# Linkable Democratic Group Signatures[⋆]

Mark Manulis[⋆⋆], Ahmad-Reza Sadeghi, and Jörg Schwenk

Horst-Görtz Institute
Ruhr-University of Bochum
D-44801, Germany
{mark.manulis, joerg.schwenk}@nds.rub.de, sadeghi@crypto.rub.de

**Abstract.** In a variety of group-oriented applications cryptographic primitives like group signatures or ring signatures are valuable methods to achieve anonymity of group members. However, in their classical form, these schemes cannot be deployed for applications that simultaneously require (i) to avoid centralized management authority like group manager and (ii) the signer to be anonymous only against non-members while group members have rights to trace and identify the signer.
The idea of recently introduced *democratic group signatures* is to provide these properties. Based on this idea we introduce a group-oriented signature scheme that allows the group members to trace the identity of any other group member who issued a signature while non-members are only able to link the signatures issued by the same signer without tracing. For this purpose the signature scheme assigns to every group member a unique pseudonym that can be used by any non-member verifier to communicate with the anonymous signer from the group. We present several group-oriented application scenarios where this kind of linkability is essential.
We propose a concrete linkable democratic group signature scheme for two-parties, prove its security in the random oracle model, and describe how to modularly extend it to the multi-party case.

**Key words:** democratic group signatures, anonymity, pseudonymity, linkability, group communication

## 1 Introduction

Designing anonymity-preserving cryptographic schemes for group-oriented applications, such as group signatures and ring signatures (c.f. Section 2), has been an appealing subject of the research. In this paper we consider group communication scenarios between members of a group and non-members where the anonymity of group members should be preserved only against non-members, but is not desired within the group. In other words, signed messages on behalf of the group must be verifiable by group members and non-members, but the signer's identity should be revealed only by other group members. Since a non-member may receive signed messages from different group members, it is essential for non-members to distinguish between signatures produced by the same signer (linkability of signatures) in order to respond accordingly. In this context it is likely that the unique pseudonyms of group members may be used to provide this kind of linkability for non-members. Further, we consider communication scenarios where no central management authorities needed who controls the initialisation of the group and the communication process.
In their classical form group and ring signatures cannot be deployed for those applications that require the mentioned properties. Recently the idea of a *democratic group signature* was introduced in [20] that allows group members to initialise the group and maintain it over dynamic group changes in a collective manner without relying on any centralized

[⋆] This is the full version of the paper which appears at ISPEC 2006, Lecture Notes of Computer Science, Volume 3903. ©Springer-Verlag, 2006.
[⋆⋆] The author was sponsored by the European Commission through IST-2002-507932 ECRYPT.

authority like group manager. However, the proposed model is too strong for the communication scenarios considered in this paper. This is because the model in [20] requires unlinkability of issued group signatures. Clearly, if signatures are unlinkable then non-members are not able to distinguish between messages signed by the anonymous communication partner from the group, and are, therefore, not able to respond to the signer. Therefore, in this paper we propose a relaxed model for *linkable democratic group signatures*. As a consequence more efficient schemes can be designed. In the following we describe some communication scenarios for linkable democratic group signatures.

*Consultative Group Decision.* Consider a group of people $\mathcal{G}$ who has to make their common decision based on anonymous consultations with third parties in $\mathcal{T}$. Every member of $\mathcal{G}$ is allowed to initiate an anonymous consultation with any party $t_i \in \mathcal{T}$ such that only other group members can identify it and follow the consultation process. Any party $t_i$ may be queried from different members at the same time and must, therefore, be able to determine the sender to respond accordingly. Concrete applications are: anonymous review and discussion of submitted papers by the program committee, selection of participants for a project tender based on their proposals. Group members may use linkable democratic group signatures to sign their queries to $t_i$. If referees want to discuss several topics of the submitted paper with its authors without revealing their identities, then they can use the scheme to sign their messages. The authors are able to distinguish between queries of different referees using pseudonyms and respond accordingly.

*Distributed Group Control.* Consider a scenario of the distributed control, e.g., in a battlefield. Given set $\mathcal{F}$ of field devices and set $\mathcal{C}$ of control devices, such that any device in $\mathcal{C}$ can send its orders to any device in $\mathcal{F}$ with respect to some control policy $P$. Any field device that receives an order must be able to respond to the control device that has issued the order. For strategic reasons field devices should not know, which control device has issued the order, but still must be able to verify that the order comes from an authentic control device in $\mathcal{C}$. Any other device in $\mathcal{C}$ must know which orders have been sent and also which control device is the issuer of a certain order. The latter allows the control devices to exclude a malicious control device whose orders are not conform with $P$. The solution based on linkable democratic group signatures is that all control devices from $\mathcal{C}$ form a group, whereas field devices are considered as non-members. Any control device $c_i$ broadcasts its signed order so that upon its reception any party in $\mathcal{C}$ and in $\mathcal{F}$ can verify whether the order is from a $c_i \in \mathcal{C}$ but only members of $\mathcal{C}$ can determine the identity of $c_i$ while members of $\mathcal{F}$ can communicate with $c_i$ over its pseudonym.

*Anonymous Intergroup Communication.* Linkable democratic group signatures allow authentic anonymous communication between members of multiple groups. Members of different groups can exchange signed messages without revealing own identities such that only members of their groups are able to open their signatures. For example, two teams want to carry out a fair two-player game competition (e.g., chess). Assume that players of one team know strengths and weaknesses of players of another team. Thus, if a player knows its opponent then it might benefit from this knowledge during the game. Therefore, to achieve fairness it is preferable that players do not know their opponents. However, once the game is started, both players have to be able to verify that all game moves come from the same opponent. After competition is finished, both teams should be able to figure out the authentic result, and reveal the identities of the opponent players for each pair.

It is thinkable to realize linkable democratic group signatures using certified pseudonyms [19] issued by a certification authority (CA) to every group member. However, this trivial construction requires an active involvement of the CA during the initialisation and is, therefore, opposed to the assumed trust model. The solution with certified pseudonyms is also impracticable if the group has to be formed spontaneously, e.g., teams in the example of anonymous intergroup communication may be built ad-hoc for each game from a pool of players. In this case a solution without active involvement of the CA is preferable.

**Organization** In Section 2 we discuss the main differences between linkable democratic group signatures and other known group-oriented signatures. The formal model and security requirements of linkable democratic group signatures are described in Section 3. We propose a concrete scheme for two parties and prove its security in the random oracle model in Section 4. In Section 5 we describe a straightforward extension for the multi-party case.

## 2 Related Work

Although there exist various group-oriented digital signatures, linkable democratic group signatures differ from them in trust relationship and signer's anonymity as discussed in the following.

*Group signatures* introduced by Chaum *et. al.* [11] and later studied and improved in [10, 9, 2, 7, 4, 8, 16] allow members of a group to sign messages on behalf of the group so that it is not possible to distinguish who the actual signer is. Nevertheless, there exists a designated *group manager* that can open the signature, i.e., reveal the identity of its signer. We are interested in two security requirements of group signatures, namely the *unlinkability* and the *anonymity*. The first one means that no party except for the group manager should be able to relate two or more signatures as being produced by the same signer. The second one means that no party except for the group manager should be able to reveal the signer's identity [2]. There are several differences between group signatures and linkable democratic group signatures: Firstly, in group signatures all group members trust the group manager that computes all group secrets. This is obviously is not the case in the trust model of linkable democratic group signatures. Secondly, group signatures are unlinkable. Thus, group members and non-members that receive a signed message are not able to relate it to any previously received signed message. The only party that is able to do this is the group manager. Obviously, this is an obstacle for the communication in scenarios described above. Thirdly, the anonymity requirement of group signatures allows only the group manager to reveal the identity of the signer from the signature. In linkable democratic group signatures, however, all group members are allowed to do this.

*Traceable signatures* introduced by Kiayias *et. al.* [15] extend group signatures as follows. There exist several members, called *tracers*, that receive a *tracing trapdoor* from the group manager and are able to link all group signatures that correspond to that tracing trapdoor. The tracing trapdoor reveals no information about the identity of the signer. Thus, tracers are able to collect all signatures of a certain group member without being able to identify him, while the group signatures produced by other group members remain unlinkable, unless the group manager reveals their tracing trapdoors too. Like in classical group signatures only group manager is able to identify the signer. Another property of traceable signatures is that group members are able to produce claiming proofs for their group signatures, i.e. a group member can convince any verifier that he is the author of a certain signed message. In the context of linkable democratic group signatures tracing trapdoors can be seen as pseudonyms that allow tracers to follow the communication by linking signed messages of the same authors. Since, by definition, tracing trapdoors are only known to the group manager, other group members and also non-members are not able to distinguish signatures generated by the same member. The property of traceable signatures that allows members to claim their signatures can be utilized to overcome this obstacle, i.e., after the signer has generated a signature and a claiming proof he sends the signature to all communication participants (group members and non-members), but sends the claiming proof only to group members that are then able to identify the signer during the verification of the proof. This can be achieved by encryption of the proof with the secret group key. Obviously, the drawback of this realisation is the increased communication and computation due to the additional generation and distribution of claiming proofs and their encryption, besides that the group manager is still the only authority that can identify the signer.

*Ring signatures* introduced by Rivest *et. al.* [21] and further studied in [18] and [5] do not require any group manager to form a group. For the signature generation every user builds a set of public keys that includes his public key and the public keys of other users. A generated signature does not reveal the public key of the signer, but a set of public keys of all possible signers. Therefore, ring signatures cannot be used for a direct communication between a verifier and a signer. Additionally, ring signatures provide unconditional anonymity, i.e., no party can reveal the signer's identity. Linkable democratic group signatures are similar to ring signatures in the sense that no group manager is required to initialise a group, and that a verifier knows the set of possible signers without being able to identify the signer. However, the differences are that in linkable democratic group signatures: (i) every member actively participates in the initialisation process, (ii) a non-member verifier is able to respond to the signer using the signer's pseudonym, and (iii) group members are able to identify the signer.

*Democratic group signatures* introduced recently in [20] result from changes to the standard model of group signatures caused by elimination of the group manager's role and distribution of the tracing rights to individuals. Group members initialise the group and control it over dynamic changes in a collective manner. Every group member has individual right to trace issued signatures to their signers. To provide individual tracing rights group members agree on a tracing trapdoor. The signer's anonymity is provided against non-members. However, the model in [20] requires unlinkability of signatures. This property is an obstacle for the direct communication between a non-member and an anonymous group member, because the non-member would no more be able to distinguish the messages of the same signer. Therefore, we relax the model of democratic group signatures to provide the linkability of generated signatures. Additionally, this results in a more efficient construction, i.e., the signatures in our scheme have the length of $O(1)$ whereas the signatures in [20] have the length of $O(n)$.

## 3 Model

### 3.1 Notations and Terminology

For a string $x$, $|x|$ denotes its length. For a finite set or tuple $X$, $|X|$ denotes its size. We use $\{,\}$ to specify a set, and $(,)$ to specify a tuple. $\epsilon$ denotes an empty string, tuple or set according to the context. We use $x := y$ to define value of $x$ to be $y$ where $y$ can be a value, a tuple or a set. If $y$ is a tuple then $x \overset{\$}{:=} y$ specifies a random permutation of values in $y$.
If $T$ is a randomized algorithm or protocol then $[T(X)]$ denotes the set of random variables having positive probability of being output by $T$ on input $X$, and $Y \overset{\$}{\leftarrow} T(X)$ denotes a concrete output of $T$ on the same input with freshly generated coins. For a finite set $X$, $x \overset{\$}{\leftarrow} X$ denotes the algorithm that samples an element uniformly at random from $X$. If $T$ is deterministic then $Y \leftarrow T(X)$ denotes the output of $T$ on input $X$.
We say one-argument function $v : \mathbb{N} \rightarrow \mathbb{N}$ is negligible if for any positive polynomial *poly* there exists $l' \in \mathbb{N}$, such that for all $l \in \mathbb{N}$, $l > l'$: $v(l) < 1/poly(l)$. We say two-argument function $w : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ is negligible if for any polynomial time function $f : \mathbb{N} \rightarrow \mathbb{N}$ for that exists a positive polynomial *poly*, such that $f(k) \leq poly(k), \forall k \in \mathbb{N}$, the one-argument function $w_f(l) = w(l, f(l))$ is negligible for all $l \in \mathbb{N}$. This definition of two-argument negligibility introduced in [2] is used in our scheme to show the negligibility of functions which are parameterized with a security parameter and a number of group members.
In the context of the security requirement $<\mathrm{req}>$ we use $\mathbf{Exp}_A^{<\mathrm{req}>}$ to denote an experiment with an algorithm $A$ aiming to break this requirement. By $\mathbf{Adv}_A^{<\mathrm{req}>}$ we denote the advantage function of $A$.

### 3.2 Linkable Democratic Group Signatures: Algorithms and Protocols

**Definition 1.** A *linkable democratic group signature scheme* $\mathcal{LDGS} = \{$*Setup*, *Sign*, *Verify*, *Trace*$\}$ is a digital signature scheme that consists of:

– A randomized protocol *Setup* between all members that takes as input a security parameter $l \in \mathbb{N}$ and a number of members $n \in \mathbb{N}$. The public output is a set of *identities ID* and a set of *pseudonyms PS* of group members. The identity and the pseudonym of a member $i$ are denoted $id_i$ and $ps_i$, respectively. The private output for each member $i$ is the individual secret signing key $sk_i$ from the set *SK*.
– A randomized algorithm *Sign* that on input a secret signing key $sk \in SK$ and a message $m$, outputs a signature $\sigma$ on $m$.
– A deterministic algorithm *Verify* that on input a candidate signature $\sigma$, a message $m$, and the set of pseudonyms *PS* outputs either a pseudonym *ps* if it accepts the signature or $\perp$ if it rejects it.
– A deterministic algorithm *Trace* that on input a candidate signature $\sigma$, a message $m$, any secret signing key $sk \in SK$, the set of pseudonyms *PS*, and the set of identities *ID* outputs either an identity *id* or the symbol $\perp$ if a failure occurs. Since every $sk_i \in SK$ can be used as input to the signing and tracing algorithms there should exist a *tracing trapdoor* as part of every $sk_i$.

*Remark 1.* In linkable democratic group signatures the scheme is initialised in a collective manner by all group members. Note that members may be in possession of certified public keys for the authentication of their messages during the *Setup* protocol, but we stress that no third trusted party like a certification authority (CA) is actively involved in the protocol.

Note that the stronger model of democratic group signatures in [20] that requires unlinkability of signatures considers additional protocols for dynamic groups, i.e., protocols that handle joins and exclusions of group members. However, in dynamic groups the property of linkability would allow non-members to reveal identities of group members that have joined to the group or have been excluded from it. In case of join a non-member may record the identity of the joining member. Then it can simply figure out which pseudonyms in the changed group formation are linkable to pseudonyms of the previous group formation, and so find out the pseudonym which cannot be linked, i.e. the pseudonym of the joined member. Similar, in case of the exclusion a non-member can record the identity of the excluded group member, then relate the pseudonyms from the previous group formation to the pseudonyms of the changed group formation, and figure out the pseudonym of the excluded member. Therefore, we consider linkable democratic group signatures only for static groups.

## 3.3 Trust Model and Assumptions

After the scheme is initialised every group member must be trusted not to reveal its secret signing key or any information that can be used to link any $id_i$ and $ps_i$ for the same $i$ to any other party. The setup protocol must also be resistant against attacks that aim to reveal such links. We require also the existence of at least one honest group member who acts according to the setup protocol and does not take part in any collusion of group members trying to cheat against non-members.

## 3.4 Security Requirements

**Definition 2 (Correctness).** *A linkable democratic group signature scheme $\mathcal{LDGS}$ from Definition 1 is correct if for all $l, n \in \mathbb{N}$, $(ID, PS, SK) \in [Setup(l,n)]$, $sk_i, sk_j \in SK$, $id_i \in ID$, $ps_i \in PS$, $m \in \{0,1\}^*$, and $\sigma \xleftarrow{\$} Sign(sk_i, m)$ :*

$$Verify(\sigma, m, PS) = ps_i \wedge Trace(\sigma, m, sk_j, PS, ID) = id_i$$

The following security requirements are derived from the *full-anonymity* and *full-traceability* requirements proposed by Bellare in [2] for group signature schemes and have been modified with respect to the model of linkable democratic group signatures. The modified requirements subsume the standard requirements on anonymity, collusion-resistance, framing and unforgeability requirements. We require from a linkable democratic group signature scheme, once initialised, to be resistant against the following attacks.

**Anonymity** Informally, in an anonymity attack against a linkable democratic group signature scheme the adversary tries to figure out the identity $id_i$ of the signer of a signature. The formal definition is given by experiment $\mathbf{Exp}_A^{\mathrm{anon-b}}(l,n)$ in Figure 1. Adversary *A* operates in two stages: *choose* and *guess*. In the *choose* stage it is given the set of identities *ID*, the set of pseudonyms *PS* produced by the protocol *Setup*$(l,n)$, and outputs two identities $id_0, id_1$, a message $m \in \{0,1\}^*$, and some state information St to be used in the second stage of the attack. In the *guess* stage *A* is given St, and a signature $\sigma_b$ of the member with $id_b$ on $m$ where $b$ is a random bit. In order to cover chosen message attacks *A* is also allowed to query oracle *Sign*$^*$ on any message $m^*$ of its choice. The oracle answers with the signature $\sigma_{b^*}$ on $m^*$. At the end of the stage *A* outputs bit $d$ as a guess for $b$.

**Definition 3.** *The anonymity advantage of A is given by the function*

$$\mathbf{Adv}_A^{\mathrm{anon}}(l,n) = \Pr[\mathbf{Exp}_A^{\mathrm{anon-1}}(l,n)=1] - \Pr[\mathbf{Exp}_A^{\mathrm{anon-0}}(l,n)=1].$$

*A linkable democratic group signature scheme $\mathcal{LDGS}$ from Definition 1 is anonymous if for any polynomial time adversary A the anonymity advantage $\mathbf{Adv}_A^{\mathrm{anon}}(l,n)$ is negligible in terms of negligibility of two-argument functions.*

**Traceability** Informally, in a traceability attack against a linkable democratic group signature scheme the adversary *A* is allowed to corrupt members (obtain their secret signing keys), and tries to generate a forged signature that either cannot be traced to any member or can be traced to an uncorrupted member. The formal definition is given by experiment $\mathbf{Exp}_A^{\mathrm{trace}}(l,n)$ in Figure 1. Adversary *A* operates in two stages: *corrupt* and *forge*. *A* starts its attack by adaptively corrupting a set $ID_c$ of members. The adversary has free choice of the identities and the number of corrupted members. At the end of the *corrupt* stage the set $ID_c$ contains identities of corrupted members. In the stage *forge* adversary knowing the set $SK_c$ of secret signing keys of corrupted members outputs a forgery $(\sigma, m)$. The experiment returns 1 if $\sigma$ fulfils verification requirements, but the tracing algorithm outputs either $\perp$ or an identity $id_i$ of an uncorrupted member. In both stages adversary is given access to the signing oracle *Sign* that outputs a signature of the member with $id_i$ on a query $(id_i, m')$.

**Definition 4.** *The traceability advantage of A is given by the function*

$$\mathbf{Adv}_A^{\mathrm{trace}}(l,n) = \Pr[\mathbf{Exp}_A^{\mathrm{trace}}(l,n)=1].$$

*A linkable democratic group signature scheme $\mathcal{LDGS}$ from Definition 1 is traceable if for any polynomial time adversary A the traceability advantage $\mathbf{Adv}_A^{\mathrm{trace}}(l,n)$ is negligible in terms of negligibility of two-argument functions.*

### 3.5 Discussion on Sizes

Since every member has its unique identity, pseudonym and secret signing key we follow that the lower size bounds of *ID*, *PS* and *SK* are equal to $n$. To the contrary, it is preferable that the size of a generated signature $\sigma$ remains constant, i.e., does not grow with $n$.

## 4 A Two-Party Linkable Democratic Group Signature Scheme

In this section we present a linkable democratic group signature scheme for two parties. A straightforward extension for a multi-party case is given later in Section 5. After description of required cryptographic primitives we specify the protocols and algorithms of the scheme and prove its security in the random oracle model.

```
$\mathbf{Exp}_A^{\mathrm{anon-b}}(l,n)$:
    $(ID, PS, SK) \xleftarrow{\$} Setup(l,n)$;
    $(\mathrm{St}, id_0, id_1, m) \xleftarrow{\$} A(choose, ID, PS)$;
    $b \xleftarrow{\$} \{0,1\}$; $\sigma \xleftarrow{\$} Sign(sk_b, m)$;
    $d \xleftarrow{\$} A^{Sign^*(\cdot)}(guess, \mathrm{St}, \sigma)$;
    return $d$;
```

```
$\mathbf{Exp}_A^{\mathrm{trace}}(l,n)$:
    $(ID, PS, SK) \xleftarrow{\$} Setup(l,n)$;
    $\mathrm{St} := (ID, PS)$; $ID_c := \epsilon$; $SK_c := \epsilon$; $next := 1$;
    Until $(next = 0)$ do
        $(next, \mathrm{St}, id_i) \xleftarrow{\$} A^{Sign(\cdot,\cdot)}(corrupt, \mathrm{St}, SK_c)$;
        If $next = 1$ then $ID_c \leftarrow ID_c \cup \{id_i\}$; $SK_c \leftarrow SK_c \cup \{sk_i\}$ EndIf
    EndUntil;
    $(\sigma, m) \xleftarrow{\$} A^{Sign(\cdot,\cdot)}(forge, \mathrm{St})$;
    If $Verify(\sigma, m, PS) = \bot$ then return 0 EndIf;
    If $Trace(\sigma, m, PS, sk_j) = \bot$ for any $0 \le j < n$ then return 1 EndIf;
    $id_i \leftarrow Trace(\sigma, m, PS, sk_j)$;
    If $id_i \notin ID_c$ and $Sign(\cdot,\cdot)$ was not queried on $(id_i, m)$ then return 1
        else return 0 EndIf;
```

**Fig. 1.** Experiments used to define anonymity and traceablity of $\mathcal{LDGS}$

## 4.1 Cryptographic Primitives

**Proof of the Equality of Two Discrete Logarithms** Let $\mathbb{G} :=< g, p, q >$ be a cyclic subgroup of the multiplicative group $\mathbb{Z}_p^*$ with generator $g$ and order $q$, where $q$ and $p$ are large prime numbers with $q|(p-1)$. In our scheme we use a non-interactive zero-knowledge (NIZK) proof system $(P, V)$ for the equality of two discrete logarithms. A probabilistic proving algorithm $P$ on input $x = (g_0, g_1, y_0, y_1) \in \mathbb{G}^4$ and $w \in \mathbb{Z}_q$ outputs a non-interactive proof $\pi$ for the equality $w = \log_{g_0}(y_0) = \log_{g_1}(y_1)$. A deterministic verifying algorithm $V$ on input $x$, and a proof $\pi$ checks whether $\pi$ is a correct proof and outputs either 1 or 0 (1 if $V$ accepts the proof). Both algorithms have access to a random oracle $R$ from the family of random oracles (c.f. [3]). The proof system $(P, V)$ must satisfy the usual requirements of completeness, soundness and zero-knowledge.
Note that the equality of two discrete logarithms in the context of digital signature schemes has already been used by Goh and Jarecki in [13]. Their scheme utilizes a NIZK proof system for the signature generation and is proven to be secure in the random oracle model. Our scheme is designed to work with any NIZK proof system for the equality of two discrete logarithms in the random oracle model, e.g., the efficient scheme used in [13] can be applied. In this case the random oracle $R$ in the description of our algorithms can be replaced by a cryptographic secure hash function.
Due to some technical reasons concerning the zero-knowledge simulation we have to use *adaptive* NIZK proof systems in our security proof. In Appendix A we provide a more detailed description and the definition of adaptive NIZK proof systems in the random oracle model based on the description of these systems in the common reference string model along the lines in [12] and [22].

**Simultaneous Decisional Diffie-Hellman Assumption** In the following we introduce a cryptographic assumption which we call a *simultaneous decisional Diffie-Hellman (SDDH)* assumption. This assumption is an extension of a well-known Decisional Diffie-Hellman (DDH) assumption [6] and is essential for the anonymity property of our scheme as shown below.

**Definition 5 (SDDH Assumption).** *Given a cyclic subgroup $\mathbb{G} :=< g, p, q >$ and security parameter $l$ as above. Let $A_s$ be a polynomial time distinguisher that tries to distinguish between two distributions $D'_0$ and $D'_1$ according to the following experiment:*

```
$\mathbf{Exp}_{A_s}^{\mathrm{sddh-b}}(l)$:
    $n \in \mathbb{N}; y \xleftarrow{\$} \mathbb{Z}_q$;
    For $i = 0$ to $n-1$ do $x_i, r_i \xleftarrow{\$} \mathbb{Z}_q$ EndFor;
    $D'_0 := (g, g^{x_0}, \ldots, g^{x_{n-1}}, g^y, g^{r_0}, \ldots, g^{r_{n-1}})$;
    $D'_1 := (g, g^{x_0}, \ldots, g^{x_{n-1}}, g^y, g^{x_0 y}, \ldots, g^{x_{n-1} y})$;
    $b \xleftarrow{\$} \{0, 1\}$;
    $d \xleftarrow{\$} A_s(D'_b)$;
    return $d$;
```

*The advantage function of $A_s$ is defined as*

$$\mathbf{Adv}_{A_s}^{\mathrm{sddh}}(l) = \Pr[\mathbf{Exp}_{A_s}^{\mathrm{sddh-1}}(l) = 1] - \Pr[\mathbf{Exp}_{A_s}^{\mathrm{sddh-0}}(l) = 1] \tag{1}$$

*The SDDH assumption is that $\mathbf{Adv}_{A_s}^{\mathrm{sddh}}(l)$ is negligible.*

**Theorem 1 (SDDH $\Leftrightarrow$ DDH).** *The SDDH assumption is equivalent to the Decisional Diffie-Hellman (DDH) assumption. Formally, for all distinguishers A, $A_s$ and for all $l \in \mathbb{N}$ : $\mathbf{Adv}_{A_s}^{\mathrm{sddh}}(l) = \mathbf{Adv}_{A}^{\mathrm{ddh}}(l)$.*

*Proof.* The proof is given in Appendix B.1.

The idea of SDDH assumption in the context of our scheme is that every group member $i$ has a temporary private key $x_i \in \mathbb{Z}_q$ and the corresponding public key $g^{x_i}$ which is used as identity $id_i$. In our scheme both group members compute the same tracing trapdoor $k \in \mathbb{Z}_q$ as part of their secret signing keys and the blinded version $g^k$ is public. The pseudonym $ps_i$ of a group member $i$ contains a value $g^{x_i k}$. According to Definition 1 a verifier obtains the signer's pseudonym. In order to provide anonymity the verifier must not be able to distinguish which temporary public key $g^{x_i}$ corresponds to the obtained signer's pseudonym. Considered that $y$ is the tracing trapdoor $k$, the distribution $D'_1$ from the experiment $\mathbf{Exp}_{A_s}^{\mathrm{sddh-b}}(l)$ consists of public keys and pseudonyms of all group members. Intuitively, if an adversary is able to distinguish between $D'_1$ and $D'_0$ then it is able to distinguish which pseudonym corresponds to which temporary public key.

## 4.2 Parameters

All computations in our scheme are performed in the cyclic subgroup $\mathbb{G} :=< g, p, q >$ of the multiplicative group $Z_p^*$ with generator $g$ and prime order $q$, such that $q|(p-1)$. In our scheme we use a NIZK proof system $(P, V)$ with a *random oracle R* as described in Section 4.1 and a publicly known strong collision-resistant hash function $H : \{0,1\}^* \rightarrow \mathbb{Z}_p$. We assume that members agree on the description of $\mathbb{G}$, *R* and $H$ before they begin with the setup procedure. All protocols and algorithms of our scheme implicitly know the description of $\mathbb{G}$ and *R*.
Every participant $i$ is equipped with an identity certificate that contains its authenticated public key $pkey_i$. The corresponding private key $skey_i$ is known only to this participant. We assume that the key pair $(pkey_i, skey_i)$ is used by member $i$ to authenticate his messages during the setup protocol. Each secret signing key $sk_i$ consists of the member's temporary *private key* $x_i$ and the *tracing trapdoor* $k$ that is known only to the group members. For simplicity the identity $id_i$ of member $i$ is given by its temporary *public key* $y_i$, such that $y_i \leftarrow g^{x_i}$, hence $ID := \{y_0, y_1\}$. The pseudonym $ps_i$ of member $i$ is a tuple $(bk, \tilde{y}_i)$, where $bk \leftarrow g^k$ is the *blinded common tracing trapdoor*, and $\tilde{y}_i \leftarrow y_i^k$ is its *pseudonym token*.

## 4.3 Protocols and Algorithms

**Protocol** *Setup***:** Both participants 0 and 1 perform protocol *Setup* in Figure 2. We require that all information sent by a participant $i$ during this protocol is signed with a secure digital

$$\begin{array}{rcl}
\text{Member 0} & & \text{Member 1} \\
\hline
x_0 \xleftarrow{\$} \mathbb{Z}_q;\ y_0 \leftarrow g^{x_0} & & x_1 \xleftarrow{\$} \mathbb{Z}_q;\ y_1 \leftarrow g^{x_1} \\
& \xrightarrow{(y_0, Sig_{skey_0}(y_0))} & \\
& \xleftarrow{(y_1, Sig_{skey_1}(y_1))} & \\
k \leftarrow y_1^{x_0};\ bk \leftarrow g^k & & k \leftarrow y_0^{x_1};\ bk \leftarrow g^k \\
\forall i = 0,1 : \tilde{y}_i \leftarrow y_i^k,\ ps_i := (bk, \tilde{y}_i) & & \forall i = 0,1 : \tilde{y}_i \leftarrow y_i^k,\ ps_i := (bk, \tilde{y}_i) \\
ID := \{y_0, y_1\};\ PS := \{ps_0, ps_1\}; & & ID := \{y_0, y_1\};\ PS := \{ps_0, ps_1\}; \\
sk_0 := (x_0, k, ps_0); & & sk_1 := (x_1, k, ps_1);
\end{array}$$

**Fig. 2.** Protocol *Setup* with $n := 2$

signature scheme using its private key *skey*$_i$, denoted $Sig_{skey_i}()$, and is verified by every recipient. For simplicity we omit the indication of the verification procedure.
In the beginning of the protocol both participants 0 and 1 choose own temporary key pairs $(x_0, y_0)$ and $(x_1, y_1)$, respectively, exchange the temporary public keys in an authentic manner and compute the tracing trapdoor $k$ as a secret shared key according to the authenticated Diffie-Hellman key agreement. Every member blinds the tracing trapdoor to $bk$ and computes the pseudonym tokens $\tilde{y}_0$ and $\tilde{y}_1$. In order to allow third parties to verify signatures, the sets *ID* and *PS* have to be made public in an authentic manner such that no third party can relate any $ps_i \in PS$ to $id_i \in ID$ for the same $i$. For this purpose the order of pseudonyms in *PS* must be randomized, denoted $PS \overset{\$}{:=} \{ps_0, ps_1\}$. A simple possibility to provide authenticity of *ID* and *PS* is that one member, w.l.o.g. member 0, publishes the computed sets signed with *skey*$_0$ and member 1 verifies the correctness of the published sets and complains upon any inconsistency. If no complains occur then *ID* and *PS* are authentic.

*Remark 2.* To simplify the security proof we assume that published sets *ID* and *PS* remain correct during the whole lifetime of the scheme.

**Algorithms *Sign*, *Verify*, *Trace*:** Algorithms *Sign*, *Verify*, and *Trace* in Figure 3 allow members to generate and verify signatures and trace signer's identities according to Definition 1. The signer $i$ uses its secret signing key $sk_i$ to generate a signature on a message $m$. It hashes the message $m$ with a random string $r$ to obtain $h$, and computes $z \leftarrow h^{x_i}$, where $x_i$ is its temporary private key. In order to bind the signature to own pseudonym $ps_i = (bk, \tilde{y}_i)$ the signer computes a non-interactive zero-knowledge prove $\pi$ of $\log_h(z) = \log_{bk}(\tilde{y}_i)$ using the proving algorithm *P*. The idea behind this is that $\pi$ can only be constructed by the signer who is in possession of the temporary private key $x_i = \log_h(z) = \log_{bk}(\tilde{y}_i)$. This is important for the traceability (unforgeability) property of the scheme. The signature of member $i$ consists of $r$, $z$, $ps_i$ and $\pi$. Note that instead of including $ps_i$ in the signature it is sufficient to include only the index of $ps_i$ in *PS*. The verification is done by recomputing the hash value $h$, and by the verification of the proof $\pi$. The verification algorithm returns the pseudonym *ps* of the signer if it accepts the signature or $\perp$ if the signature is rejected. In order to trace the signature to the identity of its signer the knowledge of the tracing trapdoor $k$ is required. Since $k$ is part of every secret signing key $sk_i$, every member $i$ is able to perform the tracing algorithm. After the signature is verified the identity of its signer is computed as $y \leftarrow \tilde{y}^{1/k}$, where $1/k$ is the inverse value of the tracing trapdoor and $\tilde{y}$ is the pseudonym token of the signer obtained from the signature. This is possible due to the construction of the latter as $\tilde{y} = y^k$.

```
Sign(sk_i, m):
    Parse $sk_i$ as $(x_i, k, ps_i)$; Parse $ps_i$ as $(bk, \tilde{y}_i)$;
    $r \xleftarrow{\$} \mathbb{Z}_p$; $h \xleftarrow{\$} H(m, r)$; $z \leftarrow h^{x_i}$;
    $\pi \xleftarrow{\$} P(R, h, bk, z, \tilde{y}_i, x_i)$;
    $\sigma := (r, z, ps_i, \pi)$; return $\sigma$;

Verify(σ, m, PS):
    Parse $\sigma$ as $(r, z, ps, \pi)$; Parse $ps$ as $(bk, \tilde{y})$;
    If $ps \notin PS$ then return $\perp$ EndIf;
    $h \xleftarrow{\$} H(m, r)$;
    If $V(R, h, bk, z, \tilde{y}, \pi) = 1$ then return $ps$
    else return $\perp$ EndIf;

Trace(σ, m, sk, PS, ID):
    If $Verify(\sigma, m, PS) = \perp$ then return $\perp$ EndIf;
    Parse $\sigma$ as $(r, z, ps, \pi)$; Parse $ps$ as $(bk, \tilde{y})$; Parse $sk$ as $(x, k, ps')$; $y \leftarrow \tilde{y}^{1/k}$;
    If $y \in ID$ then return $y$ else return $\perp$ EndIf;
```

**Fig. 3.** Algorithms *Sign, Verify, Trace*

## 4.4 Security Analysis

Let $(P, V)$ be an adaptive[1] NIZK proof system for the equality of two discrete logarithms in the random oracle model, and let $2{-}\mathcal{LDGS} = (Setup, Sign, Verify, Trace)$ be a two-party linkable democratic group signature scheme from our construction in Section 4.3.

**Theorem 2.** *The two-party linkable democratic group signature scheme $2{-}\mathcal{LDGS}$ is correct.*

*Proof.* According to Definition 2 we have to show that

$$Verify(\sigma, m, PS) = ps_i \wedge Trace(\sigma, m, sk_j, PS, ID) = id_i$$

for all $l, n \in \mathbb{N}$, $(ID, PS, SK) \in [Setup(l, n)]$, $sk_i, sk_j \in SK$, $id_i \in ID$, $ps_i \in PS$, $m \in \{0,1\}^*$, and $\sigma \xleftarrow{\$} Sign(sk_i, m)$. The proof of the correctness is obvious from the constructions given in figures 2 and 3. The signature can be parsed as $\sigma = (r, z, ps_i, \pi)$. After the verification of the pseudonym (i.e., $ps_i \in PS$) and of the proof $\pi$, the pseudonym $ps_i$ is returned by the verification algorithm. The tracing algorithm checks first whether the signature $\sigma$ is verifiable. This step ensures also the validity of the pseudonym $ps_i$. The computation of the identity $y_i \leftarrow \tilde{y}_i^{1/k}$ from the pseudonym token $\tilde{y}_i$ is valid according to the construction $\tilde{y}_i \leftarrow y_i^k$. The identity $id_i$ is returned by the tracing algorithm after its verification (i.e., $id_i \in ID$). Thus, $2{-}\mathcal{LDGS}$ is correct. □

In the following we prove anonymity and traceability of our linkable democratic group signature scheme in the random oracle model.

**Theorem 3.** *The two-party linkable democratic group signature scheme $2{-}\mathcal{LDGS}$ is anonymous in the random oracle model assuming that $(P, V)$ is an adaptive NIZK proof system for the equality of two discrete logarithms and that the SDDH assumption holds in $\mathbb{G} = <g, p, q>$ .*

*Proof (Sketch).* The complete proof is given in Appendix C.1. To prove the anonymity of $2{-}\mathcal{LDGS}$ in the random oracle model we suppose that $H$ is a random oracle and show that for any polynomial time adversary $A$ against the anonymity property it is possible to construct a polynomial time distinguisher $D$ against the zero-knowledge property of the NIZK proof system $(P, V)$ (i.e., $D$ distinguishes between simulated and real proofs), and adversary $A_s$ that is able to break the SDDH assumption, such that for all $l \in \mathbb{N}$

$$\mathbf{Adv}_A^{\text{anon}}(l, 2) < 2\mathbf{Adv}_D^{\text{zk}}(l) + 2\mathbf{Adv}_{A_s}^{\text{sddh}}(l) + \frac{2}{poly(l)}$$

Obviously, $\mathbf{Adv}_A^{\text{anon}}(l, 2)$ is negligible in terms of the negligibility of a two-argument function. Hence, $2{-}\mathcal{LDGS}$ is anonymous. □

[1] For the security analysis we require the adaptivity of $(P, V)$.

**Theorem 4.** *The two-party linkable democratic group signature scheme $2-\mathcal{LDGS}$ is traceable in the random oracle model assuming that $(P, V)$ is an adaptive NIZK proof system for the equality of two discrete logarithms and that the Computational Diffie-Hellman (CDH) assumption holds in $\mathbb{G} =< g, p, q >$.*

*Proof (Sketch).* The complete proof is given in Appendix C.2. To prove the traceability of $2-\mathcal{LDGS}$ in the random oracle model we suppose that $H$ is a random oracle and show that for any polynomial time adversary $A$ against the traceability property it is possible to construct a polynomial time adversary $A_c$ that is able to break the CDH [6] assumption in the random oracle model, such that for all $l \in \mathbb{N}$

$$\mathbf{Adv}_A^{\mathrm{trace}}(l, 2) < 2\mathbf{Adv}_{A_c}^{\mathrm{cdh}}(l) + \frac{3}{poly(l)}$$

Obviously, $\mathbf{Adv}_A^{\mathrm{trace}}(l, 2)$ is negligible in terms of the negligibility of a two-argument function. Hence, $2-\mathcal{LDGS}$ is traceable. $\square$

## 5 Extension to a Multi-Party Linkable Democratic Group Signature

Obviously, the tracing process in $2-\mathcal{LDGS}$ is trivial, because one member upon receiving a signature which it has not generated immediately knows that another member is the signer, i.e., the execution of the tracing algorithm is not required. However, tracing becomes more interesting in a multi-party case. Our $2-\mathcal{LDGS}$ scheme can be immediately extended to a multi-party case using the idea proposed for democratic group signatures in [20]. The clue is to replace an authenticated Diffie-Hellman protocol in the computation of the tracing trapdoor by an authenticated Diffie-Hellman based group key agreement protocol. Amongst so-called *contributory group key agreement* (CGKA) protocols there exists protocols like [14] and [17] where each member $i$ contributes its temporary public key $y_i = g^{x_i}$ for the interactive computation of the secret shared key $k$ (common tracing trapdoor). To keep the extended $n-\mathcal{LDGS}$ scheme secure the applied CGKA protocol should satisfy the security requirements stated in [14]. For some practical details and comparison of known CGKA protocols we refer to [1]. Note that the algorithms *Sign*, *Verify* and *Trace* of $2-\mathcal{LDGS}$ remain unchanged in $2-\mathcal{LDGS}$ because $x_i$, and $k$ are part of the secret signing key $sk_i$, and $\tilde{y}_i$ can still be computed as $y_i^k$.

## 6 Conclusion

In this paper we have proposed a model for linkable democratic group signatures that can be applied in communication scenarios for groups where a centralized control by a trusted authority is undesirable and the anonymity of the signer is required only against non-members while other group members have rights to trace and identify the signer. By the assignment of a unique pseudonym to every group member any non-member verifier is able to communicate with the anonymous signer from the group.

## References

1. Y. Amir, Y. Kim, C. Nita-Rotaru, and G. Tsudik. On the performance of group key agreement protocols. *ACM Transactions on Information and System Security*, 7(3):457–488, 2004.
2. M. Bellare, D. Micciancio, and B. Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In *Advances in Cryptology (EUROCRYPT 2003),Lecture Notes in Computer Science*, volume 2656, pages 614–629. Springer-Verlag, 2003.
3. M. Bellare and P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *Proceedings of the 1st ACM conference on Computer and communications security*, pages 62–73. ACM Press, 1993.

4. M. Bellare, H. Shi, and C. Zhang. Foundations of group signatures: The case of dynamic groups. In *CT-RSA, Lecture Notes in Computer Science*, volume 2656, pages 136–153. Springer-Verlag, 2005.

5. A. Bender, J. Katz, and R. Morselli. Ring signatures: Stronger definitions, and constructions without random oracles. In *Theory of Cryptography Conference 2006 (to appear)*, Lecture Notes in Computer Science. Springer-Verlag, 2006.

6. D. Boneh. The Decision Diffie-Hellman problem. In *ANTS-III: Proceedings of the Third International Symposium on Algorithmic Number Theory*, pages 48–63. Springer-Verlag, 1998.

7. D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In *Advances in Cryptology—CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 41–55. Springer-Verlag, 2004.

8. J. Camenisch and J. Groth. Group signatures: Better efficiency and new theoretical aspects. In *SCN, Lecture Notes in Computer Science*, volume 3352, pages 120–133. Springer-Verlag, 2004.

9. J. Camenisch and M. Michels. A group signature scheme with improved efficiency. In *Proceedings of the International Conference on the Theory and Applications of Cryptology and Information Security*, pages 160–174. Springer-Verlag, 1998.

10. J. Camenisch and M. Stadler. Efficient group signature schemes for large groups. In *Proceedings of the 17th Annual International Cryptology Conference on Advances in Cryptology, Lecture Notes in Computer Science*, volume 1294, pages 410–424. Springer-Verlag, 1997.

11. D. Chaum and E. van Heyst. Group signatures. In *Advances in Cryptology (EUROCRYPT 1991),Lecture Notes in Computer Science*, volume 547, pages 257–265. Springer-Verlag, 1991.

12. U. Feige, D. Lapidot, and A. Shamir. Multiple non-interactive zero knowledge proofs under general assumptions. *SIAM Journal on Computing*, 29(1):1–28, September 1999.

13. E.-J. Goh and S. Jarecki. A signature scheme as secure as the Diffie-Hellman problem. In *Advances in Cryptology - EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 401–415. Springer-Verlag, 2003.

14. J. Katz and M. Yung. Scalable protocols for authenticated group key exchange. In *Advances in Cryptology (CRYPTO 2003),Lecture Notes in Computer Science*, volume 2729 of *Lecture Notes in Computer Science*, pages 110–125. Springer-Verlag, 2003.

15. A. Kiayias, Y. Tsiounis, and M. Yung. Traceable signatures. In *Advances in Cryptology (EUROCRYPT 2004)*, volume 3027 of *Lecture Notes in Computer Science*, pages 571–589. Springer-Verlag, 2004.

16. A. Kiayias and M. Yung. Group signatures with efficient concurrent join. In *Advances in Cryptology - EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 198–214. Springer-Verlag, 2005.

17. Y. Kim, A. Perrig, and G. Tsudik. Tree-based group key agreement. *ACM Transactions on Information and System Security*, 7(1):60–96, 2004.

18. J. K. Liu, V. K. Wei, and D. S. Wong. Linkable spontaneous anonymous group signature for ad hoc groups (extended abstract). In *Information Security and Privacy: 9th Australasian Conference, ACISP 2004.*, volume 3108 of *Lecture Notes in Computer Science*, pages 325–335. Springer-Verlag, 2004.

19. A. Lysyanskaya, R. L. Rivest, A. Sahai, and S. Wolf. Pseudonym systems. In *Proceedings of the 6th Annual International Workshop on Selected Areas in Cryptography*, pages 184–199. Springer-Verlag, 2000.

20. M. Manulis. Democratic Group Signatures (On an Example of Joint Ventures - Fast Abstract). In *Fast Abstracts Proceedings of ACM Symposium on Information, Computer and Communications Security (ASIACCS'06)*. ACM Press, 2006. Full version at: http://eprint.iacr.org/2005/446.

21. R. L. Rivest, A. Shamir, and Y. Tauman. How to leak a secret. In *ASIACRYPT '01: Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security*, volume 2248 of *Lecture Notes in Computer Science*, pages 552–565. Springer-Verlag, 2001.

22. A. Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *FOCS '99: Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, page 543. IEEE Computer Society, 1999.

## A Adaptive Non-Interactive Zero-Knowledge Proof Systems in the Random Oracle Model

For the security proof of our scheme we use an adaptive non-interactive zero-knowledge (NIZK) proof system of membership in $\mathcal{NP}$ languages as part of sign and verify algorithms. In the following we give a short description of such proof systems in the random oracle model. Our description is similar to the description of adaptive NIZK proof systems in the common reference string model as presented in [12] and [22].
We start with the description of the binary $\mathcal{NP}$ relation $\rho : \{0,1\}^* \times \{0,1\}^*$. It is required that the membership of $(x, w) \in \rho$ is decidable in polynomial time and that there exists a polynomial *poly*, such that $|w| \leq \mathit{poly}(|x|)$. Let value $l \leftarrow |x|$ be the security parameter for the hardness of $\rho$. The value $x$ is called a *theorem*, and $w$ its *witness*. For any such $\mathcal{NP}$ relation $\rho$ there exists an associated $\mathcal{NP}$ language

$$L_\rho = \{x | \exists w : (x, w) \in \rho\}.$$

A non-interactive (NI) proof system for $\mathcal{NP}$ languages consists of two polynomial time algorithms *P* and *V*, where proving algorithm *P* is randomized and verifying algorithm *V* deterministic. Both algorithms have access to the *random oracle* $R : \{0,1\}^* \rightarrow \{0,1\}^\infty$ from the family of random oracles, denoted $2^\infty$. On input a random oracle *R*, a theorem $x$, and its witness $w$, such that $(x, w) \in \rho$, algorithm *P* outputs a non-interactive proof $\pi$ of the membership of $x \in L_\rho$ with respect to *R*. The input to the algorithm *V* is a random oracle *R*, a theorem $x$ and a proof $\pi$. *V* verifies whether $\pi$ is a correct proof for the membership of $x \in L_\rho$ and outputs either 1 or 0 (1 if *V* accepts the proof). NI proof systems must satisfy the requirements on *completeness* (if $x \in L_\rho$ then proof $\pi$ computed by *P* must be accepted by *V*) and *soundness* (if $x \notin L_\rho$ then for any proving algorithm $P^*$ the probability that *V* accepts its proof $\pi$ is negligible).
Non-interactive zero-knowledge (NIZK) proof systems have additionally a *zero-knowledge* requirement, which involves the existence of the probabilistic polynomial time algorithm *S*, called a *simulator*. The input to the simulator *S* is the theorem $x$ (equivalent to the input to *P* and *V*), however, without its associated witness $w$. The output of *S* consists of the simulation of the random oracle[2] $R'$ and the simulated non-interactive proof $\pi'$, such that algorithm *V* accepts $\pi'$ if $x \in L_\rho$. Another part of the requirement is that the simulated random oracle and simulated non-interactive proofs must be computationally indistinguishable from the real random oracle and the real proofs computed by *P*.
*Adaptive* NIZK proof systems strengthen the notion of NIZK proof systems in two ways. The soundness requirement is changed so that any proving algorithm $P^*$ is allowed to choose a theorem $x \notin L_\rho$ after having access to the random oracle *R* and that *V* still accepts its proofs only with negligible probability. For the zero-knowledge requirement the simulator *S* has to output the simulation of the random oracle before it is given a theorem $x$, for which it has then to simulate the non-interactive proof $\pi'$. For this purpose we assume that the simulator operates in two stages, *generate* and *prove*, as shown in Definition 6. In the stage *generate* *S* outputs $R'$. In the stage *prove* the simulator is given the theorem $x$, and has to output the simulated proof.

**Definition 6 (An Adaptive NIZK Proof System).** *An adaptive non-interactive zero-knowledge proof system for a $\mathcal{NP}$ language $L_\rho$ is given by two polynomial time algorithms $(P, V)$ if there exists a polynomial poly such that the following conditions are satisfied:*

[2] Since random oracle is an infinite object the simulator cannot output it directly. Bellare *et. al.* provide in [3] the construction of the simulated random oracle using a special *random oracle completion* operation that takes as input a finite sequence of input/output pairs of strings precomputed by *S* and fills the rest at random. The oracle $R'$ constructed in this way is a random subject to the constraint that on inputs that are parts of the precomputed pairs it returns the corresponding outputs, but on any other input its output is fully random.

1. *Completeness:* $\forall(x, w) \in \rho$, $\forall l \in \mathbb{N}$, $|x| \leq l$

$$\Pr[R \xleftarrow{\$} 2^{\infty}; \pi \xleftarrow{\$} P(R, x, w) : V(R, x, \pi) = 1] = 1$$

2. *Soundness:* $\forall P^*$, $\forall l \in \mathbb{N}$, $|x| \leq l$

$$\Pr[R \xleftarrow{\$} 2^{\infty}; (x, \pi) \xleftarrow{\$} P^*(R) : V(R, x, \pi) = 1 \wedge x \notin L_\rho] < \frac{1}{poly(l)}$$

3. *Zero-Knowledge: Let S and D be two algorithms. Consider the following experiments where D may query a black-box oracle* Prove() *on any* $(x, w) \in \rho$*:*

```
$\mathbf{Exp}_D^{\mathrm{zk}-1}(l)$:                  |  $\mathbf{Exp}_D^{\mathrm{zk}-0}(l)$:
    $R \xleftarrow{\$} 2^{\infty}$;                   |      $R' \xleftarrow{\$} S(generate, l)$;
    $d \xleftarrow{\$} D^{Prove(\cdot,\cdot)}(R)$;    |      $d \xleftarrow{\$} D^{Prove(\cdot,\cdot)}(R')$;
    return $d$;                                       |      return $d$;
$\mathrm{Prove}(x, w)$:                               |  $\mathrm{Prove}(x, w)$:
    $\pi \xleftarrow{\$} P(R, x, w)$;                 |      $\pi' \xleftarrow{\$} S(prove, R', x)$;
    return $\pi$;                                     |  return $\pi'$;
```

*We require that there exists a polynomial time simulator S such that for any polynomial time distinguisher D the following is negligible in l:*

$$\mathbf{Adv}_D^{\mathrm{zk}}(l) = \Pr[\mathbf{Exp}_D^{\mathrm{zk}-1}(l) = 1] - \Pr[\mathbf{Exp}_D^{\mathrm{zk}-0}(l) = 1] \qquad (2)$$

$\mathbf{Adv}_D^{\mathrm{zk}}(l)$ *denotes the advantage of D in distinguishing the real experiment* $\mathbf{Exp}_D^{\mathrm{zk}-1}(l)$ *from the simulated experiment* $\mathbf{Exp}_D^{\mathrm{zk}-0}(l)$*. (Note that in* $\mathbf{Exp}_D^{\mathrm{zk}-1}(l)$ *oracle* Prove() *contains the proving algorithm P, whereas in* $\mathbf{Exp}_D^{\mathrm{zk}-0}(l)$ *it contains the simulator S.)*

This definition of the advantage function is intuitively equivalent to the requirement of indistinguishability between random oracles and their simulation, and between real proofs and simulated proofs. The output bit $d$ of the distinguisher $D$ is a guess of the environment (experiment) in which it is running, either simulated ($d = 0$) or real ($d = 1$). The experiments in the zero-knowledge part of the definition represent the *adaptive indistinguishability* test from [12].

According to [12] and [22] an adaptive NIZK proof system exists for any $\mathcal{NP}$ language $L_\rho$ under the assumption that one-way trapdoor permutations exist for the associated $\mathcal{NP}$ relation $\rho$.

The $\mathcal{NP}$ relation $\rho$ which is used in our construction can be formally described as $\rho : \mathbb{G}^4 \times \mathbb{Z}_q$ with

$$(x, w) \in \rho \Longleftrightarrow x := (g_0, g_1, y_0, y_1), \{g_i, y_i\} \in \mathbb{G}, i \in \{0, 1\} \wedge w := \log_{g_0}(y_0) = \log_{g_1}(y_1),$$

and the associated $\mathcal{NP}$ language is $L_\rho := \{x | \exists w : (x, w) \in \rho\}$.

Obviously, $\rho$ specifies the equality of two discrete logarithms and the one-way trapdoor permutation associated with $\rho$ is given by the discrete logarithm function log. Hence, there exists an adaptive NIZK proof system $(P, V)$ of the membership in $L_\rho$.

## B SDDH assumption

### B.1 Proof of Theorem 1 (SDDH $\Leftrightarrow$ DDH)

The proof of equivalence of both assumptions consists of two reductions, SDDH $\leq$ DDH and DDH $\leq$ SDDH. Let $\mathbf{Exp}_A^{\mathrm{ddh-b}}(l)$ denote the experiment where a polynomial time distinguisher $A$ tries to distinguish between two distributions $D_0 = (g, g^x, g^y, g^r)$ and $D_1 = (g, g^x, g^y, g^{xy})$ with $x, y, r \in \mathbb{Z}_q$, i.e., $A$ tries to break the DDH assumption [6] by guessing which distribution $D_b$ it has received. By $\mathbf{Adv}_A^{\mathrm{ddh}}(l) = \Pr[\mathbf{Exp}_A^{\mathrm{ddh-1}}(l) = 1] - \Pr[\mathbf{Exp}_A^{\mathrm{ddh-0}}(l) = 1]$ we denote the success probability of $A$. The hardness of the DDH assumption requires that $\mathbf{Adv}_A^{\mathrm{ddh}}(l)$ is negligible.

The reduction SDDH $\leq$ DDH is trivial: $A_s$ receives $D'_b = (g, g^{x_0}, \ldots, g^{x_{n-1}}, g^y, g^{s_0}, \ldots, g^{s_{n-1}})$ where for all $j \in [0, n-1]$ $s_j$ is either a random value from $\mathbb{Z}_q$ (case $b = 0$) or a product $x_j y$ (case $b = 1$); $A_s$ picks $i \overset{\$}{\leftarrow} \{0, \ldots, n-1\}$, and passes the tuple $D_b = (g, g^{x_i}, g^y, g^{s_i})$ over to $A$ in the experiment $\mathbf{Exp}_A^{\mathrm{ddh-b}}(l)$. The output of $A$ is also the output of $A_s$.

In the following we show that for any $A_s$ being able to break the SDDH assumption, one can construct a polynomial time distinguisher $A$ against the DDH assumption (Figure 4), such that $\forall l \in \mathbb{N} : \mathbf{Adv}_A^{\mathrm{ddh}}(l) = \mathbf{Adv}_{A_s}^{\mathrm{sddh}}(l)$. $A$ uses $A_s$ as a black-box. The input to $A$ is

```
$A(g, g^x, g^y, g^s)$:
    $n \overset{\$}{\leftarrow} \mathbb{N}$; For $i = 1$ to $n - 1$ do $u_i \overset{\$}{\leftarrow} \mathbb{Z}_q$ EndFor;
    $D'_b := (g, (g^x)^{u_0}, (g^x)^{u_1}, \ldots, (g^x)^{u_{n-1}}, g^y, (g^s)^{u_0}, (g^s)^{u_1}, \ldots, (g^s)^{u_{n-1}})$;
    $d \overset{\$}{\leftarrow} A_s(D'_b)$;
    return $d$;
```

**Fig. 4.** Construction of the distinguisher $A$ (DDH $\leq$ SDDH)

a distribution $D_b$ from experiment $\mathbf{Exp}_A^{\mathrm{ddh-b}}(l)$. In case $b = 0$ value $s$ is a random value from $\mathbb{Z}_q$, whereas in case $b = 1$ it equals to the product $xy$. $A$ extends the distribution $D_b$ to a distribution $D'_b$ by picking random values $u_i \overset{\$}{\leftarrow} \mathbb{Z}_q$ for all $i \in [0, n-1]$ and using them as exponents for $g^x$ and $g^s$, i.e., computing the values $g^{xu_i}$ and $g^{su_i}$, such that $su_i$ is either a random value from $\mathbb{Z}_q$ or the product $xyu_i$ depending on the initial value for $b$. The SDDH distribution $D'_b$ is then passed over to $A_s$ that outputs bit $d$, which is also the output of $A$. In case that $s$ is a random value from $\mathbb{Z}_q$ the distribution $D'_b$ is indistinguishable from the distribution $D'_0$ in $\mathbf{Exp}_{A_s}^{\mathrm{sddh-b}}(l)$ in Definition 5 because the values $g^{xu_i}$ and $g^{su_i}$ are indistinguishable for any $i \in [0, n-1]$. In case that $s$ is a product $xy$ the distribution $D'_b$ is indistinguishable from the distribution $D'_1$ since the same dependencies between $g^{xu_i}$ and $g^{su_i}$ are preserved. It is obvious, that $A$ guesses correctly the distribution $D_b$ it has been given, exactly when $A_s$ guesses correctly the distribution $D'_b$ it has received from $A$, i.e., if $A_s$ wins in the experiment $\mathbf{Exp}_{A_s}^{\mathrm{sddh-b}}(l)$, no matter what $b$ is. In the following we compute the advantage of $A$, i.e., $\mathbf{Adv}_A^{\mathrm{ddh}}(l) = \Pr[\mathbf{Exp}_A^{\mathrm{ddh-1}}(l) = 1] - \Pr[\mathbf{Exp}_A^{\mathrm{ddh-0}}(l) = 1]$, where $\Pr[\mathbf{Exp}_A^{\mathrm{ddh-1}}(l) = 1] = \Pr[A_s(D'_1) = 1]$ and $\Pr[\mathbf{Exp}_A^{\mathrm{ddh-0}}(l) = 1] = \Pr[A_s(D'_0) = 1]$. Considering that $\Pr[A_s(D'_1) = 1] = \Pr[\mathbf{Exp}_{A_s}^{\mathrm{sddh-1}}(l) = 1]$ and $\Pr[A_s(D'_0) = 1] = \Pr[\mathbf{Exp}_{A_s}^{\mathrm{sddh-0}}(l) = 1]$ with Equation (1) we obtain:

$$\mathbf{Adv}_A^{\mathrm{ddh}}(l) = \Pr[\mathbf{Exp}_{A_s}^{\mathrm{sddh-1}}(l) = 1] - \Pr[\mathbf{Exp}_{A_s}^{\mathrm{sddh-0}}(l) = 1] = \mathbf{Adv}_{A_s}^{\mathrm{sddh}}(l)$$

$\square$

## C Security Proof of $2-\mathcal{LDGS}$

### C.1 Proof of Theorem 3 (Anonymity)

By the assumption $(P, V)$ is an adaptive NIZK proof system for the equality of two discrete logarithms (Definition 6), and sets *ID* and *PS* remain correct after being published in the *Setup* protocol (Remark 2), such that $\forall y_i \in ID, ps_i \in PS : ps_i = (bk, \tilde{y}_i) \wedge bk = g^k \wedge \tilde{y}_i = y_i^k$.
We show that for any polynomial time adversary $A$ being able to attack the anonymity of $2-\mathcal{LDGS}$ it is possible to construct a polynomial time distinguisher $D$ that distinguishes between simulated and real proofs, and an adversary $A_s$ that is able to break the SDDH assumption (Definition 5) in the random oracle model, such that for all $l \in \mathbb{N}$

$$\mathbf{Adv}_{A}^{\text{anon}}(l, 2) < 2\mathbf{Adv}_{D}^{\text{zk}}(l) + 2\mathbf{Adv}_{A_s}^{\text{sddh}}(l) + \frac{2}{poly(l)} \tag{3}$$

By the assumption on the security of $(P, V)$ and SDDH, all advantage functions on the right side are negligible, and so is the function on the left. Thus, $\mathbf{Adv}_{A}^{\text{anon}}(l, 2)$ is negligible in terms of negligibility of a two-argument function, and it follows that $2-\mathcal{LDGS}$ is anonymous. Besides the random oracle *R* $A$ can query oracle *H* (for the hash function *H* from algorithms *Sign* and *Verify*) on tuples of the form $(r, m)$.

THE DISTINGUISHER FOR ZERO-KNOWLEDGE. The distinguisher $D$ (Figure 5) starts by initialising the scheme with *Setup* protocol that it performs for both members 0 and 1. $D$ chooses the description of $\mathbb{G} :=< g, p, q >$. The random oracle $R$ is supplied to $D$ by the environment in which it is run, so that it can be either simulated or real. All new queries of $A$ to the random oracle *H* are answered by $D$ at random. After $D$ has initialised the scheme it runs $A$ against the anonymity of $2-\mathcal{LDGS}$. $D$ passes the tuple $(\mathbb{G}, R, PS, ID)$ over to $A$. In the *choose* stage $A$ outputs a message $m$. $D$ computes the signature $\sigma_b$ on $m$ using a secret signing key $sk_b$ for a random $b \xleftarrow{\$} \{0, 1\}$ as follows. First, $D$ computes values $r$, $h$ and $z$ according to *Sign* algorithm, and then queries black-box *Prove* on the tuple $((h, bk, z, \tilde{y}_b), x_b)$, and obtains a proof $\pi$ for $\log_h(z) = \log_{bk}(\tilde{y}_b)$, which is either simulated or real depending on the environment in which $D$ is running. $D$ includes $\pi$ in $\sigma_b$. In the *guess* stage $A$ may query oracle *Sign*$^*$ on any message $m^*$ of its choice. The oracle answers with the signature $\sigma_{b^*}$ on $m^*$ computed as $\sigma_b$ above. At the end of the stage $A$ outputs bit $d$. The output of $D$ is 1 if $d = b$ and 0 otherwise. In the following we compute the advantage of $D$ in breaking the zero-knowledge property of *(P,V,S)* according to Equation (2).

First, we consider experiment $\mathbf{Exp}_{D}^{\text{zk}-1}(l)$. In this experiment the random oracle *R* is real and $\pi$ is computed by the real proving algorithm *P*. Thus, $\sigma_b$ perfectly emulates the signature returned by the signing algorithm *Sign*. $D$ outputs 1 whenever $A$ is able to distinguish between these signatures. We write $A(\textit{guess})$ for the outcome of the *guess* stage of $A$, when $\sigma_b$ is a perfectly emulated signature. In the following we compute the probability of $D$ distinguishing its environment for this case, i.e. ouputting 1:

$$\begin{aligned}
\Pr[\mathbf{Exp}_{D}^{\text{zk}-1}(l) = 1] &= \Pr[A(\textit{guess}) = 1 | b = 1]\Pr[b = 1] + \Pr[A(\textit{guess}) = 0 | b = 0]\Pr[b = 0] \\
&= \frac{1}{2}\Pr[\mathbf{Exp}_{A}^{\text{anon}-1}(l, 2) = 1] + \frac{1}{2}\Pr[\mathbf{Exp}_{A}^{\text{anon}-0}(l, 2) = 0] \\
&= \frac{1}{2}\Pr[\mathbf{Exp}_{A}^{\text{anon}-1}(l, 2) = 1] + \frac{1}{2}(1 - \Pr[\mathbf{Exp}_{A}^{\text{anon}-0}(l, 2) = 1]) \\
&= \frac{1}{2} + \frac{1}{2}(\Pr[\mathbf{Exp}_{A}^{\text{anon}-1}(l, 2) = 1] - \Pr[\mathbf{Exp}_{A}^{\text{anon}-0}(l, 2) = 1]) \\
&= \frac{1}{2} + \frac{1}{2}\mathbf{Adv}_{A}^{\text{anon}}(l, 2)
\end{aligned} \tag{4}$$

Secondly, we consider experiment $\mathbf{Exp}_{D}^{\mathrm{zk}-0}(l)$. In this experiment $D$ is running in the simulated environment, i.e., the random oracle $R$ and proof $\pi$ are produced by the simulator $S$. Thus, $\sigma$ is a signature-like looking tuple. We classify these tuples into two distributions $E_b$, $b = \{0, 1\}$, depending on the pseudonym $ps_b$ that is part of $\sigma_b$. $D$ outputs 1 whenever $A$ is able to distinguish between these signature-like looking tuples. We write $A(E_0)$ and $A(E_1)$ for the outcome of the *guess* stage of $A$, when $\sigma_b$ is sampled from distribution $E_0$ and $E_1$, respectively.

$$\begin{aligned}\Pr[\mathbf{Exp}_{D}^{\mathrm{zk}-0}(l) = 1] &= \Pr[A(E_1) = 1]\Pr[b = 1] + \Pr[A(E_0) = 0]\Pr[b = 0] \\ &= \frac{1}{2}(\Pr[A(E_1) = 1] + \Pr[A(E_0) = 0])\end{aligned} \tag{5}$$

In order to estimate this probability precisely we relate it to the probability of breaking the SDDH assumption by the adversary $A_s$ that uses $A$, assuming that $A$ is able to distinguish between signatures sampled from $E_b$. The construction of such $A_s$ is given in Figure 5 and described in the following.

ADVERSARY AGAINST SDDH ASSUMPTION. The idea is to construct $A_s$ that challenges $A$ on signature-like looking tuples from distributions $E_b$, $b = \{0, 1\}$. According to experiment $\mathbf{Exp}_{A_s}^{\mathrm{sddh-b}}(l)$ from Definition 5 $A_s$ is given a distribution $D' = (g, g^{x_0}, g^{x_1}, g^y, g^{s_0}, g^{s_1})$, such that $s_i$, $i \in \{0, 1\}$ is either a random value from $\mathbb{Z}_q$ or equals to the product $x_i y$. $A_s$ knows the description of $\mathbb{G} :=< g, p, q >$ from the environment that has produced $D'$. $A_s$ obtains random oracle $R$ from the simulator's $S$ *generate* stage, defines $y_i \leftarrow g^{x_i}$, $bk \leftarrow g^y$ (unknown $y$ is considered to be the tracing trapdoor $k$) and $\tilde{y}_i \leftarrow g^{s_i}$, and passes the tuple $(\mathbb{G}, R, PS, ID)$ over to $A$. $A_s$ simulates $H$ by answering any new query of $A$ at random. In order to compute the signature-like looking tuple from distribution $E_b$ adversary $A_s$ selects $r, u \overset{\$}{\leftarrow} \mathbb{Z}_q$, computes $h \leftarrow bk^u$ and $z \leftarrow \tilde{y}_b^u$, obtains from $S$ the simulated proof $\pi$ for $\log_h(z) = \log_{bk}(\tilde{y}_b)$, sets $H(r, m)$ to $h$ and returns $\sigma_b := (r, z, ps_b, \pi)$ to $A$. In the *guess* stage $A$ may query oracle *Sign*$^*$ on any message $m^*$ of its choice. The oracle answers with the signature $\sigma_{b^*}$ on $m^*$ computed as $\sigma_b$ above. At the end of the stage $A$ outputs bit $d$. The output of $A_s$ is 1 if $d = b$ and 0 otherwise. In the following we compute the advantage of $A_s$ in breaking the SDDH assumption according to Equation (1).

First, we consider experiment $\mathbf{Exp}_{A_s}^{\mathrm{sddh}-1}(l)$. In this experiment $A_s$ receives distribution $D' = (g, g^{x_0}, g^{x_1}, g^y, g^{x_0 y}, g^{x_1 y})$, hence $\tilde{y}_i = y_i^k$ holds. The signature $\sigma_b$ that $A_s$ passes over to $A$ contains real value for $ps_b$ but simulated proof $\pi$, and is therefore sampled from distribution $E_b$. $A_s$ outputs 1 whenever $A$ is able to distinguish between these signatures-like looking tuples. In the following we compute the probability of $A_s$ breaking the SDDH assumption in this case, i.e., $\mathbf{Exp}_{A_s}^{\mathrm{sddh}-1}(l)$ outputs 1:

$$\begin{aligned}\Pr[\mathbf{Exp}_{A_s}^{\mathrm{sddh}-1}(l) = 1] &= \Pr[A(E_1) = 1]\Pr[b = 1] + \Pr[A(E_0) = 0]\Pr[b = 0] \\ &= \frac{1}{2}(\Pr[A(E_1) = 1] + \Pr[A(E_0) = 0])\end{aligned}$$

With Equation (5) we obtain:

$$\Pr[\mathbf{Exp}_{A_s}^{\mathrm{sddh}-1}(l) = 1] = \Pr[\mathbf{Exp}_{D}^{\mathrm{zk}-0}(l) = 1] \tag{6}$$

Secondly, we consider the experiment $\mathbf{Exp}_{A_s}^{\mathrm{sddh}-0}(l)$. In this experiment $A_s$ receives distribution $D' = (g, g^{x_0}, g^{x_1}, g^y, g^{r_0}, g^{r_1})$, where $r_i$ are random values from $\mathbb{Z}_q$. Hence, pseudonym tokens $\tilde{y}_i$ are random values and $\tilde{y}_i \neq y_i^k$. The signature $\sigma_b$ that $A_s$ passes over to $A$ contains the random value for $ps_b$ and the simulated proof $\pi$. For this kind of signature-like looking tuples we specify further two distributions $E'_b$, $b = \{0,1\}$. $A_s$ outputs 1 whenever $A$ is able to distinguish between signature-like looking tuples sampled from $E'_b$. We write $A(E'_0)$ and $A(E'_1)$ for the outcome of the *guess* stage of $A$, when $\sigma$ is sampled from distribution $E'_0$ and $E'_1$, respectively. Since signatures of $E'_b$ consist of all random values and simulated proofs $A$ can distinguish them only with a probability that is negligibly greater than that of a random guess. Hence,

$$\begin{aligned}\Pr[\mathbf{Exp}_{A_s}^{\text{sddh}-0}(l) = 1] &= \Pr[A(E'_1) = 1]\Pr[b = 1] + \Pr[A(E'_0) = 0]\Pr[b = 0] \\ &= \frac{1}{2}(\Pr[A(E'_1) = 1] + \Pr[A(E'_0) = 0]) \\ &< \frac{1}{2}\left(\frac{1}{2} + \frac{1}{poly(l)} + \frac{1}{2} + \frac{1}{poly(l)}\right) \\ &= \frac{1}{2} + \frac{1}{poly(l)}\end{aligned} \tag{7}$$

Distinguisher $D^{Prove(\cdot,\cdot)}(R, l)$:

```
    Choose $\mathbb{G} :=< g, p, q >$; $(ID, PS, SK) \xleftarrow{\$} Setup(l, 2)$;
    $(\mathrm{St}, m) \xleftarrow{\$} A^{H(\cdot,\cdot)}(choose, \mathbb{G}, R, PS, ID)$;
    $b \xleftarrow{\$} \{0,1\}$;
    $r \xleftarrow{\$} \mathbb{Z}_p$; $h \xleftarrow{\$} H(r, m)$; $z \leftarrow h^{x_b}$;
    $\pi \xleftarrow{\$} Prove((h, bk, z, \tilde{y}_b), x_b)$; [black-box query]
    $\sigma_b := (r, z, ps_b, \pi)$; $d \xleftarrow{\$} A^{Sign^*(\cdot), H(\cdot,\cdot)}(guess, \mathrm{St}, \sigma_b)$;
    If $d = b$ then return 1 else return 0 EndIf;
```

Adversary $A_s(D')$:

```
    Parse $D'$ as $(g, g^{x_0}, g^{x_1}, g^y, g^{s_0}, g^{s_1})$; Get $\mathbb{G} :=< g, p, q >$;
    $R \xleftarrow{\$} S(generate, l)$; $bk \leftarrow g^y$;
    $\forall i \in \{0,1\} : y_i \leftarrow g^{x_i}$, $\tilde{y}_i \leftarrow g^{s_i}$, $ps_i := (bk, \tilde{y}_i)$;
    $ID := \{y_i\}$; $PS \overset{\$}{:=} \{ps_i\}$;
    $(\mathrm{St}, m) \xleftarrow{\$} A^{H(\cdot,\cdot)}(choose, \mathbb{G}, R, PS, ID)$;
    $b \xleftarrow{\$} \{0,1\}$;
    $r \xleftarrow{\$} \mathbb{Z}_p$, until $H(r, m)$ has not been previously queried;
    $u \xleftarrow{\$} \mathbb{Z}_q$; $h \leftarrow bk^u$; $z \leftarrow \tilde{y}_b^u$; define $H(r, m) := h$;
    $\pi \xleftarrow{\$} S(prove, (h, bk, z, \tilde{y}_b))$; [note, $\log_h(z) = \log_{bk}(\tilde{y}_b)$]
    $\sigma_b := (r, z, ps_b, \pi)$; $d \xleftarrow{\$} A^{Sign^*(\cdot), H(\cdot,\cdot)}(guess, \mathrm{St}, \sigma_b)$;
    If $d = b$ then return 1 else return 0 EndIf;
```

**Fig. 5.** Distinguisher *D* and adversary $A_s$

PUTTING IT ALL TOGETHER. In the following we show how to relate the probabilities for the outcome of experiments described above with the advantage of $A$ in breaking the anonymity of $2-\mathcal{LDGS}$. We start with the advantage of distinguisher $D$ according to Equation (2):

$$\mathbf{Adv}_D^{\text{zk}}(l) = \Pr[\mathbf{Exp}_D^{\text{zk}-1}(l) = 1] - \Pr[\mathbf{Exp}_D^{\text{zk}-0}(l) = 1]$$

With Equations (4) and (6) we get:

$$\mathbf{Adv}_D^{\text{zk}}(l) = \frac{1}{2} + \frac{1}{2}\mathbf{Adv}_A^{\text{anon}}(l, 2) - \Pr[\mathbf{Exp}_{A_s}^{\text{sddh}-1}(l) = 1]$$

Now, we add to both sides of the above equation the left side of Equation (7) and transform it:

$$\begin{aligned}&\mathbf{Adv}_D^{\text{zk}}(l) + \Pr[\mathbf{Exp}_{A_s}^{\text{sddh}-0}(l) = 1] \\ &\quad = \frac{1}{2} + \frac{1}{2}\mathbf{Adv}_A^{\text{anon}}(l, 2) - (\Pr[\mathbf{Exp}_{A_s}^{\text{sddh}-1}(l) = 1] - \Pr[\mathbf{Exp}_{A_s}^{\text{sddh}-0}(l) = 1])\end{aligned}$$

With Equation (1) and additional transformations we get:

$$\mathbf{Adv}_A^{\mathrm{anon}}(l,2) = 2\mathbf{Adv}_D^{\mathrm{zk}}(l) + 2\mathbf{Adv}_{A_s}^{\mathrm{sddh}}(l) + 2\Pr[\mathbf{Exp}_{A_s}^{\mathrm{sddh}-0}(l) = 1] - 1$$

We obtain the desired inequality using (7):

$$\begin{aligned}\mathbf{Adv}_A^{\mathrm{anon}}(l,2) &< 2\mathbf{Adv}_{D_s}^{\mathrm{zk}}(l) + 2\mathbf{Adv}_{A_s}^{\mathrm{sddh}}(l) + 2\left(\frac{1}{2} + \frac{1}{\mathit{poly}(l)}\right) - 1\\ &= 2\mathbf{Adv}_D^{\mathrm{zk}}(l) + 2\mathbf{Adv}_{A_s}^{\mathrm{sddh}}(l) + \frac{2}{\mathit{poly}(l)}\end{aligned}$$

□

## C.2 Proof of Theorem 4 (Traceability)

By the assumption $(P, V)$ is an adaptive NIZK proof system for the equality of two discrete logarithms, and sets *ID* and *PS* remain correct after being published in the *Setup* protocol, such that $\forall y_i \in \mathit{ID}, \mathit{ps}_i \in \mathit{PS} : \mathit{ps}_i = (bk, \tilde{y}_i) \wedge bk = g^k \wedge \tilde{y}_i = y_i^k$.
We show that for any polynomial time adversary $A$ against the traceability of $2\!-\!\mathcal{LDGS}$ it is possible to construct a polynomial time adversary $A_c$ that is able to break the CDH assumption ([6]) in the random oracle model, such that for all $l \in \mathbb{N}$

$$\mathbf{Adv}_A^{\mathrm{trace}}(l,2) < 2\mathbf{Adv}_{A_c}^{\mathrm{cdh}}(l) + \frac{3}{\mathit{poly}(l)}. \quad (8)$$

All functions on the right side are negligible, and so is the function on the left. Thus, $\mathbf{Adv}_A^{\mathrm{trace}}(l,2)$ is negligible in terms of negligibility of a two-argument function, and it follows that $2\!-\!\mathcal{LDGS}$ is traceable.

Let $A$ be a traceability adversary against $2\!-\!\mathcal{LDGS}$ and let $\mathit{ID}_c$ be a set of identities of members corrupted by $A$ during its attack. $A$ wins the traceability experiment, i.e., $\mathbf{Exp}_A^{\mathrm{trace}}(l) = 1$, if it returns $(\sigma, m)$ according to

Case 0: $\mathit{Verify}(\sigma, m, \mathit{PS}) = \mathit{ps}$ and $\mathit{Trace}(\sigma, m, \mathit{PS}, \mathrm{sk}) = \perp$ or
Case 1: $\mathit{Verify}(\sigma, m, \mathit{PS}) = \mathit{ps}$ and $\mathit{Trace}(\sigma, m, \mathit{PS}, \mathrm{sk}) = \mathit{id}$ and $\mathit{id} \notin \mathit{ID}_c$ and $\mathit{Sign}(\cdot,\cdot)$ *was not queried on* $(\mathit{id}, m)$

With respect to the construction of $2\!-\!\mathcal{LDGS}$ $A$ wins in the traceability experiment if it can produce forgeries of the following types.

TYPE 0. The forgery is given by $(\sigma = (r, z, \mathit{ps}, \pi), m)$ where $\mathit{ps} = (bk, \tilde{y})$ such that case 0 is occured, i.e., $\mathit{Verify}(\sigma,\ m,\ \mathit{PS}) = \mathit{ps}$ and $\mathit{Trace}(\sigma, m, \mathit{PS}, \mathrm{sk}) = \perp$. According to algorithm *Trace* in Figure 3 case 0 can only occur if $y = \tilde{y}^{1/k}$ and $y \notin \mathit{ID}$. By the assumption that sets *ID* and *PS* remain correct after being published in the *Setup* protocol the probability that such $y$ exists is negligible, thus for any security parameter $l$ we can bound the probability that $A$ produces a forgery of type 0 (event is denoted by $F_0$) as follows:

$$\Pr[F_0] = \Pr[\exists y \notin \mathit{ID} \exists \mathit{ps} = (bk, \tilde{y}) \in \mathit{PS} : \tilde{y} = y^k] < \frac{1}{\mathit{poly}(l)} \quad (9)$$

In case 1 we distinguish between the following two types of a forgery $(\sigma = (r, z, \mathit{ps}, \pi), m)$:

TYPE 1. The forgery is given by $(\sigma = (r, z, \mathit{ps}, \pi), m)$ according to case 1 where $\mathit{ps} = (bk, \tilde{y})$ with $V(R, h, bk, z, \tilde{y}, \pi) = 1$ and $\log_h(z) \neq \log_{bk}(\tilde{y})$. According to the soundness requirement of the adaptive NIZK proof system $(P, V)$ we can bound the probability that

$A$ produces a forgery of type 1 (event is denoted by $F_1$) for any security parameter $l$ as follows:

$$\Pr[F_1] = \Pr[A \text{ outputs } (\sigma, m),\ \sigma = (r, z, \mathit{ps}, \pi),\ \mathit{ps} = (\mathit{bk}, \tilde{y}),\ h \leftarrow H(m, r) : V(R, h, \mathit{bk}, z, \tilde{y}, \pi) = 1 \text{ and } \log_h(z) \neq \log_{\mathit{bk}}(\tilde{y})] < \frac{1}{\mathit{poly}(l)} \qquad (10)$$

TYPE 2. The forgery is given by $(\sigma = (r, z, \mathit{ps}, \pi), m)$ according to case 1 where $\mathit{ps} = (\mathit{bk}, \tilde{y})$ with $V(R, h, \mathit{bk}, z, \tilde{y}, \pi) = 1$ and $\log_h(z) = \log_{\mathit{bk}}(\tilde{y})$.
Let $\mathbf{Exp}^{\mathrm{cdh}}_{A_c}(l)$ denote the experiment where a polynomial time adversary $A_c$ given a tuple $(g^a, g^b)$ with $a, b \in \mathbb{Z}_q$ tries to compute $g^{ab}$, and lets say succeeds if $\mathbf{Exp}^{\mathrm{cdh}}_{A_c}(l)$ returns 1. Obviously, $A$ tries to break the CDH assumption [6]. By $\mathbf{Adv}^{\mathrm{cdh}}_{A_c}(l) = \Pr[\mathbf{Exp}^{\mathrm{cdh}}_{A_c}(l) = 1]$ we denote the success probability of $A$. The hardness of the CDH assumption requires that $\mathbf{Adv}^{\mathrm{cdh}}_{A_c}(l)$ is negligible.
In the following we show that for any polynomial time adversary $A$ being able to construct forgeries of type 2 (event is denoted by $F_2$) there exists a polynomial time adversary $A_c$ against the CDH assumption in the random oracle model, such that for any $l \in \mathbb{N}$

$$\Pr[F_2] < \mathbf{Adv}^{\mathrm{cdh}}_{A_c}(l) + \frac{1}{\mathit{poly}(l)} \qquad (11)$$

ADVERSARY AGAINST THE CDH ASSUMPTION. $A_c$ in Figure 6 uses $A$ as black-box. We show that if $A$ outputs a forgery $(\sigma, m)$ of type 2 according to experiment $\mathbf{Exp}^{\mathrm{trace}}_{A}(l)$ then $A_c$ is able to break the CDH assumption. $A_c$ is given a pair $(g^a, g^b)$ and obtains the description of $\mathbb{G} :=< g, p, q >$ from the environment. It obtains a random oracle $R$ from the simulator, chooses bit $w$, and sets the temporary public key $y_w := g^a$ (note $A_c$ does not know $x_w = a$). With this choice $A_c$ tries to guess the identity $\mathit{id}_w$ that will be returned by algorithm *Trace* on input the forgery $(\sigma, m)$ produced by $A$ at the end of the interaction. Then $A_c$ generates temporary private and public keys of the second member, i.e., $x_{1-w}$ and $y_{1-w}$, computes the tracing trapdoor $k$ and its blinded version $\mathit{bk}$, and corresponding sets $\mathit{ID}$, $\mathit{PS}$ and $\mathit{SK}$. $A_c$ passes the tuple $(\mathbb{G}, R, \mathit{PS}, \mathit{ID})$ over to $A$. In its *corrupt* stage $A$ is allowed to request secret signing keys $\mathit{sk}_i \in \mathit{SK}$ (note that if $\mathit{sk}_w$ is requested then $A_c$ aborts, since it failed to guess the identity of a member whose signature will be forged). $A$'s requests of the type $(m, r)$ to the oracle $H$ are handled by $A_c$ as follows. On every new query $A_c$ picks random $d \overset{\$}{\leftarrow} \mathbb{Z}_q$, computes $h \leftarrow (g^b)^d$, saves tuple $((m, r), h, d)$ in the history list of $H$, and returns $h$. $A$'s requests of the type $(\mathit{id}_i, m)$ to the signing oracle *Sign* are answered by $A_c$ as follows. If $\mathit{id}_i = \mathit{id}_{1-w}$ then $A_c$ computes $\sigma$ using original signing algorithm *Sign*, because the corresponding secret signing key $\mathit{sk}_{1-w}$ was generated by $A_c$. If, however, $\mathit{id}_i = \mathit{id}_w$ then $A_c$ has to create a signature without knowing the secret signing key $\mathit{sk}_w$. For this purpose it selects random $r \overset{\$}{\leftarrow} \mathbb{Z}_p$, such that $H$ has not yet been queried on $(m, r)$, then it selects random $t \overset{\$}{\leftarrow} \mathbb{Z}_q$, computes $z \leftarrow y_w^t$, $h \leftarrow g^t$ and sets $H(m, r) := h$, then simulates the proof $\pi$ for $\log_h(z) = \log_{\mathit{bk}}(\tilde{y}_w)$, and returns $\sigma := (r, z, \mathit{ps}_w, \pi)$. It is obvious that $S$ can be used to simulate $\pi$ because $\log_h(z) = \log_{\mathit{bk}}(\tilde{y}_w)$ is a valid theorem, i.e, $\tilde{y}_w = y_w^k = (g^a)^k = \mathit{bk}^a$ and $z = y_w^t = (g^a)^t = h^a$. At the end of stage *forge* $A$ returns a valid forgery $(\sigma = (r, z, \mathit{ps}, \pi), m)$. $A_c$ tries to translate it into computing $g^{ab}$ as follows: if $\mathit{Trace}(\sigma, m, \mathit{PS}, \mathrm{sk}) = \perp$ then abort, else $\mathit{id}_i \leftarrow \mathit{Trace}(\sigma, m, \mathit{PS}, \mathit{sk})$; if $\mathit{id}_i \neq \mathit{id}_w$ or $A$ has not queried $H$ on $(m, r)$ then abort, else compute $g^{ab} \leftarrow z^{1/d}$ where $d$ stemms from the corresponding tuple $((m, r), h, d)$ in the history list of $H$. The computation of $g^{ab}$ is correct if $z = h^a$ where $h = H(m, r) = (g^b)^d$, thus $\log_h(z) = \log_{\mathit{bk}}(\tilde{y}_w)$ should be satisfied.
In the following we compute the advantage of $A_c$ in breaking the CDH assumption. Thus, we have to compute the success probability of $A_c$ outputting $g^{ab}$. Obviously, $A_c$ succeeds whenether $A$ outputs a successful forgery $(\sigma = (r, z, \mathit{ps}_w, \pi), m)$ such that $\mathit{id}_w \leftarrow$

```
Adversary $A_c(g^a, g^b)$:
  Get $\mathbb{G} := < g, p, q >$; $R \xleftarrow{\$} S(\textit{generate}, l)$;
  $w \xleftarrow{\$} \{0,1\}$;
  $y_w \leftarrow g^a$; $x_{1-w} \xleftarrow{\$} \mathbb{Z}_q$; $y_{1-w} \leftarrow g^{x_{1-w}}$; $ID := \{y_w, y_{1-w}\}$;
  $k \leftarrow y_w^{x_{1-w}}$; $bk \leftarrow g^k$; $\tilde{y}_w \leftarrow y_w^k$; $\tilde{y}_{1-w} \leftarrow y_{1-w}^k$; $ps_w := (bk, \tilde{y}_w)$; $ps_{1-w} := (bk, \tilde{y}_{1-w})$; $PS \overset{\$}{:=} \{ps_w, ps_{1-w}\}$;
  $sk_w := (\perp, k, ps_w)$; $sk_{1-w} := (x_{1-w}, k, ps_{1-w})$; $SK := \{sk_w, sk_{1-w}\}$;
  Run $A$:
    – if in the corrupt stage $A$ requests $sk_w$ then abort;
    – when $A$ makes a Sign-oracle query $(id_{1-w}, m)$ do $\sigma \xleftarrow{\$} Sign(sk_{1-w}, m)$; return $\sigma$ to $A$;
    – when $A$ makes a Sign-oracle query $(id_w, m)$ do
      $r \xleftarrow{\$} \mathbb{Z}_p$; If $H$ has already been queried on $(m, r)$ then retry EndIf;
      $t \xleftarrow{\$} \mathbb{Z}_q$; $z \leftarrow y_w^t$; $h \leftarrow g^t$; define $H(m, r) := h$; $\pi \xleftarrow{\$} S(\textit{prove}, (h, bk, z, \tilde{y}_w))$; return $\sigma := (r, z, ps_w, \pi)$ to $A$;
    – when $A$ makes a H-oracle query $(m, r)$ do
      If $H$ has already been queried on $(m, r)$ then return $h$ from the history list to $A$ EndIf;
      $d \xleftarrow{\$} \mathbb{Z}_q$; $h \leftarrow (g^b)^d$; define $H(m, r) := h$; save $((m, r), h, d)$ in the history list; return $h$ to $A$;
  Until $A$ stops and returns $(\sigma, m)$;
  Parse $\sigma$ as $(r, z, ps_i, \pi)$;
  If $Trace(\sigma, m, PS, \mathrm{sk}) = \perp$ then abort EndIf;
  $id_i \leftarrow Trace(\sigma, m, PS, sk)$;
  If $id_i \neq id_w$ then abort EndIf;
  If $H$ has not been queried on $(m, r)$ then abort EndIf;
  Find tupel $((m, r), h, d)$ in the history list of $H$; Return $g^{ab} \leftarrow z^{1/d}$;
```

**Fig. 6.** Adversary $A_c$

$Trace(\sigma, m, PS, sk)$ and $\log_h(z) = \log_{bk}(\tilde{y}_w)$ (notice this is equivalent to type 2 forgeries, i.e, event $F_2$), and oracle $H$ has been queried on $(m, r)$ (event is denoted HQ), and $A_c$ has correctly guessed $w$. The probability that $A_c$ correctly guesses $w$ is 1/2 and is independent from other events, thus we obtain:

$$\mathbf{Adv}_{A_c}^{\mathrm{cdh}}(l) = \Pr[\mathbf{Exp}_{A_c}^{\mathrm{cdh}}(l) = 1] = \frac{1}{2}\Pr[F_2 \wedge \mathrm{HQ}] \tag{12}$$

Observe that $\Pr[F_2 \wedge \mathrm{HQ}] = \Pr[F_2] - \Pr[F_2 \wedge \neg\mathrm{HQ}]$. $\Pr[F_2 \wedge \neg\mathrm{HQ}]$ is given by the probability that $A$ outputs a forgery of type 2 without querying oracle $H$. Thus, $A$ has to find $z$ such that $z^{-a} = h = H(m, r)$ (note $A$ is not in possession of $a$). This probability is negligible. With Equation (12) we obtain $\Pr[F_2 \wedge \neg\mathrm{HQ}] = \Pr[F_2] - 2\mathbf{Adv}_{A_c}^{\mathrm{cdh}}(l) < 1/poly(l)$. This is equivalent to

$$\Pr[F_2] < 2\mathbf{Adv}_{A_c}^{\mathrm{cdh}}(l) + \frac{1}{poly(l)} \tag{13}$$

PUTTING IT ALL TOGETHER. Using Equations (9), (10) and (13) we can bound the advantage of $A$ as desired in Equation (8):

$$\begin{aligned}
\mathbf{Adv}_A^{\mathrm{trace}}(l, 2) &= \Pr[A \text{ wins experiment } \mathbf{Exp}_A^{\mathrm{trace}}(l)] \\
&= \Pr[F_0 \text{ or } F_1 \text{ or } F_2] \\
&< \frac{1}{poly(l)} + \frac{1}{poly(l)} + 2\mathbf{Adv}_{A_c}^{\mathrm{cdh}}(l) + \frac{1}{poly(l)} \\
&= 2\mathbf{Adv}_{A_c}^{\mathrm{cdh}}(l) + \frac{3}{poly(l)}
\end{aligned}$$

□