This is the full version of the extended abstract which appears in Proceedings of the International Conference on Applied Cryptography and Network Security 2016 (ACNS 2016).

Foundations of Fully Dynamic Group Signatures*

Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos†, Essam Ghadafi, and Jens Groth

{jonathan.bootle.14, andrea.cerulli.13, pyrros.chaidos.10, e.ghadafi, j.groth}@ucl.ac.uk
University College London, London, UK

Abstract. Group signatures are a central cryptographic primitive that has received a considerable amount of attention from the cryptographic community. They allow members of a group to anonymously sign on behalf of the group. Membership is overseen by a designated group manager. There is also a tracing authority that can revoke anonymity by revealing the identity of the signer if and when needed, to enforce accountability and deter abuse. For the primitive to be applicable in practice, it needs to support fully dynamic groups, i.e. users can join and leave at any time. In this work we take a close look at existing security definitions for fully dynamic group signatures. We identify a number of shortcomings in existing security definitions and fill the gap by providing a formal rigorous security model for the primitive. Our model is general and is not tailored towards a specific design paradigm and can therefore, as we show, be used to argue about the security of different existing constructions following different design paradigms. Our definitions are stringent and when possible incorporate protection against maliciously chosen keys. In the process, we identify a subtle issue inherent to one design paradigm, where new members might try to implicate older ones by means of back-dated signatures. This is not captured by existing models. We propose some inexpensive fixes for some existing constructions to avoid the issue.

Keywords. Group Signatures, Security Definitions.

1 Introduction

Group signatures, put forward by Chaum and van Heyst [CvH91], are a fundamental cryptographic primitive allowing a member of a group (administered by a designated manager) to anonymously sign messages on behalf of the group. In the case of a dispute, a designated tracing manager can revoke anonymity by revealing the signer. In many settings

*The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement n. 307937 and EPSRC grant EP/J009520/1.

†Was supported by an EPSRC scholarship (EP/G037264/1 – Security Science DTC).

it is desirable to offer flexibility in joining and leaving the group. In static group signatures [BMW03], the group population is fixed once and for all at the setup phase. Partially dynamic group signatures [BSZ05, KY06] allow the enrolment of members in the group at any time but members cannot leave once they have joined. A challenging problem in group signatures is that of revocation, i.e. allowing removal of members from the group.

Related Work. After their introduction, a long line of research on group signatures has emerged. In the early years, security of group signatures was not well understood and early constructions were proven secure via informal arguments using various interpretations of their requirements.

Bellare et al. [BMW03] formalized the security definitions for static groups. In their model, the group manager (which also acts as the tracing authority) needs to be fully trusted. Later on, Bellare et al. [BSZ05] and Kiayias and Yung [KY06] provided formal security definitions for the more practical partially dynamic case. Also, [BSZ05] separated the tracing role from the group management. In both [BSZ05, KY06] models, members cannot leave the group once they have joined. More recently, Sakai et al. [SSE+12] strengthened the security definitions for partially dynamic groups by defining *opening soundness*, ensuring that a valid signature only traces to one user.

Group Signatures Without Revocation. Constructions of group signatures in the random oracle model [BR93] include [CS97, CM98, ACJT00, BBS04, CL04, CG04, NS04, FI05, FY04, KY05, DP06, BCN+10]. Constructions not relying on random oracles include [ACHdM05, Gro06, BW06, Gro07, BW07, AHO10].

Group Signatures With Revocation. Since revocation is an essential feature of group signatures, many researchers investigated the different approaches via which such a feature can be realized. One approach is for the group manager to change the group public key when members are removed and issue new group signing keys to all remaining legitimate members or allow them to update their old signing keys accordingly. This is the approach adopted by e.g. [TX03, CL02].

Bresson and Stern [BS01] realize revocation by requiring that the signer proves at the time of signing that her group membership certificate is not among those contained in a public revocation list. Another approach, which was adopted by e.g. [CL02, TX03, DKNS04, Ngu05], uses accumulators, i.e. functions that map a set of values into a fixed-length string and permit efficient proofs of membership.

Boneh, Boyen and Shacham [BBS04] showed that their static group signature scheme supports revocation since it allows members to update their signing keys according to the changes in the group without the involvement of the manager. Camenisch and Groth [CG04] also gave a construction that supports revocation. Song [Son01] gave a fully dynamic group signature with forward security.

A different approach for revocation known as *Verifier Local Revocation* (VLR), which needs relaxation of some of the security requirements, considered by Brickell [Bri04], was subsequently formalized by Boyen and Shacham [BS04] and further used in e.g. [NF05, LV09, LLNW14]. In VLR, the revocation information (i.e. revocation lists) is only sent to the verifiers (as opposed to both verifiers and signers) who can check whether a particular signature was generated by a revoked member. A similar approach is also used in Direct Anonymous Attestation (DAA) protocols [BCC04]. *Traceable Signatures* [KTY04] extend this idea, as the group manager can release a trapdoor for each member, enabling their signatures to be traced back to the individual user.

More recently, Libert, Peters and Yung [LPY12b, LPY12a] gave a number of efficient constructions of group signatures supporting revocation without requiring random oracles by utilizing the subset cover framework [NNL01] that was originally used in the context of broadcast encryption.

Shortcomings in Existing Models & Motivation. While the security of the static and partially dynamic group settings has been rigorously formulated [BMW03, BSZ05, KY06, SSE+12] and is now well understood, unfortunately, the security of their fully dynamic groups counterpart, which is more relevant to practice, has received less attention and is still lacking. In particular, the different design paradigms assume different (sometimes informal) models which do not necessarily generalize to other design approaches. This resulted in various models, the majority of which lack rigour. As a consequence, it can be difficult to compare the merits of the different constructions in terms of their security guarantees. Moreover, existing models place a large amount of trust in the different authorities and assume that their keys are generated honestly. This does not necessarily reflect scenarios arising in real applications. Furthermore, some existing models, as we show, fail to take into account some attacks which might be problematic for some applications of the primitive.

"He Who Controls the Present Controls the Past", (George Orwell). Consider a scenario where the new leadership of an organisation or country wants to justify an unpopular policy (e.g. layoffs or removal of personal freedoms). A way to do that would be to back-date documents

justifying the policy: thus, any animosity for the policy would be towards the old leadership. The new leadership is only maintaining the status quo.

Re-framing this in technical terms, we show that the notion of traceability in existing models following the revocation list approach, where the group manager periodically publishes information (i.e. revocation lists) about members excluded from the group, is too weak. In those models, the life of the scheme spans over different intervals (epochs) at the start of which the manager updates the revocation lists. Signatures in those models are bound to a specific epoch. It is vital for functionality that old valid signatures (i.e. those produced at earlier epochs by then-legitimate members) are accepted by the verification algorithm.

The issue we identify in those models is that they allow members who joined at recent epochs to sign messages w.r.t earlier epochs during which they were not members of the group. In a sense this may be considered as an attack against traceability, as those members were not in the group at that interval. Technically however, the scenario we describe is allowed by the model: the underlying issue is a gap between one's interpretation of group signatures and what the definition implies. Our expectation is that a signature bound to epoch τ was produced by a member of the group *at that time*. Current definitions however, allows for all past, current, and future members, as long as they were not revoked at time τ.

One may dismiss this attack as theoretical, since the old leadership might appeal to the opener. However, this might not always be possible: the opener may be controlled by the new leadership, or in a business setting an outgoing CEO or board member might be disinterested or disincentivized from pursuing the issue. Another possible criticism might be that the weakness is trivial, and would be silently fixed in any construction using the model.

We show that some state of the art constructions, as [NFHF09, LPY12b, LPY12a], are susceptible to this attack. Specifically, their membership certificates are not bound to the epochs of their issuance. As a result, a member can sign w.r.t. earlier epochs. We stress that neither the authors of those schemes claimed their schemes were immune against such an issue nor that their models were supposed to capture such an attack. Thus, such an issue might not be a problem for the applications they originally had in mind, but only in a more general case.

In order to have strong security guarantees from the different constructions, a rigorous and unified security model is necessary. This is the aim of this work as we believe this is a challenging problem that needs to be addressed, especially given the relevance of the primitive.

Our Contribution. We take a close look at the security definitions of fully dynamic group signatures. We provide a rigorous security model that generalizes to the different design paradigms. In particular, our model covers both accumulator based and revocation list based approaches. Our model offers stringent security definitions and takes into account some attacks which were not considered by existing models. We give different flavors of our security definitions which capture both cases when the authorities' keys are adversarially generated and when such keys are honestly generated. We also show that our security definitions imply existing definitions for static and partially dynamic group signatures.

In the process, we identify a subtle difference between accumulator based and revocation list based approaches. Specifically, we identify a simple attack against traceability inherent to constructions following the latter approach and which is not captured by existing models. The attack allows a group member of to sign w.r.t. intervals prior to her joining the group. The security notion modelled by current definitions prevents users from signing only if they are explicitly revoked.

To address this, our traceability definition models a stricter security notion: users are not authorised to sign unless they are non-revoked and are active (i.e. part of the group) at the time interval associated with the signature. We note this is already implied in the accumulator based approach: the signer proves membership in the current version of the group at the time of signing. We also propose a number of possible fixes to this issue in some existing schemes.

Finally, we show that a fully dynamic group signature scheme obtained from the generic construction of accountable ring signatures given in [BCC+15] is secure w.r.t. the stronger variant of our security definitions.

Paper Organization. We present our model for fully dynamic group signatures in Section 2 and show that it implies existing definitions for static and partially dynamic group signatures. In Section 3 we analyse the security of three existing fully dynamic group signature schemes in our model.

Notation. A function $\nu(\cdot) : \mathbb{N} \to \mathbb{R}^+$ is negligible in the security parameter λ if for every polynomial $p(\cdot)$ and all sufficiently large values of λ, it holds that $\nu(\lambda) < \frac{1}{p(\lambda)}$. Given a probability distribution Y, we denote by $x \leftarrow Y$ the operation of selecting an element according to Y. If M is a probabilistic machine, we denote by $M(x_1, \ldots, x_n)$ the output distribution of M on inputs $(x_1, \ldots, x_n)$. By $[n]$ we denote the set $\{1, \ldots, n\}$. By PPT we mean running in probabilistic polynomial time in the relevant security parameter. For algorithms X and Y, $(x, y) \leftarrow \langle \mathsf{X}(a), \mathsf{Y}(b) \rangle$ denotes the

the joint execution of X (with input a) and Y (with input b) where at the end X outputs x, whereas Y outputs y. By $\mathsf{X}^{\langle\cdot,\mathsf{Y}(b)\rangle}(a)$, we denote the invocation of Y (with input b) by X (with input a). Note that X does not get the private output of Y.

2 Syntax and Security of Fully Dynamic Group Signatures

The parties involved in a Fully Dynamic Group Signature (FDGS) are: a group manager $\mathcal{GM}$ who authorizes who can join the group; a tracing manager $\mathcal{TM}$ who can revoke anonymity by opening signatures; a set of users, each with a unique identity $\mathsf{uid} \in \mathbb{N}$, who are potential group members. Users can join/leave the group at any time at the discretion of the group manager. We assume the group manager will regularly publish some information info_τ, associated with a distinct index τ (hereafter referred to as epoch). We assume that τ can be recovered given info_τ and vice versa (i.e. there is bijection between the epochs and associated information). The information depicts changes to the group, for instance, it could include the current members of the group (as in accumulator-based constructions) or those who have been excluded from the group (as, e.g. required by constructions based on revocation lists). As in existing models, we assume that anyone can verify the well-formedness and authenticity of the published group information. By combining the group information for the current epoch with that of the preceding one, any party can identify the list of members who have been revoked at the current epoch. We assume that the epochs preserve the order in which their corresponding information was published. More precisely, for all $\tau_1, \tau_2 \in \mathcal{T}$ ($\mathcal{T}$ being the space of epochs) we require that $\tau_1 < \tau_2$ if info_{τ_1} preceded info_{τ_2}.

Unlike existing models, which assume honestly generated authorities' keys, we separate the generation of the authorities' keys from that of the public parameters, which might need to be generated by a trusted party. This allows us (where appropriate) to define stringent security that protects against adversarial authorities who might generate their keys maliciously. Our definitions can be adapted straight away to work for the weaker setting where authorities' keys are generated honestly as in existing models. For the sake of generality, we define the group key generation as a joint protocol between the group and tracing managers. Clearly, it is desirable in some cases to avoid such interaction and allow authorities to

generate their own keys independently. This is a special case of our general definition where the protocol is regarded as two one-sided protocols.

An $\mathcal{FDGS}$ scheme consists of the following polynomial-time algorithms:

- $\mathsf{GSetup}(1^\lambda) \to \mathsf{param}$: is run by a trusted third party. On input a security parameter λ, it outputs public parameters param. The algorithm also initializes the registration table $\mathbf{reg}$.
- $\langle \mathsf{GKGen}_{\mathcal{GM}}(\mathsf{param}), \mathsf{GKGen}_{\mathcal{TM}}(\mathsf{param}) \rangle$: is an interactive protocol between algorithms $\mathsf{GKGen}_{\mathcal{GM}}$ and $\mathsf{GKGen}_{\mathcal{TM}}$ run by $\mathcal{GM}$ and $\mathcal{TM}$, respectively, to generate their respective private keys as well as the rest of the group public key gpk. The input to both algorithms is the public parameters param. If completed successfully, the private output of $\mathsf{GKGen}_{\mathcal{GM}}$ is a secret manager key msk, whereas its public output is a public key mpk, and the initial group information info. The private output of $\mathsf{GKGen}_{\mathcal{TM}}$ is the secret tracing key tsk, whereas its public output is a public key tpk. The group public key is then set to $\mathsf{gpk} := (\mathsf{param}, \mathsf{mpk}, \mathsf{tpk})$.
- $\mathsf{UKGen}(1^\lambda) \to (\mathbf{usk}[\mathsf{uid}], \mathbf{upk}[\mathsf{uid}])$: outputs a secret/public key pair ($\mathbf{usk}[\mathsf{uid}]$, $\mathbf{upk}[\mathsf{uid}]$) for user uid. We assume the public key table $\mathbf{upk}$ to be publicly available (possibly via PKI) so that anyone can get authentic copies of it.
- $\langle \mathsf{Join}(\mathsf{info}_{\tau_{\text{current}}}, \mathsf{gpk}, \mathsf{uid}, \mathbf{usk}[\mathsf{uid}]), \mathsf{Issue}(\mathsf{info}_{\tau_{\text{current}}}, \mathsf{msk}, \mathsf{uid}, \mathbf{upk}[\mathsf{uid}]) \rangle$: is an interactive protocol between a user uid (who has already obtained a personal key pair, i.e. ran the UKGen algorithm) and the group manager $\mathcal{GM}$. Upon successful completion, uid becomes a member of the group. The final state of the Issue algorithm is stored in the registration table at index uid (i.e. $\mathbf{reg}[\mathsf{uid}]$), whereas that of the Join algorithm is stored in $\mathbf{gsk}[\mathsf{uid}]$. The epoch τ_{current} is part of the output of both parties.

  We assume that the protocol takes place over a secure (i.e. private and authentic) channel. The protocol is initiated by calling Join. The manager may update the group information after running this protocol. The registration table $\mathbf{reg}$ stores additional information used by the group manager and the tracing manager for updating and tracing, depending on the scheme specifics.
- $\mathsf{UpdateGroup}(\mathsf{gpk}, \mathsf{msk}, \mathsf{info}_{\tau_{\text{current}}}, \mathcal{S}, \mathbf{reg}) \to \mathsf{info}_{\tau_{\text{new}}}$: is run by the group manager to update the group information while also advancing the epoch. It takes as input the group manager's secret key msk, a (possibly empty) set $\mathcal{S}$ of active members to be removed from the group and the registration table $\mathbf{reg}$, it outputs a new group information

$\mathsf{info}_{\tau_{\text{new}}}$ and might also update the registration table **reg**. If there has been no changes to the group information, the algorithm returns $\perp$ to indicate that no new information has been issued. The algorithm aborts if any $\mathsf{uid} \in \mathcal{S}$ has not run the join protocol.

- $\mathsf{Sign}(\mathsf{gpk}, \mathbf{gsk}[\mathsf{uid}], \mathsf{info}_\tau, m) \rightarrow \Sigma$: on input the group public key gpk, a user's group signing key $\mathbf{gsk}[\mathsf{uid}]$, the group information info_τ at epoch τ, and a message m, outputs a group signature Σ on m by the group member uid. If the user owning $\mathbf{gsk}[\mathsf{uid}]$ is not an active member of the group at epoch τ, the algorithm returns $\perp$.
- $\mathsf{Verify}(\mathsf{gpk}, \mathsf{info}_\tau, m, \Sigma) \rightarrow 1/0$: is a deterministic algorithm checking whether Σ is a valid group signature on m at epoch τ and outputs a bit accordingly.
- $\mathsf{Trace}(\mathsf{gpk}, \mathsf{tsk}, \mathsf{info}_\tau, \mathbf{reg}, m, \Sigma) \rightarrow (\mathsf{uid}, \pi_{\text{Trace}})$: is a deterministic algorithm which is run by the tracing manager. It returns an identity $\mathsf{uid} > 0$ of the group member who produced Σ plus a proof π_{Trace} attesting to this fact. If the algorithm is unable to trace the signature to a particular group member, it returns $(0, \pi_{\text{Trace}})$ to indicate that it could not attribute the signature.
- $\mathsf{Judge}(\mathsf{gpk}, \mathsf{uid}, \mathsf{info}_\tau, \pi_{\text{Trace}}, \mathbf{upk}[\mathsf{uid}], m, \Sigma) \rightarrow 1/0$: is a deterministic algorithm which on input the group public key gpk, a user identity uid, the group information at epoch τ, a tracing proof π_{Trace}, the user's public key $\mathbf{upk}[\mathsf{uid}]$ (which is $\perp$ if it does not exist), a message m, and a signature Σ, outputs 1 if π_{Trace} is a valid proof that uid produced Σ, and outputs 0 otherwise.

Additional Algorithm. We will also use the following polynomial-time algorithm which is only used in the security games to ease composition.

$\mathsf{IsActive}(\mathsf{info}_\tau, \mathbf{reg}, \mathsf{uid}) \rightarrow 1/0$: returns 1 if the user uid is an active member of the group at epoch τ and 0 otherwise.

2.1 Security of Fully Dynamic Group Signatures

The security requirements of a fully dynamic group signature are: *correctness*, *anonymity*, *non-frameability*, *traceability* and *tracing soundness*. To define those requirements, we use a set of games in which the adversary has access to a set of oracles. The following global lists are maintained: HUL is a list of honest users; CUL is a list of corrupt users whose personal secret keys have been chosen by the adversary; BUL is a list of bad users whose personal and group signing keys have been revealed to the adversary; SL is

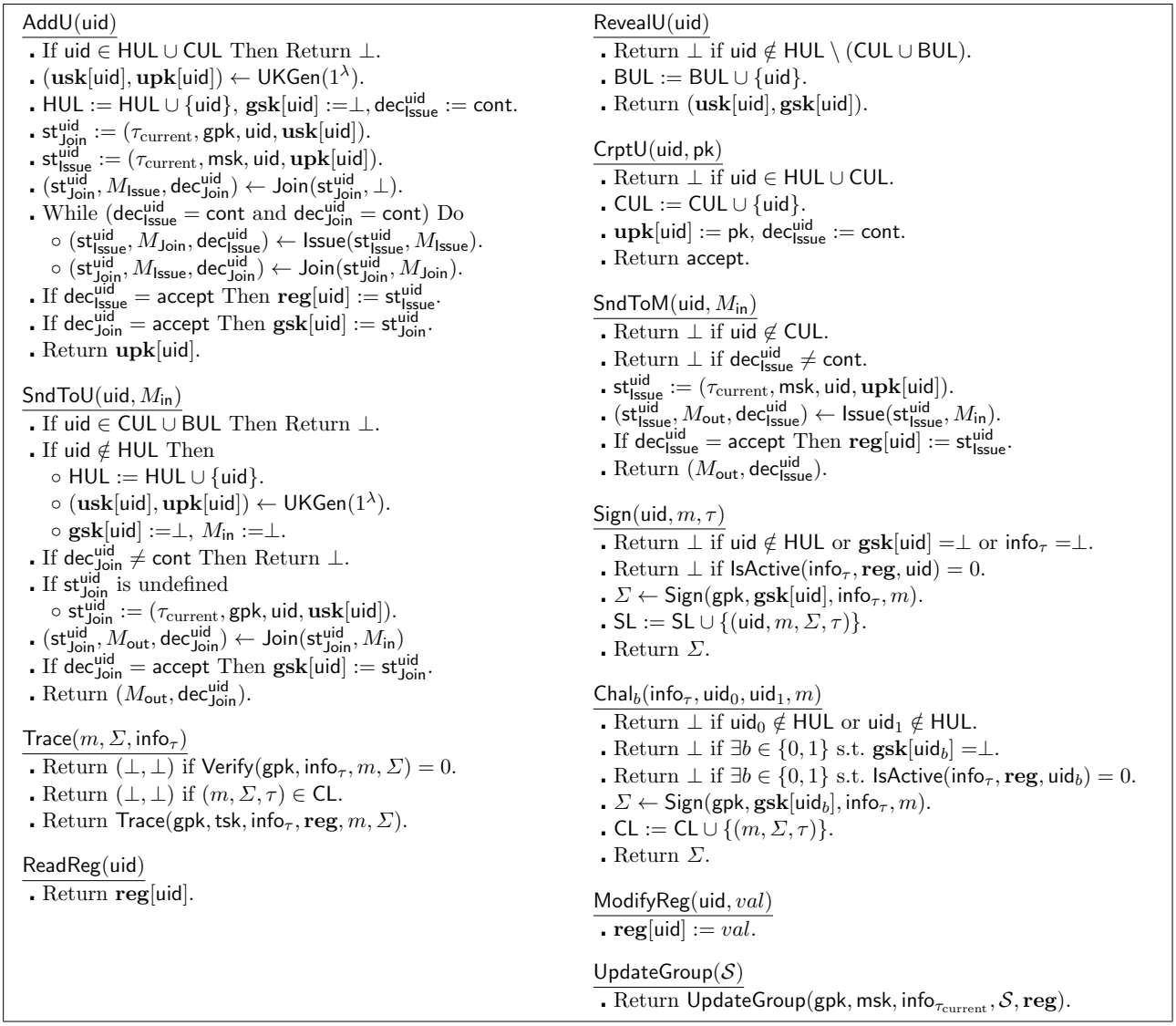

Fig. 1. Details of the oracles used in the security games

a list of signatures obtained from the Sign oracle; CL is a list of challenge signatures obtained from the challenge oracle.

The details of the following oracles are given in Fig. 1.

AddU(uid) adds an honest user uid to the group at the current epoch.

$\mathsf{CrptU}(\mathsf{uid}, \mathsf{pk})$ creates a new corrupt user whose public key $\mathbf{upk}[\mathsf{uid}]$ is chosen by the adversary. This is called in preparation for calling the SndToM oracle.

$\mathsf{SndToM}(\mathsf{uid}, M_{\mathsf{in}})$ used to engage in the $\mathsf{Join\text{-}Issue}$ protocol with the honest, Issue-executing group manager.

$\mathsf{SndToU}(\mathsf{uid}, M_{\mathsf{in}})$ used to engage in the $\mathsf{Join\text{-}Issue}$ protocol with an honest, Join-executing user uid on behalf of the corrupt group manager.

$\mathsf{ReadReg}(\mathsf{uid})$ returns the registration information $\mathbf{reg}[\mathsf{uid}]$ of user uid.

$\mathsf{ModifyReg}(\mathsf{uid}, val)$ modifies the entry $\mathbf{reg}[\mathsf{uid}]$, setting $\mathbf{reg}[\mathsf{uid}] := val$. For brevity we will assume $\mathsf{ModifyReg}$ also provides the functionality of $\mathsf{ReadReg}$.

$\mathsf{RevealU}(\mathsf{uid})$ returns the personal secret key $\mathbf{usk}[\mathsf{uid}]$ and group signing key $\mathbf{gsk}[\mathsf{uid}]$ of group member uid.

$\mathsf{Sign}(\mathsf{uid}, m, \tau)$ returns a signature on the message m by the group member uid for epoch τ assuming the corresponding group information info_τ is defined.

$\mathsf{Chal}_b(\mathsf{info}_\tau, \mathsf{uid}_0, \mathsf{uid}_1, m)$ is a left-right oracle for defining anonymity. The adversary chooses an epoch τ, the group information info_τ, two identities $(\mathsf{uid}_0, \mathsf{uid}_1)$, and a message m and receives a group signature by member uid_b for $b \leftarrow \{0, 1\}$ for the chosen epoch. It is required that both challenge users are active members at epoch τ. The adversary can only call this oracle once.

$\mathsf{Trace}(m, \Sigma, \mathsf{info}_\tau)$ returns the identity of the signer of the signature Σ on m w.r.t. info_τ if the signature was not obtained from the Chal_b oracle.

$\mathsf{UpdateGroup}(\mathcal{S})$ allows the adversary to update the group. $\mathcal{S}$ here is the set of the active members to be removed from the group.

The following security requirements are defined by the games in Fig. 2.

Correctness. This requirement guarantees that signatures produced by honest, non-revoked users are accepted by the Verify algorithm and that the honest tracing manager can identify the signer of such signatures. In addition, the Judge algorithm accepts the tracing manager's decision.

Formally, an $\mathcal{FDGS}$ scheme is *(perfectly) correct* if for all $\lambda \in \mathbb{N}$, the advantage

$$\mathsf{Adv}^{\mathrm{Corr}}_{\mathcal{FDGS},\mathcal{A}}(\lambda) := \Pr[\mathbf{Exp}^{\mathrm{Corr}}_{\mathcal{FDGS},\mathcal{A}}(\lambda) = 1]$$

is negligible (in λ) for all adversaries $\mathcal{A}$.

Note that the above definition of (perfect) correctness protects against even unbounded adversaries. If computational correctness suffices, i.e. when we consider correctness only against computationally-bounded adversaries, we can drop the last three lines from the correctness game in Fig. 2.

Experiment: $\mathbf{Exp}^{\mathrm{Corr}}_{\mathcal{FDGS},\mathcal{A}}(\lambda)$
- $\mathsf{param} \leftarrow \mathsf{GSetup}(1^\lambda)$; $\mathsf{HUL} := \emptyset$.
- $\Big((\mathsf{msk},\mathsf{mpk},\mathsf{info}),(\mathsf{tsk},\mathsf{tpk})\Big) \leftarrow \langle \mathsf{GKGen}_{\mathcal{GM}}(\mathsf{param}), \mathsf{GKGen}_{\mathcal{TM}}(\mathsf{param})\rangle$.
- $\mathsf{gpk} := (\mathsf{param},\mathsf{mpk},\mathsf{tpk})$.
- $\Big(\mathsf{uid}, m, \tau\Big) \leftarrow \mathcal{A}^{\mathsf{AddU},\mathsf{ReadReg},\mathsf{UpdateGroup}}\Big(\mathsf{gpk},\mathsf{info}\Big)$.
- If $\mathsf{uid} \notin \mathsf{HUL}$ or $\mathbf{gsk}[\mathsf{uid}] = \perp$ or $\mathsf{info}_\tau = \perp$ or $\mathsf{IsActive}(\mathsf{info}_\tau, \mathbf{reg}, \mathsf{uid}) = 0$ Then Return 0.
- $\Sigma \leftarrow \mathsf{Sign}(\mathsf{gpk}, \mathbf{gsk}[\mathsf{uid}], \mathsf{info}_\tau, m)$.
- If $\mathsf{Verify}(\mathsf{gpk}, \mathsf{info}_\tau, m, \Sigma) = 0$ Then Return 1.
- $(\mathsf{uid}^*, \pi_{\mathrm{Trace}}) \leftarrow \mathsf{Trace}(\mathsf{gpk}, \mathsf{tsk}, \mathsf{info}_\tau, \mathbf{reg}, m, \Sigma)$.
- If $\mathsf{uid} \neq \mathsf{uid}^*$ Then Return 1.
- If $\mathsf{Judge}(\mathsf{gpk}, \mathsf{uid}, \mathsf{info}_\tau, \pi_{\mathrm{Trace}}, \mathbf{upk}[\mathsf{uid}], m, \Sigma) = 0$ Then Return 1, Else Return 0.

Experiment: $\mathbf{Exp}^{\text{Anon-}b}_{\mathcal{FDGS},\mathcal{A}}(\lambda)$
- $\mathsf{param} \leftarrow \mathsf{GSetup}(1^\lambda)$; $\mathsf{HUL},\mathsf{CUL},\mathsf{BUL},\mathsf{SL},\mathsf{CL} := \emptyset$.
- $\Big(\mathsf{st}_{\mathsf{init}},\mathsf{msk},\mathsf{mpk},\mathsf{info}\Big) \leftarrow \mathcal{A}^{\langle\cdot,\mathsf{GKGen}_{\mathcal{TM}}(\mathsf{param})\rangle}(\mathsf{init}:\mathsf{param})$.
- Return 0 if $\mathsf{GKGen}_{\mathcal{TM}}$ did not accept or $\mathcal{A}$'s output is not well-formed.
- Parse the output of $\mathsf{GKGen}_{\mathcal{TM}}$ as $(\mathsf{tsk},\mathsf{tpk})$ and set $\mathsf{gpk} := (\mathsf{param},\mathsf{mpk},\mathsf{tpk})$.
- $b^* \leftarrow \mathcal{A}^{\mathsf{AddU},\mathsf{CrptU},\mathsf{SndToU},\mathsf{RevealU},\mathsf{Trace},\mathsf{ModifyReg},\mathsf{Chal}_b}\Big(\mathsf{play}:\mathsf{st}_{\mathsf{init}},\mathsf{gpk}\Big)$.
- Return b^*.

Experiment: $\mathbf{Exp}^{\text{Non-Frame}}_{\mathcal{FDGS},\mathcal{A}}(\lambda)$
- $\mathsf{param} \leftarrow \mathsf{GSetup}(1^\lambda)$; $\mathsf{HUL},\mathsf{CUL},\mathsf{BUL},\mathsf{SL} := \emptyset$.
- $(\mathsf{st}_{\mathsf{init}},\mathsf{info},\mathsf{msk},\mathsf{mpk},\mathsf{tsk},\mathsf{tpk}) \leftarrow \mathcal{A}(\mathsf{init}:\mathsf{param})$.
- Return 0 if $\mathcal{A}$'s output is not well-formed otherwise set $\mathsf{gpk} := (\mathsf{param},\mathsf{mpk},\mathsf{tpk})$.
- $\Big(m,\Sigma,\mathsf{uid},\pi_{\mathrm{Trace}},\mathsf{info}_\tau\Big) \leftarrow \mathcal{A}^{\mathsf{CrptU},\mathsf{SndToU},\mathsf{RevealU},\mathsf{Sign},\mathsf{ModifyReg}}\Big(\mathsf{play}:\mathsf{st}_{\mathsf{init}},\mathsf{gpk}\Big)$.
- If $\mathsf{Verify}(\mathsf{gpk},\mathsf{info}_\tau,m,\Sigma) = 0$ Then Return 0.
- If $\mathsf{Judge}(\mathsf{gpk},\mathsf{uid},\mathsf{info}_\tau,\pi_{\mathrm{Trace}},\mathbf{upk}[\mathsf{uid}],m,\Sigma) = 0$ Then Return 0.
- If $\mathsf{uid} \notin \mathsf{HUL} \setminus \mathsf{BUL}$ or $(\mathsf{uid},m,\Sigma,\tau) \in \mathsf{SL}$ Then Return 0 Else Return 1.

Experiment: $\mathbf{Exp}^{\mathrm{Trace}}_{\mathcal{FDGS},\mathcal{A}}(\lambda)$
- $\mathsf{param} \leftarrow \mathsf{GSetup}(1^\lambda)$; $\mathsf{HUL},\mathsf{CUL},\mathsf{BUL},\mathsf{SL} := \emptyset$.
- $\Big(\mathsf{st}_{\mathsf{init}},\mathsf{tsk},\mathsf{tpk}\Big) \leftarrow \mathcal{A}^{\langle\mathsf{GKGen}_{\mathcal{GM}}(\mathsf{param}),\cdot\rangle}(\mathsf{init}:\mathsf{param})$.
- Return 0 if $\mathsf{GKGen}_{\mathcal{GM}}$ did not accept or $\mathcal{A}$'s output is not well-formed.
- Parse the output of $\mathsf{GKGen}_{\mathcal{GM}}$ as $(\mathsf{msk},\mathsf{mpk},\mathsf{info})$. Set $\mathsf{gpk} := (\mathsf{param},\mathsf{mpk},\mathsf{tpk})$.
- $\Big(m,\Sigma,\tau\Big) \leftarrow \mathcal{A}^{\mathsf{AddU},\mathsf{CrptU},\mathsf{SndToM},\mathsf{RevealU},\mathsf{Sign},\mathsf{ModifyReg},\mathsf{UpdateGroup}}\Big(\mathsf{play}:\mathsf{st}_{\mathsf{init}},\mathsf{gpk},\mathsf{info}\Big)$.
- If $\mathsf{Verify}(\mathsf{gpk},\mathsf{info}_\tau,m,\Sigma) = 0$ Then Return 0.
- $(\mathsf{uid},\pi_{\mathrm{Trace}}) \leftarrow \mathsf{Trace}(\mathsf{gpk},\mathsf{tsk},\mathsf{info}_\tau,\mathbf{reg},m,\Sigma)$.
- If $\mathsf{IsActive}(\mathsf{info}_\tau,\mathbf{reg},\mathsf{uid}) = 0$ Then Return 1.
- If $\mathsf{uid} = 0$ or $\mathsf{Judge}(\mathsf{gpk},\mathsf{uid},\mathsf{info}_\tau,\pi_{\mathrm{Trace}},\mathbf{upk}[\mathsf{uid}],m,\Sigma) = 0$ Then Return 1 Else Return 0.

Experiment: $\mathbf{Exp}^{\text{Trace-Sound}}_{\mathcal{FDGS},\mathcal{A}}(\lambda)$
- $\mathsf{param} \leftarrow \mathsf{GSetup}(1^\lambda)$; $\mathsf{CUL} := \emptyset$.
- $(\mathsf{st}_{\mathsf{init}},\mathsf{info},\mathsf{msk},\mathsf{mpk},\mathsf{tsk},\mathsf{tpk}) \leftarrow \mathcal{A}(\mathsf{init}:\mathsf{param})$.
- Return 0 if $\mathcal{A}$'s output is not well-formed otherwise set $\mathsf{gpk} := (\mathsf{param},\mathsf{mpk},\mathsf{tpk})$.
- $\Big(m,\Sigma,\{\mathsf{uid}_i,\pi_{\mathrm{Trace}_i}\}_{i=1}^2,\mathsf{info}_\tau\Big) \leftarrow \mathcal{A}^{\mathsf{CrptU},\mathsf{ModifyReg}}\Big(\mathsf{play}:\mathsf{st}_{\mathsf{init}},\mathsf{gpk}\Big)$.
- If $\exists i \in \{1,2\}$ s.t. $\mathsf{Verify}(\mathsf{gpk},\mathsf{info}_\tau,m,\Sigma) = 0$ Then Return 0.
- If $\mathsf{uid}_1 = \mathsf{uid}_2$ or $\mathsf{uid}_1 = \perp$ or $\mathsf{uid}_2 = \perp$ Then Return 0.
- If $\exists i \in \{1,2\}$ s.t. $\mathsf{Judge}(\mathsf{gpk},\mathsf{uid}_i,\mathsf{info}_\tau,\pi_{\mathrm{Trace}_i},\mathbf{upk}[\mathsf{uid}_i],m,\Sigma) = 0$ Then Return 0.
- Return 1.

Fig. 2. Security games for fully dynamic group signatures

Computational correctness of the Trace and Judge algorithms is implied by the other requirements.

(Full) Anonymity. This requires that signatures do not reveal the identity of the group member who produced them. In the game, the adversary, $\mathcal{A}$, can corrupt any user and fully corrupt the group manager by choosing her key. We require that both challenge users are active members of the group at the chosen epoch. Also, note that a Trace query on the challenge signature will fail.

As $\mathcal{A}$ can learn the personal secret and group signing keys of any user, including the challenge users, our definition captures full key exposure attacks.

The adversary chooses an epoch, the group information for that epoch, a message and two group members and gets a signature by either member and wins if she correctly guesses the member. Without loss in generality, we allow the adversary a single call to the challenge oracle. A hybrid argument (similar to that used in [BSZ05]) can be used to prove that this is sufficient.

Formally, an $\mathcal{FDGS}$ scheme is *(fully) anonymous* if for all $\lambda \in \mathbb{N}$, the advantage $\mathsf{Adv}^{\text{Anon}}_{\mathcal{FDGS},\mathcal{A}}(\lambda)$ is negligible (in λ) for all PPT adversaries $\mathcal{A}$, where

$$\mathsf{Adv}^{\text{Anon}}_{\mathcal{FDGS},\mathcal{A}}(\lambda) := \left|\Pr[\mathbf{Exp}^{\text{Anon-0}}_{\mathcal{FDGS},\mathcal{A}}(\lambda) = 1] - \Pr[\mathbf{Exp}^{\text{Anon-1}}_{\mathcal{FDGS},\mathcal{A}}(\lambda) = 1]\right|.$$

Non-Frameability. This ensures that even if the rest of the group as well as the tracing and group managers are fully corrupt, they cannot produce a signature that can be attributed to an honest member who did not produce it.

In the game, the adversary can fully corrupt both the group and tracing managers. She even chooses the keys of both managers. Thus, our definition is stronger than existing models. We just require that the framed member is honest.

Formally, an $\mathcal{FDGS}$ scheme is *non-frameable* if for all $\lambda \in \mathbb{N}$, the advantage

$$\mathsf{Adv}^{\text{Non-Frame}}_{\mathcal{FDGS},\mathcal{A}}(\lambda) := \Pr[\mathbf{Exp}^{\text{Non-Frame}}_{\mathcal{FDGS},\mathcal{A}}(\lambda) = 1]$$

is negligible (in λ) for all PPT adversaries $\mathcal{A}$.

Remark 1. In the game variant we give in Fig. 2, we allow the adversary to generate the tracing manager's key herself. While, as we show later, there are schemes which satisfy this strong variant of the definition, such

definition might be too strong to be satisfied by some existing schemes. A weaker variant of the definition is where the tracing key is generated by the challenger rather than the adversary. This requires replacing lines 2-4 in the game in Fig. 2 by the following:

- $(\mathsf{st}_{\mathsf{init}}, \mathsf{info}, \mathsf{msk}, \mathsf{mpk}) \leftarrow \mathcal{A}^{\langle \cdot, \mathsf{GKGen}_{\mathcal{TM}}(\mathsf{param})\rangle}(\mathsf{init} : \mathsf{param})$.
- Return 0 if $\mathcal{A}$'s output is not well-formed or $\mathsf{GKGen}_{\mathcal{TM}}$ did not accept.
- Let $(\mathsf{tsk}, \mathsf{tpk})$ be the output of $\mathsf{GKGen}_{\mathcal{TM}}$. Set $\mathsf{gpk} := (\mathsf{param}, \mathsf{mpk}, \mathsf{tpk})$.
- $\Big(m, \Sigma, \mathsf{uid}, \pi_{\mathrm{Trace}}, \mathsf{info}_\tau\Big) \leftarrow \mathcal{A}^{\mathsf{CrptU},\mathsf{SndToU},\mathsf{RevealU},\mathsf{Sign},\mathsf{ModifyReg}}\Big(\mathsf{play} : \mathsf{st}_{\mathsf{init}}, \mathsf{gpk}, \mathsf{tsk}\Big)$.

Traceability. This ensures that the adversary cannot produce a signature that cannot be traced to an active member of the group at the chosen epoch. In the game, the adversary can corrupt any user and even chooses the tracing key of the tracing manager. The adversary is not given the group manager's secret key as this would allow her to create dummy users which are thus untraceable. Note that unlike [LPY12b, LPY12a, NFHF09], our definition captures that a member of the group should not be able to sign w.r.t. epochs prior to her joining the group since we do not restrict the adversary's forgery to be w.r.t. to the current epoch (i.e. the current version of the group information). The adversary wins if she produces a signature whose signer cannot be identified or is an inactive member at the chosen epoch. The adversary also wins if the Judge algorithm does not accept the tracing decision on the forgery.

Formally, an $\mathcal{FDGS}$ scheme is *traceable* if for all $\lambda \in \mathbb{N}$, the advantage

$$\mathsf{Adv}^{\mathrm{Trace}}_{\mathcal{FDGS},\mathcal{A}}(\lambda) := \Pr[\mathbf{Exp}^{\mathrm{Trace}}_{\mathcal{FDGS},\mathcal{A}}(\lambda) = 1]$$

is negligible (in λ) for all PPT adversaries $\mathcal{A}$.

Remark 2. To get an honestly-generated tracing key variant of the game in Fig. 2, we replace lines 2-5 in the game in Fig. 2 by the following lines:

- $\Big((\mathsf{msk}, \mathsf{mpk}, \mathsf{info}), (\mathsf{tsk}, \mathsf{tpk})\Big) \leftarrow \langle \mathsf{GKGen}_{\mathcal{GM}}(\mathsf{param}), \mathsf{GKGen}_{\mathcal{TM}}(\mathsf{param})\rangle$.
- Set $\mathsf{gpk} := (\mathsf{param}, \mathsf{mpk}, \mathsf{tpk})$.
- $\Big(m, \Sigma, \tau\Big) \leftarrow \mathcal{A}^{\mathsf{AddU},\mathsf{CrptU},\mathsf{SndToM},\mathsf{RevealU},\mathsf{Sign},\mathsf{ModifyReg},\mathsf{UpdateGroup}}\Big(\mathsf{play} : \mathsf{st}_{\mathsf{init}}, \mathsf{gpk}, \mathsf{info}, \mathsf{tsk}\Big)$.

Tracing Soundness. As recently defined by [SSE+12] in the context of partially dynamic group signatures, this requirement ensures that even if both the group and the tracing managers as well as all members of the group collude, they cannot produce a valid signature that traces to two different members. Such a requirement is vital for many applications. For

example, applications where signers get rewarded or where we need to stop abusers shifting blame to others.

In the definition, the adversary can fully corrupt all parties involved and wins if she produces a valid signature and valid tracing proofs that the signature traces to different (possibly corrupt) users. We may also consider a stronger variant where the adversary wins by producing a signature that traces to different epochs.

Formally, an $\mathcal{FDGS}$ scheme has *tracing soundness* if for all $\lambda \in \mathbb{N}$,

$$\mathsf{Adv}^{\text{Trace-Sound}}_{\mathcal{FDGS},\mathcal{A}}(\lambda) := \Pr[\mathbf{Exp}^{\text{Trace-Sound}}_{\mathcal{FDGS},\mathcal{A}}(\lambda) = 1]$$

is negligible (in λ) for all PPT adversaries $\mathcal{A}$.

Remark 3. To get an honestly-generated tracing key variant of the game in Fig. 2, we replace lines 2-4 in the game in Fig. 2 by the following lines:

– $\Big(\mathsf{st}_{\mathsf{init}}, \mathsf{msk}, \mathsf{mpk}, \mathsf{info}\Big) \leftarrow \mathcal{A}^{\langle\cdot,\mathsf{GKGen}_{\mathcal{TM}}(\mathsf{param})\rangle}(\mathsf{init} : \mathsf{param})$.
– Return 0 if $\mathsf{GKGen}_{\mathcal{TM}}$ did not accept or $\mathcal{A}$'s output is not well-formed.
– Parse the output of $\mathsf{GKGen}_{\mathcal{TM}}$ as $(\mathsf{tsk}, \mathsf{tpk})$ and set $\mathsf{gpk} := (\mathsf{param}, \mathsf{mpk}, \mathsf{tpk})$.
– $\Big(m, \Sigma, \{\mathsf{uid}_i, \pi_{\mathrm{Trace}_i}\}_{i=1}^{2}, \mathsf{info}_\tau\Big) \leftarrow \mathcal{A}^{\mathsf{CrptU},\mathsf{ModifyReg}}\Big(\mathsf{play} : \mathsf{st}_{\mathsf{init}}, \mathsf{gpk}, \mathsf{tsk}\Big)$.

2.2 Comparison with Existing Models

Models used by accumulator-based constructions, e.g. [BS01, CL02, TX03, AST01, Ngu05, NFHF09], the vast majority of which are stated informally, are specific to that particular design paradigm and do not generalize to other construction approaches. Moreover, most of the them do not take into account some of the attacks that arise in a more formal setting. For instance, some models only protect against partially but not fully corrupt tracing managers and do not capture the tracing soundness requirement. On the other hand, models used by other design approaches, e.g. [NFHF09, LPY12b, LPY12a] are also specific to those approaches and have their own shortcomings. For instance, as discussed earlier, the models used by the state-of-the-art constructions by Libert et al. [LPY12b, LPY12a] and Nakanishi et al. [NFHF09] do not prevent a group member from being able to sign w.r.t. time intervals before she joined the group. This is an attack that can be problematic in some applications of the primitive. In the traceability game used in [NFHF09] as well as the misidentification game used in [LPY12b, LPY12a], the adversary is required to output a signature that is valid w.r.t. the current interval (epoch) and therefore

the definitions do not capture the attack we highlight. We stress that the authors of the concerned models never claimed that their models cover such an attack as it might not be a problem for their intended applications.

The traceability issue we shed light on does not apply to accumulator based models. In these settings, when the group changes, an update is published containing a list of the currently active group members and most constructions work by having the signer prove membership in such a list. Therefore, even if a malicious member tries to sign w.r.t. an earlier version of the group information, she still has to prove she is a member of the group at the concerned interval.

In addition [NFHF09, LPY12b, LPY12a] only consider a partially but not fully corrupt tracing manager in the non-frameability game. Moreover, they do not capture the requirement that a signature should only trace to one member (i.e. tracing soundness). The latter is vital for many applications of the primitive.

Another distinction from existing models is that our model allows maliciously generated authorities' keys when applicable. Therefore, it offers more stringent security than existing models which rely on such keys being generated honestly.

2.3 Recovering Other Models

We give security reductions which relate our model to other well-known models for group signatures. All these models assume honest key generation, for both group and tracing managers, which is a special case of our model. We consider three models. First, the model for static group signatures given in [BMW03]. We then consider two models for partially dynamic groups from [BSZ05] and [KY06].

Static Group Signatures [BMW03]. We note that we can recover static group signatures [BMW03] from our group signatures. We fix the group manager as the designated opener and include tsk in the group master secret key. In the setup, group members generate their key pairs and interact with the group manager to join the group. Their Open algorithm does not output proofs, as their model does not use a Judge algorithm, so we define a variant of our non-frameability game from Fig. 2 where we replace the last 4 lines in the game in Fig. 2 by the ones in Fig. 3.

This gives a sensible and compatible definition which allows us to recover the model from the fully dynamic scheme.

Static group signatures are just fully dynamic group signatures with no joining, issuing, or group updates. Correctness follows trivially from the

– $\Big(m, \Sigma, \mathsf{info}_\tau\Big) \leftarrow \mathcal{A}^{\mathsf{CrptU},\mathsf{SndToU},\mathsf{RevealU},\mathsf{Sign},\mathsf{ModifyReg}}\Big(\mathsf{play} : \mathsf{st}_{\mathsf{init}}, \mathsf{gpk}\Big)$.
– If $\mathsf{Verify}(\mathsf{gpk}, \mathsf{info}_\tau, m, \Sigma) = 0$ Then Return 0.
– $(\mathsf{uid}, \pi_{\mathrm{Trace}}) \leftarrow \mathsf{Trace}(\mathsf{gpk}, \mathsf{tsk}, \mathsf{info}_\tau, \mathbf{reg}, m, \sigma)$
– If $\mathsf{uid} \notin \mathsf{HUL} \setminus \mathsf{BUL}$ or $(\mathsf{uid}, m, \Sigma, \tau) \in \mathsf{SL}$ Then Return 0 Else Return 1.

Fig. 3. Modified non-frameability game.

correctness of the fully dynamic group signature scheme. [BMW03]-full-anonymity follows from (full) anonymity of the fully dynamic group signature scheme, while [BMW03]-full-traceability follows from our traceability and non-frameability requirements. We now give an explicit construction and security reductions which show how our model relates to [BMW03].

$\underline{\mathsf{GS}_{GKg}(1^\lambda, 1^n)} \rightarrow (\mathsf{GS}_{\mathsf{gpk}}, \mathsf{GS}_{\mathsf{gmsk}}, \mathsf{GS}_{\mathbf{gsk}})$
▪ $\mathsf{param} \leftarrow \mathsf{GSetup}(1^\lambda)$.
▪ $\Big((\mathsf{msk}, \mathsf{mpk}, \mathsf{info}), (\mathsf{tsk}, \mathsf{tpk})\Big) \leftarrow \langle \mathsf{GKGen}_{\mathcal{GM}}(\mathsf{param}), \mathsf{GKGen}_{\mathcal{TM}}(\mathsf{param})\rangle$.
▪ Set $\mathsf{gpk} := (\mathsf{param}, \mathsf{mpk}, \mathsf{tpk})$.
▪ For each user:
  ◦ Run $(\mathbf{usk}[\mathsf{uid}], \mathbf{upk}[\mathsf{uid}]) \leftarrow \mathsf{UKGen}(1^\lambda)$.
  ◦ Run $\langle \mathsf{Join}(\mathsf{info}, \mathsf{gpk}, \mathsf{uid}, \mathbf{usk}[\mathsf{uid}]), \mathsf{Issue}(\mathsf{info}, \mathsf{msk}, \mathsf{uid}, \mathbf{upk}[\mathsf{uid}])\rangle$.
▪ Set $\mathsf{GS}_{\mathsf{gmsk}} := (\mathsf{gpk}, \mathsf{tsk}, \mathsf{info}, \mathbf{reg})$, $\mathsf{GS}_{\mathsf{gpk}} := (\mathsf{gpk}, \mathsf{info})$
▪ Set $\mathsf{GS}_{\mathbf{gsk}[\mathsf{uid}]} := (\mathsf{gpk}, \mathbf{gsk}[\mathsf{uid}], \mathsf{info})$.

$\underline{\mathsf{GS}_{GSig}(\mathsf{GS}_{\mathbf{gsk}[\mathsf{uid}]}, m)} \rightarrow \sigma$
▪ Parse $\mathsf{GS}_{\mathsf{gpk}}$ as $(\mathsf{gpk}, \mathsf{info})$ and $\mathsf{GS}_{\mathbf{gsk}[\mathsf{uid}]}$ as $(\mathsf{gpk}, \mathbf{gsk}[\mathsf{uid}], \mathsf{info})$.
▪ Return $\mathsf{Sign}(\mathsf{gpk}, \mathbf{gsk}[\mathsf{uid}], \mathsf{info}, m)$.

$\underline{\mathsf{GS}_{GVf}(\mathsf{GS}_{\mathsf{gpk}}, m, \sigma)} \rightarrow 0/1$
▪ Parse $\mathsf{GS}_{\mathsf{gpk}}$ as $(\mathsf{gpk}, \mathsf{info})$.
▪ Return $\mathsf{Verify}(\mathsf{gpk}, \mathsf{info}, m, \sigma)$.

$\underline{\mathsf{GS}_{Open}(\mathsf{GS}_{\mathsf{gmsk}}, m, \sigma)} \rightarrow \{\mathsf{uid}\} \cup \{\perp\}$
▪ Parse $\mathsf{GS}_{\mathsf{gpk}}$ as $(\mathsf{gpk}, \mathsf{info})$ and $\mathsf{GS}_{\mathsf{gmsk}}$ as $(\mathsf{gpk}, \mathsf{tsk}, \mathsf{info}, \mathbf{reg})$.
▪ Run $(\mathsf{uid}, \pi_{\mathrm{Trace}}) \leftarrow \mathsf{Trace}(\mathsf{gpk}, \mathsf{tsk}, \mathsf{info}_\tau, \mathbf{reg}, m, \sigma)$.
▪ Return uid.

Fig. 4. Static group signatures [BMW03] from our group signatures.

Theorem 1. *The construction of Fig. 4 is a secure static group signature scheme in the sense of [BMW03].*

Proof. Correctness is straightforward to verify. Next we prove full-anonymity. Suppose that there exists an efficient adversary $\mathcal{B}$ who successfully breaks the [BMW03]-full-anonymity of the static group signature scheme with probability that is not negligible, with respect to a particular group of users. We construct an efficient adversary $\mathcal{A}$ for the (full) anonymity of the fully dynamic group signature scheme. While initializing the fully dynamic scheme, $\mathcal{A}$ behaves honestly, so that all parameters are honestly generated. Next, $\mathcal{A}$ adds users to the scheme using the AddU oracle to create the group of users for $\mathcal{B}$, and learns their signing keys using the $\mathsf{RevealU}$ oracle. Now, $\mathcal{A}$ is able to start $\mathcal{B}$ on input $(\mathsf{GS}_{\mathsf{gpk}}, \mathsf{GS}_{\mathbf{gsk}})$. $\mathcal{A}$ can answer $\mathcal{B}$'s Open queries using her Trace oracle. When $\mathcal{B}$ outputs his challenge (i_0, i_1, m), $\mathcal{A}$ calls $\mathsf{Chal}_b(\mathsf{info}, i_0, i_1, m)$ and gets a challenge signature σ^* which she forwards to $\mathcal{B}$ as the challenge signature. Again, using the Trace oracle, $\mathcal{A}$ can answer $\mathcal{B}$'s Open queries as long as they do not involve the challenge signature σ^*. Eventually, when $\mathcal{B}$ outputs his guess b^*, $\mathcal{A}$ returns b^* in her game. Clearly, if $\mathcal{B}$ wins his game, $\mathcal{A}$ also wins her game with the same probability. Therefore, the (full) anonymity of the full dynamic scheme implies full-anonymity of the static scheme.

Finally, we turn our attention to full-traceability. Suppose that there exists an efficient adversary $\mathcal{B}$ who successfully breaks the [BMW03]-full-traceability of the static group signature scheme w.r.t. a particular group of users with non-negligible probability. Using $\mathcal{B}$ which produces with non-negligible probability (m, σ) which cannot be opened in [BMW03]-full-traceability game we construct an efficient adversary $\mathcal{A}_1$ against the traceability of the fully dynamic group signature scheme. Adversary $\mathcal{A}_1$ behaves honestly while initializing the fully dynamic scheme, so that all parameters are honestly generated. As part of the initialization, $\mathcal{A}_1$ generates the tracing key tsk for the fully dynamic scheme, so she knows $\mathsf{GS}_{\mathsf{gmsk}}$. Next, $\mathcal{A}_1$ adds users to the scheme using the AddU oracle to create the group of users for $\mathcal{B}$. Now, $\mathcal{A}_1$ starts $\mathcal{B}$ on input $(\mathsf{GS}_{\mathsf{gmsk}}, \mathsf{GS}_{\mathsf{gpk}})$. Adversary $\mathcal{A}_1$ answers signature queries from $\mathcal{B}$ using her own Sign oracle if the user requested by $\mathcal{B}$ has not been already corrupted. If the secret key of the user in question has already been revealed to $\mathcal{B}$, $\mathcal{A}_1$ uses the user's secret signing key to answer signing queries w.r.t. that user. $\mathcal{A}_1$ is also able to answer $\mathcal{B}$'s corrupt queries using her $\mathsf{RevealU}$ oracle. When $\mathcal{B}$ outputs (m, σ), the probability that σ cannot be opened is non-negligible, so $\mathcal{A}_1$ can use these to output (m, σ, τ) to break the traceability of the fully dynamic scheme.

Therefore, if traceability holds, we conclude that when $\mathcal{B}$ produces (m, σ) and successfully breaks full-traceability, we have with overwhelming probability that $\mathsf{Open}(\mathsf{gmsk}, m, \sigma) = i$ for some i. This will allow us to construct an efficient adversary $\mathcal{A}_2$ against the non-frameability of the fully dynamic group signature scheme. Similarly to $\mathcal{A}_1$, adversary $\mathcal{A}_2$ behaves honestly while initializing the fully dynamic scheme, so that all parameters are honestly generated. As part of the initialization, $\mathcal{A}_2$ gets the tracing manager's key tsk for the fully dynamic scheme, so she knows $\mathsf{GS}_{\mathsf{gmsk}}$. Note that unlike $\mathcal{A}_1$, $\mathcal{A}_2$ additionally has the full dynamic group signature group manager's secret key msk. $\mathcal{A}_2$ can add users to the group using her SndToU oracle. Now, $\mathcal{A}_2$ starts $\mathcal{B}$ on input $(\mathsf{GS}_{\mathsf{gmsk}}, \mathsf{GS}_{\mathsf{gpk}})$. $\mathcal{A}_2$ answers $\mathcal{B}$'s sign queries using her own Sign oracle if the user in question is honest or directly if she knows the secret key of the user. Also, she can reveal the user's secret key requested by $\mathcal{B}$ using her $\mathsf{RevealU}$ oracle. Now, when $\mathcal{B}$ breaks full-traceability, he outputs (m, σ) such that $\mathsf{Open}(\mathsf{gmsk}, m, \sigma) = i$ for some i, with overwhelming probability. It follows that (m, σ) is a valid signature, and that $\mathcal{B}$ did not request a signature for m from user i, nor did $\mathcal{B}$ request the signing key of i. Therefore, $\mathcal{A}_2$ can use (m, σ) to break the variant of non-frameability (Fig. 3) of the fully dynamic group signature scheme, with probability negligibly different from the success probability of $\mathcal{B}$. This shows that full-traceability of the static scheme of [BMW03] is implied by the traceability and non-frameability of the fully dynamic scheme. □

Partially Dynamic Group Signatures [BSZ05]. Fully dynamic group signatures also imply the partially dynamic group signatures of [BSZ05] in the case where nobody is removed from the group. Anonymity, non-frameability and traceability all follow from our corresponding definitions. Correctness follows trivially from the correctness of the fully dynamic group signature scheme.

We now give an explicit construction (in Fig. 5) and security reductions which show how our model relates to [BSZ05].

Theorem 2. *The construction in Fig. 5 is a secure partially dynamic group signature scheme in the sense of [BSZ05].*

Proof. Correctness follows trivially from the correctness of the fully dynamic group signature scheme.

Next we prove anonymity. Suppose that there exists an efficient adversary $\mathcal{B}$ who successfully breaks the [BSZ05]-anonymity of the partially dynamic group signature scheme. We construct an efficient adversary $\mathcal{A}$

$\underline{\mathsf{DGS}_{GKg}(1^\lambda) \to (\mathsf{DGS}_{\mathsf{gpk}}, \mathsf{DGS}_{\mathsf{ik}}, \mathsf{DGS}_{\mathsf{ok}})}$
- $\mathsf{param} \leftarrow \mathsf{GSetup}(1^\lambda)$.
- $\Big((\mathsf{msk}, \mathsf{mpk}, \mathsf{info}), (\mathsf{tsk}, \mathsf{tpk})\Big) \leftarrow \langle \mathsf{GKGen}_{\mathcal{GM}}(\mathsf{param}), \mathsf{GKGen}_{\mathcal{TM}}(\mathsf{param})\rangle$.
- Set $\mathsf{gpk} := (\mathsf{param}, \mathsf{mpk}, \mathsf{tpk}, \mathsf{info})$.
- Set $\mathsf{DGS}_{\mathsf{gpk}} := \mathsf{gpk}$, $\mathsf{DGS}_{\mathsf{ik}} := (\mathsf{gpk}, \mathsf{msk})$ and $\mathsf{DGS}_{\mathsf{ok}} := (\mathsf{gpk}, \mathsf{tsk})$.

$\underline{\mathsf{DGS}_{UKg}(1^\lambda) \to (\mathsf{DGS}_{\mathbf{upk}[i]}, \mathsf{DGS}_{\mathbf{usk}[i]})}$
- Return $\mathsf{UKGen}(1^\lambda)$.

$\underline{\langle \mathrm{Join}, \mathrm{Iss}\rangle}$
- Run $\langle \mathsf{Join}(\mathsf{info}_{\tau_{\mathrm{current}}}, \mathsf{gpk}, i, \mathbf{usk}[i]), \mathsf{Issue}(\mathsf{info}_{\tau_{\mathrm{current}}}, \mathsf{msk}, i, \mathbf{upk}[i])\rangle$, with the issuer modifying $\mathbf{reg}[i]$ if accepting, and the Join algorithm outputting $\mathsf{DGS}_{\mathbf{gsk}}[i] = (\mathsf{gpk}, \mathbf{gsk}[i], \mathsf{info}_\tau)$.

$\underline{\mathsf{DGS}_{GSig}(\mathsf{DGS}_{\mathsf{gpk}}, \mathsf{DGS}_{\mathbf{gsk}[i]}, m) \to \sigma}$
- Parse $\mathsf{DGS}_{\mathsf{gpk}}$ as gpk and $\mathsf{DGS}_{\mathbf{gsk}}[i]$ as $(\mathsf{gpk}, \mathbf{gsk}[i], \mathsf{info}_\tau)$.
- $\Sigma \leftarrow \mathsf{Sign}(\mathsf{gpk}, \mathbf{gsk}[i], \mathsf{info}_{\tau_{\mathrm{current}}}, m)$.
- Return $\sigma = (\Sigma, \tau_{\mathrm{current}})$.

$\underline{\mathsf{DGS}_{GVf}(\mathsf{DGS}_{\mathsf{gpk}}, m, \sigma) \to 0/1}$
- Parse $\mathsf{DGS}_{\mathsf{gpk}}$ as gpk and σ as (Σ, τ).
- Return $\mathsf{Verify}(\mathsf{gpk}, \mathsf{info}_\tau, m, \Sigma)$.

$\underline{\mathsf{DGS}_{Open}(\mathsf{DGS}_{\mathsf{gpk}}, \mathsf{DGS}_{\mathsf{ok}}, \mathsf{DGS}_{\mathbf{reg}}, m, \sigma) \to (i, \pi_{\mathrm{Trace}})}$
- Set $\mathsf{DGS}_{\mathsf{gpk}}$ as gpk, $\mathsf{DGS}_{\mathsf{ok}}$ as $(\mathsf{gpk}, \mathsf{tsk})$ and σ as (Σ, τ).
- Return $\mathsf{Trace}(\mathsf{gpk}, \mathsf{tsk}, \mathsf{info}_\tau, \mathbf{reg}, m, \Sigma)$.

$\underline{\mathsf{DGS}_{Judge}(\mathsf{DGS}_{\mathsf{gpk}}, j, \mathsf{DGS}_{\mathbf{upk}[j]}, m, \sigma, \pi_{\mathrm{Trace}}) \to 0/1}$
- Parse $\mathsf{DGS}_{\mathsf{gpk}}$ as gpk and σ as (Σ, τ).
- Return $\mathsf{Judge}(\mathsf{gpk}, j, \mathsf{info}_\tau, \pi_{\mathrm{Trace}}, \mathbf{upk}[j], m, \Sigma)$.

Fig. 5. Group signatures [BSZ05] from our group signatures.

for the anonymity of the fully dynamic group signature scheme. While initializing the fully dynamic scheme, $\mathcal{A}$ behaves honestly, so that all parameters are honestly generated, but $\mathcal{A}$ knows the group manager's secret key msk, and thus knows $\mathsf{DGS}_{\mathsf{ik}}$. Adversary $\mathcal{A}$ sets $\mathsf{DGS}_{\mathsf{gpk}} := \mathsf{gpk} = (\mathsf{param}, \mathsf{mpk}, \mathsf{tpk}, \mathsf{info})$ and starts $\mathcal{B}$ on $(\mathsf{DGS}_{\mathsf{gpk}}, \mathsf{DGS}_{\mathsf{ik}})$. Adversary $\mathcal{B}$ has

access to oracles $\mathsf{Ch}, \mathsf{Open}, \mathsf{SndToU}, \mathsf{WReg}, \mathsf{USK}, \mathsf{CrptU}$, which directly correspond to the oracles that $\mathcal{A}$ has access to, namely $\mathsf{Chal}_b, \mathsf{Trace}, \mathsf{SndToU}, \mathsf{ModifyReg}, \mathsf{RevealU}, \mathsf{CrptU}$. Therefore, $\mathcal{A}$ can simulate all necessary oracles for $\mathcal{B}$. When $\mathcal{B}$ calls the challenge oracle Ch on (i_0, i_1, m), adversary $\mathcal{A}$ calls her challenge oracle Chal_b on $(\mathsf{info}_{\tau_{\text{current}}}, i_0, i_1, m)$ where τ_{current} is the current epoch. Once $\mathcal{A}$ gets back a signature Σ from her oracle, she passes $\sigma = (\Sigma, \tau_{\text{current}})$ to $\mathcal{B}$ as the answer. Whenever $\mathcal{B}$ outputs his final bit guess b^*, $\mathcal{A}$ returns b^* as her answer. We have that $\mathcal{A}$ breaks anonymity of the fully dynamic group signature whenever $\mathcal{B}$ successfully breaks anonymity of the partially dynamic scheme. Thus, the former anonymity definition implies the latter.

Next, we prove traceability. Assume there exists an efficient adversary $\mathcal{B}$ who successfully breaks the [BSZ05]-traceability of the partially dynamic scheme. We construct an efficient adversary $\mathcal{A}$ against the traceability of the fully dynamic group signature scheme. While initializing the fully dynamic scheme, $\mathcal{A}$ behaves honestly, so that all parameters are honestly generated, but $\mathcal{A}$ knows the tracing manager's secret key tsk, so she knows $\mathsf{DGS}_{\mathsf{ok}}$. Adversary $\mathcal{A}$ sets $\mathsf{DGS}_{\mathsf{gpk}} := \mathsf{gpk} = (\mathsf{param}, \mathsf{mpk}, \mathsf{tpk}, \mathsf{info})$ and starts $\mathcal{B}$ on $(\mathsf{DGS}_{\mathsf{gpk}}, \mathsf{DGS}_{\mathsf{ok}})$. Adversary $\mathcal{B}$ has access to oracles $\mathsf{SndToI}, \mathsf{AddU}, \mathsf{RReg}, \mathsf{USK}, \mathsf{CrptU}$, which directly correspond to some of the oracles $\mathcal{A}$ has access to, namely $\mathsf{SndToM}, \mathsf{AddU}, \mathsf{ReadReg}, \mathsf{RevealU}, \mathsf{CrptU}$. Therefore, $\mathcal{A}$ can simulate all necessary oracles for $\mathcal{B}$. When $\mathcal{B}$ outputs (m, σ) where $\sigma = (\Sigma, \tau)$, $\mathcal{A}$ returns (m, Σ, τ) in her game. Now, whenever $\mathcal{B}$ breaks [BSZ05]-traceability, $\mathcal{A}$ breaks the traceability of the fully dynamic scheme. This shows that traceability of fully dynamic schemes implies the same for partially dynamic schemes, in the sense of [BSZ05].

Finally, we prove non-frameability. Suppose that there exists an efficient adversary $\mathcal{B}$ who successfully breaks the [BSZ05]-non-frameability of the partially dynamic group signature scheme. We construct an efficient adversary $\mathcal{A}$ against the non-frameability of the fully dynamic group signature scheme. While initializing the fully dynamic scheme, $\mathcal{A}$ behaves honestly, so that all parameters are honestly generated, but $\mathcal{A}$ knows the tracing manager's secret key tsk and the issuer's secret key msk. Therefore, $\mathcal{A}$ knows $\mathsf{DGS}_{\mathsf{ik}}, \mathsf{DGS}_{\mathsf{ok}}$. Adversary $\mathcal{A}$ sets $\mathsf{DGS}_{\mathsf{gpk}} := \mathsf{gpk} = (\mathsf{param}, \mathsf{mpk}, \mathsf{tpk}, \mathsf{info})$ and starts $\mathcal{B}$ on $(\mathsf{DGS}_{\mathsf{gpk}}, \mathsf{DGS}_{\mathsf{ok}}, \mathsf{DGS}_{\mathsf{ik}})$. Adversary $\mathcal{B}$ has access to oracles $\mathsf{SndToU}, \mathsf{WReg}, \mathsf{GSig}, \mathsf{USK}, \mathsf{CrptU}$, which directly correspond to some of the oracles that $\mathcal{A}$ has access to, namely $\mathsf{SndToU}, \mathsf{ModifyReg}, \mathsf{Sign}, \mathsf{RevealU}, \mathsf{CrptU}$. Therefore, $\mathcal{A}$ can simulate all necessary oracles for $\mathcal{B}$. Whenever $\mathcal{B}$ outputs $(m, \sigma, i, \pi_{\text{Trace}})$ where $\sigma = (\Sigma, \tau)$, $\mathcal{A}$ returns $(m, \Sigma, i, \pi_{\text{Trace}}, \mathsf{info}_\tau)$ in her game. Therefore, whenever

$\mathcal{B}$ breaks the non-frameability of the partially dynamic scheme, $\mathcal{A}$ succeeds in breaking the non-frameability of the fully dynamic scheme. This shows that non-frameability of fully dynamic schemes implies the same for partially dynamic schemes, in the sense of [BSZ05]. □

Partially Dynamic Group Signatures [KY06]. Finally, we consider the partially-dynamic model of [KY06]. We fix the group manager as the designated opener and set $(\mathsf{msk}, \mathsf{tsk})$ to be the group master secret key. Our group info and registration table generalize their public state string. Their Join algorithm runs our user key-generation and $\mathsf{Join}/\mathsf{Issue}$ algorithms. The membership certificate is then the user's public key along with the group information, and the membership secret is the user's private key. Again, their Open algorithm does not output proofs, and the model does not have a judge algorithm. Therefore, as in the case of [BMW03] we modify our non-frameability game from Fig. 2 where we replace the last 4 lines in the game in Fig. 2 with those in Fig. 3.

Correctness follows trivially from the correctness of the fully dynamic group signature scheme. Security against misidentification-attacks follows from traceability, security against framing-attacks follows from non-frameability, and anonymity follows from the (full) anonymity of the fully dynamic group signature.

We now give an explicit construction (in Fig. 6) and security reductions which show how our model relates to [KY06].

Theorem 3. *The construction in Fig. 6 is is a secure partially dynamic group signature scheme in the sense of [KY06].*

Proof. We begin by proving correctness. User tagging soundness and join soundness are both trivial by construction of the `Setup` and `Join` algorithms. Signing soundness and opening soundness both follow trivially from the correctness of the fully dynamic group signature scheme.

Next we prove security against misidentification-attacks. Suppose that $\mathcal{B}$ is an efficient adversary against the misidentification-attack game of [KY06]. As stated in [KY06], without loss of generality, we may consider an adversary who controls all users in the system, and wins the misidentification-attack game by providing a signature which fails to open to any user. We construct an efficient adversary $\mathcal{A}$ against the traceability of the fully dynamic group signature scheme. While initializing the fully dynamic scheme, $\mathcal{A}$ behaves honestly, so that all parameters are honestly generated, but $\mathcal{A}$ knows the tracing key tsk. In the misidentification-attack game, $\mathcal{B}$ has access to oracles $\mathcal{Q}_{pub}, \mathcal{Q}_{read}, \mathcal{Q}_{open}$ and $\mathcal{Q}_{a-join}$. Adversary

$\underline{\texttt{Setup}}(1^\lambda) \to (\mathcal{Y}, \mathcal{S})$
. $\mathsf{param} \leftarrow \mathsf{GSetup}(1^\lambda)$.
. $\Big((\mathsf{msk}, \mathsf{mpk}, \mathsf{info}), (\mathsf{tsk}, \mathsf{tpk})\Big) \leftarrow \langle \mathsf{GKGen}_{\mathcal{GM}}(\mathsf{param}), \mathsf{GKGen}_{\mathcal{TM}}(\mathsf{param})\rangle$.
. Set $\mathsf{gpk} := (\mathsf{param}, \mathsf{mpk}, \mathsf{tpk})$.
. Set $\mathcal{S} := (\mathsf{msk}, \mathsf{tsk})$, $\mathcal{Y} := (\mathsf{gpk}, \mathsf{info})$.
. Set $\mathsf{St} := (\mathsf{St}_{users}, \mathsf{St}_{trans}) := (\emptyset, \mathbf{reg})$.

$\underline{\texttt{Join}}$
This is an interactive protocol between a user and the group manager.
. The user runs $(\mathbf{usk}[\mathsf{uid}], \mathbf{upk}[\mathsf{uid}]) \leftarrow \mathsf{UKGen}(1^\lambda)$.
. Both parties run $\langle \mathsf{Join}(\mathsf{info}_{\tau_{\text{current}}}, \mathsf{gpk}, \mathsf{uid}, \mathbf{usk}[\mathsf{uid}]), \mathsf{Issue}(\mathsf{info}_{\tau_{\text{current}}}, \mathsf{msk}, \mathsf{uid}, \mathbf{upk}[\mathsf{uid}])\rangle$.
. The user receives private output $\langle i, \mathsf{cert}_i, \mathsf{sec}_i\rangle := \langle \mathsf{uid}, (\mathbf{upk}[\mathsf{uid}], \mathsf{info}_\tau), \mathbf{gsk}[\mathsf{uid}]\rangle$
. Set $\mathsf{transcript}_i := \mathbf{reg}[\mathsf{uid}]$.
. After a successful execution, the state is updated, with $\mathsf{St}_{users} := \mathsf{St}_{users} \cup \{i\}$ and $\mathsf{St}_{trans} := \mathsf{St}_{trans} || \langle i, \mathsf{transcript}_i\rangle$.

$\underline{\texttt{Sign}}(\mathcal{Y}, \mathsf{cert}_i, \mathsf{sec}_i, m) \to \sigma$
. Parse $\mathcal{Y}$ as $(\mathsf{gpk}, \mathsf{info})$ and sec_i as $\mathbf{gsk}[\mathsf{uid}]$.
. $\Sigma \leftarrow \mathsf{Sign}(\mathsf{gpk}, \mathbf{gsk}[\mathsf{uid}], \mathsf{info}_{\tau_{\text{current}}}, m)$.
. Return $\sigma = (\Sigma, \tau_{\text{current}})$.

$\underline{\texttt{Verify}}(\mathcal{Y}, m, \sigma) \to \top/\bot$
. Parse $\mathcal{Y}$ as $(\mathsf{gpk}, \mathsf{info})$ and σ as (Σ, τ).
. Let $b = \mathsf{Verify}(\mathsf{gpk}, \mathsf{info}_\tau, m, \Sigma)$.
. Return $\top$ if $b = 1$, or $\bot$ if $b = 0$.

$\underline{\texttt{Open}}(m, \sigma, \mathcal{Y}, \mathcal{S}, \mathsf{St}) \to i \in \mathsf{St}_{users} \cup \{\bot\}$
. Parse $\mathcal{Y}$ as $(\mathsf{gpk}, \mathsf{info})$, $\mathcal{S}$ as $(\mathsf{msk}, \mathsf{tsk})$, σ as (Σ, τ), and St_{trans} as $\mathbf{reg}$.
. Run $(i, \pi_{\text{Trace}}) \leftarrow \mathsf{Trace}(\mathsf{gpk}, \mathsf{tsk}, \mathsf{info}_\tau, \mathbf{reg}, m, \Sigma)$.
. Return i if $i \in \mathsf{St}_{users}$, or $\bot$ if $i = 0$.

Fig. 6. Group signatures [KY06] from our group signatures.

$\mathcal{A}$ can simulate all of those oracles using the information available to her in her traceability game as well as the oracles she has access to in her game. The oracle $\mathcal{Q}_{pub}$ is easy to simulate. The oracle $\mathcal{Q}_{read}$ corresponds directly to $\mathsf{ReadReg}$. The oracle $\mathcal{Q}_{open}$ can be easily simulated by $\mathcal{A}$, since $\mathcal{A}$ possesses the tracing key tsk. Also, $\mathcal{Q}_{a-join}$ can be simulated using

CrptU and SndToM. Therefore, $\mathcal{A}$ can run $\mathcal{B}$, successfully simulating all oracles, and when $\mathcal{B}$ outputs (m, σ), where $\sigma = (\Sigma, \tau)$, $\mathcal{A}$ outputs (m, Σ, τ) as her answer in her traceability game. Whenever $\mathcal{B}$ succeeds in breaking security against misidentification attacks, $\mathcal{A}$ breaks traceability. Therefore, traceability of the fully dynamic group signature implies security against misidentification attacks.

Next, we prove security against framing-attacks. Suppose that $\mathcal{B}$ is an efficient adversary against the framing-attack game of [KY06]. We construct an efficient adversary $\mathcal{A}$ against the non-frameability of the fully dynamic group signature scheme. While initializing the fully dynamic scheme, $\mathcal{A}$ behaves honestly, so that all parameters are honestly generated, but $\mathcal{A}$ knows $\mathsf{msk}, \mathsf{tsk}$. In his game, $\mathcal{B}$ has access to oracles $\mathcal{Q}_{pub}, \mathcal{Q}_{key}, \mathcal{Q}_{b-join}, \mathcal{Q}_{read}, \mathcal{Q}_{write}, \mathcal{Q}_{sign}$. Using the information available to her in her game as well as the oracles she has access to in her non-frameability game, $\mathcal{A}$ can simulate all of $\mathcal{B}$'s oracles. The oracle $\mathcal{Q}_{pub}$ is easy to simulate. The oracle $\mathcal{Q}_{key}$ is easy to simulate since $\mathcal{A}$ knows msk and tsk. The oracle $\mathcal{Q}_{b-join}$ directly corresponds to SndToU. Oracle $\mathcal{Q}_{read}, \mathcal{Q}_{write}$ are easily simulated by $\mathcal{A}$ since $\mathcal{A}$ possesses msk and has access to the ModifyReg oracle. Also, $\mathcal{A}$ can simulate the $\mathcal{Q}_{sign}$ oracle for $\mathcal{B}$ using her own Sign oracle. This means that $\mathcal{A}$ can run $\mathcal{B}$ to obtain (m, σ) where $\sigma = (\Sigma, \tau)$. Note in [KY06], users created by $\mathcal{Q}_{b-join}$ are honest users whose private keys have not been revealed to $\mathcal{B}$. Therefore, a user uid framed by $\mathcal{B}$ will be in $\mathsf{HUL} \setminus \mathsf{BUL}$, unless $\mathcal{A}$ explicitly calls her RevealU oracle on uid. Given the output (m, σ) produced by $\mathcal{B}$, $\mathcal{A}$ outputs $(m, \Sigma, \mathsf{info}_\tau)$ as her answer in her game. Adversary $\mathcal{A}$ successfully breaks the non-frameability of the fully dynmic group signatures whenever $\mathcal{B}$ wins in his framing-attack game. Therefore, non-frameability of our fully dynamic group signatures implies security against framing attacks in the sense of [KY06].

Finally, we prove anonymity. Suppose $\mathcal{B}$ is an efficient adversary against the anonymity game of [KY06]. We construct an efficient adversary $\mathcal{A}$ against the anonymity of the fully-dynamic group signature scheme. While initializing the fully dynamic scheme, $\mathcal{A}$ behaves honestly, so that all parameters are honestly generated, but $\mathcal{A}$ knows msk. In the anonymity game of [KY06], $\mathcal{B}$ has access to oracles $\mathcal{Q}_{pub}, \mathcal{Q}_{a-join}, \mathcal{Q}_{read}, \mathcal{Q}_{open}$. The oracle $\mathcal{Q}_{pub}$ is easy to simulate. The oracle $\mathcal{Q}_{a-join}$ can be easily simulated using CrptU and knowledge of msk. Similarly, $\mathcal{Q}_{read}$ can be simulated using msk. Lastly, $\mathcal{Q}_{open}$ can be simulated by $\mathcal{A}$ by making use of her own Trace oracle. This means that $\mathcal{A}$ can simulate the anonymity game for $\mathcal{B}$, using Chal_b on the current epoch τ to provide $\mathcal{B}$ with a challenge signature

$\sigma = (\Sigma, \tau)$. Now, when $\mathcal{B}$ returns a bit b^*, $\mathcal{A}$ returns that as her answer in her game. It is clear that whenever $\mathcal{B}$ succeeds in winning the [KY06] anonymity game, $\mathcal{A}$ breaks (full) anonmity of the fully dynamic group signature. Therefore, anonymity of fully dynamic group signatures implies that of partially dynamic group signatures in the sense of [KY06]. □

3 On the Security of Some Existing Schemes

Here we take a closer look at some of the existing fully dynamic schemes and investigate whether or not they are secure using our proposed model.

We show that the state-of-the-art certificate-based schemes in [LPY12b, LPY12a] and [NFHF09] are all susceptible to an attack against traceability which allows any user to sign w.r.t. an epoch predating her joining. In our model this directly breaks traceability, as the signature is w.r.t. an epoch in which the signer was not active. We note that our attack does not contradict the original security proofs of the schemes, but instead highlights that our definition is stronger. We also show that it is easy to repair the schemes at a reasonable cost.

At first glance, our attack is the dual of a well known issue with many revocation systems. If a user is revoked and anonymity is maintained, the revoked user is able to produce back-dated signatures that still verify. The difference here is that while the revoked user *was* authorized to be part of the group for the epoch in question, in our attack the signing user was in fact *not* authorized to sign for the group. If the adversary is able to block the opening of this signature (e.g. via legal action), its existence would implicitly frame the group's past membershipas the signature would be attributed to them.

3.1 Libert et al. Schemes [LPY12b, LPY12a]

In [LPY12a], users are assigned leaves of a complete binary tree and given a membership certificate containing a unique tag identifying the user, and a commitment to the path from the root to the user's leaf in the tree. Note that the certificate is not bound to the epoch at which the user joined the group. In fact, users joining does not change info_τ or the epoch τ itself.

Revocation is based on the subset difference method [NNL01], using disjoint sets S_{k_i,u_i} for $i = 1, \ldots, m$ which cover non-revoked users. Sets are represented by two nodes, a node k_i and one of its descendants node u_i, and cover all leaves of the sub-tree rooted at node k_i which are not leaves of the sub-tree rooted at u_i. Revocations trigger epoch changes with info_τ updated with a new cover.

To sign, the group member anonymously proves that she holds a membership certificate, and that the node indicated by the certificate belongs to one of those sets. More precisely, the user proves that her leaf is a descendant of node k_i but not a descendant of node u_i for some $i \in [m]$.

Since user certificates are not bound to epochs and leaves are covered until their corresponding users are revoked, it is simple to break traceability: a user can join and then produce a signature for an epoch that predates her joining. A similar argument also applies to the variant of the scheme given in [LPY12b].

Theorem 4. *The fully dynamic scheme of Libert et al. [LPY12a] does not satisfy our traceability definition even w.r.t. honestly generated tracing manager's keys.*

Proof. Consider the following strategy in the traceability experiment: the adversary asks to join as a user uid_1 at epoch τ_1. User uid_1 gets assigned the leaf l_1. Then at a later epoch, τ_2, the adversary asks to join as a second user uid_2. Finally, the adversary signs using the credentials of uid_2 but for epoch τ_1.

We can check by inspection that all subproofs in the back-dated signature go through. The crucial observation is that at epoch τ_1, the leaf l_2 is not revoked and thus must be covered by one of the S_{k_i,u_i} sets. As the proof verifies and uid_2 used a legitimate certificate, opening the signature will be successful and indicate uid_2 as the signer. The adversary wins, as uid_2 was not active at epoch τ_1. □

A possible countermeasure against the above attack is to regard unassigned leaves as revoked until they are assigned. This is simple to do as the scheme does not bound the number of revoked users. We do however need to re-examine the number of subsets required to express this, as the $2|\mathcal{R}| - 1$ bound for $|\mathcal{R}|$ revoked users may now seem impractical. If we assume leaves are allocated sequentially to users, we can bound the number of subsets by $2|\mathcal{R}_1| + log(|\mathcal{N} \setminus \mathcal{R}_2|)$ where $\mathcal{R}_2$ is the set of leaves pending allocation and $\mathcal{R}_1$ is the set of leaves allocated to users who were later revoked. Thus, our fix is only marginally more expensive than the base system and much more efficient than a naive analysis would indicate.

If proving set membership/intervals can be done efficiently (and depending on how the epoch counter is implemented), another possible fix is to bind membership certificates to the join epoch and then get the signer to prove that their join epoch is not later than the signing epoch.

3.2 Nakanishi et al. Scheme [NFHF09]

The scheme of Nakanishi et al. [NFHF09] is another certificate-based scheme in the random oracle model. It achieves constant time for both signing and signature verification, relative to the size of the group and the number of revoked users.

A user's group membership certificate consists of a signature on (x, ID) produced by the group manager, where x is a secret owned by the user and ID is a unique integer the manager assigned to her. The group manager can revoke users by issuing revocation lists info_τ. Each list consists of a sequence of *open* integer intervals (R_i, R_{i+1}) signed by the manager, whose endpoints are all the revoked ID's. At each epoch τ, a signer fetches the current info_τ and proves, as part of the signature, that her ID is contained in one interval of the revocation list. If the ID lies between two revoked users' identities, it means it is not an endpoint and so she has not been revoked.

As in other certificate-based constructions, verifiers only know of revoked members, not active ones and, similarly to [LPY12a], the time of joining is not taken into account. This allows users to sign with respect to any epoch prior to joining the group, which represents an attack against our traceability definition.

Theorem 5. *The Nakanishi et al. [NFHF09] fully dynamic group signature scheme does not satisfy our traceability definition.*

Proof. Let $\mathcal{A}$ be an adversary against the traceability game. The adversary adds user uid to the group at epoch τ. Since the user is not revoked, her ID is not an endpoint in any interval of the revocation list info_τ, as for all previous epochs. Therefore, $\mathcal{A}$ could easily produce valid signatures for uid to any epoch $\bar{\tau} < \tau$. Since these signatures trace back to a user which was inactive at the interval with which the signature is associated, $\mathcal{A}$ succeeds in the traceability game. □

The scheme could be easily immunized against the above attack. A first solution, as for [LPY12a], is to initialize the revocation list with all ID's of users that have not joined the group yet. When the manager assigns an ID to a new user, he updates **reg** and the revocation list info_τ. This way, the signature size is not affected. On the other hand, revocation lists are now proportional to the size of the maximum number of users, instead of the number of revoked users.

An alternative countermeasure requires the group manager to include the joining epochs in the certificates by signing $(x, \mathrm{ID}, \tau_{\mathrm{join}})$, where x is a

secret owned by user ID and τ_{join} is the joining epoch. A signer then needs to include in the signature a proof that τ_{join} is not greater than the signing epoch. To realize the latter, one can use membership proof techniques from [TS06, CCS08] which are already used in the original scheme. This would increase the cost of signing and verifying by only a constant factor. The new membership proof would require the group manager to provide signatures for every elapsed epoch, which could be appended, for instance, to the revocation list. This makes revocation lists grow linearly with the number of revoked users as well as the number of epochs.

3.3 Bootle et al. Scheme [BCC$^+$15]

Recently, Bootle et al. [BCC$^+$15] gave a generic construction of accountable ring signatures, where every signature can be traced back to a user in the ring. They also showed how one can obtain fully dynamic group signatures from accountable ring signatures. In addition, they gave an efficient instantiation in the random oracle model that is based on the DDH assumption. Their instantiation yields signatures of logarithmic size (w.r.t. the size of the ring), while signing is quasi-linear, and signature verification requires a linear number of operations. Bootle et al. claimed that their instantiation is more efficient than existing group signature schemes based on standard assumptions.

Each user has a secret key and an associated verification key. To sign, users first encrypt their verification key. Then, via a membership proof, they provide a signature of knowledge showing that the verification key belongs to the ring, and that they know the corresponding secret key. We now reproduce their definitions and prove their construction is secure w.r.t. the stronger variant of our model.

Accountable Ring Signatures. Bootle et al. [BCC$^+$15] define an accountable ring signature scheme over a PPT setup ARS_{Setup} as a tuple of polynomial time algorithms ($\mathsf{ARS}_{OKGen}, \mathsf{ARS}_{UKGen}, \mathsf{ARS}_{Sign}, \mathsf{ARS}_{Vfy}$, $\mathsf{ARS}_{Open}, \mathsf{ARS}_{Judge}$).

$\mathsf{ARS}_{Setup}(1^\lambda)$**:** Given the security parameter, produces public parameters pp used (sometimes implicitly) by the rest of the scheme. The public parameters define key spaces PK, DK, VK, SK with efficient algorithms for sampling and deciding membership.

$\mathsf{ARS}_{OKGen}(pp)$**:** Given the public parameters pp, produces a public key $pk \in PK$ and secret key $dk \in DK$ for an opener. Without loss

of generality, we assume dk defines pk deterministically and write $pk = \mathsf{ARS}_{OKGen}(pp, dk)$ when computing pk from dk.

$\mathsf{ARS}_{UKGen}(pp)$**:** Given the public parameters pp, produces a verification key $vk \in VK$ and a secret signing key $sk \in SK$ for a user. We can assume sk deterministically determines vk and write $vk = \mathsf{ARS}_{UKGen}(pp, sk)$ when computing vk from sk.

$\mathsf{ARS}_{Sign}(pk, m, R, sk)$**:** Given an opener's public key, a message, a ring (i.e. a set of verification keys) and a secret key, produces a ring signature σ. The algorithm returns the error symbol $\perp$ if the inputs are malformed, i.e., if $pk \notin PK, R \not\subset VK, sk \notin SK$ or $vk = \mathsf{ARS}_{UKGen}(pp, sk) \notin R$.

$\mathsf{ARS}_{Vfy}(pk, m, R, \sigma)$**:** Given an opener's public key, a message, a ring and a signature, returns 1 if accepting the signature and 0 otherwise. We assume the algorithm always returns 0 if the inputs are malformed, i.e., if $pk \notin PK$ or $R \not\subset VK$.

$\mathsf{ARS}_{Open}(m, R, \sigma, dk)$**:** Given a message, a ring, a ring signature and an opener's secret key, returns a verification key vk and a proof ψ that the owner of vk produced the signature. If any of the inputs are invalid, i.e., $dk \notin DK$ or σ is not a valid signature using $pk = \mathsf{ARS}_{OKGen}(pp, dk)$, the algorithm returns $\perp$.

$\mathsf{ARS}_{Judge}(pk, m, R, \sigma, vk, \psi)$**:** Given an opener's public key, a message, a ring, a signature, a verification key and a proof, returns 1 if accepting the proof and 0 otherwise. We assume the algorithm returns 0 if σ is invalid or $vk \notin R$.

An accountable ring signature scheme [BCC+15] should be correct, fully unforgeable, anonymous and traceable as defined below.

Definition 1 (Perfect correctness). *An accountable ring signature scheme is perfectly correct if for any PPT adversary* $\mathcal{A}$

$$\Pr\left[\begin{array}{c} pp \leftarrow \mathsf{ARS}_{Setup}(1^\lambda); (vk, sk) \leftarrow \mathsf{ARS}_{UKGen}(pp); \\ (pk, m, R) \leftarrow \mathcal{A}(pp, sk); \sigma \leftarrow \mathsf{ARS}_{Sign}(pk, m, R, sk) \ : \\ \textit{If } pk \in PK, R \subset VK, vk \in R \text{ then } \mathsf{ARS}_{Vfy}(pk, m, R, \sigma) = 1 \end{array}\right] = 1.$$

Full unforgeability ensures that an adversary, who may control the opener, can neither falsely accuse an honest user of producing a ring signature nor forge ring signatures on behalf of an honest ring. The former should hold even when all other users in the ring are corrupt.

Definition 2 (Full Unforgeability). *An accountable ring signature scheme is fully unforgeable if for any PPT adversary* $\mathcal{A}$

$$\Pr\left[\begin{array}{c} pp \leftarrow \mathsf{ARS}_{Setup}(1^\lambda); (pk, vk, m, R, \sigma, \psi) \leftarrow \mathcal{A}^{\mathrm{UKGen,Sign,RevealU}}(pp) : \\ \Big(vk \in Q_{\mathrm{UKGen}} \setminus Q_{\mathrm{RevealU}} \ \wedge \ (pk, vk, m, R, \sigma) \notin Q_{\mathrm{Sign}} \\ \wedge \mathsf{ARS}_{Judge}(pk, m, R, \sigma, vk, \psi) = 1\Big) \\ \vee \ \Big(R \subset Q_{\mathrm{UKGen}} \setminus Q_{\mathrm{RevealU}} \ \wedge \ (pk, \cdot, m, R, \sigma) \notin Q_{\mathrm{Sign}} \\ \wedge \ \mathsf{ARS}_{Vfy}(pk, m, R, \sigma) = 1\Big) \end{array}\right] \approx 0.$$

- UKGen *runs* $(vk, sk) \leftarrow \mathsf{ARS}_{UKGen}(pp)$ *and returns* vk. Q_{UKGen} *is the set of verification keys* vk *that have been generated by this oracle.*
- Sign *is an oracle that on query* (pk, vk, m, R) *checks if* $vk \in R \cap Q_{\mathrm{UKGen}}$*, in which case returns* $\sigma \leftarrow \mathsf{ARS}_{Sign}(pk, m, R, sk)$. Q_{Sign} *contains the queries and responses* (pk, vk, m, R, σ).
- RevealU *is an oracle that when queried on* $vk \in Q_{\mathrm{UKGen}}$ *returns the corresponding signing key* sk. Q_{RevealU} *is the list of verification keys* vk *for which the corresponding signing key has been revealed.*

Anonymity ensures that a signature does not reveal the identity of the ring member who produced it without the opener explicitly wanting to open the particular signature. The definition below implies anonymity against full key exposure attacks [BKM09] as in the game the adversary is allowed to choose the secret signing keys of the users.

Definition 3 (Anonymity). *An accountable ring signature scheme is anonymous if for any PPT adversary* $\mathcal{A}$

$$\Pr\left[\begin{array}{c} pp \leftarrow \mathsf{ARS}_{Setup}(1^\lambda); b \leftarrow \{0,1\}; (pk, dk) \leftarrow \mathsf{ARS}_{OKGen}(pp) : \\ \mathcal{A}^{\mathrm{Chal}_b, \mathrm{Open}}(pp, pk) = b \end{array}\right] \approx \frac{1}{2}.$$

- Chal_b *is an oracle that the adversary can only call once.*
  On query (m, R, sk_0, sk_1) *it runs* $\sigma_0 \leftarrow \mathsf{ARS}_{Sign}(pk, m, R, sk_0)$; $\sigma_1 \leftarrow \mathsf{ARS}_{Sign}(pk, m, R, sk_1)$. *If* $\sigma_0 \neq \perp$ *and* $\sigma_1 \neq \perp$ *it returns* σ_b*, otherwise it returns* $\perp$.
- Open *is an oracle that on a query* (m, R, σ) *returns* $\mathsf{ARS}_{Open}(m, R, \sigma, dk)$. *If* σ *was obtained by calling* Chal_b *on* (m, R)*, the oracle returns* $\perp$.

Traceability ensures that the specified opener can always identify the ring member who produced a signature and that she is able to produce a valid proof for her decision.

Definition 4 (Traceability). *An accountable ring signature scheme is traceable if for any PPT adversary* $\mathcal{A}$

$$\Pr\left[\begin{array}{c} pp \leftarrow \mathsf{ARS}_{Setup}(1^\lambda); (dk, m, R, \sigma) \leftarrow \mathcal{A}(pp); \\ pk \leftarrow \mathsf{ARS}_{OKGen}(pp, dk); (vk, \psi) \leftarrow \mathsf{ARS}_{Open}(m, R, \sigma, dk) : \\ \mathsf{ARS}_{Vfy}(pk, m, R, \sigma) = 1 \ \wedge \ \mathsf{ARS}_{Judge}(pk, m, R, \sigma, vk, \psi) = 0 \end{array}\right] \approx 0.$$

Tracing soundness ensures that a signature cannot trace to two different users; only one person can be identified as the signer even when all users as well as the opener are fully corrupt. Similarly to the setting of group signatures [SSE+12], this requirement is vital for some applications, e.g., where users might be rewarded for signatures they produced, or to avoid shifting blame when signatures are used as evidence of abuse.

Definition 5 (Tracing Soundness). *An accountable ring signature scheme satisfies tracing soundness if for any PPT adversary* $\mathcal{A}$

$$\Pr\left[\begin{array}{c} pp \leftarrow \mathsf{ARS}_{Setup}(1^\lambda); (m, \sigma, R, pk, vk_1, vk_2, \psi_1, \psi_2) \leftarrow \mathcal{A}(pp) : \\ \forall i \in \{1, 2\}, \ \mathsf{ARS}_{Judge}(pk, m, R, \sigma, vk_i, \psi_i) = 1 \wedge \ vk_1 \neq vk_2 \end{array}\right] \approx 0.$$

Security in our model. Here we will show that a fully dynamic group signature scheme obtained from the generic accountable ring signature construction of [BCC+15] is secure w.r.t. our security model.

In what follows, note that each epoch τ specifies an instance of the accountable ring signature scheme with ring R_τ. Algorithms from the accountable ring signature scheme are labelled with ARS. We assume that the epoch can be appended to a message using the $||$ operation, and removed again without ambiguity.

An accountable ring signature scheme does not involve a group manager. Hence, to construct a group signature, we assume the existence of a functionality GMg, which allows a group manager's key-pair to be derived as $(\mathsf{gmk}, \mathsf{msk}) \leftarrow \mathrm{GMg}(1^\lambda)$, and initialises a group information board which is visible to all parties, but can only be modified by a party with msk.

The construction of a fully dynamic group signature from an accountable ring signature is presented in Fig. 7.

Theorem 6. *The generic group signature scheme construction from accountable ring signatures of [BCC+15] satisfies our definitions for a secure, fully-dynamic group signature scheme.*

$\underline{\mathsf{GSetup}(1^\lambda) \to \mathsf{param}}$
. Compute $\mathsf{param} := pp \leftarrow \mathsf{ARS}_{Setup}(1^\lambda)$.
. Initialize **reg** and a counter $\tau := 0$.

$\underline{\langle \mathsf{GKGen}_{\mathcal{GM}}(\mathsf{param}), \mathsf{GKGen}_{\mathcal{TM}}(\mathsf{param}) \rangle}$
. The group and tracing managers participate in an interactive protocol to generate $(\mathsf{tpk}, \mathsf{tsk}) \leftarrow \mathsf{ARS}_{OKGen}(pp)$ and $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{GMg}(1^\lambda)$.
. Set $\mathsf{gpk} := (\mathsf{param}, \mathsf{mpk}, \mathsf{tpk})$, and initialise info.

$\underline{\mathsf{UKGen}(1^\lambda) \to (\mathbf{usk}[\mathsf{uid}], \mathbf{upk}[\mathsf{uid}])}$
. Return $(\mathbf{usk}[\mathsf{uid}], \mathbf{upk}[\mathsf{uid}]) \leftarrow \mathsf{ARS}_{UKGen}(pp)$.

$\underline{\langle \mathsf{Join}(\tau, \mathsf{gpk}, \mathsf{uid}, \mathbf{usk}[\mathsf{uid}]), \mathsf{Issue}(\tau, \mathsf{msk}, \mathsf{uid}, \mathbf{upk}[\mathsf{uid}]) \rangle}$
. The group manager and a new user uid who already has a key-pair undergo an interactive protocol to register uid.
. Upon completion, the group manager uses msk to update **reg** with the transcript.
. Set $\mathbf{gsk}[\mathsf{uid}] = \mathbf{usk}[\mathsf{uid}]$.

$\underline{\mathsf{UpdateGroup}(\mathsf{gpk}, \mathsf{msk}, \mathsf{info}_{\text{current}}, \mathcal{S}, \mathbf{reg}) \to \mathsf{info}_{\text{new}}}$
. If R_{new} is different from R_{current} then increment the counter τ
. The new group information $\mathsf{info}_{\text{new}}$ contains the current group R_{new} and the public keys of the members as well as the counter τ.

$\underline{\mathsf{Sign}(\mathsf{gpk}, \mathbf{gsk}[\mathsf{uid}], \mathsf{info}_\tau, m) \to \Sigma}$
. Denote the group for the current epoch by R_τ. This is part of info_τ.
. Return $\Sigma \leftarrow \mathsf{ARS}_{Sign}(\mathsf{tpk}, m||\tau, R_\tau, \mathbf{gsk}[\mathsf{uid}])$.

$\underline{\mathsf{Verify}(\mathsf{gpk}, \mathsf{info}_\tau, m, \Sigma) \to 1/0}$
. Return $\mathsf{ARS}_{Vfy}(\mathsf{tpk}, m||\tau, R_\tau, \Sigma)$.

$\underline{\mathsf{Trace}(\mathsf{gpk}, \mathsf{tsk}, \mathsf{info}_\tau, \mathbf{reg}, m, \Sigma) \to (\mathsf{uid}, \pi_{\text{Trace}})}$
. Return $\mathsf{ARS}_{Open}(m||\tau, R_\tau, \Sigma, \mathsf{tsk})$.

$\underline{\mathsf{Judge}(\mathsf{gpk}, \mathsf{uid}, \mathsf{info}_\tau, \pi_{\text{Trace}}, \mathbf{upk}[\mathsf{uid}], m, \Sigma) \to 1/0}$
. Return $\mathsf{ARS}_{Judge}(\mathsf{tpk}, m||\tau, R_\tau, \Sigma, \mathbf{upk}[\mathsf{uid}], \pi_{\text{Trace}})$.

$\underline{\mathsf{IsActive}(\mathsf{info}_\tau, \mathbf{reg}, \mathsf{uid})}$
. Check info_τ to see if user uid is in R_τ, and returns $0/1$ accordingly.

Fig. 7. Construction of a fully dynamic group signature from an accountable ring signature [BCC+15] .

Proof. We begin by proving correctness. For simplicity and without loss in generality, we reduce the computational variant of the correctness requirement, i.e. where the last three lines in the correctness game in Fig. 2 are dropped (see Section 2.1), to the perfect correctness of the accountable ring signature. Let $\mathcal{A}$ be an adversary against the (computational) correctness of the fully dynamic group signature scheme. We construct an adversary $\mathcal{B}$ against the (perfect) correctness of the accountable ring signature scheme. On receiving (pp, sk) from his game, $\mathcal{B}$ sets $\mathsf{param} = pp$ and chooses $(\mathsf{tpk}, \mathsf{tsk})$ and $(\mathsf{mpk}, \mathsf{msk})$ by himself. He also initializes $\mathbf{reg}$ and info. Note that $\mathcal{B}$ can compute vk corresponding to sk by computing $vk = \mathsf{ARS}_{UKGen}(pp, sk)$. Now, $\mathcal{B}$ sets $\mathsf{gpk} := (\mathsf{param}, \mathsf{mpk}, \mathsf{tpk})$ and starts $\mathcal{A}$ on $(\mathsf{gpk}, \mathsf{info})$. Since $\mathcal{B}$ knows msk, he can simulate all oracle queries for $\mathcal{A}$. Let q be a polynomial upper bound on the number of AddU queries $\mathcal{A}$ can make in her game. Adversary $\mathcal{B}$ randomly chooses an index $i \leftarrow \{1, \ldots, q\}$ and sets the challenge key pair (vk, sk) of his game as the keys of the i-th user uid^* queried to the AddU oracle: $(\mathbf{usk}[\mathsf{uid}^*], \mathbf{upk}[\mathsf{uid}^*]) := (sk, vk)$. By construction, user keys do not depend on uid, so the simulation is perfect. Whenever $\mathcal{A}$ succeeds in breaking correctness by returning (uid, m, τ), $\mathcal{B}$ succeeds when $\mathsf{uid} = \mathsf{uid}^*$ by returning $(\mathsf{tpk}, m||\tau, R_\tau)$. Note that i is independent of the success probability of $\mathcal{A}$. Hence, the success probability of $\mathcal{B}$ is that of $\mathcal{A}$ divided by a factor of q, and thus correctness of the fully dynamic group signature scheme follows.

For anonymity, suppose that $\mathcal{A}$ is an adversary against the anonymity of the group signature scheme. We construct an adversary $\mathcal{B}$ against the anonymity of the accountable ring signature scheme. Adversary $\mathcal{B}$ gets $\mathsf{param} := pp$ and pk from his anonymity game. He sets $\mathsf{tpk} := pk$ and interacts with $\mathcal{A}$ on behalf of the tracing manager of the fully dynamic group signature scheme to get $(\mathsf{msk}, \mathsf{mpk}, \mathsf{info})$. Note that $\mathcal{B}$ does not know dk corresponding to pk and hence does not know the secret tracing key tsk of the tracing manager. $\mathcal{B}$ now sets $\mathsf{gpk} := (\mathsf{param}, \mathsf{mpk}, \mathsf{tpk})$ and initializes $\mathbf{reg}$. For all oracles except Trace and Chal_b, $\mathcal{B}$ is able to simulate all queries of $\mathcal{A}$ by himself and return the result, since $\mathcal{B}$ knows all necessary keys. If $\mathcal{A}$ queries $(m, \Sigma, \mathsf{info}_\tau)$ to her Trace oracle, $\mathcal{B}$ can simulate the oracle by querying $(m||\tau, R_\tau, \Sigma)$ to his Open oracle and forwards the answer to $\mathcal{A}$. If $\mathcal{A}$ queries $(\mathsf{info}_\tau, \mathsf{uid}_0, \mathsf{uid}_1, m)$ to her Chal_b oracle, $\mathcal{B}$ simulates the oracle by querying $(m||\tau, R_\tau, \mathbf{gsk}[\mathsf{uid}_0], \mathbf{gsk}[\mathsf{uid}_1])$ to his Chal_b oracle and returns the answer to $\mathcal{A}$. Eventually, when $\mathcal{A}$ outputs her guess b^*, $\mathcal{B}$ returns b^* as his answer in his game. It is clear that both adversaries have the same advantage. Therefore, by anonymity of the accountable

ring signature scheme, the fully dynamic group signature scheme satisfies our full anonymity definition.

Next, we prove non-frameability. Let $\mathcal{A}$ be an adversary against the non-frameability of the fully dynamic group signature scheme. We construct an adversary $\mathcal{B}$ attacking the full-unforgeability of the accountable ring signature scheme. Given pp, $\mathcal{B}$ starts $\mathcal{A}$ on $\mathsf{param} := pp$ and receives $(\mathsf{info}, \mathsf{msk}, \mathsf{mpk}, \mathsf{tsk}, \mathsf{tpk})$ using which $\mathcal{B}$ can compute gpk which he forwards to $\mathcal{A}$. Now, for all oracles except SndToU, $\mathsf{RevealU}$ and Sign, $\mathcal{B}$ is able to simulate all queries of $\mathcal{A}$ by himself and return the result. For the other three oracles, $\mathcal{B}$ can simulate those successfully by consulting his own oracles UKGen, Sign and RevealU if necessary. Eventually, when $\mathcal{A}$ outputs $(m, \Sigma, \mathsf{uid}, \pi_{\mathrm{Trace}}, \mathsf{info}_\tau)$, $\mathcal{B}$ returns $(\mathsf{tpk}, \mathbf{upk}[\mathsf{uid}], m||\tau, R_\tau, \Sigma, \pi_{\mathrm{Trace}})$ in his game. Clearly, if $\mathcal{A}$ wins her game, $\mathcal{B}$ succeeds in breaking the full unforgeability of the accountable ring signature since his output satisfies the first clause of the winning condition.

Next, we prove traceability. Let $\mathcal{A}$ be an adversary against the traceability of the fully dynamic group signature scheme. We construct an adversary $\mathcal{B}$ against the traceability of the accountable ring signature scheme. Given pp, $\mathcal{B}$ sets $\mathsf{param} := pp$, generates $(\mathsf{msk}, \mathsf{mpk})$ and initializes **reg** and info. He interacts with $\mathcal{A}$ on behalf the group manager in order to obtain $(\mathsf{tsk}, \mathsf{tpk})$. He sets $\mathsf{gpk} := (\mathsf{param}, \mathsf{mpk}, \mathsf{tpk})$ and forwards $(\mathsf{gpk}, \mathsf{info})$ to $\mathcal{A}$. Now, $\mathcal{B}$ is able to simulate all of the oracle queries of $\mathcal{A}$. Eventually, when $\mathcal{A}$ outputs (m, Σ, τ) where Σ is a valid signature, $\mathcal{B}$ returns $(\mathsf{tsk}, m||\tau, R_\tau, \Sigma)$ in his game. We have three cases to consider depending on how $\mathcal{A}$ wins her game. If uid is inactive group member at epoch τ, then $\mathsf{uid} \notin R_\tau$. If $\mathsf{uid} = 0$ then again $\mathsf{uid} \notin R_\tau$. The final case is that the group signature judge algorithm returns 0. The probability of success of $\mathcal{B}$ is the same as that of $\mathcal{A}$. Thus, by the traceability of the accountable ring signature scheme, the fully dynamic group signature scheme satisfies traceability.

Finally, we prove tracing-soundness. Let $\mathcal{A}$ be an adversary against the tracing-soundness of the fully dynamic group signature scheme. We construct an adversary $\mathcal{B}$ against the tracing-soundness of the accountable ring signature scheme. Given pp, $\mathcal{B}$ initializes **reg** and sets $\mathsf{param} := pp$. He starts $\mathcal{A}$ on param to get $(\mathsf{info}, \mathsf{msk}, \mathsf{mpk}, \mathsf{tsk}, \mathsf{tpk})$. He sets $\mathsf{gpk} := (\mathsf{param}, \mathsf{mpk}, \mathsf{tpk})$ which he then forwards to $\mathcal{A}$. Note that all $\mathcal{A}$'s oracle calls can be simulated by $\mathcal{B}$ since he has the required keys. Eventually, $\mathcal{A}$ halts by responding with a tuple $\Big(m, \Sigma, \mathsf{uid}_1, \pi_{\mathrm{Trace}_1}, \mathsf{uid}_2, \pi_{\mathrm{Trace}_2}, \mathsf{info}_\tau\Big)$. By construction, Σ returned by $\mathcal{A}$ is a valid accountable ring signature on

m that traces to two different users. Adversary $\mathcal{B}$ returns $\Big(m||\tau, \Sigma, \mathsf{tpk}, \mathbf{upk}[\mathsf{uid}_1], \mathbf{upk}[\mathsf{uid}_2], \pi_{\mathrm{Trace}_1}, \pi_{\mathrm{Trace}_2}\Big)$ as his output in his game. Clearly, if $\mathcal{A}$ wins her game, $\mathcal{B}$ wins his game with the same advantage. Thus, if the accountable ring signature scheme satisfies tracing soundness so does the fully dynamic group signature scheme. □

References

[ACHdM05] G. Ateniese, J. Camenisch, S. Hohenberger, and B. de Medeiros. *Practical group signatures without random oracles. IACR Cryptology ePrint Archive*, 2005.

[ACJT00] G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. *A practical and provably secure coalition-resistant group signature scheme.* In *Advances in Cryptology - CRYPTO*. 2000.

[AHO10] M. Abe, K. Haralambiev, and M. Ohkubo. *Signing on elements in bilinear groups for modular protocol design. IACR Cryptology ePrint Archive*, 2010.

[AST01] G. Ateniese, D. Song, and G. Tsudik. *Quasi-efficient revocation of group signatures. IACR Cryptology ePrint Archive*, 2001:101, 2001.

[BBS04] D. Boneh, X. Boyen, and H. Shacham. *Short group signatures.* In *Advances in Cryptology - CRYPTO*. 2004.

[BCC04] E. F. Brickell, J. Camenisch, and L. Chen. *Direct anonymous attestation.* In *Conference on Computer and Communications Security - CCS*. 2004.

[BCC+15] J. Bootle, A. Cerulli, P. Chaidos, E. Ghadafi, J. Groth, and C. Petit. *Short accountable ring signatures based on DDH.* In *Computer Security - ESORICS*. 2015.

[BCN+10] P. Bichsel, J. Camenisch, G. Neven, N. P. Smart, and B. Warinschi. *Get shorty via group signatures without encryption.* In *Security and Cryptography for Networks - SCN*. 2010.

[BKM09] A. Bender, J. Katz, and R. Morselli. *Ring signatures: Stronger definitions, and constructions without random oracles. Journal of Cryptology*, 22(1):114, 2009.

[BMW03] 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.

[BR93] M. Bellare and P. Rogaway. *Random oracles are practical: A paradigm for designing efficient protocols.* In *Conference on Computer and Communications Security - CCS*. 1993.

[Bri04] E. Brickell. *An efficient protocol for anonymously providing assurance of the container of a private key. Submitted to the Trusted Computing Group*, 2004.

[BS01] E. Bresson and J. Stern. *Efficient revocation in group signatures.* In *Public Key Cryptography - PKC*. 2001.

[BS04] D. Boneh and H. Shacham. *Group signatures with verifier-local revocation.* In *Conference on Computer and Communications Security - CCS*. 2004.

[BSZ05] M. Bellare, H. Shi, and C. Zhang. *Foundations of group signatures: The case of dynamic groups.* In *Topics in Cryptology - CT-RSA*. 2005.

[BW06] X. Boyen and B. Waters. *Compact group signatures without random oracles.* In *Advances in Cryptology - EUROCRYPT*. 2006.

[BW07] X. Boyen and B. Waters. *Full-domain subgroup hiding and constant-size group signatures.* In *Public Key Cryptography - PKC*. 2007.

[CCS08] J. Camenisch, R. Chaabouni, and A. Shelat. *Efficient protocols for set membership and range proofs.* In *Advances in Cryptology - ASIACRYPT*. 2008.

[CG04] J. Camenisch and J. Groth. *Group signatures: Better efficiency and new theoretical aspects.* In *Security in Communication Networks - SCN*. 2004.

[CL02] J. Camenisch and A. Lysyanskaya. *Dynamic accumulators and application to efficient revocation of anonymous credentials.* In *Advances in Cryptology - CRYPTO*. 2002.

[CL04] J. Camenisch and A. Lysyanskaya. *Signature schemes and anonymous credentials from bilinear maps.* In *Advances in Cryptology - CRYPTO*. 2004.

[CM98] J. Camenisch and M. Michels. *A group signature scheme with improved efficiency.* In *Advances in Cryptology - ASIACRYPT*. 1998.

[CS97] J. Camenisch and M. Stadler. *Efficient group signature schemes for large groups (extended abstract).* In *Advances in Cryptology - CRYPTO*. 1997.

[CvH91] D. Chaum and E. van Heyst. *Group signatures.* In *Advances in Cryptology - EUROCRYPT*. 1991.

[DKNS04] Y. Dodis, A. Kiayias, A. Nicolosi, and V. Shoup. *Anonymous identification in ad hoc groups.* In *Advances in Cryptology - EUROCRYPT*. 2004.

[DP06] C. Delerablée and D. Pointcheval. *Dynamic fully anonymous short group signatures.* In *Progressing Cryptology - VIETCRYPT*. 2006.

[FI05] J. Furukawa and H. Imai. *An efficient group signature scheme from bilinear maps.* In *Information Security and Privacy - ACISP*. 2005.

[FY04] J. Furukawa and S. Yonezawa. *Group signatures with separate and distributed authorities.* In *Security in Communication Networks - SCN*. 2004.

[Gro06] J. Groth. *Simulation-sound NIZK proofs for a practical language and constant size group signatures.* In *Advances in Cryptology - ASIACRYPT*. 2006.

[Gro07] J. Groth. *Fully anonymous group signatures without random oracles.* In *Advances in Cryptology - ASIACRYPT*. 2007.

[KTY04] A. Kiayias, Y. Tsiounis, and M. Yung. *Traceable signatures.* In *Advances in Cryptology - EUROCRYPT*. 2004.

[KY05] A. Kiayias and M. Yung. *Group signatures with efficient concurrent join.* In *Advances in Cryptology - EUROCRYPT*. 2005.

[KY06] A. Kiayias and M. Yung. *Secure scalable group signature with dynamic joins and separable authorities. IJSN*, 1(1/2):24, 2006.

[LLNW14] A. Langlois, S. Ling, K. Nguyen, and H. Wang. *Lattice-based group signature scheme with verifier-local revocation.* In *Public-Key Cryptography - PKC*. 2014.

[LPY12a] B. Libert, T. Peters, and M. Yung. *Group signatures with almost-for-free revocation.* In *Advances in Cryptology - CRYPTO*. 2012.

[LPY12b] B. Libert, T. Peters, and M. Yung. *Scalable group signatures with revocation.* In *Advances in Cryptology - EUROCRYPT*. 2012.

[LV09] B. Libert and D. Vergnaud. *Group signatures with verifier-local revocation and backward unlinkability in the standard model.* In *Cryptology and Network Security - CANS*. 2009.

[NF05] T. Nakanishi and N. Funabiki. *Verifier-local revocation group signature schemes with backward unlinkability from bilinear maps.* In *Advances in Cryptology - ASIACRYPT*. 2005.

[NFHF09] T. Nakanishi, H. Fujii, Y. Hira, and N. Funabiki. *Revocable group signature schemes with constant costs for signing and verifying.* In *Public Key Cryptography - PKC*. 2009.

[Ngu05] L. Nguyen. *Accumulators from bilinear pairings and applications.* In *Topics in Cryptology - CT-RSA*. 2005.

[NNL01] D. Naor, M. Naor, and J. Lotspiech. *Revocation and tracing schemes for stateless receivers.* In *Advances in Cryptology - CRYPTO*. 2001.

[NS04] L. Nguyen and R. Safavi-Naini. *Efficient and provably secure trapdoor-free group signature schemes from bilinear pairings.* In *Advances in Cryptology - ASIACRYPT*. 2004.

[Son01] D. X. Song. *Practical forward secure group signature schemes.* In *Conference on Computer and Communications Security - CCS*. 2001.

[SSE+12] Y. Sakai, J. C. N. Schuldt, K. Emura, G. Hanaoka, and K. Ohta. *On the security of dynamic group signatures: Preventing signature hijacking.* In *Public Key Cryptography - PKC*. 2012.

[TS06] I. Teranishi and K. Sako. *k-times anonymous authentication with a constant proving cost.* In *Public Key Cryptography - PKC*. 2006.

[TX03] G. Tsudik and S. Xu. *Accumulating composites and improved group signing.* In *Advances in Cryptology - ASIACRYPT*. 2003.