Adding Controllable Linkability to Pairing-Based Group Signatures For Free*

Daniel Slamanig, Raphael Spreitzer, and Thomas Unterluggauer

Graz University of Technology, IAIK
{daniel.slamanig,raphael.spreitzer,thomas.unterluggauer}@iaik.tugraz.at

Abstract. Group signatures, which allow users of a group to anonymously produce signatures on behalf of the group, are an important cryptographic primitive for privacy-enhancing applications. Over the years, various approaches to enhanced anonymity management mechanisms, which extend the standard feature of opening of group signatures, have been proposed.
In this paper we show how pairing-based group signature schemes (PB-GSSs) based on the sign-and-encrypt-and-prove (SEP) paradigm can be generically transformed in order to support one particular enhanced anonymity management mechanism, *i.e.*, we propose a transformation that turns every such PB-GSS into a PB-GSS with *controllable linkability*. Basically, this transformation replaces the public key encryption scheme used for identity escrow within a group signature scheme with a modified all-or-nothing public key encryption with equality tests scheme (denoted AoN-PKEET*) instantiated from the respective public key encryption scheme. Thereby, the respective trapdoor is given to the linking authority as a linking key. The appealing benefit of this approach in contrast to other anonymity management mechanisms (such as those provided by traceable signatures) is that controllable linkability can be added to PB-GSSs based on the SEP paradigm *for free*, *i.e.*, it neither influences the signature size nor the computational costs for signers and verifiers in comparison to the scheme without this feature.

Keywords: Controllable linkability, generic transformation, pairing-based group signature schemes.

1 Introduction

The concept of group signature schemes (GSSs) has been introduced by Chaum and van Heyst [19] in 1991. Members within a predefined group are able to sign messages on behalf of the group anonymously. Verifiers can determine whether a signature indeed has been produced by a member of the group, but are not able to determine the actual identity of the signer. However, in case of dispute the so-called group manager (GM) is able to open a given signature in order to determine the identity of the actual signer in case of dispute. Early works of GSSs considered only a static setting [6], where the group is fixed at the time of the setup, whereas more recent constructions consider dynamic groups [7], *i.e.*, new members may be added and possibly deleted to and from the group over time. Moreover, in some cases it is also desirable to have distributed authorities, *i.e.*, one party only receives the opening key and a distinct party receives the issuing key required to add new members or to revoke existing members.

For both, the static as well as the dynamic setting, there are constructions under generic assumptions [6, 7] based on the *sign-and-encrypt-and-prove* (SEP) paradigm [16]. In schemes following this paradigm, a user on joining a group receives a signature from the GM, which is known as the membership certificate. A group signature produced by a user is then an encryption of the membership certificate of the user (or some related information – we will always use the term membership certificate henceforth) under the GM's public key and a non-interactive zero-knowledge proof of knowledge of

*An extended abstract of this paper appears in the proceedings of the 17th International Conference on Information Security (ISC 2014).

well-formedness of the respective values. A common way to instantiate this efficiently is to use honest-verifier zero-knowledge proofs of knowledge made non-interactive via the Fiat-Shamir transform to obtain a group signature scheme secure in the random oracle model.

Today, pairing-based group signature schemes (PB-GSSs) [8, 10, 11, 14, 23, 26, 30, 41] are prevalent, since they are far more efficient regarding bandwidth and computational efficiency than earlier constructions relying on the strong RSA assumption [2, 3] especially for higher security levels. Essentially, most of the PB-GSSs are variations of the BBS scheme [10] or the BBS$^+$ scheme [5], secure under the SDH (or a slightly modified SDH$^+$) assumption but could also be built from CL signatures [14] relying on the LRSW assumption. The core schemes (without the revocation feature) thereby essentially differ in the use of various IND-CPA or IND-CCA secure public key encryption schemes that support efficient zero-knowledge proofs of knowledge about encrypted plaintexts (generally denoted as verifiable encryption [15]) and are secure in the random oracle model (ROM). With the exception of [8], which is based on [14] and requires no encryption but a testing approach linear in the size of the group members to open signatures, all existing schemes follow the SEP paradigm. For the sake of completeness, we also want to mention that besides the inefficient generic constructions [6, 7] relying on general non-interactive zero-knowledge proofs, there are also more efficient standard-model constructions. For instance, both constructions in [12, 28] make use of the Groth Sahai proof system [29].

Over the years, various approaches to enhanced anonymity management mechanisms for group signatures extending the standard opening feature of group signatures have been proposed. For instance, in various applications such as Direct Anonymous Attestation (DAA), anonymous credentials, offline anonymous e-cash, or e-voting it may be desirable to be able (under certain circumstances) to link different signatures of the same anonymous signer. Clearly, a naive approach to realize such a feature is to always involve the group manager, who can, given two signatures, open both signatures and decide whether they have been produced by the same signer. However, involving the group manager for such tasks may not be desirable and it is interesting to design extensions to schemes that allow to answer such questions without the group manager, either publicly or by means of another dedicated but less powerful entity, *i.e.*, an entity which does not need to be in possession of the secret key of the group manager. For instance, it may be desirable to either support linkability across different groups in the context of multi group signatures [4] or to publicly link signatures of users without identifying them [38] or even to allow public tracing for signers that have produced a number of signatures above a certain threshold $k \geq 2$ [47].

Besides these authority-free linking approaches, there also exist schemes, and in particular [30, 31], supporting so-called *controllable linkability*. This means that there are designated linking authorities (LAs), *i.e.*, parties in possession of a so-called *linking key*, who are able to link two signatures by means of this key, but no verifier is able to do so. A LA thereby can *only* decide whether two given signatures have been issued by the same *unknown* signer, *i.e.*, signers stay anonymous. Hwang *et al.* [30, 31] argued that the concept of controllable linkability can be useful in vehicular adhoc networks (VANETs), for instance, to prevent Sybil attacks. Furthermore, they argue that this concept can be beneficial in the context of data mining, e.g., service providers might want to establish statistics regarding buying patterns, while still preserving the customer's privacy. The concept of controllable linkability can also be useful in the context of "smart cities". For instance, public transport systems can support anonymous traveling, e.g., a valid group signature represents showing of a valid ticket. However, service providers might also be interested in analyzing traveling patterns, *i.e.*, some kind of flow control analysis. Thereby, the concept of controllable linkability allows the service provider to efficiently link signatures (ticket showings), while the customer of the public transport system still remains anonymous.

Traceable signatures [34], on the other hand, pursue to add selective traceability features to group signatures, *i.e.*, they allow an authority to compute a tracing trapdoor for every user such that only signatures produced by this user can be linked using the tracing trapdoor, but all other signatures from remaining users stay unlinkable. Furthermore, traceable signatures provide signature claiming, *i.e.*, a group member can prove that a signature has indeed been produced by her. We also note that selective tracing may also be realized by "borrowing" the verifier local revocation (VLR) feature of

VLR group signatures [11] and by distributing the revocation token of a user (that is originally used in their revocation list) as the tracing trapdoor. We provide a more detailed discussion of all these linking approaches and how they relate to our approach in Section 4.

Somewhat orthogonal to traceable signatures is the concept of group signatures with message-dependent opening [42]. Such an approach aims to restrict the power of the opener by introducing another authority (the admitter). Thereby, the admitter can issue a token which corresponds to a particular message and by using this token, the opener can extract the identity of the signer from a signature for this message, while without the token, he is not able to do so.

We note that the concept of linking is also studied in the context of ring signatures [36], *i.e.*, group signatures without a setup (ad-hoc groups) and without the feature of opening signatures (typically ring signatures provide unconditional anonymity). However, as our focus is on classical group signatures we do not investigate linkable ring signatures in this paper. In addition, their linking is public and not conducted by some dedicated authority.

Contribution. In this paper we show how PB-GSSs based on the SEP paradigm can be generically transformed in order to support controllable linkability, *i.e.*, we propose a transformation that turns any PB-GSS into a PB-GSS with controllable linkability. Basically, this transformation replaces the public key encryption scheme used for the identity escrow within a group signature scheme with a modified *all-or-nothing public key encryption with equality tests* scheme (denoted AoN-PKEET*) instantiated from the respective public key encryption scheme. Thereby, the LA receives a trapdoor and can perform trapdoor-equality tests on the ciphertexts (being part of the group signature) without learning the respective plaintexts and thus is able to link signatures without being able to identify the signers.

The techniques which are the basis for our generic approach to controllable linkability have been inspired by the works of Hwang *et al.* [30, 31] (which have been standardized in an ISO standard [32]), who realize controllable linkability for their variation of the BBS$^+$ GSS. However, firstly our generic construction is different from their proposed construction (cf. Section 3.3). Secondly, their approach is tailored towards their variation of the BBS$^+$ GSS, meaning that they neither make their design intuition explicit nor do they rely on a general building block. In contrast, our approach relies on the general building block of the newly introduced AoN-PKEET* to abstract away from a particular group signature scheme. Furthermore, our proposed approach is more efficient than the ones presented in [30, 31]. In particular, compared to the existing BBS$^+$ based instantiations of Hwang *et al.* [30, 31] we no longer have to include two ciphertexts into a group signature and also do not require their additional zero-knowledge proof of knowledge that the message contained in the second ciphertext is consistent with the specific part of the membership certificate that is used for linking.

The appealing benefit of our approach is that controllable linkability can be added to PB-GSSs based on the SEP paradigm *for free*, *i.e.*, it neither influences the signature size nor the computational costs for signers and verifiers. Therefore, we present and formalize this generic transformation based on AoN-PKEET* that can be used to turn any PB-GSS following the SEP paradigm into a GSS with controllable linkability. The used mechanisms, *i.e.*, the AoN-PKEET*, may also be of independent interest for other applications.

In comparison to other approaches such as traceable signatures, which have a different goal but may be "casted" to be used to achieve controllable linkability, our generic approach is more efficient and simplistic when only requiring controllable linkability and no other feature of traceable signatures. Finally, we note that our transformation has no influence whatsoever on the used revocation mechanism of the group signature scheme.

Outline. The remainder of this paper is organized as follows. We outline the models for dynamic GSSs, GSSs with controllable linkability, GSSs following the SEP as well as additional preliminaries in Section 2. Subsequently, in Section 3 we present the modified all-or-nothing public key encryption with equality tests primitive, the transformation of PB-GSSs into schemes with controllable linkability

and we also investigate the required security properties. In Section 4 we compare the efficiency of our transformation to controllable linkability with related approaches such as traceable signatures when used to achieve controllable linkability. Finally, we discuss potential future work in Section 5.

2 Background

In this Section, we present the dynamic model of GSSs by Bellare, Shi, and Zhang (BSZ) [7]. Then, we present an extension to the BSZ model (due to Hwang *et al.* [30]) in order to add controllable linkability to GSSs. For the sake of completeness, we also present the generic construction of GSSs based on the SEP paradigm.

2.1 Dynamic GSSs

The BSZ model [7] defines two different authorities: (1) an *opening authority* who can open signatures, and (2) an *issuing authority* who is capable of issuing signing keys to group members. We denote the keys of these authorities as master opening key (mok), and master issuing key (mik), respectively. Other involved parties are group members and verifiers.

GSSs usually refer to the group manager as an authority in charge of issuing new certificates and opening signatures. Due to the fact that the secret key for issuing certificates (mik) and the secret key for opening signatures (mok) are separated, the group manager is just a logical authority whose responsibilities might be split across two physical authorities, e.g., the *issuing authority* and the *opening authority*.

A dynamic GSS is a tuple $\mathcal{GS} = (\mathsf{GkGen}, \mathsf{UkGen}, \mathsf{Join}, \mathsf{Issue}, \mathsf{GSig}, \mathsf{GVf}, \mathsf{Open}, \mathsf{Judge})$ of polynomial time algorithms which are defined as follows [7]:

$\mathsf{GkGen}(\lambda)$**:** On input a security parameter $\lambda \in \mathbb{N}$, the algorithm generates the public parameters and outputs a tuple (gpk, mok, mik), representing the group public key, the master opening key and the master issuing key.

$\mathsf{UkGen}(\lambda)$**:** On input a security parameter $\lambda \in \mathbb{N}$, the algorithm generates a key pair ($\mathsf{usk_i}$, $\mathsf{upk_i}$).

$\mathsf{Join}(\mathsf{usk_i}, \mathsf{upk_i})$**:** On input the user's key pair ($\mathsf{usk_i}, \mathsf{upk_i}$), the algorithm interacts with the Issue algorithm and outputs the group signing key $\mathsf{gsk_i}$ of user i.

$\mathsf{Issue}(\mathsf{gpk}, \mathsf{mik})$**:** On input of the group public key gpk, and the master issuing key mik, the algorithm interacts with the Join algorithm in order to add user i to the group by adding an entry for user i to the registration table $\mathbf{reg}$.

$\mathsf{GSig}(\mathsf{gpk}, M, \mathsf{gsk_i})$**:** On input of the group public key gpk, a message $M \in \{0,1\}^*$, and a user's secret key $\mathsf{gsk_i}$, the algorithm outputs a group signature σ.

$\mathsf{GVf}(\mathsf{gpk}, M, \sigma)$**:** On input of the group public key gpk, a message M and signature σ, the algorithm verifies whether the signature σ is valid with respect to the message M and the group public key gpk and outputs `true` if the verification succeeds and `false` otherwise.

$\mathsf{Open}(\mathsf{gpk}, \mathbf{reg}, M, \sigma, \mathsf{mok})$**:** On input of the group public key gpk, a message M with valid signature σ, and the master opening key mok, the algorithm accesses the registration table $\mathbf{reg}$ to find the unique ID of the corresponding signer of σ. If a signer has been identified, the algorithm returns the ID of the signer and a publicly verifiable proof τ for the corresponding claim. Otherwise the algorithm claims that no group member produced σ.

$\mathsf{Judge}(\mathsf{gpk}, M, \sigma, ID, \mathsf{upk}, \tau)$**:** On input of the group public key gpk, the signature σ for message M, the claimed ID of the corresponding signer of σ and the corresponding public key upk as well as a proof τ that the claimed ID is indeed the signer of σ, the algorithm determines whether the proof τ holds and outputs `true` if it holds and `false` otherwise.

Note that some models consider the algorithms UkGen and Join as one algorithm, *i.e.*, the Join algorithm is used to generate the whole private key $\mathsf{gsk_i}$ composed of the part only known to the user and the part known to the group manager.

Security Properties for GSSs. In the BSZ model, for a GSS to be secure it needs to satisfy the following properties (for a formal description we refer the reader to [7]):

- **Correctness**: Signatures generated by honest group members should be valid and the Open algorithm should correctly identify the signer. Furthermore, the proof returned by the Open algorithm should be accepted by the Judge algorithm.
- **Anonymity**: The identity of the signer can only be determined by the authority in possession of the master opening key mok.
- **Traceability**: An adversary should not be able to produce a signature that either cannot be opened by the master opening authority, or the opening authority identifies the signer but cannot generate a correct proof for this claim.
- **Non-frameability**: No entity should be able to produce a correct proof that an honest user generated a signature unless this entity indeed produced this signature.

2.2 GSSs with Controllable Linkability

For *GSSs with controllable linkability* we use the model introduced by Hwang *et al.* [30, 31] that extends [7]. The model requires an additional authority, namely a *linking authority* (LA) who is capable of linking signatures. The corresponding private key of this linking authority is denoted as *master linking key* (mlk). A GSS with controllable linkability is specified as a tuple $\mathcal{GS} = (\mathsf{GkGen}, \mathsf{UkGen}, \mathsf{Join}, \mathsf{Issue}, \mathsf{GSig}, \mathsf{GVf}, \mathsf{Open}, \mathsf{Judge}, \mathsf{Link})$. Subsequently, we present the algorithms that change due to the modifications as well as the additional Link algorithm.

$\mathsf{GkGen}(\lambda)$**:** On input a security parameter $\lambda \in \mathbb{N}$, the algorithm generates the public parameters and outputs a tuple (gpk, mok, mik, mlk), representing the group public key, the master opening key, the master issuing key, and the master linking key.

$\mathsf{Link}(\mathsf{gpk}, M, \sigma, M', \sigma', \mathsf{mlk})$**:** On input of the group public key gpk, a pair of tuples (M, σ) and (M', σ'), as well as the master linking key mlk, the algorithm first verifies both signatures by calling $\mathsf{GVf}(\mathsf{gpk}, M, \sigma)$ and $\mathsf{GVf}(\mathsf{gpk}, M', \sigma')$. If both signatures are valid for messages M and M' under the group public key gpk, the algorithm uses mlk to determine whether σ and σ' have been produced by the same *unknown* signer. If both signatures are valid and can be linked to the same *unknown* signer, the algorithm returns `true` and `false` otherwise.

Modified and Additional Properties for GSSs with Controllable Linkability.

- **Correctness**: Signatures generated by honest group members should be valid, the Open algorithm should correctly identify the signer, and the proof returned by the Open algorithm should be accepted by the Judge algorithm. Furthermore, the Link algorithm should correctly link two signatures from the same *unknown* signer.
- **Linkability**: The authority in possession of the master linking key mlk should neither be able to gain any useful information for opening a signature nor for generating a Judge proof τ. Furthermore, colluding parties—including users, the linking authority, and/or the opening authority—should not be able to generate pairs of messages and signatures (M, σ) and (M', σ') that violate the correctness property mentioned above.

We provide a more detailed description of security aspects in Section 3.4 and the formal model for controllable linkability in Appendix A.

2.3 Sign-and-Encrypt-and-Proof Paradigm in the ROM

We use the representation of [37] for illustration. GSSs based on the SEP paradigm consist of a secure signature scheme $\mathcal{DS} = (\mathsf{KeyGen_s}, \mathsf{Sign}, \mathsf{Vrfy})$, and an (at least IND-CPA secure) public key encryption

scheme $\mathcal{AE} = (\mathsf{KeyGen_e}, \mathsf{Enc}, \mathsf{Dec})$. Additionally, we require zero-knowledge proofs of knowledge (PKs) which are converted to signatures of knowledge (SPKs) using the Fiat-Shamir transform in the ROM. Furthermore, let $f(\cdot)$ be a one-way function.

The group public key gpk includes a public encryption key pk_e, and a signature verification key pk_s. The master opening key mok is the decryption key sk_e, and the master issuing key mik is the signing key sk_s. During the execution of the Join protocol a user i generates a secret x_i and sends $f(x_i)$ to the issuer. The issuer in turn returns a signature $\mathsf{cert} \leftarrow \mathsf{Sign}(\mathsf{sk}_s, f(x_i))$ by signing $f(x_i)$ with the signing key sk_s.

A group signature $\sigma = (T, \pi)$ for a message M is computed as follows: Compute a ciphertext $T \leftarrow \mathsf{Enc}(\mathsf{pk}_e, X_i)$ and

$$\pi = \mathsf{SPK}\{(x_i, \mathsf{cert}) : \mathsf{cert} = \mathsf{Sign}(\mathsf{sk}_s, f(x_i)) \quad \wedge \quad T = \mathsf{Enc}(\mathsf{pk}_e, X_i)\}(M)$$

where X_i can either be $g(x_i)$ for some one-way function $g(\cdot)$ or cert. We note that the membership certificate typically refers to the issuer's signature (cert) but might also be a commitment to a user's secret that has been signed by the issuer. Nevertheless, we always refer to T as an encryption of the membership certificate.

2.4 Mathematical Preliminaries

Let $\mathbb{G}_1 = \langle g_1 \rangle$, $\mathbb{G}_2 = \langle g_2 \rangle$, and $\mathbb{G}_T$ be cyclic groups of prime order p. A bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is a map, such that $e(u^a, v^b) = e(u, v)^{ab}$ for all $u \in \mathbb{G}_1$, $v \in \mathbb{G}_2$, and $a, b \in \mathbb{Z}_p^*$, it holds that $e(g_1, g_2) \neq 1$ and e is efficiently computable.

Usually, $\mathbb{G}_1$ and $\mathbb{G}_2$ are subgroups of points on elliptic curves defined over a finite field $\mathbb{F}_q$ and an extension field $\mathbb{F}_{q^k}$, respectively. $\mathbb{G}_T$ usually represents a subgroup of the multiplicative finite field $\mathbb{F}_{q^k}^*$. The parameter q denotes the size of the underlying field and k denotes the embedding degree such that $n | q^k - 1$, with k minimal. If $\mathbb{G}_1 = \mathbb{G}_2$, then e is called *symmetric* (Type 1) and *asymmetric* (Type 2 or Type 3) otherwise. For Type 2 pairings there is an efficiently computable isomorphism $\psi : \mathbb{G}_2 \rightarrow \mathbb{G}_1$ (XDH holds in $\mathbb{G}_1$), whereas for Type 3 pairings no such efficient isomorphism is known (SXDH holds). Furthermore, a function $\epsilon : \mathbb{N} \rightarrow \mathbb{R}^+$ is called *negligible* if for all $c > 0$ there is a k_0 such that $\epsilon(k) < 1/k^c$ for all $k > k_0$. In the remainder of this paper, we use ϵ to denote such a negligible function.

For the sake of completeness, we (semiformally) state the following cryptographic assumptions which are referred to throughout this paper.

Computational Diffie-Hellman Assumption (CDH). Let $\mathbb{G}$ be a cyclic group of order p and g a generator. The CDH assumption states that given a tuple (g, g^a, g^b) it is hard to compute g^{ab}.

Computational co-Diffie-Hellman Assumption (co-CDH). Let $\mathbb{G}_1$, and $\mathbb{G}_2$ be two distinct cyclic groups and g_1 and g_2 their respective generators. The co-CDH assumption states that given a tuple (g_1, g_1^a, g_2) it is hard to compute g_2^a.

Decisional Diffie-Hellman Assumption (DDH). Let $\mathbb{G}$ be a cyclic group of prime order p and g a generator. The DDH assumption states that given (g, g^a, g^b, g^c) it is hard to decide whether $g^{ab} = g^c$.

q-Strong Diffie-Hellman Assumption (q-SDH) [10]. Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two cyclic groups of prime order p with generators g_1 and g_2. The q-SDH assumption states that given a $(q+2)$-tuple $(g_1, g_2, g_2^{\gamma}, g_2^{\gamma^2}, \ldots, g_2^{\gamma^q})$ it is hard to compute $(g_1^{\frac{1}{\gamma+x}}, x)$ with $x \in \mathbb{Z}_p^*$.

Modified q-Strong Diffie-Hellman Assumption (q-SDH$^+$) [30]. Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two cyclic groups of prime order p with generators g_1, h_1, k_1 and g_2 respectively. The q-SDH$^+$ assumption states that given a tuple $(g_1, h_1, k_1, g_1^{\gamma}, \ldots, g_1^{\gamma^q}, g_2, g_2^{\gamma})$ it is hard to compute $((g_1 h_1^y k_1^z)^{\frac{1}{\gamma+x}}, x, y, z)$ with $x, y, z \in \mathbb{Z}_p^*$.

eXternal Diffie-Hellman Assumption (XDH). Let $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ be three cyclic groups of prime order p and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ a pairing. Then, the XDH assumption states that the DDH assumption holds in $\mathbb{G}_1$.

Symmetric eXternal Diffie Hellman Assumption (SXDH). Let $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ be three distinct cyclic groups of prime order p and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a pairing. Then, the SXDH assumption states that in both groups $\mathbb{G}_1$ and $\mathbb{G}_2$ the DDH assumption holds.

Decision LINear Problem (DLIN) [10]. Let $\mathbb{G}_1$ be a cyclic group of prime order p and $u, v, h \in \mathbb{G}_1$ be three generators. The DLIN assumption states that given (u, v, h, u^a, v^b, h^c) it is hard to decide whether $a + b = c$.

3 Adding Controllable Linkability to GSSs Generically

In all PB-GSSs following the SEP paradigm, the used encryption scheme depends on the respective bilinear map setting, *i.e.*, types of used pairings, as well as whether the construction targets to achieve weak or full anonymity. The former issue is concerned with the DDH problem in the used groups. If one assumes the DDH problem to be easy in $\mathbb{G}_1$ and $\mathbb{G}_2$, then one usually relies on linear encryption variants of ElGamal encryption [10, 31] which are IND-CPA secure under the DLIN (or some related) assumption. However, if one assumes the XDH assumption to hold for the respective group used for encryption, *i.e.*, DDH is assumed to be hard in this group, then one can use standard IND-CPA secure ElGamal encryption. Weak anonymity means that anonymity is guaranteed as long as the adversary does not have access to an Open oracle and the use of IND-CPA secure encryption schemes yields weak anonymous group signatures (CPA-full-anonymity). For full anonymity (CCA-full-anonymity), *i.e.*, anonymity that holds even if the adversary has access to an Open oracle, one needs to rely on IND-CCA secure encryption schemes. For instance, [23] tweak the weak anonymous BBS variant with strong exculpability (non-frameability) [10], by replacing linear ElGamal with standard IND-CPA secure ElGamal (relying on the XDH assumption) and turn IND-CPA secure ElGamal into an IND-CCA secure encryption scheme using the Naor-Yung paradigm (double encryption) [40] in the random oracle model to achieve full anonymity. The construction in [14] on the other hand uses the IND-CCA secure ElGamal variant due to Cramer and Shoup [22] to obtain full anonymity.

3.1 Trapdoor Equality Test for Public-Key Encryption

At the heart of our controllable linkability is a means to extend IND-CPA/IND-CCA secure public key encryption schemes with a feature that allows a designated party (holding a trapdoor) to check whether two ciphertexts under the same public key contain the same message, but without being able to decrypt the ciphertexts, *i.e.*, still providing one-wayness (OW). A naive approach would be to give away the private key itself as a trapdoor and for two given ciphertexts one could simply decrypt and compare the two plaintexts. However, this would allow to fully recover the messages and this is a "feature" we do not want to have as it would give too much power to the linking authority.

Our idea is related to the concept of probabilistic public key encryption with equality tests (PKEET) [48], but differs in that their equality tests are *public* (not only feasible for one holding a trapdoor) and need to work for ciphertexts under *different* public keys. Consequently, their constructions can no longer satisfy any meaningful notion of indistinguishability. However, in our approach indistinguishability supported by the respective encryption scheme still needs to hold for all parties except the one holding the trapdoor, who clearly can test against any potential message. However, if the messages are not guessable (the message space has large enough entropy) they can still be hidden from the party holding the trapdoor, *i.e.*, provide OW-CPA security.

In [45], an authorization mechanism for users to specify who can perform a plaintext equality test has been added to PKEET, denoted as all-or-nothing PKEET (AoN-PKEET). This idea is related to our idea, but their focus is on allowing a semi-trusted proxy to compare ciphertexts for two distinct

parties by obtaining a trapdoor from each party. This approach also works if the two users are identical and then only one trapdoor is required. However, firstly they target applications for searchable encryption and consequently their Type-I adversary (representing the proxy holding the trapdoor(s)) is very powerful, *i.e.*, they require OW-CCA security, and also for an outsider who does not know the trapdoor(s) (Type-II adversary), they always require IND-CCA security. Secondly, due to their focus on comparison of ciphertexts from distinct users, their construction is quite involved and requires a quite costly variant of double encryption for each user. Besides inefficiency, the most important difference between the AoN-PKEET construction and our approach is that we need compatibility with efficient proofs of knowledge of encrypted messages, which are not possible in [45] and follow up work [44] as all these instantiations involve encrypting hashes of the messages. Note that this renders the approaches in [44, 45] not applicable to our setting as it cannot be efficiently proven that the correct hash of the message (membership certificate) has been included by the user and thus identity escrow is not that efficiently possible [33] to be of practical relevance for group signature schemes.

3.2 Modified All-Or-Nothing PKEET (AoN-PKEET*)

In brief, our approach may be considered as a restricted version of AoN-PKEET, where we only allow comparison of ciphertexts of the same user, *i.e.*, under the same public key. Furthermore, against a Type-I adversary (the trapdoor holder) we do not require OW-CCA but OW-CPA security[1] and against Type-II adversaries (outsiders) either IND-CPA or IND-CCA security (depending on the underlying public key encryption scheme). Additionally, we require that there are efficient (honest-verifier) zero-knowledge proofs of knowledge about messages hidden in ciphertexts, which is clearly true for all encryption schemes used for the SEP approach with PB-GSSs. As already mentioned, AoN-PKEET constructions [45] and follow up work [44] do not allow such efficient proofs of knowledge of encrypted messages, and thus renders these approaches not applicable to our setting. We denote such a modified scheme as an AoN-PKEET* scheme subsequently and first present the formal model and then discuss instantiations.

Formal Model. An AoN-PKEET* scheme $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Aut}, \mathsf{Com})$ is a conventional public key encryption scheme $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ augmented by two polynomial time algorithms $(\mathsf{Aut}, \mathsf{Com})$, which are defined as follows, and where $(\mathsf{sk}, \mathsf{pk})$ have been generated by KeyGen:

$\mathsf{Aut}(\mathsf{sk})$**:** The authorization algorithm Aut takes a private key sk and outputs a trapdoor tk.

$\mathsf{Com}(c, c', \mathsf{tk})$**:** The comparison algorithm takes two ciphertexts c and c' (of two messages m and m') produced under pk, and a trapdoor tk produced with the corresponding sk, and outputs `true` if $m = m'$ and `false` otherwise.

Security Definition. For our modified setting, the *soundness* definition of [45] reduces to the fact that besides correctness of the public key encryption scheme, we have that for all $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\lambda)$ we require that $\mathsf{Com}(\mathsf{Enc}(\mathsf{pk}, m), \mathsf{Enc}(\mathsf{pk}, m'), \mathsf{Aut}(\mathsf{sk})) = \mathtt{true}$ if and only if $m = m'$. Against a Type-I adversary, we require OW-CPA security, which is defined as follows.

Definition 1 (AoN-PKEET* OW-CPA) *For all PPT adversaries $\mathcal{A}$ and security parameters λ there is a negligible function ϵ such that:*

$$\mathsf{Pr}\left[\begin{array}{r}(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\lambda), \mathsf{tk} \leftarrow \mathsf{Aut}(\mathsf{pk}), m \xleftarrow{R} \{0,1\}^{\lambda}, \\ m^* \leftarrow \mathcal{A}(\mathsf{pk}, \mathsf{tk}, \mathsf{Enc}(\mathsf{pk}, m)) : m^* = m\end{array}\right] \leq \epsilon(\lambda).$$

[1] Assuming that the opener cannot be used as a decryption oracle is reasonable, however, we may also extend the approach to OW-CCA security.

Against a Type-II adversary, we require IND-CPA security or IND-CCA security (depending on the used public key encryption scheme) as it is defined for conventional public key encryption schemes.

Definition 2 *An AoN-PKEET* scheme is called secure if it is sound, provides OW-CPA security against Type-I adversaries and provides the respective IND-CPA/IND-CCA security provided by* $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *against Type-II adversaries.*

Constructions. Subsequently, we elaborate how an AoN-PKEET* scheme can be instantiated with various ElGamal type public key encryption schemes.

ElGamal Encryption. We consider ElGamal encryption in $\mathbb{G}_1$ in a bilinear map setting $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ such that the XDH assumption holds for $\mathbb{G}_1$. Let the private key be $\xi \in \mathbb{Z}_p^*$ and the public key be $h = g^\xi \in \mathbb{G}_1$ and a ciphertext for message $m \in \mathbb{G}_1$ be $(T_1, T_2) = (g^\alpha, mh^\alpha) \in \mathbb{G}_1^2$ for random $\alpha \in \mathbb{Z}_p^*$. Decryption works in $\mathbb{G}_1$ by computing $m = T_2/(T_1^\xi)$. Now, we describe Aut and Com.

Aut : Given ξ, the trapdoor is computed as $\mathsf{tk} = (r, t = r^\xi)$ for a random $r \in \mathbb{G}_2$.

Com : Given two ciphertexts $(T_1, T_2) = (g^\alpha, mh^\alpha)$ and $(T_1', T_2') = (g^{\alpha'}, m'h^{\alpha'})$ and trapdoor $\mathsf{tk} = (r, t = r^\xi)$ check:

$$\frac{e(T_2, r)}{e(T_1, t)} \stackrel{?}{=} \frac{e(T_2', r)}{e(T_1', t)}.$$

If the check holds, *i.e.*, we have $e(m, r) = e(m', r)$, then return `true` and `false` otherwise.

Linear Encryption. We consider linear encryption [10] in $\mathbb{G}_1$ in a bilinear map setting $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ such that the DDH may be easy in $\mathbb{G}_1$ as well as $\mathbb{G}_2$ but the DLIN assumption holds. Let the private key be $(\xi, \mu) \in (\mathbb{Z}_p^*)^2$ and the public key be $(u, v, h) \in \mathbb{G}_1^3$ such that $u^\xi = v^\mu = h$, *i.e.*, choose random h and compute $u \leftarrow h^{1/\xi}$ and $v \leftarrow h^{1/\mu}$ for random $\xi, \mu \in \mathbb{Z}_p^*$. A ciphertext for a message $m \in \mathbb{G}_1$ is $(T_1, T_2, T_3) = (u^\alpha, v^\beta, mh^{\alpha+\beta}) \in G_1^3$ for random $\alpha, \beta \in \mathbb{Z}_p^*$. Decryption works in $\mathbb{G}_1$ by computing $m = T_3/(T_1^\xi T_2^\mu)$. Now, we describe Aut and Com.

Aut : Given (ξ, μ) the trapdoor is computed as $\mathsf{tk} = (r, s = r^\xi, t = r^\mu)$ for a random $r \in \mathbb{G}_2$.

Com : Given two ciphertexts $(T_1, T_2, T_3) = (u^\alpha, v^\beta, mh^{\alpha+\beta})$ and $(T_1', T_2', T_3') = (u^{\alpha'}, v^{\beta'}, m'h^{\alpha'+\beta'})$ and trapdoor $\mathsf{tk} = (r, s = r^\xi, t = r^\mu)$ check:

$$\frac{e(T_3, r)}{e(T_1, s)e(T_2, t)} \stackrel{?}{=} \frac{e(T_3', r)}{e(T_1', s)e(T_2', t)}.$$

If the check holds, *i.e.*, we have $e(m, r) = e(m', r)$, then return `true` and `false` otherwise.

Security of the Constructions. We exemplarily analyze AoN-PKEET* instantiated with ElGamal encryption under XDH and with linear encryption (without the XDH assumption) below. Consequently, we rely on the use of Type 2 parings, *i.e.*, $\mathbb{G}_1 \neq \mathbb{G}_2$ with efficient $\psi : \mathbb{G}_2 \to \mathbb{G}_1$, and the Type 1 setting, *i.e.*, $\mathbb{G}_1 = \mathbb{G}_2$, respectively.

Lemma 1. *Under the co-CDH assumption AoN-PKEET* based on ElGamal in* $\mathbb{G}_1$ *in an XDH setting is secure.*

Lemma 2. *Under the CDH assumption in* $\mathbb{G}_1$ *AoN-PKEET* based on linear encryption in* $\mathbb{G}_1$ *is secure.*

Now we provide proofs for Lemma 1 and Lemma 2.

Proof (Lemma 1). It is obvious, that AoN-PKEET* based on ElGamal satisfies the soundness and is IND-CPA secure against Type-II adversaries. It remains to argue about the OW-CPA security against Type-I adversaries. We show how an OW-CPA adversary $\mathcal{A}$ yields an adversary $\mathcal{B}$ against the co-CDHP. Let (g_1^a, g_2^b) be a co-CDHP instance given to $\mathcal{B}$. $\mathcal{B}$ sets $\mathsf{pk} \leftarrow \psi(g_2^b)$ and $\mathsf{tk} \leftarrow (g_2^w, (g_2^b)^w)$ for random $w \in \mathbb{Z}_p^*$, chooses a random $h \in \mathbb{G}_1$ and runs $\mathcal{A}$ on pk and $c \leftarrow (g_1^a, h)$. If $\mathcal{A}$ manages to output $m^* = h/g_1^{ab}$, then $\mathcal{B}$ outputs $h/m^* = g_1^{ab}$ which is a valid solution to the co-CDHP instance. □

Note, that under the SXDH assumption, e.g., a Type 3 setting, where we do not have an efficiently computable isomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$, the reduction works analogously but to the co-CDH* problem [18]. Additionally, we have to take a look at the value $\mathsf{t} = e(m, r)$ with $r = g_2^w$ available to $\mathcal{A}$. Note that the problem given the value $\mathsf{t} \in \mathbb{G}_T$ and $r \in \mathbb{G}_2$ to recover $m \in \mathbb{G}_1$ is the fixed argument pairing inversion 1 (FAPI-1) problem and a solver for FAPI-1 implies a solver for the CDHP [27] (and the co-DHP in the above ElGamal setting).

Proof (Lemma 2). It is obvious, that AoN-PKEET* based on Linear Encryption satisfies the soundness and is IND-CPA secure against Type-II adversaries. It remains to argue about the OW-CPA security against Type-I adversaries. We show how an OW-CPA adversary $\mathcal{A}$ yields an adversary $\mathcal{B}$ against the CDHP in $\mathbb{G}_1$. Let (g^a, g^b) be a CDHP instance given to $\mathcal{B}$. Observe, that unlike the ElGamal case, here $\mathcal{A}$ can verify whether $\mathsf{pk} = (u, v, h)$ and $\mathsf{tk} = (r, r^\xi, r^\mu)$ are consistent, *i.e.*, by checking whether $e(u, r^\xi) = e(v, r^\mu) = e(h, r)$ holds. Consequently, we need to prepare the keys pk and tk more carefully in our reduction. In particular, $\mathcal{B}$ randomly chooses $x, y, z \in \mathbb{Z}_p^*$ and sets $\mathsf{pk} = (u, v, h) \leftarrow (g^x, g^y, g^b)$ as well as $\mathsf{tk} \leftarrow (g^z, h^{z1/x}, h^{z1/y})$, chooses a random $k \in \mathbb{G}_1$ and runs $\mathcal{A}$ on pk and $c \leftarrow ((g^a)^x, v^\beta, k)$ for random $\beta \in \mathbb{Z}_p^*$. Note that for $\mathcal{A}$ this simulation is indistinguishable from the real game. If $\mathcal{A}$ manages to output $m^* = k/h^{\alpha+\beta}$, then $\mathcal{B}$ outputs $(k/m^*)/(h^\beta) = h^\alpha = g^{ab}$ which is a valid solution to the CDHP instance. □

For the sake of completeness, we also mention other schemes that may be used within AoN-PKEET* schemes below.

Double ElGamal Encryption: Double ElGamal encryption as used in [23, 41] uses the Naor-Yung paradigm [40] to transform IND-CPA secure ElGamal into an IND-CCA2 secure variant in the random oracle model. The idea is to encrypt a message m twice under two independent public keys $(h_1 = g^{\xi_1})$ and $(h_2 = g^{\xi_2})$ using independent random coins α and and β to obtain $(T_1, T_2) = (g^\alpha, mh_1^\alpha)$ and $(T_3, T_4) = (g^\beta, mh_2^\beta)$. Then, one additionally computes a non-interactive zero-knowledge proof of knowledge (via Fiat-Shamir) that the two ciphertexts (T_1, T_2) and (T_3, T_4) contain the same message m, which can be realized by providing the following signature of knowledge (SoK) of the values (α, β):

$$\mathsf{SoK}\{(\alpha, \beta) : T_1 = g^\alpha \wedge T_3 = g^\beta \wedge T_2/T_4 = h_1^\alpha/h_2^\beta\}.$$

Observe, that as in standard ElGamal discussed above, when used in a bilinear map setting (and assuming XDH as done in [23]), our proposed trapdoor equality test works analogously to standard ElGamal and the trapdoor can either be $\mathsf{tk} = (r, t = r^{\xi_1})$ or $\mathsf{tk} = (r, t = r^{\xi_2})$.

Double Linear Encryption: Note, that the same approach used to turn standard ElGamal into an IND-CCA2 secure variant in the random oracle model can also be applied to linear encryption (when one does not want to rely on the XDH assumption) and the trapdoor equality test works analogously.

Cramer-Shoup Encryption: We also note that our trapdoor equality test works analogously to standard ElGamal also for the IND-CCA2 secure encryption scheme from Cramer and Shoup [22], since the first and the third element of the ciphertext form a standard ElGamal ciphertext.

3.3 Transformation to PB-GSSs with Controllable Linkability

To generically add controllable linkability to PB-GSSs following the SEP paradigm, we replace the used public key encryption scheme with its AoN-PKEET* version. The required additional linking key mlk is the trapdoor tk computed by Aut and is given to the linking authority (LA). Consequently, for linking the linking authority is given two message-signature pairs (M, σ) and (M', σ') where the signatures $\sigma = (T, \pi)$ and $\sigma' = (T', \pi')$ contain ciphertexts T and T' as well as the non-interactive proofs π and π'. Then, by running $\mathsf{Com}(T, T', \mathsf{mlk})$, LA can decide whether the two signatures have been produced by the same *unknown* signer.

In order to convert a PB-GSS $\mathcal{GS} = (\mathsf{GkGen}, \mathsf{UkGen}, \mathsf{Join}, \mathsf{Issue}, \mathsf{GSig}, \mathsf{GVf}, \mathsf{Open}, \mathsf{Judge})$ following the SEP paradigm into a PB-GSS with controllable linkability, we have to add a linking authority LA as well as an additional algorithm Link (cf. Section 2.2). Below we present the required modifications:

$\mathsf{GkGen}(\lambda)$**:** On input a security parameter $\lambda \in \mathbb{N}$, the algorithm generates the public parameters and outputs a tuple $(\mathsf{gpk}, \mathsf{mok}, \mathsf{mik}, \mathsf{mlk})$, where mok is generated by running $\mathsf{KeyGen}(\lambda)$ of the AoN-PKEET* scheme (where pk is integrated into gpk) and the master linking key is computed as $\mathsf{mlk} = \mathsf{Aut}(\mathsf{mok})$. All the remaining steps remain unchanged.

$\mathsf{Link}(\mathsf{gpk}, M, \sigma, M', \sigma', \mathsf{mlk})$**:** On input of the group public key gpk, a pair of tuples (M, σ) and (M', σ'), as well as the master linking key mlk, the algorithm first verifies both signatures via the GVf algorithm. If at least one of these verifications fails, the algorithm outputs `false`. Otherwise, the algorithm extracts the ciphertexts T and T' from σ and σ' and runs $\mathsf{Com}(T, T', \mathsf{mlk})$ and outputs whatever Com outputs.

Nevertheless, irrespective of the actual value that is being encrypted we always refer to this as an encryption of the membership certificate.

We state the following abstract assumption and discuss it below:

Assumption 1 *Honestly computed membership certificates (used for identity escrow) of users are uniformly distributed over the respective group and are unknown to the linking authority*[2]*.*

Note that for all BBS variations the membership certificates are of the form $A = g_1^{\frac{1}{x_i + \mathsf{mik}}}$ where g_1 may be the product of other group elements representing a BB signature [9]. In [24] it is shown that this represents a verifiable random function (VRF) (for a superlogarithmic sized input). Basically, in this setting an adversary controlling the input should not be able to distinguish the output of the VRF from uniformly sampled strings of equal size. However, in our setting we do not want to realize a pseudo-random function, but we only require that A is uniformly distributed over the group for uniformly at random sampled values of messages (that are not known to the linking authority). In case of BB signatures A will be a random element of the group, if the unknown value of x_i is chosen uniformly at random from the integers in the order of the group.

We note that the CL group signature scheme in [14] uses Cramer-Shoup encryption in $\mathbb{G}_T$ as they defined it over Type 1 pairings, which would not work with our approach. However, it can easily be adapted to asymmetric pairings such that the encryption scheme no longer needs to work on elements in $\mathbb{G}_T$ and then our transformation applies. Anyways, in a CL signature the membership certificate to be encrypted is not the signature (as in BBS variants), but a commitment to a user's secret is encrypted. As this secret is chosen uniformly at random, our assumption clearly holds. The above stated assumption also seems to be reasonable for other constructions that one may envision, where the membership certificate is a signature for a uniformly at random chosen message.

[2] The latter case is just to rule out trivial cases where the membership certificate that is encrypted is a well known public key. Note that in such cases the user could always commit to his secret using a base different from that in the public key and proof the equality of the respective discrete logarithms during joining.

Relation to Hwang *et al.*: The controllable linkability feature of the group signature schemes in [30, 31] inspired us to our general transformation, but differs in the following issues. They use two ciphertexts (sharing the same randomness), one for opening and one for linking, and consequently an additional ciphertext in the actual group signature. Recall, that in the BBS scheme a membership certificate is of the form $A_i = g_1^{\frac{1}{x_i+\mathsf{mik}}}$ for x_i randomly chosen by the issuer and when requiring strong exculpability $A = (g_1 h^{-z_i})^{\frac{1}{x_i+\mathsf{mik}}}$ where z_i is randomly chosen by the user and given in form of a commitment h^{-z_i} to the issuer. In the construction in [30] Hwang *et al.* add an additional issuer-chosen element y_i and their membership certificates are of the form $A = (g_1 h^{-z_i} h'^{-y_i})^{\frac{1}{x_i+\mathsf{mik}}}$. For realizing controllable linkability, a user encrypts g^{y_i} and provides an additional non-interactive zero-knowledge proof of knowledge that the unrevealed value y_i in the second ciphertext is included in the membership certificate and linking basically represents a plaintext equality check based on ElGamal encryption (the construction in [31] works analogously but uses another encryption scheme).

Consequently, they apply their equality testing for linkability not directly on the membership certificate but on the second ciphertext to the message g^{y_i} and thus do not require Assumption 1 for their LO-linkability, but satisfy this by construction. The use of two ciphertexts and an additional non-interactive proof of knowledge, however, makes the group signature more expensive from a computational as well as bandwidth perspective. Although this may also be converted into a generic transformation, it seems more natural to us to use a single ciphertext (which is already available in the group signatures) as in our proposed generic transformation and also base the linking decision on the encryption of the membership certificate as this comes at no additional cost at all.

3.4 Security Analysis

A nice feature of this generic transformation is that the linking key mlk is fully independent from the issuing key mik which is required to issue membership certificates and thus does not constitute a danger in the sense that the LA may manage it to impersonate the issuer. Since mlk is only related to mok, *i.e.*, the public key encryption scheme used to encrypt a membership certificate cert, the only issue we have to cope with is that LA is not able to trace users besides checking whether signatures have been issued by the same unknown group member. Consequently, all proofs for security properties of the respective PB-GSS that are not related to the property of linkability remain valid as well as untouched by the generic transformation.

In GSSs with controllable linkability [30, 31] the related security issues are covered by *linkability*, which consists of three security notions, called LO-linkability (Link-Only linkability), JP-Unforgeability (Judge-Proof Unforgeability), and E-linkability (Enforced linkability). We briefly present the ideas behind these linkability experiments and the formal experiments are defined in Appendix A:

LO-linkability: LO-linkability captures that a linking key should be used only for linking signatures, not for gaining useful information for opening.

JP-Unforgeability: JP-Unforgeability captures that a linking key cannot be used for generating a Judge proof.

E-linkability: E-linkability captures that colluding users should not be able to generate two message-signature pairs satisfying any of the following two conditions (even with the help of authorities such as the linking authority or the opening authority): (1) Open yields identical identities which are successfully judged, while Link outputs `false` or (2) identities output by Open are different and both are successfully judged, while Link outputs `true`.

Our construction does not change any of the security arguments from [30, 31]. The argumentation in the proofs is nearly identical whereas we require a more abstract level of argumentation for our generic transformation. We prove the following lemmas in Appendix A and clearly assume that AoN-PKEET* is based on the encryption scheme used in the respective PB-GSS.

Lemma 3. *If AoN-PKEET* is secure and PB-GSS is secure and Assumption 1 holds, the PB-GSS with controllable linkability obtained from the PB-GSS provides LO-linkability.*

Lemma 4. *If AoN-PKEET* is secure and PB-GSS is secure, the PB-GSS with controllable linkability obtained from the PB-GSS provides JP-Unforgeability.*

Lemma 5. *If AoN-PKEET* is secure and PB-GSS is secure, the PB-GSS with controllable linkability obtained from the PB-GSS provides E-linkability.*

Since all other properties remain unaffected by the controllable linkability feature, from the above lemmas we obtain the following.

Theorem 1. *If AoN-PKEET* is secure, PB-GSS is secure and Assumption 1 holds, then the generic transformation yields a secure PB-GSS with controllable linkability.*

4 Comparison with Other Approaches

In this Section we discuss other existing methods regarding enhanced anonymity management mechanisms for group signatures with respect to their capability of realizing controllable linkability. However, since these concepts have been proposed for different purposes, these implementations either lead to inefficient and impractical instantiations or cannot be employed at all to achieve controllable linkability.

Linkable Group Signatures: *Tracing-by-linking* (TbL) group signatures are group signatures where instead of the SEP paradigm (tracing-by-escrowing) that allows the group manager to open all signatures, the signer's anonymity cannot be revoked by any combination of authorities. Only if a group member signs twice (or more general $k \geq 2$ times), then his identity can be traced by any member of the public without needing any trapdoor (cf. [46]). Another direction are *link-but-not-trace* group signatures [38], where signatures contain a tag such that double signing can be detected but without the help of an authority the signer cannot be identified. We note that these approaches cannot be compared to controllable linkability, since in these approaches linking is a public operation that can detect double signing by the same entity (or more general k-time signing) while controllable linkability requires that only a dedicated party can perform the linking operation.

Traceable Signatures: Traceable signatures [34] are group signatures extending the signature opening by the group manager by 1) *user tracing*, meaning that the group manager can publish a tracing trapdoor which allows to trace all signatures of the respective user, and 2) *signature claiming*, meaning that the producer of a signature can provably claim that the signature has been produced by her. Constructions have been proposed in the ROM [34] extending the strong RSA group signature of [2] or from pairings [20, 41] as well as in the standard model [35]. The idea to realize user tracing is that the group manager publishes a user specific trapdoor, and given a signature and the trapdoor everyone can check whether specific elements of the signature and the trapdoor satisfy a certain relation. Consequently, using this functionality to implement controllable linkability could be realized by giving a tracing trapdoor for every user of the group to a linking authority. Given two signatures, the linking authority can take every user trapdoor and check this relation for both signatures. If the relation is satisfied for the same trapdoor, the signatures are linked and are from different anonymous group members otherwise. Obviously, this requires computational costs linear in the number of group members for linking operations, which can soon become impractical for larger groups and additionally the group manager needs to communicate tracing trapdoors of newly joined users frequently (and in time) to the linking authority.

Chow [21] employs the idea of "join once, spend many" from a compact e-cash scheme [13] to develop an alternative approach of traceable signatures, denoted as real traceable signatures. He argues that all previous traceable signatures are not optimal since checking whether a signature has been issued by the

user of the corresponding tracing trapdoor requires additional computations by the party performing the check for every signature and results in a high volume of bandwidth and computations. The idea behind his alternative construction is that a tracing trapdoor allows to deterministically recompute every tag of a user's signature (essentially the tracing trapdoor is the seed for a verifiable random function [24]) and thus to trace. Consequently, the tracing authority can compute a list of tags and send them to every verifier who can then check if the tag of the signature can be found in the given list. However, the drawback of the approach in [21] is that only ℓ unlinkable signatures can be issued by every user in the system and every signature requires an additional zero-knowledge range proof per signature, *i.e.*, that the used value for the tag is less than ℓ, where ℓ is a fixed parameter in the system. Consequently, signatures get more expensive the larger ℓ is and so does the size of the list of tags per user. When used to realize controllable linkability, lists of tags for every member of the group may be precomputed and given to the linking authority and the linking of two signatures would then be a lookup if the tags of the two signatures in question can be found on the same list. We note that for large groups and large ℓ the size of the collection of lists for all users can be quite significant and, as in case of traceable signatures, for every joining user such a list for the new user has to be communicated to the linking authority. Moreover, we consider the application of this approach not suitable for controllable linkability, as the respective group signature scheme does only support ℓ unlinkable group signatures and consequently does not constitute a standard group signature scheme supporting an arbitrary number of unlinkable signatures.

Public Key Anonymous Tag Systems: Abe *et al.* [1] further generalized the approach from [21] to a primitive denoted as *public key anonymous tag system* which can be used to construct traceable signatures. Essentially, [1] use so called *link tokens*, where an authority with a dedicated link key can compute these link tokens for specific users that should be traceable. Given such a token for a specific member, signatures produced by this member can be publicly traced by anyone. Furthermore, their scheme allows all members to compute their link tokens without knowledge of the link key. Thus, signers can always link their own signatures which might not be desirable in some scenarios. An additional zero-knowledge proof based on this link token and the user's private key allows signers to claim and non-signers to deny being the signatory of a message. While this property might be useful in some applications, it also leads to the unavoidable attack that $N - 1$ colluding members can reveal the actual signer of a message by proving that they did not sign a specific message. Again, controllable linkability can be implemented with public key anonymous tag systems, but the linking requires costs linear in the size of the number of group members and cannot be precomputed (as it is the case with traceable signatures). In addition, the link tokens required to perform this linking must be distributed to the linking authority as soon as new members join the group. Furthermore, as public key anonymous tag systems by construction support signature claiming (by the signer) as well as denial of signatures (by any group member) the so obtained group signatures have additional features which may not be desired.

Verifier Local Revocation: In verifier local revocation (VLR) for group signatures [11], verifiers are given a revocation list in order to determine whether a signature stems from an already revoked member. More precisely, the verifier needs to test generated signatures against all entries on this revocation list, which means that the computational effort for the verifier grows linearly in the number of revoked users. Now, similar to how traceable signatures could be used for controllable linkability, the group manager could give a list with tokens for *all* group members to the linking authority, which given two signatures can test each of the signatures against the list and if the match happens on the same positions in the two runs then the signature has been produced by the same anonymous signer. Consequently, as in the case of traceable signatures to obtain a decision whether two signatures are linked requires computational costs linear in the size of the group of users and also as it is the case for traceable signatures the necessity to deliver a user token to the linking authority after every joining operation of a user. Besides, the scheme of [11] can only be implemented with Type 4 pairings, which

Table 1. Comparison of related concepts regarding their applicability for controllable linkability (CL).

Mechanism	Suitable for CL	Overhead for LA		
		Update on JOIN	Link	Memory
Tracing-by-linking group signatures [46]	✘	-	-	-
Link-but-not-trace group signatures [38]	✘	-	-	-
Traceable signatures [34]	✓	✓	$\mathcal{O}(N)$	$\mathcal{O}(N)$
Real traceable signatures [21]	✓	✓	$\mathcal{O}(N)$	$\mathcal{O}(\ell \cdot N)$
Public key anonymous tag systems [1]	✓	✓	$\mathcal{O}(N)$	$\mathcal{O}(N)$
Verifier local revocation [11]	✓	✓	$\mathcal{O}(N)$	$\mathcal{O}(N)$
Controlled linkability	✓	✘	$\mathcal{O}(1)$	$\mathcal{O}(1)$

are considered rather impractical [17], although the latter issue could be mitigated by alternative constructions such as [39].

Comparison: Table 1 shows a comprehensive comparison of the above outlined concepts regarding their applicability for the implementation of *controllable linkability* (CL) in group signature schemes. Although there exist concepts that sound quite similar, *i.e.*, *tracing-by-linking* as well as *link-but-not-trace* group signatures, the intention of these mechanisms is completely different, namely to prevent signers from signing more than a predefined number of k messages. Thus, these two concepts cannot be employed for the implementation of controllable linkability. The other concepts can be employed to achieve controllable linkability, but are rather inefficient in terms of communication overhead (communication of trapdoors to the linking authority) every time a new user joins the group, in terms of additional costs for the actual linking of signatures as well as in terms of memory requirements to achieve the desired functionality of linking. Note that we do not mention the trivial approach of opening group signatures for the purpose of linking due to the serious privacy concerns.

In conclusion, when requiring controllable linkability as the enhanced anonymity management mechanisms within group signature schemes, the proposed generic construction to convert pairing-based group signature schemes (PB-GSSs) based on the sign-and-encrypt-and-prove (SEP) paradigm into group signature schemes with controllable linkability is superior to "emulating" controllable linkability by means of features provided by other existing constructions. In particular, the approach proposed in this paper comes *for free*, meaning that it does not add additional costs to signature generation and verification, does not enlarge the signature size, and the linking authority can perform linking efficiently in constant time and without additional memory requirements. Furthermore, as the linking authority holds a single trapdoor in form of a linking key, there is no necessity to communicate user specific trapdoors from the group manager to the linking authority on every joining of a new user.

5 Future Work

It would be interesting to investigate group signatures with controllable linkability in a model such as [43], which prevents signature hijacking, being stronger than the BSZ model. Furthermore, it could be valuable to extend the model for controllable linkability in a way that it is required by the linking authority to provide a publicly verifiable proof that two signatures have indeed been produced by the same signer (similar to the proof τ required by the Open algorithm). On a theoretical side, as group signature schemes secure in the dynamic group setting are known to imply public key encryption with non-interactive opening (PKENO) [25] it would also be interesting to study whether group signatures with controllable linkability imply AoN-PKEET(*).

Acknowledgments. The authors would like to thank the anonymous reviewers for their valuable comments and suggestions. Daniel Slamanig has been supported by the European Commission through

project FP7-FutureID, grant agreement number 318424. Raphael Spreitzer and Thomas Unterluggauer have been supported by the Austrian Government through the research program FIT-IT under the grant number 835917 (NewP@ss).

References

[1] M. Abe, S. S. M. Chow, K. Haralambiev, and M. Ohkubo. Double-Trapdoor Anonymous Tags for Traceable Signatures. In *ACNS*, volume 6715 of *LNCS*, pages 183–200. Springer, 2011.
[2] G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A Practical and Provably Secure Coalition-Resistant Group Signature Scheme. In *CRYPTO*, pages 255–270, 2000.
[3] G. Ateniese, D. X. Song, and G. Tsudik. Quasi-Efficient Revocation in Group Signatures. In *Financial Cryptography*, pages 183–197, 2002.
[4] G. Ateniese and G. Tsudik. Some Open Issues and New Directions in Group Signatures. In *Financial Cryptography*, pages 196–211, 1999.
[5] M. H. Au, W. Susilo, and Y. Mu. Constant-Size Dynamic k-TAA. In *SCN*, volume 4116 of *LNCS*, pages 111–125. Springer, 2006.
[6] M. Bellare, D. Micciancio, and B. Warinschi. Foundations of Group Signatures: Formal Definitions, Simplified Requirements, and a Construction Based on General Assumptions. In *EUROCRYPT*, pages 614–629, 2003.
[7] M. Bellare, H. Shi, and C. Zhang. Foundations of Group Signatures: The Case of Dynamic Groups. In *CT-RSA*, pages 136–153, 2005.
[8] P. Bichsel, J. Camenisch, G. Neven, N. P. Smart, and B. Warinschi. Get Shorty via Group Signatures without Encryption. In *SCN*, pages 381–398, 2010.
[9] D. Boneh and X. Boyen. Short Signatures Without Random Oracles. In *EUROCRYPT*, pages 56–73, 2004.
[10] D. Boneh, X. Boyen, and H. Shacham. Short Group Signatures. In *CRYPTO*, pages 41–55, 2004.
[11] D. Boneh and H. Shacham. Group Signatures with Verifier-Local Revocation. In *ACM CCS*, pages 168–177, 2004.
[12] X. Boyen and B. Waters. Compact Group Signatures Without Random Oracles. In *EUROCRYPT*, pages 427–444, 2006.
[13] J. Camenisch, S. Hohenberger, and A. Lysyanskaya. Compact E-Cash. In *EUROCRYPT*, volume 3494 of *LNCS*, pages 302–321. Springer, 2005.
[14] J. Camenisch and A. Lysyanskaya. Signature Schemes and Anonymous Credentials from Bilinear Maps. In *CRYPTO*, volume 3152 of *LNCS*, pages 56–72. Springer, 2004.
[15] J. Camenisch and V. Shoup. Practical Verifiable Encryption and Decryption of Discrete Logarithms. In *CRYPTO*, volume 2729 of *LNCS*, pages 126–144. Springer, 2003.
[16] J. Camenisch and M. Stadler. Efficient Group Signature Schemes for Large Groups. In *CRYPTO*, volume 1294 of *LNCS*, pages 410–424. Springer, 1997.
[17] S. Chatterjee, D. Hankerson, and A. Menezes. On the Efficiency and Security of Pairing-Based Protocols in the Type 1 and Type 4 Settings. In *WAIFI*, pages 114–134, 2010.
[18] S. Chatterjee and A. Menezes. On cryptographic protocols employing asymmetric pairings - The role of ψ revisited. *Discrete Applied Mathematics*, 159(13):1311–1322, 2011.
[19] D. Chaum and E. van Heyst. Group Signatures. In *EUROCRYPT*, pages 257–265, 1991.
[20] S. G. Choi, K. Park, and M. Yung. Short Traceable Signatures Based on Bilinear Pairings. In *IWSEC*, volume 4266 of *LNCS*, pages 88–103. Springer, 2006.
[21] S. S. M. Chow. Real Traceable Signatures. In *Selected Areas in Cryptography*, volume 5867 of *LNCS*, pages 92–107. Springer, 2009.
[22] R. Cramer and V. Shoup. A Practical Public Key Cryptosystem Provably Secure Against Adaptive Chosen Ciphertext Attack. In *CRYPTO*, pages 13–25, 1998.
[23] C. Delerablée and D. Pointcheval. Dynamic Fully Anonymous Short Group Signatures. In *VIETCRYPT*, pages 193–210, 2006.

[24] Y. Dodis and A. Yampolskiy. A Verifiable Random Function with Short Proofs and Keys. In *Public Key Cryptography*, volume 3386 of *LNCS*, pages 416–431. Springer, 2005.
[25] K. Emura, G. Hanaoka, Y. Sakai, and J. C. N. Schuldt. Group signature implies public-key encryption with non-interactive opening. *Int. J. Inf. Sec.*, 13(1):51–62, 2014.
[26] J. Furukawa and H. Imai. An Efficient Group Signature Scheme from Bilinear Maps. In *ACISP*, volume 3574 of *LNCS*, pages 455–467. Springer, 2005.
[27] S. D. Galbraith, F. Hess, and F. Vercauteren. Aspects of Pairing Inversion. *IEEE Transactions on Information Theory*, 54(12):5719–5728, 2008.
[28] J. Groth. Fully Anonymous Group Signatures Without Random Oracles. In *ASIACRYPT*, volume 4833 of *LNCS*, pages 164–180. Springer, 2007.
[29] J. Groth and A. Sahai. Efficient Non-interactive Proof Systems for Bilinear Groups. In *EUROCRYPT*, volume 4965 of *LNCS*, pages 415–432. Springer, 2008.
[30] J. Y. Hwang, S. Lee, B.-H. Chung, H. S. Cho, and D. Nyang. Short Group Signatures with Controllable Linkability. In *LightSec*, pages 44–52, March 2011.
[31] J. Y. Hwang, S. Lee, B.-H. Chung, H. S. Cho, and D. Nyang. Group signatures with controllable linkability for dynamic membership. *Inf. Sci.*, 222:761–778, 2013.
[32] Internationl Organization for Standardization (ISO). ISO/IEC 20008-2: Information technology - Security techniques - Anonymous digital signatures - Part 2: Mechanisms using a group public key, November 2013.
[33] M. Jawurek, F. Kerschbaum, and C. Orlandi. Zero-knowledge using garbled circuits: how to prove non-algebraic statements efficiently. In *ACM CCS*, pages 955–966, 2013.
[34] A. Kiayias, Y. Tsiounis, and M. Yung. Traceable Signatures. In *EUROCRYPT*, volume 3027 of *LNCS*, pages 571–589. Springer, 2004.
[35] B. Libert and M. Yung. Efficient Traceable Signatures in the Standard Model. In *Pairing*, volume 5671 of *LNCS*, pages 187–205. Springer, 2009.
[36] J. K. Liu, V. K. Wei, and D. S. Wong. Linkable Spontaneous Anonymous Group Signature for Ad Hoc Groups. In *ACISP*, volume 3108 of *LNCS*, pages 325–335. Springer, 2004.
[37] T. Nakanishi, H. Fujii, Y. Hira, and N. Funabiki. Revocable Group Signature Schemes with Constant Costs for Signing and Verifying. In *Public Key Cryptography*, LNCS, pages 463–480. Springer, 2009.
[38] T. Nakanishi, T. Fujiwara, and H. Watanabe. A Linkable Group Signature and Its Application to Secret Voting. *Trans. of Information Processing Society of Japan*, 40(7), 1999.
[39] T. Nakanishi and N. Funabiki. Verifier-Local Revocation Group Signature Schemes with Backward Unlinkability from Bilinear Maps. In *ASIACRYPT*, volume 3788 of *LNCS*, pages 533–548. Springer, 2005.
[40] M. Naor and M. Yung. Public-key Cryptosystems Provably Secure against Chosen Ciphertext Attacks. In *STOC*, pages 427–437. ACM, 1990.
[41] L. Nguyen and R. Safavi-Naini. Efficient and Provably Secure Trapdoor-Free Group Signature Schemes from Bilinear Pairings. In *ASIACRYPT*, pages 372–386, 2004.
[42] Y. Sakai, K. Emura, G. Hanaoka, Y. Kawai, T. Matsuda, and K. Omote. Group Signatures with Message-Dependent Opening. In *Pairing*, volume 7708 of *LNCS*, pages 270–294. Springer, 2012.
[43] 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*, volume 7293 of *LNCS*, pages 715–732. Springer, 2012.
[44] Q. Tang. Public key encryption schemes supporting equality test with authorisation of different granularity. *IJACT*, 2(4):304–321, 2012.
[45] Q. Tang. Public key encryption supporting plaintext equality test and user-specified authorization. *Security and Communication Networks*, 5(12):1351–1362, 2012.
[46] I. Teranishi, J. Furukawa, and K. Sako. k-Times Anonymous Authentication. In *ASIACRYPT*, volume 3329 of *LNCS*, pages 308–322. Springer, 2004.
[47] V. K. Wei. Tracing-by-Linking Group Signatures. In *ISC*, volume 3650 of *LNCS*, pages 149–163. Springer, 2005.

[48] G. Yang, C. H. Tan, Q. Huang, and D. S. Wong. Probabilistic Public Key Encryption with Equality Test. In *CT-RSA*, pages 119–131, 2010.

A Formal Model and Proofs for Controllable Linkability

In the following we present the three properties required for controllable linkability (as defined in [30, 31]), whereas we note that we have corrected some minor flaws in their definitions. In particular, in the LO-unlinkability experiment the adversary $\mathcal{A}$ is additionally required to have access to the $\texttt{AddU}(\cdot)$ oracle (missing in their papers) as otherwise there cannot be any honest user in the experiment and the winning condition can never be satisfied. Furthermore, in the JP-unforgeability experiment, we find it more natural to write that the $\mathcal{A}$ is not allowed to query $\texttt{GSig}(i_0, \cdot)$ and $\texttt{GSig}(i_1, \cdot)$ in the second phase (this seems to be implicitly equivalent to their condition of not being allowed to query $\texttt{GSig}(i_b, \cdot)$). Moreover, in their E-linkability experiment $\mathsf{upk}[i]$ should be $\mathsf{upk}[i_0]$ and $\mathsf{upk}[j]$ should be $\mathsf{upk}[i_1]$.

Definition 3 (LO-Linkability) *A group signature scheme $\mathcal{GS}$ with controllable linkability is said to be LO-linkable if for any adversary $\mathcal{A}$ and any $\lambda \in \mathbb{N}$, $\mathsf{Pr}[\mathsf{Exp}^{\mathsf{LO-link}}_{\mathcal{GS},\mathcal{A}}(\lambda) = 1] \leq \epsilon(\lambda)$, where the experiment is defined as follows:*

Experiment $\mathsf{Exp}^{\mathsf{LO-link}}_{\mathcal{GS},\mathcal{A}}(\lambda)$*:*

- $(\mathsf{gpk}, \mathsf{mok}, \mathsf{mik}, \mathsf{mlk}) \leftarrow \mathsf{GkGen}(\lambda)$, $\mathbf{CU} \leftarrow \emptyset$, $\mathbf{HU} \leftarrow \emptyset$, $\mathbf{GSet} \leftarrow \emptyset$;
- $(i_0, i_1, M) \leftarrow \mathcal{A}^{\texttt{SndToU},\texttt{AddU},\texttt{GSig},\texttt{Open},\texttt{USK},\texttt{CrptU}}(\mathsf{gpk}, \mathsf{mik}, \mathsf{mlk})$;
- $b \xleftarrow{R} \{0,1\}$, $\sigma \leftarrow \mathsf{GSig}(\mathsf{gpk}, i_b, M)$;
- $b' \leftarrow \mathcal{A}^{\texttt{SndToU},\texttt{AddU},\texttt{GSig},\texttt{Open},\texttt{USK},\texttt{CrptU}}(\mathsf{gpk}, \mathsf{mik}, i_0, i_1, M, \sigma)$;
- *If all of the following conditions hold, then return* 1 *and* 0 *otherwise*
  - $i_0, i_1 \in \mathbf{HU}$;
  - $\texttt{GSig}(i_0, \cdot)$ *and* $\texttt{GSig}(i_1, \cdot)$ *and* $\texttt{Open}(M, \sigma)$ *have not been queried;*
  - $b' = b$

Definition 4 (JP-unforgeability) *A group signature scheme $\mathcal{GS}$ with controllable linkability is said to be JP-unforgeable if for any adversary $\mathcal{A}$ and any $\lambda \in \mathbb{N}$, $\mathsf{Pr}[\mathsf{Exp}^{\mathsf{JP-UF}}_{\mathcal{GS},\mathcal{A}}(\lambda) = 1] \leq \epsilon(\lambda)$, where the experiment is defined as follows:*

Experiment $\mathsf{Exp}^{\mathsf{JP-UF}}_{\mathcal{GS},\mathcal{A}}(\lambda)$*:*

- $(\mathsf{gpk}, \mathsf{mok}, \mathsf{mik}, \mathsf{mlk}) \leftarrow \mathsf{GkGen}(\lambda)$, $\mathbf{CU} \leftarrow \emptyset$, $\mathbf{HU} \leftarrow \emptyset$, $\mathbf{GSet} \leftarrow \emptyset$;
- $(i, M) \leftarrow \mathcal{A}^{\texttt{SndToU},\texttt{AddU},\texttt{WReg},\texttt{GSig},\texttt{Open},\texttt{USK},\texttt{CrptU}}(\mathsf{gpk}, \mathsf{mik}, \mathsf{mlk})$;
- $\sigma \leftarrow \mathsf{GSig}(\mathsf{gpk}, i, M)$;
- $\tau \leftarrow \mathcal{A}^{\texttt{SndToU},\texttt{WReg},\texttt{GSig},\texttt{Open},\texttt{USK},\texttt{CrptU}}(\mathsf{gpk}, \mathsf{mik}, \mathsf{mlk}, i, M, \sigma)$;
- *If all of the following conditions hold, then return* 1 *and* 0 *otherwise*
  - $i \in \mathbf{HU}$ *and* $\mathbf{gsk}[i] \neq \emptyset$;
  - $\texttt{Open}(M, \sigma)$ *has not been queried;*
  - $\texttt{true} \leftarrow \mathsf{Judge}(\mathsf{gpk}, \mathbf{upk}[i], M, \sigma, \tau)$

Definition 5 (E-Linkability) *A group signature scheme $\mathcal{GS}$ with controllable linkability is said to be E-linkable if for any adversary $\mathcal{A}$ and any $\lambda \in \mathbb{N}$, $\mathsf{Pr}[\mathsf{Exp}^{\mathsf{E-link}}_{\mathcal{GS},\mathcal{A}}(\lambda) = 1] \leq \epsilon(\lambda)$, where the experiment is defined as follows:*

Experiment $\mathsf{Exp}^{\mathsf{E-link}}_{\mathcal{GS},\mathcal{A}}(\lambda)$*:*

- $(\mathsf{gpk}, \mathsf{mok}, \mathsf{mik}, \mathsf{mlk}) \leftarrow \mathsf{GkGen}(\lambda)$, $\mathbf{CU} \leftarrow \emptyset$, $\mathbf{HU} \leftarrow \emptyset$, $\mathbf{GSet} \leftarrow \emptyset$;
- $(M_0, \sigma_0, M_1, \sigma_1) \leftarrow \mathcal{A}^{\texttt{SndToU},\texttt{AddU},\texttt{RReg},\texttt{GSig},\texttt{USK},\texttt{CrptU}}(\mathsf{gpk}, \mathsf{mok}, \mathsf{mlk})$;
- *If* $\mathsf{GVf}(\mathsf{gpk}, M_b, \sigma_b) = \texttt{false}$ *for* $b = 0, 1$ *then return* 0;

- $(i_0, \tau_{i_0}) \leftarrow \mathsf{Open}(\mathsf{gpk}, \mathbf{REG}, M_0, \sigma_0, \mathsf{mok})$
- $(i_1, \tau_{i_1}) \leftarrow \mathsf{Open}(\mathsf{gpk}, \mathbf{REG}, M_1, \sigma_1, \mathsf{mok})$
- *If* $\mathtt{false} \leftarrow \mathsf{Judge}(\mathsf{gpk}, M_0, \sigma_0, i_0, \mathbf{upk}[i_0], \tau_{i_0})$ *or* $\mathtt{false} \leftarrow \mathsf{Judge}(\mathsf{gpk}, M_1, \sigma_1, i_0, \mathbf{upk}[i_1], \tau_{i_1})$ *then return* 0*;*
- *If* $i_0 \neq i_1$ *and* $\mathtt{true} \leftarrow \mathsf{Link}(\mathsf{gpk}, M_0, \sigma_0, M_1, \sigma_1, \mathsf{mlk})$ *then return 1;*
- *else if* $i_0 = i_1$ *and* $\mathtt{false} \leftarrow \mathsf{Link}(\mathsf{gpk}, M_0, \sigma_0, M_1, \sigma_1, \mathsf{mlk})$ *then return 1;*
- *else return 0;*

Where **CU**, **HU**, **GSet** represent the sets of corrupted users, honest users and the set of message-signature pairs. Furthermore, **REG** is the list of transcripts generated by the join process. The oracles $\mathtt{SndToU}, \mathtt{AddU}, \mathtt{WReg}, \mathtt{RReg}, \mathtt{GSig}, \mathtt{USK}$ and $\mathtt{CrptU}$ represent a send to issuer oracle (by a corrupted user), an add user oracle, a write registration table oracle, a read registration table oracle, a group signing oracle, a user secret key oracle and a corrupt user oracle (cf. [30, 31] for details).

A.1 Proof for LO-Linkability

Proof (Lemma 3). An adversary $\mathcal{A}$ that is able to determine whether a challenge signature has been issued by i_0 or i_1 has done so without calling Open for the challenge signature, not knowing usk_0 and usk_1 as well as not having issued any signature query for i_0 or i_1 to the $\mathtt{GSig}$ oracle. Note that the adversary knows mlk in the first phase but is not allowed to corrupt challenged users, *i.e.*, cannot obtain the secret keys nor public keys upk_0 and upk_1, nor to obtain signatures or call the opening. The adversary no longer has access to mlk in the second phase. Since the adversary does not know the user's secrets and thus cannot recompute the membership certificates contained in T_b of $\sigma_b = (T_b, \pi_b)$ and under Assumption 1, if $\mathcal{A}$ is able to win the experiment, $\mathcal{A}$ obviously can be turned into an adversary against the ciphertext indistuinguishability of the public key encryption scheme underlying the AoN-PKEET* scheme, which contradicts the assumption that AoN-PKEET* is secure. □

A.2 Proof for JP-Unforgeability

Proof (Lemma 4). In PB-GSSs following the sign-and-encrypt-and-prove paradigm, the proof τ output by the Open algorithm represents a non-interactive honest-verifier zero-knowledge proof of knowledge of equality of the output certificate with the certificate in the ciphertext (which is usually a simple proof of equality of discrete logarithms). An efficient adversary $\mathcal{A}$ that wins the JP-unforgeability experiment can be used to extract the private key corresponding to the public key encryption key in gpk and mlk. The reduction can embed the problem instance into the public key in gpk and mlk and simulating the proofs required by the Open queries by programming the random oracle. The private key can then be extracted from $\mathcal{A}$ by using standard rewinding techniques for the proof of knowledge in the reduction. □

A.3 Proof for E-Linkability

Proof (Lemma 5). If an adversary $\mathcal{A}$ outputs two pairs (M_0, σ_0) and (M_1, σ_1) such that $(i_b, \tau_b) \leftarrow \mathsf{Open}(\mathsf{gpk}, \mathbf{reg}, M_b, \sigma_b, \mathsf{mok})$ and $\mathtt{true} \leftarrow \mathsf{Judge}(\mathsf{gpk}, M_b, \sigma_b, i_b, \mathsf{upk}_{i_b}, \tau_b)$ for $b = 0, 1$, then we have two cases:

Case 1: In the first case we have that $i_0 = i_1$ but $\mathtt{false} \leftarrow \mathsf{Link}(\mathsf{gpk}, \mathsf{M}_0, \sigma_0, \mathsf{M}_1, \sigma_1, \mathsf{mlk})$. This means that for $\sigma_0 = (T_0, \pi_0)$ and $\sigma_1 = (T_1, \pi_1)$ both ciphertexts T_0 and T_1 are encryptions of the membership certificate of the same user ($i_0 = i_1$ follows from Open), but $\mathtt{false} \leftarrow \mathsf{Com}(T_0, T_1, \mathsf{mlk})$. This, however, contradicts the soundness of the AoN-PKEET* scheme.

Case 2: In the second case we have that $i_0 \neq i_1$ but $\mathtt{true} \leftarrow \mathsf{Link}(\mathsf{gpk}, \mathsf{M}_0, \sigma_0, \mathsf{M}_1, \sigma_1, \mathsf{mlk})$. This means that for $\sigma_0 = (T_0, \pi_0)$ and $\sigma_1 = (T_1, \pi_1)$ the ciphertexts T_0 and T_1 are encryptions of distinct certificates of the users $i_0 \neq i_1$. However, as $\mathsf{true} \leftarrow \mathsf{Com}(T_0, T_1, \mathsf{mlk})$, this, again, contradicts the soundness of the AoN-PKEET* scheme.

□