queried basename to its challenger and uses the received tag, whereas for the remaining queries, it uses independent random keys. Eventually, $\mathcal{B}$ outputs whatever $\mathcal{A}$ outputs.

If the bit that $\mathcal{B}$'s challenger flipped is 0 then the first $j+1$ queries are answered using user i's secret key; the game $\mathcal{A}$ is playing is thus G_{j+1}. Whereas if the challenger's bit is 1 then $\mathcal{A}$ is playing G_j. Thus by the security of the LIT scheme, the difference between two games is negligible and thus so is the difference between Games 3 and 4.

Game 5. The difference between this game and the previous one is that after running Join for user i, we discard the obtained $\mathfrak{gsk}_i$ and replace it with a blind signature on a random secret key. Observe that the game is now independent of the bit b.

If the difference between Games 4 and 5 was non-negligible, we could build an adversary $\mathcal{B}$ that breaks weak blindness (see Figure 5) of the scheme BS as follows. Adversary $\mathcal{B}$ receives $\mathsf{param}, \mathsf{pk}_{\mathsf{BS}}$ and $\mathsf{sk}_{\mathsf{BS}}$ from its challenger and hands it to $\mathcal{A}$ as $\mathfrak{gmpk} = \mathsf{pk}_{\mathsf{BS}}$ and $\mathfrak{gmsk} = \mathsf{sk}_{\mathsf{BS}}$. It simulates Game 4 for $\mathcal{A}$, except when joining user i, it relays $\mathcal{A}$'s messages impersonating the issuer to its challenger. After obtaining σ_0 from the challenger, $\mathcal{B}$ sets $\mathfrak{gsk}_i := \sigma_0$ 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 $\mathfrak{gsk}_i$ is a randomization of the signature that $\mathcal{A}$ issued; $\mathcal{A}$ plays thus Game 4. If the bit was 1 then $\mathfrak{gsk}_i$ is set to a signature on a random message, i.e. a random LIT key, which means that $\mathcal{A}$ is playing Game 5.

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 LIT key was used elsewhere in the experiment. We have proved the following theorem:

Theorem 2. *If the underlying blind signature scheme is weakly blind, the signature proof of knowedge is zero-knowledge, and the LIT is indistinguishable then our pre-DAA scheme has the anonymity property.*

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 signature/message/basename triple that verifies and a collection of secret keys such that all transcripts accepted by the honest issuer identify to one of these keys, but the signature does not.

Intuitively, if from the signature proof of knowledge that the adversary returns we extract the secret key $\mathfrak{sk}$, we get a blind-signature/message forgery: $\mathfrak{sk}$ has never been signed by the issuer since it is different from all the keys associated to the transcripts. There is one issue that needs to be taken care of: if the adversary registers the same key twice then the simulator makes two oracle calls, but obtains only one signed message; it would therefore not break blind-signature unforgeability. This is why we require the blind signature to be issuer-simulatable. We make this argument more formal. Let $\mathcal{A}$ be an adversary for the first traceability game; we construct an adversary $\mathcal{B}$ winning the unforgeability game of the blind signature as follows.

Receive a public key $\mathsf{pk}_{\mathsf{BS}}$ from the challenger and pass it to $\mathcal{A}$ as $\mathfrak{gmpk}$. Adversary $\mathcal{B}$ simulates multiple instances of $\mathsf{SimIssue}_{\mathsf{BS}}$, one for each new value f that $\mathcal{A}$ sends when it asks to join a user. If the value has not been sent by $\mathcal{A}$ before, $\mathcal{B}$ provides $\mathsf{SimIssue}_{\mathsf{BS}}$ with an issuing query by forwarding it to its own $\mathsf{Issue}_{\mathsf{BS}}$ oracle. Note that if $\mathcal{A}$ queries a value f again then $\mathsf{SimIssue}_{\mathsf{BS}}$ simulates $\mathsf{Issue}_{\mathsf{BS}}$ without making $\mathcal{B}$ query its oracle. By the last point of Definition 2, from the response of its $\mathsf{Issue}_{\mathsf{BS}}$ oracle, $\mathcal{B}$ can derive the blind signature σ'_i corresponding to each f_i.

Let $(\sigma, m, \mathsf{bsn}, \mathfrak{sk}'_1, \ldots, \mathfrak{sk}'_l)$ be the adversary's output and let $\sigma = (\tau, \sigma', \Sigma)$. Since σ is valid on m and bsn, by the soundness of Σ, we can extract $\mathfrak{sk}^*$ on which σ' is valid and for which τ is valid on bsn (if $\mathsf{bsn} \neq \perp$). We have thus $\mathsf{Identify}_{\mathsf{S}}(\sigma, m, \mathsf{bsn}, \mathfrak{sk}^*) = 1$. Since $\mathsf{Identify}_{\mathsf{S}}$ outputs 0 for all $\mathfrak{sk}'_i$, we have $\mathfrak{sk}^* \neq \mathfrak{sk}'_i$ for all $1 \leq i \leq l$. Since for every transcript $\mathcal{T}$, $\mathcal{A}$ has output an $\mathfrak{sk}'_i$ that satisfies $\mathsf{Identify}_{\mathsf{T}}(\mathcal{T}, \mathfrak{sk}'_i)$, we have that for every j there exists i such that $f_j = f(sk'_i)$. Adversary $\mathcal{B}$ can thus form pairs $(\mathfrak{sk}'_1, \sigma'_1), \ldots, (\mathfrak{sk}'_k, \sigma'_k)$ such that all $\mathfrak{sk}'_i$ are different and σ'_i is a valid blind signature on $\mathfrak{sk}'_i$, where k is the number of $\mathsf{Issue}_{\mathsf{BS}}$ queries $\mathcal{B}$ has made. Adversary $\mathcal{B}$ can thus output $((\mathfrak{sk}'_1, \sigma'_1), \ldots, (\mathfrak{sk}'_k, \sigma'_k), (\mathfrak{sk}^*, \sigma'))$, which breaks the blind-signature unforgeability property.

Traceability game 2. The adversary must output $(\sigma_0, m_0, \sigma_1, m_1, \mathsf{bsn}, \mathfrak{sk}')$ such that σ_0 is valid on m_0 and bsn, σ_1 is valid on m_1 and bsn, both signatures are identified with $\mathfrak{sk}'$, but they do not link. For $b = 0, 1$, let τ_b be the tag contained in σ_b. Since winning the game implies $\mathsf{bsn} \neq \perp$ and both signatures identify with $\mathfrak{sk}'$, we have $\mathsf{Tag}_{\mathsf{LIT}}(\mathsf{bsn}, \mathfrak{sk}') = \tau_0$ and $\mathsf{Tag}_{\mathsf{LIT}}(\mathsf{bsn}, \mathfrak{sk}') = \tau_1$ (by the definition of $\mathsf{Identify}_{\mathsf{S}}$). On the other hand, since σ_0 and σ_1 do not link, we have $\tau_0 \neq \tau_1$ (by the definition of Link). Together, this is a contradiction.

Non-frameability game 1. To win this game, the adversary must output a tuple $(\sigma, i, m, \mathsf{bsn})$ such that σ 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.

As in the anonymity proof, we define Game 1, which picks a random user i and aborts if the "framed" user is not this user i. In Game 2, we simulate the SPKs when answering the Sign queries; and in Game 3 we replace the tags in these queries by tags under randomly chosen keys (or reuse the tag if a signature for a basename is queried multiple times). By the same arguments as in the proof of anonymity we have that if the adversary has non-negligible advantage in the first non-frameability game then this still holds if we demand that the forgery be for a randomly fixed user i, if we simulate the SPKs, and if we replace every new LIT tag with a tag for a random key.

Game 3 can now be reduced to inversion of the one-way function f. Let $c = f(\mathfrak{sk})$ be given by a challenger who chose $\mathfrak{sk}$ uniformly at random; the challenge is to return $\mathfrak{sk}$. To simulate Game 3, we use c as the blinded secret key of user i in the Join protocol. This is possible, as we required our blind signature scheme be such that the user only needs to know f of the message. Note that $\mathfrak{sk}$ is not required anywhere in the simulation, as the tags are for random keys and the SPKs are simulated.

If the adversary is successful then it has never queried the signing oracle for user i on m and bsn. The simulator has thus never produced a SPK on $(\mathsf{bsn}\|m)$. By soundness of the SPK, the simulator can thus extract the witness $\mathfrak{sk}$ and solve the inversion challenge. If the adversary wins Game 3 then $\mathfrak{sk}$ is the key that user i used in the Join protocol, which is the preimage of c.

Non-frameability game 2. Let $\mathcal{A}$ be an adversary winning the second non-frameability game by outputting

$$(\sigma_0 = (\tau_0, \sigma'_0, \Sigma_0), m_0, \mathsf{bsn}_0, \sigma_1 = (\tau_1, \sigma'_1, \Sigma_1), m_1, \mathsf{bsn}_1, \mathfrak{sk}) \ .$$

We show that $\mathcal{A}$ can be used to break the (weak) linkability property of the LIT, i.e. to output either $(m_0, m_1, \mathfrak{sk})$ s.t. $m_0 \neq m_1$ and $\mathsf{Tag}_{\mathsf{LIT}}(m_0, \mathfrak{sk}) = \mathsf{Tag}_{\mathsf{LIT}}(m_1, \mathfrak{sk})$, or to output $(m, \mathfrak{sk}_0, \mathfrak{sk}_1)$ s.t. $\mathfrak{sk}_0 \neq \mathfrak{sk}_1$ and $\mathsf{Tag}_{\mathsf{LIT}}(m, \mathfrak{sk}_0) = \mathsf{Tag}_{\mathsf{LIT}}(m, \mathfrak{sk}_1)$.

Since $\mathcal{A}$ wins the game, both signatures have to be valid, and moreover they must link for bsn_b for some $b \in \{0,1\}$. We thus have the following (where the last 3 equations follow from the definition of Link):

$$\mathsf{GVf}(\mathfrak{gmpk}, \sigma_0, m_0, \mathsf{bsn}_0) = 1 \tag{1}$$
$$\mathsf{GVf}(\mathfrak{gmpk}, \sigma_1, m_1, \mathsf{bsn}_1) = 1 \tag{2}$$
$$\mathsf{GVf}(\mathfrak{gmpk}, \sigma_{b-1}, m_{b-1}, \mathsf{bsn}_b) = 1 \tag{3}$$
$$\mathsf{bsn}_b \neq \perp \tag{4}$$
$$\tau_0 = \tau_1 =: \tau \tag{5}$$

From the definition of GVf and soundness of the SPK, we can extract witnesses $\mathfrak{sk}'_0$ and $\mathfrak{sk}'_1$ from Σ_0 and Σ_1, respectively.

Let us assume that $\mathsf{bsn}_{1-b} = \perp$. By (1) and (2) we have $\mathsf{GVf}(\mathfrak{gmpk}, \sigma_{1-b}, m_{1-b}, \mathsf{bsn}_{1-b}) = 1$, and thus $\tau_{1-b} = \emptyset$ (by the definition of GVf). On the other hand, by (3) and (4) and soundness of Σ_{1-b} we have $\mathsf{Tag}_{\mathsf{LIT}}(\mathsf{bsn}_b, \mathfrak{sk}_{b-1}) = \tau_{b-1}$, which contradicts $\tau_{1-b} = \emptyset$. We have thus $\mathsf{bsn}_{1-b} \neq \perp$.

From (1), (2) and (5) and by soundness of the SPKs Σ_0 and Σ_1, the following holds for the extracted keys $\mathfrak{sk}'_0$ and $\mathfrak{sk}'_1$:

$$\mathsf{Verify}_{\mathsf{BS}}(\mathfrak{sk}'_0, \sigma'_0, \mathfrak{gmpk}) = 1 \qquad \mathsf{Tag}_{\mathsf{LIT}}(\mathsf{bsn}_0, \mathfrak{sk}'_0) = \tau \tag{6}$$
$$\mathsf{Verify}_{\mathsf{BS}}(\mathfrak{sk}'_1, \sigma'_1, \mathfrak{gmpk}) = 1 \qquad \mathsf{Tag}_{\mathsf{LIT}}(\mathsf{bsn}_1, \mathfrak{sk}'_1) = \tau \tag{7}$$

Similarly, by (3) we have $\mathsf{Tag}_{\mathsf{LIT}}(\mathsf{bsn}_b, \mathfrak{sk}'_{b-1}) = \tau$. This yields $\mathsf{bsn}_0 = \mathsf{bsn}_1$, as otherwise $(\mathsf{bsn}_0, \mathsf{bsn}_1, \mathfrak{sk}_{b-1})$ would break weak linkability. With overwhelming probability we thus have

$$\mathsf{bsn}_0 = \mathsf{bsn}_1 =: \mathsf{bsn} \quad \text{such that} \quad \mathsf{bsn} \neq \perp \ . \tag{8}$$

Given (8), the only remaining condition for the adversary to have won the game is by having

$$\mathsf{Identify}_{\mathsf{S}}(\sigma_0, m_0, \mathsf{bsn}, \mathfrak{sk}) = 1 \tag{9}$$
$$\mathsf{Identify}_{\mathsf{S}}(\sigma_1, m_1, \mathsf{bsn}, \mathfrak{sk}) = 0 \tag{10}$$

From (9) we have

$$\mathsf{Tag}_{\mathsf{LIT}}(\mathsf{bsn}, \mathfrak{sk}) = \tau \ , \tag{11}$$

so for (10) to hold, we must have $\mathsf{Verify}_{\mathsf{BS}}(\mathfrak{sk}, \sigma'_1, \mathfrak{gmpk}) = 0$. Together with (7), this implies $\mathfrak{sk} \neq \mathfrak{sk}'_1$ Thus $(\mathsf{bsn}, \mathfrak{sk}'_1, \mathfrak{sk})$ breaks weak linkability by (7) and (11).

8 Instantiating the Primitives

8.1 Mathematical preliminaries

Before instantiating our primitives we first need to introduce some necessary mathematics, and associated hard problems.

Bilinear groups. Bilinear groups are a set of three groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$, of prime order p, along with a bilinear map (a deterministic function) $\hat{t}$ which takes as input one element in $\mathbb{G}_1$ and one element in $\mathbb{G}_2$ and outputs an element in $\mathbb{G}_T$. We shall write $\mathbb{G}_1$ and $\mathbb{G}_2$ additively (with identity element 0), and $\mathbb{G}_T$ multiplicatively (with identity element 1), and write $\mathbb{G}_1 = \langle P_1 \rangle, \mathbb{G}_2 = \langle P_2 \rangle$, for two explicitly given generators P_1 and P_2. We define $\mathcal{P} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{t}, P_1, P_2)$ to be the set of pairing group parameters.

The function $\hat{t}$ must have the following three properties:

1. Bilinearity: $\forall Q_1 \in \mathbb{G}_1, \forall Q_2 \in \mathbb{G}_2, \forall x, y \in \mathbb{Z}$, we have

$$\hat{t}([x]Q_1, [y]Q_2) = \hat{t}(Q_1, Q_2)^{xy} \ .$$

2. Non-Degeneracy: The value $\hat{t}(P_1, P_2)$ generates $\mathbb{G}_T$.

3. The function $\hat{t}$ is efficiently computable.

In practice there are a number of different types of bilinear groups one can take, each giving rise to different algorithmic properties and different hard problems. Following [28] we categorise pairings into three distinct types (other types are possible, but the following three are the main ones utilised in practical protocols).

- **Type 1**: This is the symmetric pairing setting in which $\mathbb{G}_1 = \mathbb{G}_2$.
- **Type 2**: Here we have $\mathbb{G}_1 \neq \mathbb{G}_2$, but there is an efficiently computable isomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$ where $\psi(P_2) = P_1$.
- **Type 3**: Again $\mathbb{G}_1 \neq \mathbb{G}_2$, but now there is no known efficiently computable isomorphism.

In this paper we shall always consider Type-3 pairings. Such pairings can be efficiently realised; by taking $\mathbb{G}_1$ to be the set of points of order p of an elliptic curve over $\mathbb{F}_q$ with "small" embedding degree k; by taking $\mathbb{G}_2$ to be the set of points of order p on a twist of the same elliptic curve over $\mathbb{F}_{q^e}$, for some divisor e of k; and $\mathbb{G}_T$ to be the subgroup of order p in the finite field $\mathbb{F}_{q^k}$.

For a security parameter λ we let $\mathsf{Setup}_{\mathsf{Grp}}(1^\lambda)$ denote an algorithm which produces a pairing group instance $\mathcal{P}$ of Type-3. Note that for Type-3 pairings the DDH problem is believed to be hard in both $\mathbb{G}_1$ and $\mathbb{G}_2$.

Definition 3 (Decision Diffie-Hellman assumption in $\mathbb{G}_i$)**.** *The DDH assumption in $\mathbb{G}_i$ is said to hold if the following difference of probabilities is negligible in the security parameter λ, for all adversaries $\mathcal{A}$ and all parameter sets $\mathcal{P}$ output by* $\mathsf{Setup}_{\mathsf{Grp}}(1^\lambda)$*:*

$$\begin{aligned}\mathsf{Adv}_{\mathsf{DDH},\mathcal{A}}(\lambda) = \Pr\left[x, y, z \leftarrow \mathbb{F}_p, X = [x]P_1, Y = [y]P_1, Z = [z]P_1 : \mathcal{A}(X, Y, Z, \mathcal{P}) = 1\right] \\ - \Pr\left[x, y \leftarrow \mathbb{F}_p, X = [x]P_1, Y = [y]P_1, Z = [x \cdot y]P_1 : \mathcal{A}(X, Y, Z, \mathcal{P}) = 1\right] \quad .\end{aligned}$$

Definition 4 (Computational Diffie-Hellman assumption in $\mathbb{G}_i$)**.** *The CDH assumption holds in $\mathbb{G}_i$ if the following probability is negligible in the security parameter λ, for all adversaries $\mathcal{A}$ and all parameter sets $\mathcal{P}$ output by* $\mathsf{Setup}_{\mathsf{Grp}}(1^\lambda)$*:*

$$\mathsf{Adv}_{\mathsf{CDH},\mathcal{A}}(\lambda) = \Pr\left[x, y \leftarrow \mathbb{F}_p, X = [x]P_1, Y = [y]P_1 : \mathcal{A}(X, Y, \mathcal{P}) = [x \cdot y]P_1\right] \quad .$$

8.2 CL signatures and the LRSW assumptions

All of our basic constructions build upon the pairing-based Camenisch-Lysyanskaya signature scheme [18].

Definition 5 (Camenisch–Lysyanskaya signature scheme)**.** *The CL signature scheme is defined by the following triple of algorithms given an output $\mathcal{P}$ of* $\mathsf{Setup}_{\mathsf{Grp}}(1^\lambda)$.

- $\mathsf{KeyGen}(\mathcal{P})$*: Set $\mathsf{sk}_{CL} \leftarrow (x, y) \in \mathbb{F}_p^2$ and $\mathsf{pk}_{CL} \leftarrow (X, Y) = ([x]P_2, [y]P_2)$.*
- $\mathsf{Sign}(m, \mathsf{sk}_{CL})$*: Select $A \in \mathbb{G}_1 \setminus \{0\}$, and then set $B \leftarrow [y]A$, $C = [x + m \cdot x \cdot y]A$. Output (A, B, C).*
- $\mathsf{Verify}(m, (A, B, C), \mathsf{pk}_{CL})$*: This output 1 if and only if $\hat{t}(B, P_2) = \hat{t}(A, Y)$ and $\hat{t}(C, P_2) = \hat{t}(A, X) \cdot \hat{t}(B, X)^m$.*

The EF-CMA security of the CL signature scheme is seen to be equivalent to the hardness of the LRSW problem introduced in [32]; although the problems in [32] and [18] are presented slightly differently, see later for a discussion on this. The LRSW problem was originally given in the context of Type-1 pairings only, however the following generalisation to arbitrary pairing groups is immediate:

Definition 6 (LRSW assumption from [18])**.** *If $\mathcal{A}$ is an algorithm which is given access to an oracle $\mathcal{O}_{[x]P_2,[y]P_2}(\cdot)$ that on input of $f \in \mathbb{F}_p$ outputs $(A, B, C) = (A, [y]A, [x + f \cdot x \cdot y]A)$, for some random $A \in \mathbb{G}_1 \setminus \{0\}$, we let Q denote the set of queries made by $\mathcal{A}$ to $\mathcal{O}_{[x]P_2,[y]P_2}(\cdot)$.*

The LRSW assumption holds for the output of $\mathsf{Setup}_{\mathsf{Grp}}$ *if for all probabilistic polynomial-time adversaries $\mathcal{A}$, and all outputs of* $\mathsf{Setup}_{\mathsf{Grp}}$*, the following probability is negligible in the security parameter λ:*

$$\begin{aligned}\mathsf{Adv}_{\mathsf{LRSW},\mathcal{A}}(\lambda) = \Pr\big[\, & x \leftarrow \mathbb{F}_p,\ y \leftarrow \mathbb{F}_p,\ X \leftarrow [x]P_2,\ Y \leftarrow [y]P_2, \\ & (Q, m, A, B, C) \leftarrow \mathcal{A}^{\mathcal{O}_{X,Y}(\cdot)}(\mathcal{P}, X, Y)\ :\ m \notin Q\ \wedge\ m \in \mathbb{F}_p \setminus \{0\}\ \wedge \\ & A \in \mathbb{G}_1 \setminus \{0\}\ \wedge\ B = [y]A\ \wedge\ C = [x + m \cdot x \cdot y]A\,\big] \quad .\end{aligned}$$

In [32] it was shown that the LRSW assumption holds in the generic group model and is independent of the DDH assumption. Our protocols will require certain strengthenings of the LRSW assumption, in particular the so-called blind-LRSW (B-LRSW) assumption introduced in [22], and recently used in [30]. The B-LRSW assumption can also be shown to hold in the generic group model.

Definition 7 (B-LRSW assumption). *If $\mathcal{A}$ is an algorithm which is given access to an oracle $\mathcal{O}^B_{[x]P_2,[y]P_2}(\cdot)$ that on input of $M = [m]P_1 \in \mathbb{G}_1$ outputs $(A, B, C) = (A, [y]A, [x + m \cdot x \cdot y]A)$, for some random $A \in \mathbb{G}_1 \setminus \{0\}$, we let Q denote the set of queries made by $\mathcal{A}$ to $\mathcal{O}^B_{[x]P_2,[y]P_2}(\cdot)$.*

The B-LRSW assumption holds for the output of $\mathsf{Setup}_{\mathsf{Grp}}$ *if for all probabilistic polynomial-time adversaries $\mathcal{A}$, and all outputs of* $\mathsf{Setup}_{\mathsf{Grp}}$*, the following probability is negligible in the security parameter λ:*

$$\begin{aligned}\mathsf{Adv}_{\text{B-LRSW},\mathcal{A}}(\lambda) = \Pr[\, & x \leftarrow \mathbb{F}_p,\ y \leftarrow \mathbb{F}_p,\ X \leftarrow [x]P_2,\ Y \leftarrow [y]P_2,\\ & (Q, m, A, B, C) \leftarrow \mathcal{A}^{\mathcal{O}^B_{X,Y}(\cdot)}(\mathcal{P}, X, Y) : [m]P_1 \notin Q \ \wedge\ m \in \mathbb{F}_p \setminus \{0\} \ \wedge\\ & A \in \mathbb{G}_1 \setminus \{0\} \ \wedge\ B = [y]A \ \wedge\ C = [x + m \cdot x \cdot y]A \,] \ .\end{aligned}$$

Our second randomizable weakly blind signature scheme outputs CL-style signatures consisting of quadruples (A, B, C, D). To show security of this blind signature scheme requires us to state a new variant of the LRSW assumption. We call this the *blind 4-LRSW assumption*, and it is the natural extension of the B-LRSW assumption given earlier.

Definition 8 (B-4-LRSW assumption). *If $\mathcal{A}$ is an algorithm which is given access to an oracle $\mathcal{O}^{B\text{-}4}_{[x]P_2,[y]P_2}(\cdot)$ that on input of $M = [m]P_1 \in \mathbb{G}_1$ outputs $(A, B, C, D) = (A, [y]A, [x + m \cdot x \cdot y]A, [y \cdot m]A)$, for some random $A \in \mathbb{G}_1 \setminus \{0\}$, we let Q denote the set of queries made by $\mathcal{A}$ to $\mathcal{O}^{B\text{-}4}_{[x]P_2,[y]P_2}(\cdot)$.*

The B-4-LRSW assumption is said to hold for the output of $\mathsf{Setup}_{\mathsf{Grp}}$ *if for all probabilistic polynomial-time adversaries $\mathcal{A}$, and all outputs of* $\mathsf{Setup}_{\mathsf{Grp}}$*, the following probability is negligible in the security parameter λ:*

$$\begin{aligned}\mathsf{Adv}_{\text{B-4-LRSW},\mathcal{A}}(\lambda) = \Pr[\, & x \leftarrow \mathbb{F}_p,\ y \leftarrow \mathbb{F}_p,\ X \leftarrow [x]P_2,\ Y \leftarrow [y]P_2,\\ & (Q, m, A, B, C, D) \leftarrow \mathcal{A}^{\mathcal{O}^{B\text{-}4}_{X,Y}(\cdot)}(\mathcal{P}, X, Y) :\\ & [m]P_1 \notin Q \ \wedge\ m \in \mathbb{F}_p \setminus \{0\} \ \wedge\ A \in \mathbb{G}_1 \setminus \{0\} \ \wedge\ B = [y]A \ \wedge\\ & C = [x + m \cdot x \cdot y]A \ \wedge\ D = [y \cdot m]A \,]\end{aligned}$$

Note that this assumption is not completely new: The original LRSW assumption from [32] uses an oracle which outputs triples of the form (A, C, D), wheres the one from [18] given above outputs triples of the form (A, B, C); where $A \in \mathbb{G}_1$, $B = [y]A$, $C = [x + m \cdot x \cdot yA]$ and $D = [y \cdot m]A$ as above. The output of the LRSW adversary is similarly (A, C, D) or (A, B, C). These two such formulations are equivalent if the message m are known to the oracle. Since we require a blind oracle (i.e. only $M = [m]P$ is passed to the oracle and not m) the two formulations are distinct, and the B-4-LRSW assumption is the natural combination of the two standard ways of presenting the LRSW assumption.

In addition, in [1] and [3] Ateniese et al. defined a strong LRSW assumption (i.e. the adversary is not required to output m) using 5-tuples where the fourth element is the same as ours. The B-4-LRSW assumption is also similar to the q-Hidden LRSW assumption from [27], in which the adversary obtains q-tuples similar to our 4-tuples. The main difference is that in [27] the tuples are given as input to the adversary, whereas we allow the adversary to obtain tuples on M of his choosing.

8.3 Non-interactive zero-knowledge proofs

At various points in our schemes we will require Non-Interactive Zero-Knowledge (NIZK) proofs of statements relating various (unknown) discrete logarithms in finite abelian groups of prime order p. To express the creation of such a proof we will write, for example,

$$\mathsf{NIZK}(\{a, b\}, A = [a]P, B = [b]Q, C = [ab]R) \ ,$$

for public group elements A, B, C, P, Q and R, where $A, P \in \mathbb{G}$, $B, Q \in \mathbb{G}'$, $C, R \in \mathbb{G}''$, for possibly distinct groups $\mathbb{G}, \mathbb{G}', \mathbb{G}''$ of the same order, the elements a and b being the unknown discrete logarithms. If we write $\Sigma \leftarrow \mathsf{NIZK}(\{a, b\}, A = [a]P, B = [ab]Q)$ we mean: let Σ denote the corresponding NIZK proof. To verify this proof using the public data A, B, C, P, Q and R we will use the notation $\mathsf{Verify}(\Sigma, \{A, B, C, P, Q, R\})$.

In the Random Oracle Model (ROM), we can obtain such NIZK proofs by applying the Fiat–Shamir heuristic to the Sigma protocol which provides an interactive proof of the same statement. We note that by programming the underlying random oracle in a security proof, a simulator can always simulate such a proof, even if it does not know the relevant discrete logarithms. In addition, in the ROM such proofs are proofs of knowledge, since by rewinding, and applying the forking lemma, we can always extract the witness.

8.4 Two example Randomizable Weakly Blind Signature schemes

We give two examples, both of which are based on the signature scheme of Camenisch and Lysyanskaya [18]. Both make use of the same $\mathsf{Setup}_{\mathsf{BS}}$, $\mathsf{KeyGen}_{\mathsf{BS}}$ and $\mathsf{Request}^0_{\mathsf{BS}}$ algorithms given in Figure 9. From these basic operations we define two randomizable weakly blind signature schemes as follows.

Scheme 1. Our first scheme is given in Figure 10. The NIZK proof required in the scheme is assumed to be constructed via the use of the Fiat–Shamir heuristic applied to the underlying Sigma-protocol. At first sight the NIZK proof issued by the signer seems superfluous, as the user, who knows m, could simply verify the CL signature (A, B, C). However, in our security proof for weak blindness the simulator will need to determine whether an adversarial signer is acting correctly without access to the value of m. Thus by requiring the adversary to output a NIZK proof of correctness of the formation of the entry C we can bypass the need for the simulator to know m. Zero-knowledge of this proof will guarantee issuer simulatability, as given one signature we can simulate many issuing sessions by randomizing the signature and simulating the proof. We note that the request/issue protocol is a one-round protocol and hence (if the protocol is secure) it will be secure under parallel composition (unlike multi-round protocols).

Theorem 3. *If the DDH assumption holds in $\mathbb{G}_1$ and the NIZK proof used is sound then the above scheme satisfies the weak blindness property. More formally, in the random oracle model, for any adversary $\mathcal{A}$ against the weak blindness property of Scheme 1 there are adversaries $\mathcal{B}$ and $\mathcal{C}$ against the DDH problem in $\mathbb{G}_1$ and the soundness property of the NIZK respectively, such that*

$$\mathsf{Adv}^{\mathit{weak\text{-}blind}}_{\mathsf{BS},\mathcal{A}}(\lambda) = \mathsf{Adv}^{DDH}_{\mathcal{B}}(\lambda) + \mathsf{Adv}^{NIZK\text{-}soundness}_{\mathcal{C}}(\lambda) \ .$$

Proof. Let $(R = [\alpha]P_1, S = [\beta]P_1, T = [\gamma]P_1)$ be the input to adversary $\mathcal{B}$, where (α, β) are independent uniform random elements of $\mathbb{F}_p$, and either $\gamma = \alpha \cdot \beta$, or γ is also an independent uniform random element of $\mathbb{F}_p$. Algorithm $\mathcal{B}$'s goal is to determine which of the two possibilities has been given to it.

Algorithm $\mathcal{B}$ picks secret keys $x, y \leftarrow \mathbb{F}_p$ for the blind signature scheme and passes these to Algorithm $\mathcal{A}$. Then Algorithm $\mathcal{B}$ requests a blind signature on the unknown element α, by passing the element R to $\mathcal{A}$ as the message ρ_0 in the Request phase. Algorithm $\mathcal{A}$ will then respond with a tuple (A, B, C, Σ). By checking the proof Σ Algorithm $\mathcal{B}$ is guaranteed that the value (A, B, C) returned is a valid signature on the unknown element α. If it is not then we can construct an adversary $\mathcal{C}$ against the soundness of the NIZK proof. That the NIZK proofs are sound follows since they are constructed via the Fiat–Shamir heuristic from Sigma protocols. From now on we assume (A, B, C) has been computed correctly.

To produce the challenge for $\mathcal{A}$, Algorithm $\mathcal{B}$ now forms the challenge $(A^*, B^*, C^*) \leftarrow (S, [y]S, [x]S + [x \cdot y]T)$ and passes back to $\mathcal{A}$ the pair of tuples (A, B, C) and (A^*, B^*, C^*). Since (A, B, C) is a valid signature on α, we claim that if $T = [\alpha \cdot \beta]P_1$, 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 $\mathcal{B}$ will solve DDH with essentially the same advantage as that of $\mathcal{A}$ against the weak blindness game. We finally need to justify the claim:

$\mathsf{Setup}_{\mathsf{BS}}(1^\lambda)$:

- $\mathcal{P} \leftarrow \mathsf{Setup}_{\mathsf{Grp}}(1^\lambda)$.
- $\mathcal{M} := \mathbb{F}_p \setminus \{0\}$.
- $\mathsf{param} \leftarrow (\mathcal{P}, \mathcal{M})$.
- Output param.

$\mathsf{KeyGen}_{\mathsf{BS}}(1^\lambda)$:

- $x, y \leftarrow \mathbb{F}_p$.
- $X \leftarrow [x]P_2$.
- $Y \leftarrow [y]P_2$.
- $\mathsf{sk}_{\mathsf{BS}} \leftarrow (x, y)$, $\mathsf{pk}_{\mathsf{BS}} \leftarrow (X, Y)$.
- Output $(\mathsf{pk}_{\mathsf{BS}}, \mathsf{sk}_{\mathsf{BS}})$.

$\mathsf{Request}^0_{\mathsf{BS}}(m, \mathsf{pk}_{\mathsf{BS}})$:

- $Q_m \leftarrow [m]P_1$.
- $\rho_0 \leftarrow Q_m$, $\mathsf{St}^0_R \leftarrow (Q_m, m)$.
- Output $(\rho_0, \mathsf{St}^0_R)$.

Figure 9: Common $\mathsf{Setup}_{\mathsf{BS}}$, $\mathsf{KeyGen}_{\mathsf{BS}}$ and $\mathsf{Request}_{\mathsf{BS}}$ algorithms for our two blind signature schemes

```
$\mathsf{Issue}^1_{\mathsf{BS}}(\rho_0, \mathsf{sk}_{\mathsf{BS}})$:
  • Parse $\rho_0$ as $Q_m$.
  • $a \leftarrow \mathbb{F}_p \setminus \{0\}$.
  • $A \leftarrow [a]P_1$
  • $B \leftarrow [a \cdot y]P_1$
  • $C \leftarrow [a \cdot x]P_1 + [a \cdot x \cdot y]Q_m$
  • $\Sigma \leftarrow \mathsf{NIZK}(\{a, x, y\}$ :
        $A = [a]P_1, X = [x]P_2, Y = [y]P_2,$
        $C = [a \cdot x]P_1 + [a \cdot x \cdot y]Q_m)$.
  • $\beta_1 \leftarrow (A, B, C, \Sigma)$.
  • Output $\beta_1$.

$\mathsf{Verify}_{\mathsf{BS}}(m, \sigma, \mathsf{pk}_{\mathsf{BS}})$:
  • Parse $\sigma$ as $(A, B, C)$.
  • If $A = 0$ or $\hat{t}(A, Y) \neq \hat{t}(B, P_2)$
    or $\hat{t}(C, P_2) \neq \hat{t}(A, X) \cdot \hat{t}(B, X)^m$
      • Return 0.
  • Return 1.

$\mathsf{Request}^1_{\mathsf{BS}}(\beta_1, \mathsf{St}^0_R, \mathsf{pk}_{\mathsf{BS}})$:
  • Parse $\beta_1$ as $(A, B, C, \Sigma)$.
  • Parse $\mathsf{St}^0_R$ as $(Q_m, m)$.
  • If $A = 0$ or $\hat{t}(A, Y) \neq \hat{t}(B, P_2)$
    or $\mathsf{Verify}(\Sigma, \{A, C, X, Y,$
                    $P_1, P_2, Q_m\}) = 0$
      • Return $\perp$.
  • $\sigma \leftarrow (A, B, C)$.
  • Output $\sigma$.

$\mathsf{Randomize}_{\mathsf{BS}}(\sigma)$:
  • Parse $\sigma$ as $(A, B, C)$.
  • $r \leftarrow \mathbb{F}_p$.
  • $A' \leftarrow [r]A, B' \leftarrow [r]B, C' \leftarrow [r]C$.
  • Return $(A', B', C')$.
```

Figure 10: Scheme 1

- Case $\gamma = \alpha \cdot \beta$: 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 assumed assumed that (A, B, C) are computed correctly, there is some value $a \in \mathbb{F}_p$ such that $A = [a]P_1, B = [ya]P_1$ and $C = [a(x + xy\alpha)]P_1$.

  A rerandomization of (A, B, C) has the form $([r]A, [r]B, [r]C)$ for $r \in_R \mathbb{F}_p$, whereas the triple we sent was of the form $([\beta]P_1, [y \cdot \beta]P_1, [x \cdot \beta + x \cdot y \cdot \alpha \cdot \beta]P_1)$. We substitute $r = \beta/a$ in and note that because β is independent of everything else and uniformly random, r is again uniform in $\mathbb{F}_p$. Thus the challenge is also a correct CL signature on the message α This exactly corresponds to case $b = 0$ of the weak-blindness game.

- Case γ is random: Here we argue identically up to the point where we send our triple. This will be $([\beta]P_1, [y \cdot \beta]P_1, [x \cdot \beta + x \cdot y \cdot \gamma]P_1)$ and we substitute $\delta = \gamma/\beta$ getting $([\beta]P_1, [y \cdot \beta]P_1, [\beta \cdot (x + x \cdot y \cdot \delta)]P_1)$. Because γ is independent and uniformly distributed, δ is again uniformly distributed. Therefore we have a correct CL signature on a random message δ which exactly corresponds to case $b = 1$.

□

Theorem 4. *If the B-LRSW assumption holds then the above scheme is unforgeable. More formally, in the random oracle model, if $\mathcal{A}$ is an adversary against the forgeability game of Scheme 1, then there exists an adversary $\mathcal{B}$ against the B-LRSW assumption such that*

$$\mathsf{Adv}^{\textit{forge}}_{\mathsf{BS},\mathcal{A}}(\lambda) = \mathsf{Adv}^{\textit{B-LRSW}}_{\mathcal{B}}(\lambda) \ .$$

Proof. Let $\mathcal{B}$ have as input the public keys (X, Y) which it passes to $\mathcal{A}$. Algorithm $\mathcal{A}$ proceeds to make a series of oracle calls to the blind signature issuer. To obtain a valid (A, B, C) pair for a challenge Q_m, Algorithm $\mathcal{B}$ uses its blind LRSW oracle to obtain the tuple (A, B, C). Then, since $\mathcal{A}$ is operating in the random oracle model, it can produce a fake simulated NIZK proof Σ which $\mathcal{A}$ cannot distinguish from a genuine proof. The tuple (A, B, C, Σ) is passed back to Algorithm $\mathcal{A}$.

Eventually $\mathcal{A}$ will terminate with a tuple $((m_1, \sigma_1), \ldots, (m_{k+1}, \sigma_{k+1}))$ for the forgery game. If $\mathcal{A}$ is successful in winning its game then all entries verify, and so the σ_i are correct LRSW-tuples for the entries m_i. 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 CL signature that was never queried to the B-LRSW oracle and output it. Therefore $\mathcal{B}$ breaks the B-LRSW assumption with the same advantage as $\mathcal{A}$ has of creating a forgery. □

```
$\mathsf{Issue}^1_{\mathsf{BS}}(\rho_0, \mathsf{sk}_{\mathsf{BS}})$:
  • Parse $\rho_0$ as $Q_m$.
  • $a \leftarrow \mathbb{F}_p \setminus \{0\}$.
  • $A \leftarrow [a]P_1$
  • $B \leftarrow [a \cdot y]P_1$
  • $C \leftarrow [a \cdot x]P_1 + [a \cdot x \cdot y]Q_m$
  • $D \leftarrow [a \cdot y]Q_m$.
  • $\Sigma \leftarrow \mathsf{NIZK}(\{t\} : B = [t]P_1, D = [t]Q_m)$.
  • $\beta_1 \leftarrow (A, B, C, D, \Sigma)$.
  • Output $\beta_1$.

$\mathsf{Request}^1_{\mathsf{BS}}(\beta_1, \mathsf{St}^0_R, \mathsf{pk}_{\mathsf{BS}})$:
  • Parse $\beta_1$ as $(A, B, C, D, \Sigma)$.
  • Parse $\mathsf{St}^0_R$ as $(Q_m, m)$.
  • If $A = 0$ or $\hat{t}(A, Y) \neq \hat{t}(B, P_2)$
    or $\hat{t}(C, P_2) \neq \hat{t}(A + D, X)$
    or $\mathsf{Verify}(\Sigma, \{B, P_1, D, Q_m\}) = 0$
      • Return $\perp$.
  • $\sigma \leftarrow (A, B, C, D)$.
  • Output $\sigma$.

$\mathsf{Verify}_{\mathsf{BS}}(m, \sigma, \mathsf{pk}_{\mathsf{BS}})$:
  • Parse $\sigma$ as $(A, B, C, D)$.
  • If $A = 0$ or $\hat{t}(A, Y) \neq \hat{t}(B, P_2)$
    or $\hat{t}(C, P_2) \neq \hat{t}(A + D, X)$ or $D \neq [m]B$
      • Return 0.
  • Return 1.

$\mathsf{Randomize}_{\mathsf{BS}}(\sigma)$:
  • Parse $\sigma$ as $(A, B, C, D)$.
  • $r \leftarrow \mathbb{F}_p$.
  • $A' \leftarrow [r]A$, $B' \leftarrow [r]B$.
  • $C' \leftarrow [r]C$, $D' \leftarrow [r]D$.
  • Return $(A', B', C', D')$.
```

Figure 11: Scheme 2

Theorem 5. *If the proof* NIZK *used in* $\mathsf{Issue}^1_{\mathsf{BS}}$ *has the zero-knowledge property then Scheme 1 is issuer simulatable.*

Proof. Let $\mathcal{A}$ be an adversary that interacts an arbitrary number of times with $\mathsf{Issue}_{\mathsf{BS}}$, each time for the same message m. From $\mathsf{Request}^0_{\mathsf{BS}}$ we have that each time the message sent to the issuer is $[m]P_1$. We construct a simulator $\mathsf{SimIssue}_{\mathsf{BS}}$ that on input $[m]P_1$ forwards this to its (one-time) $\mathsf{Issue}_{\mathsf{BS}}$ oracle to obtain (A, B, C, Σ), where Σ is a proof of knowledge of the signing key and the randomness a of the signature. The simulator forwards this to $\mathcal{A}$ when called for the first time and for all succeeding calls does the following. It produces a randomization $(A', B', C') \leftarrow \mathsf{Randomize}_{\mathsf{BS}}(A, B, C)$ of the original signature and makes a simulated proof Σ' of knowledge for (A', B', C'). Since randomized signatures are distributed like freshly computed signatures and since NIZK is zero-knowledge, $\mathcal{A}$ cannot detect the difference to interacting with a genuine $\mathsf{Issue}_{\mathsf{BS}}$ oracle. □

Finally, it is easily seen that Scheme 1 also satisfies the last point of Definition 2: it is a one-round scheme, and the first message is $f(m) = Q_m = [m]P_1$; the output of $\mathsf{Issue}^1_{\mathsf{BS}}$ contains a ready signature (A, B, C), and the proof Σ, whose verification only requires $f(m)$, guarantees validity of the signature.

Scheme 2. In our application of these randomizable weakly blind signature schemes we do not use the verification algorithm directly, but will prove correctness by providing a NIZK proofs of knowledge of the value m which verifies the equation. Thus a verification equation which applies m to elements of G_T is going to be more computationally expensive to run. This motivates our second scheme in Figure 11. The element D in the verification equation allows us to generate a simpler NIZK proof of knowledge of the value m on which the signature is valid. Note that the user could generate D from the (A, B, C) tuple by computing $D \leftarrow [m]B$ rather than having D come from the signer. However, if we do this then the protocol is not simulatable. Thus, whilst the scheme might look strange at first sight, when it is applied to our group-signature-like construction it results in a more efficient scheme. This however comes at the cost of forgeability security being based on an even less standard underlying hard problem.
Using variants of the proofs given for Scheme 1 we can show the following theorems. Analogously, it follows that Scheme 2 satisfies the compatibility requirements of Definition 2.

Theorem 6. *If the DDH assumption is hard in $\mathbb{G}_1$, and the NIZK proof used is sound, then Scheme 2 satisfies the weak blindness property. More formally, in the random oracle model, for any adversary $\mathcal{A}$ against the weak blindness property of Scheme 2 there are adversaries $\mathcal{B}$ and $\mathcal{C}$ against the DDH problem in $\mathbb{G}_1$ and the soundness property of the NIZK respectively, such that*

$$\mathsf{Adv}^{weak\text{-}blind}_{\mathsf{BS},\mathcal{A}}(\lambda) = \mathsf{Adv}^{DDH}_{\mathcal{B}}(\lambda) + \mathsf{Adv}^{NIZK\text{-}soundness}_{\mathcal{C}}(\lambda) \ .$$

$\mathsf{KeyGen}_{\mathsf{LIT}}(1^\lambda)$:

- $\mathcal{M}_{\mathsf{LIT}} = \{0,1\}^*$.
- $\mathfrak{sk} \leftarrow \mathbb{F}_p$.

$\mathsf{Tag}_{\mathsf{LIT}}(m, \mathfrak{sk})$:

- $\tau \leftarrow [\mathfrak{sk}]H_1(m)$.

Figure 12: The BLS based Linkable Indistinguishable Tag

Theorem 7. *If the B-4-LRSW assumption holds then Scheme 2 is unforgeable. More formally, in the random oracle model, if $\mathcal{A}$ is an adversary against the forgeability game of Scheme 2, then there exists an adversary $\mathcal{B}$ against the B-4-LRSW assumption such that*

$$\mathsf{Adv}^{forge}_{\mathsf{BS},\mathcal{A}}(\lambda) = \mathsf{Adv}^{B\text{-}4\text{-}LRSW}_{\mathcal{B}}(\lambda) \ .$$

Theorem 8. *If the proof* NIZK *used in* $\mathsf{Issue}^1_{\mathsf{BS}}$ *has the zero-knowledge property, then Scheme 2 is issuer simulatable.*

8.5 An example of Linkable Indistinguishable Tags

We can always consider a deterministic digital signature scheme as a symmetric keyed MAC function, by ignoring the public key. In our constructions of group signature-like schemes we will require Linkable Indistinguishable Tags which allow efficient zero-knowledge proofs of knowledge of the underlying key, given a message/tag pair. These will be easier to construct from digital signature schemes, when considered in a similar way as symmetric key functions.

Our construction of a linkable indistinguishable tag is in the ROM and is based on the BLS [9] signature scheme, although our instantiation is for any (additive) finite abelian group $\mathbb{G}$ of prime order p. The construction is given in Figure 12, and it makes use of a hash function $H_1 : \{0,1\}^* \longrightarrow \mathbb{G}$. We call this construction the BLS-LIT.

Theorem 9. *In the ROM for all adversaries $\mathcal{A}$ against the indistinguishability property of the BLS-LIT in an arbitrary finite abelian group $\mathbb{G}$, there is an adversary $\mathcal{B}$ against DDH in $\mathbb{G}$ such that*

$$\mathsf{Adv}^{f\text{-}\mathsf{IND}}_{\mathsf{LIT},\mathcal{A}}(\lambda) \leq q_H \cdot \mathsf{Adv}^{\mathsf{DDH}}_{\mathcal{P},\mathcal{B}}(\lambda) \ ,$$

where q_H denotes an upper bound on the number of hash function, tag and verify queries which $\mathcal{A}$ makes in total, and f is the function $x \mapsto [x]P_1$.

Proof. Let $(P, [x]P, [y]P, [z]P)$ denote the input to the adversary $\mathcal{B}$, for unknown values x, y, z. The aim of $\mathcal{B}$ is to determine whether $z = x \cdot y$ or not. Algorithm $\mathcal{B}$ first calls the adversary $\mathcal{A}$ with input $c = f(x) = [x]P$. We can assume that $\mathcal{A}$ calls H_1 on the message m^* before the first stage of $\mathcal{A}$ terminates, and we can assume that H_1 is called on a message before every adversarial call to the tag or verify oracles. We select $i^* \in \{1, \ldots, q_H\}$ to be the "critical query" to the hash function. Algorithm $\mathcal{B}$ then responds to the various queries of $\mathcal{A}$ as follows:

HASH QUERIES. Algorithm $\mathcal{B}$ maintains a list H_1-List consisting of triples (m, h, r). If H_1 is called for a value m for which there is already an entry $(m, h, *) \in H_1$-List, then $\mathcal{B}$ responds with the value h. If the H_1-List has $i^* - 1$ entries in it already then the entry $(m, [y]P, \perp)$ is added to H_1-List and $[y]P$ is returned to $\mathcal{A}$. Otherwise $\mathcal{B}$ generates a new random value $r \in \mathbb{F}_p$ and defines $h \leftarrow [r]P$, adds (m, h, r) to the H_1-List, and returns h to $\mathcal{A}$.

TAG QUERIES. When $\mathcal{A}$ queries a tag on a message m, we can assume that there exists either $(m, h, r) \in H_1$-List or $(m, h, \perp) \in H_1$-List. In the latter case the algorithm $\mathcal{B}$ aborts. In the former case algorithm $\mathcal{B}$ returns the tag $[r]([x]P)$ to $\mathcal{A}$.

At the end of Stage 1 of the adversary, algorithm $\mathcal{B}$ aborts if the i^*th call to the Hash oracle was not equal to the value m^* returned by $\mathcal{A}_1$. The value $\tau^* = [z]P$ is returned by $\mathcal{B}$ to the second stage of $\mathcal{A}$ as the supposed tag on m^*. Upon completion of the second stage of algorithm $\mathcal{A}$ it will respond with its guess as to whether the tag τ^* is a valid tag on the message m^* with respect to the hidden key x. Since $H_1(m^*) = [y]P$ we have that this will be the correct tag if and only if the input tuple is a valid DDH tuple. Thus $\mathcal{B}$ answers its challenger with the output of $\mathcal{A}$ and the result follows. □

Theorem 10. *The BLS-LIT is linkable in the ROM, i.e. for all adversaries $\mathcal{A}$ there is a negligible function ν such that*

$$\mathsf{Adv}^{\mathsf{LINK}}_{\mathsf{LIT},\mathcal{A}}(\lambda) \leq \nu(\lambda) \ .$$

Join		Iss
$\mathfrak{sk}$		$\mathfrak{gmsk}$
$Q_{\mathfrak{sk}} \leftarrow [\mathfrak{sk}]P_1$	$\xrightarrow{Q_{sk}}$	
		$a \leftarrow \mathbb{F}_p \setminus \{0\}$ $A \leftarrow [a]P_1$ $B \leftarrow [a \cdot y]P_1$ $C \leftarrow [a \cdot x]P_1 + [a \cdot x \cdot y]Q_{\mathfrak{sk}}$ $D \leftarrow [a \cdot y]Q_{\mathfrak{sk}}$ $t \leftarrow a \cdot y$ $r \leftarrow \mathbb{F}_p$ $R_1 \leftarrow [r]P_1, R_2 \leftarrow [r]Q_{\mathfrak{sk}}$ $c \leftarrow H_1(B \| D \| P_1 \| Q_{\mathfrak{sk}} \| R_1 \| R_2)$ $s \leftarrow r + c \cdot t \pmod p$
	$\xleftarrow{(A, B, C, D, c, s)}$	
$R_1' \leftarrow [s]P_1 - [c]B$ $R_2' \leftarrow [s]Q_{\mathfrak{sk}} - [c]D$ $c' \leftarrow H_6(B \| D \| P_1 \| Q_{\mathfrak{sk}} \| R_1' \| R_2')$ If $A = 0$ or $\hat{t}(A, Y) \neq \hat{t}(B, P_2)$ or $\hat{t}(C, P_2) \neq \hat{t}(A + D, X)$ or $c \neq c'$ Return $\perp$. $\mathfrak{gsk} \leftarrow (A, B, C, D)$		

Figure 13: The Join and Iss protocol for our specific instance of a pre-DAA scheme in the ROM

Proof. In the BLS-LIT, a tag is created as $\tau = [\mathfrak{sk}]H_1(m)$ and verifies if and only if this equation holds. No adversary can link two tags for the same message but different keys as $[\mathfrak{sk}]H_1(m) = [\mathfrak{sk}']H_1(m)$ at once implies $\mathfrak{sk} = \mathfrak{sk}'$.

If the adversary provides two messages such that $m \neq m'$ but $[\mathfrak{sk}]H_1(m) = [\mathfrak{sk}]H_1(m')$ then we conclude $H_1(m) = H_1(m')$ and have found a collision in the random oracle H_1. The probability of this happening is negligible.

We show that an adversary able to output a valid tuple $(m, \tau, \mathfrak{sk}, m', \tau', \mathfrak{sk}')$ such that $(m, \mathfrak{sk}) \neq (m', \mathfrak{sk}')$ can be used to compute discrete logarithms (DL) in $\mathbb{G}$. Let $\mathcal{A}$ be such an adversary, and let $\mathcal{B}$ be given a DL instance $(P, Q = [\alpha]P)$. $\mathcal{B}$ controls the random oracle and its goal is to output α.

We can assume that, before returning a successful tuple $(m, \tau, \mathfrak{sk}, m', \tau', \mathfrak{sk}')$, $\mathcal{A}$ has called its hash oracle on m and m'. Let q_H be an upper bound on $\mathcal{A}$'s oracle calls. $\mathcal{B}$ chooses $i^* \in \{1, \ldots, q_H\}$ uniformly at random and maintains a list H_1-List of tuples of the form (m, R, r). An oracle query for a message m is answered as follows: if H_1-List already contains an item (m, R, r) for some R, r then $\mathcal{B}$ returns R; else if H_1-List contains $i^* - 1$ items then $\mathcal{B}$ returns Q from its instance and adds $(m, Q, \perp)$ to H_1-List; otherwise $\mathcal{B}$ chooses $r \in \{1, \ldots, |\mathbb{G}|\}$, returns $R = [r]P$ and adds (m, R, r) to the list.

Let $(m, \tau, \mathfrak{sk}, m', \tau', \mathfrak{sk}')$ be $\mathcal{A}$'s output which satisfies $\tau = \tau'$, $\mathsf{Tag}_{\mathsf{LIT}}(m, \mathfrak{sk}) = \tau$ and $\mathsf{Tag}_{\mathsf{LIT}}(m', \mathfrak{sk}') = \tau$. If neither m nor m' were queried in the i^*-th oracle query then $\mathcal{B}$ aborts. Let w.l.o.g. m be the i^*-th query, and let R', r' be such that there is a tuple of the form (m', R', r') in H_1-List. (Such a tuple exists, since we assumed $\mathcal{A}$ has queried m' to its oracle and $m \neq m'$.) Since both tags verify and link, we have $[\mathfrak{sk}]H_1(m) = [\mathfrak{sk}']H_1(m')$ and thus $[\mathfrak{sk}]Q = [\mathfrak{sk}']([r']P)$. $\mathcal{B}$ can thus compute the discrete logarithm of Q in basis P as $\mathfrak{sk}' \cdot r' \cdot \mathfrak{sk}^{-1}$. $\square$

9 An example DAA and pre-DAA scheme

We instantiate our pre-DAA scheme construction with our second blind signature scheme from earlier and the BLS-Tag. We thus obtain a protocol very similar to the DAA protocols of [6, 5, 16, 21, 20, 22], whilst obtaining our strong security guarantees provided by our new model. The major differences between our pre-DAA scheme and prior DAA schemes using pairings being; Firstly, that the issuer provides a proof of knowledge rather than the user in the Join protocol; Secondly, the pre-DAA scheme merges the TPM and Host into one entity (the user);

```
$\mathsf{GSig}(\mathfrak{gsk}_i, \mathfrak{sk}_i, m, \mathsf{bsn})$:
  • Parse $\mathfrak{gsk}_i$ as $(A, B, C, D)$.
  • $l \leftarrow \mathbb{F}_p \setminus \{0\}$.
  • $(R, S, T, W) \leftarrow ([l]A, [l]B, [l]C, [l]D)$.
  • If $\mathsf{bsn} \neq \perp$
      • $J \leftarrow H_2(\mathsf{bsn})$.
      • $K \leftarrow [\mathfrak{sk}_i]J$.
  • Else
      • $J, K \leftarrow \mathcal{O}$.
  • $r \leftarrow \mathbb{F}_p$.
  • $R_1 \leftarrow [r]J$, $R_2 \leftarrow [r]S$.
  • $c \leftarrow H_3(J\|K\|S\|W\|R_1\|R_2\|\mathsf{bsn}\|m)$.
  • $s \leftarrow r + c \cdot \mathfrak{sk}_i \pmod p$.
  • $\sigma \leftarrow (K, R, S, T, W, c, s)$.

$\mathsf{GVf}(\mathfrak{gmpk}, \sigma, m, \mathsf{bsn})$:
  • Parse $\sigma$ as $(K, R, S, T, W, c, s)$.
  • $J \leftarrow \mathcal{O}$.
  • If $\mathsf{bsn} \neq \perp$
      • $J \leftarrow H_2(\mathsf{bsn})$.
  • $R_1' \leftarrow [s]J - [c]K$.
  • $R_2' \leftarrow [s]S - [c]W$.
  • $c' \leftarrow H_3(J\|K\|S\|W\|R_1'\|R_2'\|\mathsf{bsn}\|m)$.
  • If $R = 0$ or $\hat{t}(R, Y) \neq \hat{t}(S, P_2)$
    or $\hat{t}(T, P_2) \neq \hat{t}(R + W, X)$ or $c \neq c'$
      • Return 0.
  • Return 1.
```

Figure 14: The GSig and GVf algorithms for our specific instance of an pre-DAA scheme in the ROM

although we remove this restriction at the end of this section by presenting a full DAA scheme. And finally, the case of $\mathsf{bsn} = \perp$ within a signature is dealt with differently from the case of $\mathsf{bsn} \neq \perp$.

The Setup procedure picks a pairing parameter set $\mathcal{P} \leftarrow \mathsf{Setup}_{\mathsf{Grp}}(1^\lambda)$, and defines a set of hash functions, all of which will be modelled as random oracles,

$$H_1 : \{0,1\}^* \longrightarrow \mathbb{F}_p, \quad H_2 : \{0,1\}^* \longrightarrow \mathbb{G}_1, \quad H_3 : \{0,1\}^* \longrightarrow \mathbb{F}_p.$$

For the algorithm GKg, the issuer picks two elements $x, y \leftarrow \mathbb{F}_p$, forming $\mathfrak{gmsk}$. The public key $\mathfrak{gmpk}$ is $(X, Y) \leftarrow ([x]P_2, [y]P_2)$. Whilst using the algorithm UKg the user picks his secret key $\mathfrak{sk} \leftarrow \mathbb{F}_p$. Using the second blind signature scheme above, and instantiating the required NIZK using the Fiat–Shamir heuristic and the hash function H_1, we obtain the (Join, Iss) protocol of Figure 13, with the GSig and GVf algorithms being given in Figure 14.

The signature proof of knowledge in the GSig algorithm is obtained by combining the verification algorithm for the blind signature with a proof of knowledge of the actual message being signed. Notice, that the proof of knowledge is executed within the group $\mathbb{G}_1$, whereas if we used the first of our blind signature methods we would need to execute a proof of knowledge in $\mathbb{G}_T$. When, in a moment, we split this signing protocol between a resource constrained TPM and a Host, this will provide a significant advantage of using this method. Albeit this comes at the cost of a non-standard assumption of the B-4-LRSW assumption. Also note that we have, when $\mathsf{bsn} = \perp$, included a dummy component into the signature proof of knowledge so as to make the same proof be output as in the case of $\mathsf{bsn} \neq \perp$.

Finally, we need to define the algorithms $\mathsf{Identify}_\mathsf{T}(\mathcal{T}, \mathfrak{sk})$, $\mathsf{Identify}_\mathsf{S}(\sigma, m, bsn, \mathfrak{sk})$, and $\mathsf{Link}(\mathfrak{gmpk}, \sigma_0, m_0, \sigma_1, m_1, \mathsf{bsn})$. For the algorithm $\mathsf{Identify}_\mathsf{T}(\mathcal{T}, \mathfrak{sk})$, we assume the transcript $\mathcal{T}$ parses as $(Q_{\mathfrak{sk}}, A, B, C, D, c, s)$. We first check that $Q_{\mathfrak{sk}} = [\mathfrak{sk}]P_1$ and then we perform the checks on A, B, C, D, c and s performed by the user in the Join protocol in Figure 13. To execute the $\mathsf{Identify}_\mathsf{S}(\sigma, m, bsn, \mathfrak{sk})$ algorithm in order to verify whether a signature could have been produced with $\mathfrak{sk}$, we first verify it normally, then we check that $W = [\mathfrak{sk}]S$ and $K = [\mathfrak{sk}]J$. The algorithm returns 1 if and only if these checks pass. Finally, for the $\mathsf{Link}(\mathfrak{gmpk}, \sigma_0, m_0, \sigma_1, m_1, \mathsf{bsn})$ algorithm: We first check whether the two signature verify correctly, then we check that $\mathsf{bsn} \neq \perp$. If any of these checks fail then we return 0. Otherwise we take the two input signatures, $\sigma_0 = (K_0, R_0, S_0, T_0, W_0, c_0, s_0)$ and $\sigma_1 = (K_1, R_1, S_1, T_1, W_1, c_1, s_1)$, and return one if and only if $K_0 = K_1$.

Taking this pre-DAA scheme we then transform it to a fully fledged DAA scheme using the method of Section 4. We then obtain the Join/Iss and Signature protocols in Figures 15 and 16. Compared to previous schemes, in which the TPM needed to generate a proof of knowledge in the Join protocol, our scheme may be preferable computationally. In addition, due to our model our scheme has much clearer functional properties and security guarantees.

Acknowledgements

This work has been supported in part by the European Commission through the ICT Programme under Contract ICT-2007-216676 ECRYPT II, by an European Research Council Advanced Grant ERC-2010-AdG-267188-CRIPTO and by the Engineering and Physcial Sciences Research Council via grant EP/H043454/1. The fourth author has also been supported in part by a Royal Society Wolfson Merit Award.

References

[1] G. Ateniese, J. Camenisch, S. Hohenberger and B. de Medeiros. Practical group signatures without random oracles. *Cryptology ePrint Archive*. Report 2005/385, available at `http://eprint.iacr.org/2005/385`.

[2] M. Abe, S.S.M. Chow, K. Haralambiev and M. Ohkubo. Double-Trapdoor Anonymous Tags for Traceable Signatures. *Applied Cryptography and Network Security – ACNS 2011*, Springer LNCS 6715, 183–200, 2011.

[3] G. Ateniese, J. Camenisch and B. de Medeiros. Untraceable RFID tags via insubvertible encryption. *Proceedings of ACM CCS 2005*, 92–101, 2005.

[4] E. Brickell, J. Camenisch and L. Chen. Direct anonymous attestation. *Computer and Communications Security – CCS 2004*, ACM Press, 132–145, 2004.

[5] 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.

[6] 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.

[7] 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.

[8] 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`.

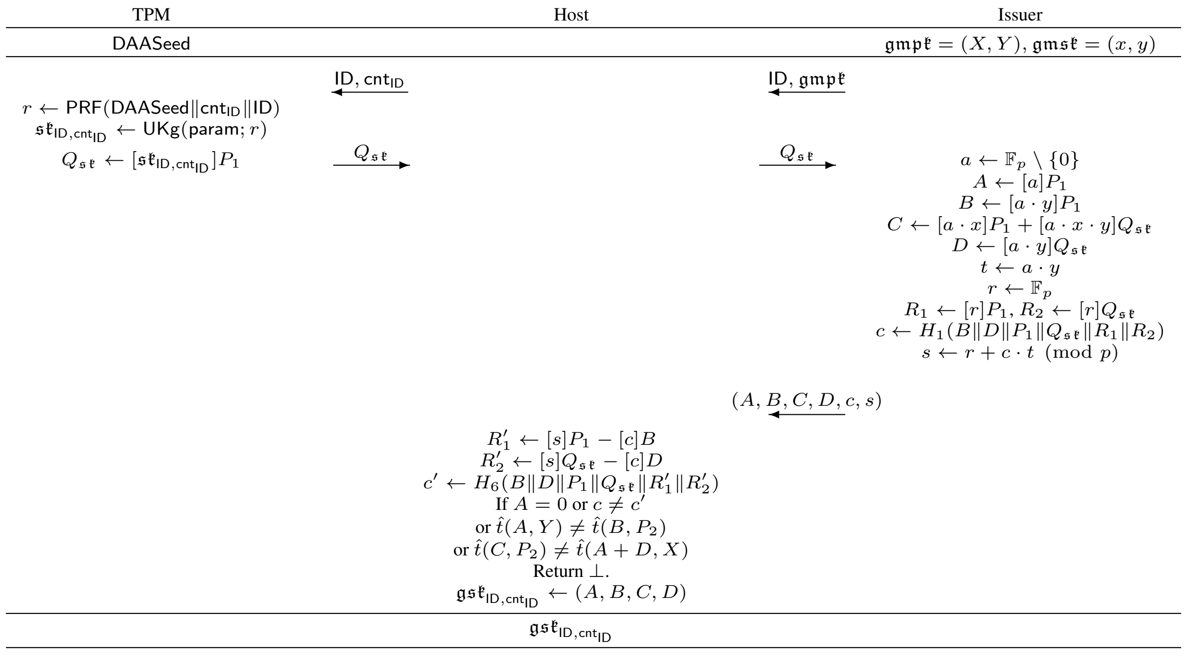
TPM
Host
Issuer
DAASeed
$\mathfrak{gmpk} = (X, Y), \mathfrak{gmsk} = (x, y)$
$\mathsf{ID}, \mathsf{cnt}_{\mathsf{ID}}$
$\mathsf{ID}, \mathfrak{gmpk}$
$r \leftarrow \mathsf{PRF}(\mathsf{DAASeed} \| \mathsf{cnt}_{\mathsf{ID}} \| \mathsf{ID})$
$\mathfrak{sk}_{\mathsf{ID},\mathsf{cnt}_{\mathsf{ID}}} \leftarrow \mathsf{UKg}(\mathsf{param}; r)$
$Q_{\mathfrak{sk}} \leftarrow [\mathfrak{sk}_{\mathsf{ID},\mathsf{cnt}_{\mathsf{ID}}}]P_1$
$Q_{\mathfrak{sk}}$
$Q_{\mathfrak{sk}}$
$a \leftarrow \mathbb{F}_p \setminus \{0\}$
$A \leftarrow [a]P_1$
$B \leftarrow [a \cdot y]P_1$
$C \leftarrow [a \cdot x]P_1 + [a \cdot x \cdot y]Q_{\mathfrak{sk}}$
$D \leftarrow [a \cdot y]Q_{\mathfrak{sk}}$
$t \leftarrow a \cdot y$
$r \leftarrow \mathbb{F}_p$
$R_1 \leftarrow [r]P_1, R_2 \leftarrow [r]Q_{\mathfrak{sk}}$
$c \leftarrow H_1(B \| D \| P_1 \| Q_{\mathfrak{sk}} \| R_1 \| R_2)$
$s \leftarrow r + c \cdot t \pmod p$
(A, B, C, D, c, s)
$R_1' \leftarrow [s]P_1 - [c]B$
$R_2' \leftarrow [s]Q_{\mathfrak{sk}} - [c]D$
$c' \leftarrow H_6(B \| D \| P_1 \| Q_{\mathfrak{sk}} \| R_1' \| R_2')$
If $A = 0$ or $c \neq c'$
or $\hat{t}(A, Y) \neq \hat{t}(B, P_2)$
or $\hat{t}(C, P_2) \neq \hat{t}(A + D, X)$
Return $\perp$.
$\mathfrak{gsk}_{\mathsf{ID},\mathsf{cnt}_{\mathsf{ID}}} \leftarrow (A, B, C, D)$
$\mathfrak{gsk}_{\mathsf{ID},\mathsf{cnt}_{\mathsf{ID}}}$

Figure 15: Our DAA Join Protocol in the ROM

TPM		Host
$\mathsf{DAASeed}$		$(\mathsf{ID}, \mathsf{cnt}_{\mathsf{ID}}, \mathfrak{gsk}_{\mathsf{ID},\mathsf{cnt}_{\mathsf{ID}}})$
		$(A, B, C, D) \leftarrow \mathfrak{gsk}_{\mathsf{ID},\mathsf{cnt}_{\mathsf{ID}}}$
		$l \leftarrow \mathbb{F}_p \setminus \{0\}$
		$(R, S, T, W) \leftarrow ([l]A, [l]B, [l]C, [l]D)$
	$\xleftarrow{\mathsf{ID}, \mathsf{cnt}_{\mathsf{ID}}, S, W, \mathsf{bsn}, m}$	
If $\mathsf{bsn} \neq \perp$ then $J \leftarrow H_2(\mathsf{bsn})$		
Else $J \leftarrow \mathcal{O}$		
$r \leftarrow \mathsf{PRF}(\mathsf{DAASeed} \| \mathsf{cnt}_{\mathsf{ID}} \| \mathsf{ID})$		
$\mathfrak{sk}_{\mathsf{ID},\mathsf{cnt}_{\mathsf{ID}}} \leftarrow \mathsf{UKg}(\mathsf{param}; r)$		
$K \leftarrow [\mathfrak{sk}_{\mathsf{ID},\mathsf{cnt}_{\mathsf{ID}}}]J$		
$k \leftarrow \mathbb{F}_p$		
$R_1 \leftarrow [k]J, R_2 \leftarrow [k]S$		
$c \leftarrow H_3(J\|K\|S\|W\|R_1\|R_2\|\mathsf{bsn}\|m)$		
$s \leftarrow k + c \cdot \mathfrak{sk}_i \pmod p$	$\xrightarrow{c, s, K}$	
		$\sigma \leftarrow (K, R, S, T, W, c, s)$

Figure 16: Our DAA Sign Protocol in the ROM

[9] D. Boneh, B. Lynn and H. Shacham. Short signatures from the Weil pairing. *Journal of Cryptology*, **17(4)**, 297–319, 2004.

[10] M. Bellare, D. Micciancio and B. Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. *Advances in Cryptology – Eurocrypt 2003*, Springer LNCS 2656, 614–629, 2003.

[11] D. Boneh and H. Shacham. Group signatures with verifier-local revocation. *Proceedings of ACM CCS 2004*, 168–177, 2004.

[12] 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.

[13] R. Canetti. Universally Composable Signatures, Certification and Authentication. *Cryptology ePrint Archive*. Report 2003/239, available at `http://eprint.iacr.org/2003/239`.

[14] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. (revised version of December 2005). *Cryptology ePrint Archive*. Report 2000/067, available at `http://eprint.iacr.org/2000/067`.

[15] X. Chen and D. Feng. Direct anonymous attestation for next generation TPM. *Journal of Computers*, **3**, 43–50. 2008.

[16] L. Chen. A DAA scheme requiring less TPM resources. *Int. Conference on Information Security and Cryptology - Inscrypt 2009.*

[17] S.S.M. Chow. Real Traceable Signatures. *Selected Areas in Cryptography – SAC 2009*, Springer LNCS 5867, 92–107, 2009.

[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] M. Chase and A. Lysyanskaya. On Signatures of Knowledge. In *Advances in Cryptology – CRYPTO 2006*, Springer LNCS 4117, 78–96, 2006.

[20] 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.

[21] L. Chen, P. Morrissey and N. P. Smart. Pairings in trusted computing. *Pairings in Cryptography – Pairing 2008*, Springer LNCS 5209, 1–17, 2008.

[22] 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`.

[23] 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.

[24] L. Chen and B. Warinschi. Security of the TCG Privacy-CA solution. *Proceedings of the 6th IEEE/IFIP International Symposium on Trusted Computing and Cmomunications – TrustCom 2010*

[25] A. Datta, A. Derek, J.C. Mitchell, A. Ramanathan and A. Scedrov. Games and the Impossibility of Realizable Ideal Functionality. In Theory of Cryptography Conference – TCC 2006, Springer LNCS 3876, 360–379, 2006.

[26] 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.

[27] M. Green and S. Hohenberger. Universally composable adaptive oblivious transfer. In *Advances in Cryptology - ASIACRYPT 2008*, Springer LNCS 5350, 179–197, 2008.

[28] S. Galbraith, K. Paterson and N.P. Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, **156**, 3113–3121, 2008

[29] J. Groth. Fully anonymous group signatures without random oracles. In *Advances in Cryptology - ASIACRYPT 2007*, Springer LNCS 4833, 164–180, 2007.

[30] E. Ghadafi and N.P. Smart. Efficient two-move blind signatures in the common reference string model. *Cryptology ePrint Archive*. Report 2010/568, available at `http://eprint.iacr.org/2010/568`.

[31] A. Juels, M. Luby and R. Ostrovsky. Security of blind digital signatures. In *Advances in Cryptology – CRYPTO '97*, Springer LNCS 1294, 150–164, 1997.

[32] A. Lysyanskaya, R. Rivest, A. Sahai and S. Wolf. Pseudonym systems. In *Selected Areas in Cryptography – SAC 99*, Springer LNCS 1758, 184–199, 1999.

[33] J.K. Liu, V.K. Wei and D.S. Wong. Linkable spontaneous anonymous group signature for ad hoc groups. *Information Security and Privacy – ACISP 2004*, Springer LNCS 3108, 325–335, 2004.

[34] D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, **13(3)**, 361–396, 2000.

[35] P.P. Tsang, M.H. Au, A. Kapadia and S.W. Smith. Blacklistable anonymous credentials: blocking misbehaving users without ttps. *Proceedings of ACM CCS 2007*, 72–81, 2007.

[36] Trusted Computing Group. TCG TPM specification 1.2. Available at `http://www.trustedcomputinggroup.org`, 2003.