Identity Based Group Signatures from Heiarchical Identity-Based Encryption

N.P. Smart and B. Warinschi

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

Abstract. A number of previous papers explored the notion of identity-based group signature. We present a generic construction of identity-based group signatures. Our construction is based on the Naor transformation of a identity-based signature out of an identity-based encryption, adjusted to hierarchical identity-based encryption. We identify sufficient conditions on the underlying HIBE so that the scheme that results from our transformation meets our security definitions. Finally, we suggest a couple of extensions enabled by our construction, one of which is to hierarchical identity-based group signatures.

1 Introduction

Identity-based cryptography as envisioned by Shamir [24] aims to ease the key distribution problem associated to standard PKIs used for asymmetric cryptosystems. The key insight is that parties can use their identities as their public keys, which in turn makes secure repositories for public keys unnecessary. This idea had been thoroughly explored in the context of standard encryption [5, 7, 12, 14, 21–23] and signature schemes [2, 10, 17] as well as in that of more complex primitives like traitor tracing [1].

In the context of group signatures, a primitive with multiple practical uses, a large proportion of the prior work did not consider the appropriate extension of the primitive to the ID-based setting. Specifically, the schemes proposed in many previous papers under the name of identity-based group signatures still use a standard public key for the group key. This is clearly a departure from the original motivation for identity based cryptography, does not properly extend identity-based signature schemes, and suffers from the standard PKI related difficulties. The reason for the name of the primitive was that the identity of group members was allowed to be an unstructured identity. Examples of such proposals include [13, 18–20, 26].

In [28] a more correct syntax and security definition is given in which identifier strings are used for both the users, and the group names themselves. Recall that in a group signature, multiple signers can produce signatures on behalf of the group without revealing information about the origin of the signature. Only a designated opener can later link these signatures to their authors using a special secret key, whereas a group manager is in charge of adding users to the group. In [28] these two functionalities are seperated, in our work we simplify the model somewhat by requiring the opener and the group manager to be the same.

Our contributions. We provide the following results on identity-based group signatures.

Security and Syntax of the primitives. We provide a simplified identity-based group signature model, which is a subset of the model in [28]. Since we work in the ID-based setting we consider a trusted authority that generates system-wide parameters. We model and explore the realistic scenario where the same set of system parameters is shared by multiple groups of signers. Users can join existent groups, and we allow for the same user can belong to multiple groups. Our security models are those of full-anonymity and full-traceability. Full-anonymity captures the idea that the identity of the signers is not revealed by signatures, and full-traceability says that the group manager can determine who created a given valid signature.

In this paper we model a simple setting where the roles of the group manager and signature openers are merged (very much like in [3]). Also in our model users do not have public keys (or independent identity based keys) and therefore no secrets, hence the group manager (opener) can always add users to groups and produce signatures on their behalf, undetected. Our simplified definition facilitates our direct HIBE based construction, which itself then can be easily seen to extend to a construction which enables hierarchies of groups. It is this HIBE based construction and its extension which is the most novel part of our work.

Generic construction based on HIBE. Clearly, one can construct an ID-based group signature schemes by appending certificates for the group public key to each signature in a standard group signature scheme. Our models can be used to analyze such constructions. However, the flexibility afforded by our syntax may lead to more efficient and/or schemes with enhanced functionality. One interesting example is schemes with group hierarchies alluded to above, and discussed further below.

We provide a generic construction based on hierarchical identity-based encryption (HIBE) [15]. The transformation that we present adapts the Naor transformation of an identity-based encryption scheme into an identity-based signature scheme and shares ideas with the Boyen-Waters construction of a *standard* group signature scheme out of a HIBE. Next, we sketch our construction and provide further details on the transformation that we designed.

Recall that in HIBEs users at the lower levels of the hierarchy can compute the decryption keys for users at the higher levels. The idea behind our construction is to set up a 4-level HIBE: on the first level we place group identities, on the second level user identities, on the third level the messages to be signed, and the fourth level is reserved for some sort of randomizers. A new group is created by extracting the key associated to the identity grpID which is then given to the group manager. To add user userID to group grpID, the manager extracts the key associated to identity (grpID, userID) which becomes the signing key of user userID for group grpID. One tempting way to produce a signature on a message m using this key, is to extract the secret key associated to hierarchical identity $(\mathrm{grpID}, \mathrm{userID}, m)$. This is essentially the approach taken by the construction of [8] which encrypts the resulting signature under the public key and uses (efficient) non-interactive zero-knowledge proofs to ensure that the construction followed the prescribed recipe. Notice that encrypting the signature is indeed needed, as the signature may leak information about userID (for example when the extraction algorithm is deterministic.) To hide the identity of the signer we use a different approach based on properties that we observe in existent HIBE constructions. Specifically, to produce a signature on message m on behalf of group grpID, a user userID extracts the secret key d associated to $(\mathrm{grpID}, \mathrm{userID}, m, \mathrm{r_{ID}})$, for a randomly chosen randomizer $\mathrm{r_{ID}}$. We observe that for existent constructions the resulting decryption key hides all information about the hierarchical identity to which it corresponds (provided that $\mathrm{r_{ID}}$ is from a big enough space). A remaining problem is that in order to verify the signature, one needs to first encrypt a message under the identity $(\mathrm{grpID}, \mathrm{userID}, m, \mathrm{r_{ID}})$ and then test that the decryption with d succeeds. Clearly, this verification procedure leaks information about userID. Instead, we observe that the encryption process of HIBEs usually compute an encryption key e associated to the hierarchical identity which is then used in an encryption algorithm. We can therefore let (e, d) play the role of a signature, provided they indeed do not reveal information about userID. We call this property that we identify and demand from the underlying HIBE *random identity hiding*. It is worth noting that all of the existent HIBE constructions, Boneh-Boyen [5], Waters [27] and Boneh-Boyen-Goh [6] satisfy this property. In addition to (e, d) a signature also contains an encryption of userID under grpID and a non-interactive proof that all of the parts fit together. We analyze a construction where this proof is obtained via the Fiat–Shamir transform, and therefore our construction is under the random oracle model.

Instantiation based on the Boneh-Boyen-Goh HIBE. We use the Boneh-Boyen-Goh HIBE to instantiate our construction. We show that our theoretical construction yields in this case an explicit identity-based group signature scheme which has a signature of fixed length, irrespective of the size of the group to which the signature is attached. Furthermore, the signature is relatively short, and computationally very efficient.

Extensions. Finally, we sketch a couple of variants of our basic construction. First, we note that by eliminating the first level of the HIBE (the level that contains group identities) we obtain a standard group signature with a standard public key as the verification key. A more interesting extension is that to a hierarchical identity-based group signatures. For standard group signatures the extension to hierarchical group managers has been investigated by Trolin and Wikström[25]. The analogous extension for the case of identity-based group signatures is beyond the goals of this paper. We sketch however how to extend our construction as to meet the intuitive goals of such an extension. The idea is to introduce additional group identity levels. Group managers can then add users to any of the subgroups of the group he manages, users can sign on behalf of any of the groups to which they belong, and signatures can be opened by the managers of these groups, or indeed any other levels in the hierarchy.

On the use of the random oracle. We end the introduction with a note on the usage of the random oracles in our construction. In our construction we use non-interactive zero-knowledge proofs obtained via the Fiat–Shamir heuristic from Σ-protocols, and thus our construction is in the random oracle model. An alternative that would yield schemes secure in the standard model could employ standard model NIZKPOKs (based on a common random string which can be placed in the system parameters), such as those in [16]. However, whilst such NIZKPOKs run in polynomial time, their performance is not very efficient when compared to constructions obtained from Σ-protocols via the Fiat–Shamir heuristic. Indeed, we have chosen to carry out our work in the random oracle model to be able to obtain the efficient implementation based on BBG, which itself requires the random oracle model to obtain non-selective ID security.

Acknowledgements The authors would like to thank G. Neven for various discussions whilst the work in this paper was carried out. The authors would like to acknowledge the support of the eCrypt-2 Network of Excellence. The first author was supported by a Royal Society Wolfson Merit Award.

2 Preliminaries

Sigma protocols. A Σ-protocol $(\mathcal{P}, \mathcal{V})$ for an NP-language L is a three-move, public coin interactive proof. We typically write (r, c, s) for a transcript of the conversation between the prover and the verifier, where r and s are the messages sent by the prover and c is the message sent by the verifier. We call r the commitment message, c the challenge message, and s the response. We write $\mathsf{CommitSpace}$ for the space to which r belongs, $\mathsf{ChallSpace}$ for the space from where c is drawn. We call a transcript accepting for x if the verification algorithm employed by the verifier, $\mathcal{V}((r, c, s), x)$ returns 1. Notice that we abuse notation and write $\mathcal{V}$ for both the verifier and its verification algorithm.

In this paper we use Σ-protocols that satisfy special-soundness: we require that there exists an extraction algorithm $\mathcal{E}$ which given two accepting transcripts (r, c, s) and (r, c', s') for x returns a witness w that $x \in L$. Furthermore, we require that the protocol be special-zero-knowledge, that is: there exists a simulator $\mathcal{S}$ which on input x and challenge c outputs (r, s) such that (r, c, s) is an accepting transcript for x. If c is selected at random from $\mathsf{ChallSpace}$ then (r, c, s) is distributed as true transcripts. We formalize this notion in Appendix B.

In addition, we also require that $(\mathcal{P}, \mathcal{V})$ have perfect completeness: for any witness w that $x \in L$ the interaction $(\mathcal{P}(x, w), \mathcal{V}(x))$ is accepting.

The Fiat–Shamir transform. The Fiat–Shamir transform is a heuristic that transforms a three move public coin into a signature. The heuristic can be used to create "signatures of knowledge" [11]: constructs which in addition to being signatures on messages, also prove knowledge of a certain secret. Essentially, given a Σ protocol $(\mathcal{P}, \mathcal{V})$ for some language L and a hash function H one can build a signature of knowledge scheme as follows. Given an element $x \in L$ and a corresponding witness w, one can produce a signature of knowledge $\mathsf{FS}^{\mathsf{H}}_{\mathcal{P}}((w, x))(m)$ by running locally the interactive proof that $x \in L$, using $c \leftarrow \mathsf{H}(r||x||m)$ as challenge. Here r is the first message produced by the prover. A bit more formally, we define $\mathsf{FS}^{\mathsf{H}}_{\mathcal{P}}((w, x), m)$ as the algorithm: $(r, state) \leftarrow \mathcal{P}(w)(x)$; $c \leftarrow \mathsf{H}(r||x||m)$, $s \leftarrow \mathcal{P}(state, c)(x)$; output (r, s). To verify that (r, s) is a signature of knowledge on message m given public information x, one runs $\mathcal{V}(r, \mathsf{H}(r||x||m), s)$ and accepts if the output is 1[1]. We do not formalize the properties signatures of knowledge satisfy. Instead, when we use them in constructions, we reduce the security of the constructions to the properties of the underlying Σ-protocol. In

[1] Notice that throught the paper we avoid cluttered notation by assuming that the statement to be verified is an implicit input to the verifier.

particular, in order for the Fiat–Shamir heuristic to work, we further require from the Σ-protocol that it has high-entropy commitments, and high-entropy challenges. Since the challenge is selected at random from the challenge space, the second condition is satisfied whenever this space is sufficiently large. We simplify the first requirement and ask that the commitment space to also be large, and that commitments are randomly distributed over this space.

3 Hierarchical Identity Based Encryption (HIBE)

In this section we recall the notion of HIBE, and introduce its variant that concerns us. Throughout the remainder of the paper we assume a set of basic identities $\mathsf{IdSp} \subseteq \{0,1\}^*$. We call $\mathsf{ID} \in \mathsf{IdSp}^l$ an l-level hierarchical identity. For clarity we denote elements of IdSp by lower case variables (e.g., $\mathsf{id}, \mathsf{id}', \mathsf{id}_1, \mathsf{id}_2, \ldots$) and hierarchical identities by upper-case variables (e.g. $\mathsf{ID}, \mathsf{ID}', \mathsf{ID}_1, \mathsf{ID}_2, \ldots$).

Hierarchical Identity Based Encryption (HIBE). A HIBE consists of four polynomial time algorithms $(\mathsf{Setup}, \mathsf{Extract}, \mathsf{Encrypt}, \mathsf{Decrypt})$:

- $\mathsf{Setup}(1^k, L)$. The setup algorithm, on input a security parameter k and a maximal number of levels L generates a master public/private key pair $(\mathrm{mpk}, \mathrm{msk})$ and a message space description $\mathcal{M}$ for an L-level HIBE.
- $\mathsf{Extract}(\mathrm{mpk}, \mathsf{ID}, d_{\mathsf{ID}'})$. The secret key extraction algorithm takes as input an identity ID and the secret key associated to a parent ID' of ID and derives a secret key d_{ID} for ID. By convention, we let $d_{()}$ (the key associated to identity $''()''$) to be msk.
- $\mathsf{Encrypt}(\mathrm{mpk}, \mathsf{ID}, \mathfrak{m}; r)$. The randomized encryption algorithm, on input the master public key mpk, a hierarchical identity ID, and message $\mathfrak{m}$ outputs an encryption *enc* of the message $\mathfrak{m}$ for identity ID using randomness r.
- $\mathsf{Decrypt}(\mathrm{d}_{\mathrm{ID}}, c)$. The decryption algorithm takes as input a secret key d_{ID} that corresponds to some hierarchical identity ID, and a ciphertext *enc* and returns the underlying plaintext (assuming that the ciphertext was encrypted using some identity ID' to which ID is a parent).

Notice that the extraction algorithm works with the secret key of any parent of the target identity (and not only with the master secret key). For correctness we require that ciphertexts created using some identity can be decrypted using a secret key associated to the identity of any of its parents, i.e.

$$\mathsf{Decrypt}(\mathsf{Extract}(\mathrm{mpk}, \mathsf{ID}_2, d_{\mathsf{ID}_1}), \mathsf{Encrypt}(\mathrm{mpk}, \mathsf{ID}_3, \mathfrak{m}; r)) = \mathfrak{m}$$

whenever ID_1 is a parent of ID_2 which in turn is a parent of ID_3 and d_{ID_1} is a secret key associated to ID_1.

In the variant of HIBE that we introduce we would like to allow parties to encrypt messages for identities which he does not know. We enable this property by making the assumption that the $\mathsf{Encrypt}(\mathrm{mpk}, \mathsf{ID}, \mathfrak{m}; r)$ algorithm works in two phases. First the encryptor obtains an encryption key e_{ID} out of the identity ID and the master public key and then the ciphertext is obtained using an underlying encryption algorithm. More precisely, we assume that $\mathsf{Encrypt}(\mathrm{mpk}, \mathsf{ID}, \mathfrak{m}; r) = \mathsf{Encr}(\mathsf{Distill}(\mathrm{mpk}, \mathsf{ID}), \mathfrak{m}; r)$ for some algorithm $\mathsf{Distill}$ for distilling keys out of identities, and some underlying encryption algorithm Encr. To define a HIBE, it is therefore required to give two algorithms $\mathsf{Distill}, \mathsf{Encr}$ instead of the single $\mathsf{Encrypt}$. We call schemes defined this way *canonical*.

In Appendix C we recap on the BBG HIBE of [6], but using the functions $\mathsf{Distill}, \mathsf{Encr}$ instead of the single $\mathsf{Encrypt}$. The BBG HIBE will be used as our example throughout since it is very efficient, and thus results in a highly efficient identity-based group signature scheme.

3.1 Security Notions

Our construction for ID-based group signatures is based on a HIBE which satisfies two security properties. In addition to the standard notion of indistinguishability against chosen-plaintext/chosen-ciphertext, the scheme should also hide the identity of a random identity. We first recall the former notion and then formalise the latter.

Definition 1 (Indistinguishability under CPA and CCA). *Indistinguishability under chosen-plaintext, and chosen-ciphertext attacks of a HIBE scheme Π, are security notions defined through the experiments* $\mathbf{Exp}_{\Pi,\mathcal{A}}^{\mathsf{ind\text{-}id\text{-}cpa}-b}(k)$ *and* $\mathbf{Exp}_{\Pi,\mathcal{A}}^{\mathsf{ind\text{-}id\text{-}cca}-b}(k)$ *that we describe below. The experiments depend on an adversary $\mathcal{A}$, and are parametrised by a bit b. In a first phase, the adversary is given as input the master public key* mpk *of a freshly generated key pair* $(\mathrm{mpk}, \mathrm{msk}) \overset{\$}{\leftarrow} \mathsf{Setup}(1^k, L)$ *as input. In a chosen-plaintext attack (*IND-ID-CPA*), the adversary is given access to a key derivation oracle that on input of an identity* $\mathsf{ID} = (\mathsf{id}_1, \ldots, \mathsf{id}_\ell)$*, returns the secret key* $d_{\mathsf{ID}} \overset{\$}{\leftarrow} \mathsf{Extract}(\mathrm{msk}, \mathsf{ID})$ *corresponding to identity* ID*. In a chosen-ciphertext attack (*IND-ID-CCA*), the adversary is additionally given access to a decryption oracle that for a given identity* $\mathsf{ID} = (\mathsf{id}_1, \ldots, \mathsf{id}_\ell)$ *and a given ciphertext enc returns the decryption*

$$\mathfrak{m} \leftarrow \mathsf{Decrypt}(\mathsf{Extract}(\mathrm{msk}, \mathsf{ID}), c).$$

At the end of the first phase, the adversary outputs a challenge messages $\mathfrak{m}* \in \{0,1\}^*$ *and a challenge identity* $\mathsf{ID}^* = (\mathsf{id}_1^*, \ldots, \mathsf{id}_{\ell^*}^*)$*, where* $0 \le \ell^* \le L$*. Both experiments then generate a challenge ciphertext* $c^* \overset{\$}{\leftarrow} \mathsf{Encrypt}(\mathrm{mpk}, \mathsf{ID}^*, \mathfrak{m}_b^*; r)$*, where b is the parameter bit,* $\mathfrak{m}_0^* = 0^{|\mathfrak{m}^*|}$ *and* $\mathfrak{m}_1^* = \mathfrak{m}^*$*, and gives c^* as input to the adversary for the second phase.*[2] *In the second phase the adversary has access to the same oracles and has to output a bit d. The experiment outputs the d. We require that in both experiment the adversary never queries the key derivation oracle on a parent identity of* ID^**, and that in the CCA experiment the pair* (ID^*, c^*) *is never sent to the decryption oracle. The advantage of the adversary is defined by:*

$$\mathbf{Adv}_{\Pi,\mathcal{A}}^{\mathsf{ind\text{-}id\text{-}xxx}}(k) = \Pr\left[\mathbf{Exp}_{\Pi,\mathcal{A}}^{\mathsf{ind\text{-}id\text{-}xxx}-1}(k) = 1\right] - \Pr\left[\mathbf{Exp}_{\Pi,\mathcal{A}}^{\mathsf{ind\text{-}id\text{-}xxx}-0}(k) = 1\right]$$

for $\mathsf{xxx} \in \{\mathsf{cpa}, \mathsf{cca}\}$*.*

We say that Π is IND-ID-CCA *secure (respectively* IND-ID-CPA*secure) if for all p.p.t. adversaries its advantage* $\mathbf{Adv}_{\Pi,\mathcal{A}}^{\mathsf{ind\text{-}id\text{-}cca}}(k)$ *(respectively* $\mathbf{Adv}_{\Pi,\mathcal{A}}^{\mathsf{ind\text{-}id\text{-}cpa}}(k)$*) is negligible.*

Random-Identity Hiding. Informally, the notion of *random identity hiding* requires that the key distilled from a hierarchical identity $\mathsf{ID} = (\mathsf{id}_1, \mathsf{id}_2, \ldots, \mathsf{id}_l)$ together with an associated decryption key, does not reveal any information about ID, as long as at least one of the basic identities id_i is chosen at random. The formalisation of this notion uses *patterns*. An l-level pattern is simply element of the set $(\mathsf{IdSp} \cup \{\star\})^l$, i.e. a hierarchical identity where some components are replaced by $\star$. We call a pattern *non-trivial* if it contains $\star$ on at least one position. For a pattern P we write $\hat{P}$ for the set $\hat{P} = \{\mathsf{ID} \mid \mathsf{ID} \in \mathsf{IdSp}^l, P_i \neq \star \Rightarrow P_i = \mathsf{ID}_i\}$ of hierarchical identities that coincide with the entries in the pattern on all positions that are not $\star$ in P.

The security game that defines random identity hiding is as follows. The adversary selects a non-trivial patterns P of level $l \le L$. The adversary is then given the pair

$$(d_{\mathsf{ID}}, e_{\mathsf{ID}}) = (\mathsf{Extract}(\mathrm{mpk}, \mathsf{ID}, ()), \mathsf{Distill}(\mathrm{mpk}, \mathsf{ID}))$$

for either a random identity ID of level l, or an identity $\mathsf{ID} \in \hat{P}$. The task of the adversary is to determine whether its input has been obtained from the given pattern, or a truly random identity. In his game, the adversary has access to essentially all the information in the system (i.e. the master secret key msk grants access to the secret key of any identity), except to the randomness used to obtain ID.

Definition 2 (Random identity hiding). *Consider the following experiment for a L-level HIBE scheme* $\Pi = (\mathsf{Setup}, \mathsf{Distill}, \mathsf{Extr}, \mathsf{Encr}, \mathsf{Decrypt})$ *and adversary $\mathcal{A}$:*

```
$\mathbf{Exp}_{\Pi,\mathcal{A}}^{\mathsf{RIdH}-b}(1^k)$
    $(\mathrm{mpk}, \mathrm{msk}) \overset{\$}{\leftarrow} \mathsf{Setup}(1^k.L)$
    $(P, \mathrm{St}) \leftarrow \mathcal{A}(\mathrm{mpk}, \mathrm{msk})$
```

[2] The definition that we use asks the adversary to tell apart encryptions of the message from the encryptions of the all-0 string of the same length. This notion is equivalent to the one in the literature.

```
$b \leftarrow \{0,1\}$
$\textit{If } b = 0 \textit{ then } \mathsf{ID}^* \overset{\$}{\leftarrow} \hat{P};$
$\textit{else } \mathsf{ID}^* \overset{\$}{\leftarrow} \mathsf{IdSp}^l \textit{ where } P \textit{ is an } l\textit{-level pattern.}$
$e^* \leftarrow \mathsf{Distill}(\mathrm{mpk}, \mathsf{ID}^*)$
$d^* \leftarrow \mathsf{Extract}(\mathrm{mpk}, \mathsf{ID}^*, d_{()})$
$b' \leftarrow \mathcal{A}(\mathrm{St}, e^*, d^*)$
$\textit{Return } b'$
```

We insist that the pattern P output by the adversary has at least one $\star$ in it.

We say that the scheme Π is random identity hiding if for any probabilistic polynomial time adversary $\mathcal{A}$ its advantage:

$$\mathbf{Adv}^{\mathsf{RIdH}}_{\Pi,\mathcal{A}}(1^k) = \Pr\left[\mathbf{Exp}^{\mathsf{RIdH}-1}_{\Pi,\mathcal{A}}(1^k) = 1\right] - \Pr\left[\mathbf{Exp}^{\mathsf{RIdH}-0}_{\Pi,\mathcal{A}}(1^k) = 1\right]$$

is negligible.

An important observation related to the generality of our results is that most of the existing HIBE constructions (e.g. BB [5],BBG [6] and Waters [27]) are both canonical and random identity hiding. We prove this for our running example of the BBG HIBE (the proof is in Appendix D.)

Theorem 1. *The Boneh-Boyen-Goh HIBE is random identity hiding.*

4 Identity Based Group Signatures

As discussed in the introduction much prior work on ID-Based group signatures has looked at the case where group members are given by "unstructured" identities, but the verification key used by the group is still a public key in the classical sense of the word. In this section we present a concept of group signatures in the ID-based setting, our security models and syntax are a subset of those of Wei et al [28]. We concentrate on the two security notions full-anonymity (the identity of the signer is hidden) and full-traceability (a signer can be identified by the group manager). We model a setting where the same set of public parameters (generated by a trusted centre) is used to setup multiple groups of signers (for different group identities).

Syntax. An *ID-based group signature scheme* consists of six polynomial time algorithms:

$$(\mathsf{Setup}, \mathsf{GrpSetUp}, \mathsf{Join}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Open}),$$

- $\mathsf{Setup}(1^k)$. This generates a master public/private key pair $(\mathrm{mpk}, \mathrm{msk})$.
- $\mathsf{GrpSetUp}(\mathrm{grpID}, \mathrm{msk})$. This algorithm on input of a string, which identifies the group; outputs a group secret key gsk. This secret key is then given to the group manager.
- $\mathsf{Join}(\mathrm{userID}, \mathrm{gsk})$. This algorithm executed by the group manager outputs a user secret key usk, which is passed to the group member. We assume that the group manager keeps a list of the member identities (say be adding them into gsk).
- $\mathsf{Sign}(m, \mathrm{usk})$. This algorithm produces a signature σ on the message m from the group for which usk corresponds.
- $\mathsf{Verify}(m, \sigma, \mathrm{mpk}, \mathrm{grpID})$. This outputs true if the signature σ is on the message m and was issued by the someone in the group grpID, otherwise it should output false.
- $\mathsf{Open}(\mathrm{gsk}, \sigma, m)$. This returns the identifier of the user who produced the signature σ on the message m. Note that in some situations the message m need not be input to the Open algorithm. This algorithm is run by the group manager.

For correctness we require that if gsk is the group secret key corresponding the group with identifier grpID, then

1. $\mathsf{Verify}\left(m, \mathsf{Sign}(m, \mathsf{Join}(\mathrm{userID}, \mathrm{gsk})), \mathrm{mpk}, \mathrm{grpID}\right) = \mathrm{true}$
2. $\mathsf{Open}\left(\mathrm{msk}, \mathsf{Sign}(m, \mathsf{Join}(\mathrm{userID}, \mathrm{gsk})), m\right) = \mathrm{userID}.$

Security models. To define the security of ID-based group signatures we extend the model introduced by Bellare et. al. [3] to this setting. Specifically, we cast the properties of *full-anonymity* (signatures do not reveal information about the signer) and *full-traceability* (the identity of the signer can be recovered by the group manager) to the ID-based setting. These security notions are well-established by now, so we will not repeat the ideas behind their design.

Anonymity is captured by an indistinguishability experiment between an adversary and the group signature. The adversary has full control over the scheme: can create new groups (and obtain the group manager's key), can add users to group (and obtain their signing keys), open signatures at will etc. These capabilities are modelled by appropriate access to several oracles. At some point the adversary outputs a group identity, two identities of group members and a message. It receives in return a signature on that messages, created with the secret key an identity selected at random between the two output by the adversary. The goal of the adversary is to guess which of the users created the signature. Of course, we impose the minimal requirements that the adversary does not know the master secret used for setup, and the opening key associated to the group under attack.

Definition 3 (Full-Anonymity). *Let* $\Pi = (\mathsf{Setup}, \mathsf{GrpSetUp}, \mathsf{Join}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Open})$ *be an identity based group signature. Consider the experiment* $\mathbf{Exp}^{\mathsf{anon}-b}_{\Pi,\mathcal{A}}(1^k)$ *that involves an adversary* $\mathcal{A}$ *and is parametrised by bit* b*. The experiment uses* $\mathrm{msk}, \mathrm{mpk}$ *as global variables. It also maintains two lists* **grpIDs** *(used to record the manager secret keys of the groups) and* **userIDs** *(used to record the secret signing keys of users, for the various groups to which they belong), as global variables. Initially both these lists are empty. During the experiment, the adversary has access to the following three oracles:*

- *Oracle* $\mathsf{GrpSetUp}(\cdot)$ *on input a query* $\mathrm{grpID} \in \mathsf{IdSp}$ *the oracle checks the list* **grpIDs** *for an entry* $(\mathrm{grpID}, \mathrm{gsk})$*. If such an entry is found, then* gsk *is returned to the adversary. Otherwise, the oracle executes* $\mathrm{gsk} \leftarrow \mathsf{GrpSetUp}(\mathrm{msk}, \mathrm{grpID})$*, adds* $(\mathrm{grpID}, \mathrm{msk})$ *to the list* **grpIDs** *and returns* gsk *to the adversary.*
- *Oracle* $\mathsf{Join}(\cdot)$ *is given as input a pair* $(\mathrm{grpID}, \mathrm{userID})$*. If the list* **grpIDs** *does not contain an element of the form* $(\mathrm{grpID}, \mathrm{gsk})$ *then the oracle executes* $\mathrm{gsk} \leftarrow \mathsf{GrpSetUp}(\mathrm{msk}, \mathrm{grpID})$ *and adds* $(\mathrm{grpID}, \mathrm{gsk})$ *to* **grpIDs**.
  Assuming now that **grpIDs** *contains an element of the form* $(\mathrm{grpID}, \mathrm{gsk})$*, if the list* **userIDs** *contains an element of the form* $((\mathrm{grpID}, \mathrm{userID}), \mathrm{usk})$ *then* usk *is returned to the adversary. Otherwise, the oracle runs* $\mathrm{usk} \leftarrow \mathsf{Join}(\mathrm{gsk}, \mathrm{userID})$ *to obtain a signing key for user identity returns the user signing key for that group.*
- *The* $\mathsf{Open}(\cdot)$ *oracle on input a tuple* $(\mathrm{grpID}, \sigma, m)$ *that consists of a group identity, a message* m *and a signature* σ *on* m *(valid for the group* grpID*), finds a pair* $(\mathrm{grpID}, \mathrm{gsk})$ *in* **grpIDs** *and then returns to the adversary* $\mathrm{userID} \leftarrow \mathsf{Open}(\mathrm{gsk}, \sigma, m)$*.*

The experiment proceeds as follows:

```
$\mathbf{Exp}^{\mathsf{anon}-b}_{\Pi,\mathcal{A}}(1^k)$
    $(\mathrm{mpk}, \mathrm{msk}) \leftarrow \mathsf{Setup}(1^k)$.
    $(\mathrm{grpID}^*, \mathrm{userID}_0, \mathrm{userID}_1, m, \mathrm{state}) \leftarrow A^{\mathsf{GrpSetUp}(),\mathsf{Join}(),\mathsf{Open}()}(\mathrm{mpk})$
    $b \leftarrow \{0, 1\}$
    $\sigma^* \leftarrow \mathsf{Sign}(m, \mathrm{usk})$, where $((\mathrm{grpID}^*, \mathrm{userID}_b), \mathrm{usk})$ is an entry in userIDs.
    $d \leftarrow A_2^{\mathsf{GrpSetUp}(),\mathsf{Join}(),\mathsf{Open}()}(\sigma^*, \mathrm{state})$.
    Return $d = b$.
```

The experiment only makes sense if the adversary is not allowed to call the $\mathsf{GrpSetUp}$ *oracle on* grpID^* *and is not allowed to call the* Open *oracle on* $(\mathrm{grpID}, \sigma^*, m^*)$*. We call such an adversary a* proper *one. We say that scheme* Π *is fully-anonymous if for any proper adversary* $\mathcal{A}$*, its advantage:*

$$\mathbf{Adv}^{\mathsf{anon}}_{\mathcal{A},\Pi}(k) = \Pr\left[\mathbf{Exp}^{\mathsf{anon}-1}_{\mathcal{A},\Pi}(k) = 1\right] - \Pr\left[\mathbf{Exp}^{\mathsf{anon}-0}_{\mathcal{A},\Pi}(k) = 1\right],$$

is negligible.

The second security property that we demand from group signatures is full-traceability: a signer, or a group of signers cannot produce a valid signature which the group manager cannot trace to one of the signers. This is a notion which itself implies the notion of unforgeability of the resulting signatures. The game that we consider involves an adversary with similar powers as the one in the previous experiment. The adversary can setup groups, add users to groups, see signatures of users of his choice, and open arbitrary signatures. In this experiment however we keep track of the set of corrupt users (users for which the adversary learns the signing key). The goal of the adversary is to produce a valid signature on a message of his choosing, which when opened by the group manager is not traced to one of the corrupt users.

Definition 4 (Full-Traceability). *The experiment* $\mathbf{Exp}^{\mathsf{trace}}_{\Pi,\mathcal{A}}(k)$ *used to define full traceability of IDGS scheme* $\Pi = (\mathsf{Setup}, \mathsf{GrpSetUp}, \mathsf{Extract}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Open})$ *involves an adversary* $\mathcal{A}$*. The experiment maintains three lists:* **corrgrpIDs** *keeps track of the corrupt identities in each of the groups of signers, and* **grpIDs** *and* **userIDs** *have the same use as in the experiment for anonymity). During the experiment the adversary may access the following five oracles:*

- *Oracle* $\mathsf{GrpSetUp}(\cdot)$ *on input a query* $(\mathrm{grpID}, type) \in \mathsf{IdSp} \times \{h, c\}$ *the oracle checks the list* **grpIDs** *for an entry* $(\mathrm{grpID}, \mathrm{gsk})$*. If such an entry is found, then* gsk *is returned to the adversary. Otherwise, the oracle executes* $\mathrm{gsk} \leftarrow \mathsf{GrpSetUp}(\mathrm{msk}, \mathrm{grpID})$*, adds* $(\mathrm{grpID}, \mathrm{msk})$ *to the list* **grpIDs***. If* $type = c$ *then it the oracle returns* gsk*.*
- *Oracle* $\mathsf{Join}(\cdot)$ *is given as input* $((\mathrm{grpID}, \mathrm{userID}), type) \in (\mathsf{IdSp} \times \mathsf{IdSp}) \times \{h, c\}$*. If the list* **grpIDs** *does not contain an element of the form* $(\mathrm{grpID}, \mathrm{gsk})$ *then the oracle computes* gsk *via* $\mathrm{gsk} \leftarrow \mathsf{GrpSetUp}(\mathrm{msk}, \mathrm{grpID})$ *and adds* $(\mathrm{grpID}, \mathrm{gsk})$ *to* **grpIDs***.*
  Assuming that **grpIDs** *contains an element of the form* $(\mathrm{grpID}, \mathrm{gsk})$*, the oracle runs* $\mathrm{usk} \leftarrow \mathsf{Join}(\mathrm{gsk}, \mathrm{userID})$*, and it adds the tuple* $((\mathrm{grpID}, \mathrm{userID}), \mathrm{usk})$ *to* **userIDs***.*
  If $type = c$ *then the oracle adds* $(\mathrm{grpID}, \mathrm{userID})$ *to* **corrgrpIDs** *and returns* usk*.*
- *Oracle* Sign *on input a tuple* $((\mathrm{grpID}, \mathrm{userID}), m)$ *the oracle searches* **userIDs** *for an entry of the form* $((\mathrm{grpID}, \mathrm{userID}), \mathrm{usk})$*. If such an entry does not exist it returns* $\perp$*. Otherwise, the oracle computes* $\sigma \leftarrow \mathsf{Sign}(\mathrm{usk}, m)$ *and returns* σ*.*
- *Oracle* Open *on input a tuple* $(\mathrm{grpID}, \sigma, m)$ *searches the* **grpIDs** *for an entry* $(\mathrm{grpID}, \mathrm{gsk})$*. If such an entry does not exist, it returns* $\perp$*. Otherwise it returns the result of* $\mathsf{Open}(\mathrm{gsk}, \sigma, m)$*.*

The experiment that defines security is as follows:

```
$\mathbf{Exp}^{\mathsf{trace}}_{\Pi,\mathcal{A}}(k)$
    $(\mathrm{mpk}, \mathrm{msk}) \leftarrow \mathsf{Setup}(1^k)$
    $(m, \sigma, \mathrm{grpID}^*) \leftarrow \mathcal{A}^{\mathsf{GrpSetUp}(),\mathsf{Join}(),\mathsf{Sign}(),\mathsf{Open}()}(\mathrm{mpk})$.
    Let $\mathrm{gsk}^* = \mathsf{Extract}(\mathrm{msk}, \mathrm{grpID}^*)$
    If $\mathsf{Verify}(m, \sigma, \mathrm{mpk}, \mathrm{grpID}^*) = \mathrm{false}$ or
        $(\mathrm{grpID}^*, \mathsf{Open}(\mathrm{gsk}^*, \sigma, m)) \in \mathbf{corrgrpIDs}$
    Then Return $0$
    Else Return $1$
```

The experiment only makes sense if the adversary does not request the group manager key for group grpID^* *(i.e. it does not make a query* $(grpID^*, c)$ *to the* $\mathsf{GrpSetUp}$ *oracle). We call such an adversary proper. The scheme* Π *is a fully traceable if for any proper adversary its advantage, defined by:*

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

is negligible.

5 Generic HIBE-based Construction of an ID-Based Group Signature

In this section we detail a generic construction of a ID-based group signature from a HIBE.

Outline: The construction is based on the following idea. We setup a four level HIBE and identify the root with the trusted authority that generates the parameters of the systems. Then, the first level corresponds to the various groups of signers. To create a new group of signers with public key grpID, the trusted authority produces the secret key associated to identity (grpID) in the HIBE and hands that as the group manager's key. This key is to be used for both adding members to the group, and as opening signatures to discover the underlying signer. To add a new group member userID to the group grpID, the group manager uses its secret key to compute the secret key associated to the hierarchical identity (grpID, userID). The resulting key $d_{(\mathrm{grpID},\mathrm{userID})}$ is the key that user userID uses to sign messages on behalf of the group grpID. User userID member of the group grpID, signs a message m as follows: it selects a random basic identity $\mathrm{r_{ID}}$ in IdSp, computes a distilled key e associated to identity $(\mathrm{grpID}, \mathrm{userID}, m, \mathrm{r_{ID}})$, and then uses its secret key to compute the decryption key d associated to e. The pair (e, d) is part of the signature that is output. The idea here is that since the HIBE is random identity hiding, the key e does not reveal any information about $(\mathrm{grpID}, \mathrm{userID}, m, \mathrm{r_{ID}})$ which is a random identity (due to the randomisation introduced by $\mathrm{r_{ID}}$.) We also need to ensure that the manager is able to recover the identity of the signer. For this we ask that the signer encrypts his identity under the identity of the group manager (i.e. under grpID) and then proves in zero-knowledge that the identity userID that had been encrypted under grpID is the same as the identity used in $(\mathrm{grpID}, \mathrm{userID}, m, \mathrm{r_{ID}})$ to distill e. Here, we use a non-interactive proof obtained from a Σ protocol via the Fiat–Shamir transform.

As pointed out in the introduction, one could avoid the random oracle by using a non-interactive simulation sound zero knowledge protocol. However, finding practically efficient instantiations of such proofs for the language that we need for our construction seems to be difficult. However, we note that using the random oracle model not only produces a gain in efficiency, the proof also becomes conceptually simpler due to the stronger properties of the proof of knowledge. Secondly, our specific constructions via the BBG HIBE uses the random oracle model, thus using the random oracle model in the overall construction does not loose us anything. We however point out that a proof of the generic construction in the standard model can be given.

The construction: We first define the NP-language that captures the desired relation between distilled keys and encrypted identities sketched above. For a fixed public key mpk, part of the parameters of a HIBE scheme ($\mathsf{Setup}, \mathsf{Distill}, \mathsf{Encr}, \mathsf{Extr}, \mathsf{Decrypt}$), and a bijection f between the space of basic identities IdSp and the plaintext space for the HIBE, we define the following NP relation:

$$\mathcal{R}((e, enc, \mathrm{grpID}, m), (\mathrm{userID}, \mathrm{r_{ID}}, r) = 1$$

if and only if

$$e = \mathsf{Distill}((\mathrm{grpID}, \mathrm{userID}, m, \mathrm{r_{ID}}), \mathrm{mpk}) \wedge enc = \mathsf{Encrypt}(\mathrm{grpID}, f(\mathrm{userID}); r))$$

Informally, an element $(e, enc, \mathrm{grpID}, m)$ of the language $L_{\mathcal{R}}$ defined by the relation $\mathcal{R}$ in the usual way satisfies the property that the user identity userID used to obtain the distilled key e equals the identity that had been encrypted under grpID to produce the ciphertext enc.

Given a canonical HIBE scheme ($\mathsf{Setup}H, \mathsf{Distill}, \mathsf{Extr}, \mathsf{Encr}, \mathsf{Decrypt}$), a Σ-protocol $(\mathcal{P}, \mathcal{V})$ for the language $L_{\mathcal{R}}$ above, and a hash function H (which we model as a random oracle) we construct an ID-based group signature scheme $\mathsf{GS}(\mathsf{HIBE}, (\mathcal{P}, \mathcal{V}), \mathsf{H}) = (\mathsf{Setup}G, \mathsf{GrpSetUp}, \mathsf{Join}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Open})$.
The algorithms are summarised in Figure 1. They work as follows.

Setup.The parameter setup algorithm $\mathsf{Setup}G$ simply runs the setup algorithm for the underlying HIBE scheme, and sets up a 4-level HIBE with public key mpk and secret key msk. The secret key of the trusted authority is set to msk.

Group setup. To setup a new group for identity grpID, the authority hands over to the group manager the secret key d_{grpID} associated to the hierarchical identity (grpID). User userID is added to the group of signers with public identity grpID by giving him the key $d_{(\mathrm{grpID},\mathrm{userID})}$ associated to the hierarchical identity (grpID, userID). Notice that this key enables the user userID to compute the associated key of any hierarchical identity to which (grpID, userID) is a parent.

```
$\mathsf{Setup}_G(1^k)$
    $(\mathrm{mpk}, \mathrm{msk}) \leftarrow \mathsf{Setup}_H(1^k, 4)$
    Return $(\mathrm{mpk}, \mathrm{msk})$
```

```
$\mathsf{GrpSetUp}(\mathrm{msk}, \mathrm{grpID})$
    $e \leftarrow \mathsf{Distill}((\mathrm{grpID}))$;
    $d_{\mathrm{grpID}} \leftarrow \mathsf{Extr}(\mathrm{msk}, e)$
    Return $(\mathrm{grpID}, d_{\mathrm{grpID}})$
```

```
$\mathsf{Sign}(m, (\mathrm{grpID}, \mathrm{userID}, \mathrm{d_{ID}}))$
    $\mathrm{r_{ID}} \leftarrow \mathsf{IdSp}$
    $e \leftarrow \mathsf{Distill}((\mathrm{grpID}, \mathrm{userID}, m, \mathrm{r_{ID}}), \mathrm{mpk})$
    $d \leftarrow \mathsf{Extr}(\mathrm{d_{ID}}, e)$
    $enc \leftarrow \mathsf{Encrypt}(\mathrm{mpk}, \mathrm{grpID}, f(\mathrm{userID}); r)$
    $\pi \leftarrow \mathsf{FS}_{\mathcal{P}}((e, enc, \mathrm{grpID}, m),$
                $(\mathrm{userID}, \mathrm{r_{ID}}, r))(m)$
    Return $(e, d, enc, \pi)$
```

```
$\mathsf{Verify}(m, \sigma, \mathrm{mpk}, \mathrm{grpID})$
    Parse $\sigma$ as $(e, d, enc, (r, s))$.
    If $\mathcal{V}(r, H(\mathrm{mpk}||e||enc||m||r), s) = 0$
        Then Return 0
    Else
        $m \leftarrow \mathcal{M}$
        If $m = \mathsf{Decrypt}(d, \mathsf{Encr}(e, m))$
            Then Return 1
            Else Return 0
```

```
$\mathsf{Open}(\mathrm{gsk}, \sigma, m)$
    Parse $\sigma$ as $(e, d, enc, (r, s))$
    Output $f^{-1}(\mathsf{Decrypt}(\mathrm{gsk}, c))$
```

```
$\mathsf{Join}((\mathrm{grpID}, d_{\mathrm{grpID}}), \mathrm{userID})$
    $e \leftarrow \mathsf{Distill}((\mathrm{grpID}, \mathrm{userID}), \mathrm{mpk})$
    $d \leftarrow \mathsf{Extr}(d_{\mathrm{grpID}}, (\mathrm{grpID}, \mathrm{userID}))$
    Return $(\mathrm{grpID}, \mathrm{userID}, d)$
```

Fig. 1. Generic construction of an ID-based group signature scheme from a canonical HIBE

Signing. To produce a signature on message m, user userID uses distills the public key e associated to $(\mathrm{grpID}, \mathrm{userID}, m, \mathrm{r_{ID}})$ (for a randomly chosen $\mathrm{r_{ID}}$) and uses his secret key to compute an associated decryption key d. Next, he encrypts the identity userID under the identity of the group. Finally, it uses the Fiat–Shamir transform to produce a non-interactive zero knowledge proof Σ that $(e, d, enc, \mathrm{grpID}, m)$ belong to the language $L_{\mathcal{R}}$ described above. The signature is then (e, d, enc, Σ).

Verification. A signature (e, d, enc, Σ) for message m and public key grpID is verified by first checking that Σ proves that $(e, enc, \mathrm{grpID}, m) \in L_{\mathcal{R}}$, and then checking that d is a valid decryption key for e. The second part of the verification is done by encrypting a random message under e and decrypting the resulting ciphertext with d.

Open. To open a signature (e, d, enc, Σ) for message m, the group manager grpID decrypts e using his secret key, and obtains the encrypted identity which it then outputs.

Instantiation based on BBG HIBE. In Appendix E we present our generic construction applied to the BBG HIBE in detail. The rest of the main body of the paper is devoted to showing that our generic construction meets our security definitions.

6 Security of our Construction

In this section we discuss the security of our generic construction. we start with the anonymity property. The intuition here is that a signature (e, d, enc, Σ) does not leak information about the identity of its creator since e is obtained from a random identity, the encryption enc hides its underlying plaintext, and Σ is a zero-knowledge proof. Since our construction uses the Fiat–Shamir heuristic, in addition to the above conditions we also need to require that the underlying proof system has high-entropy commitment and challenges (or alternatively, that the commitments and challenges are distributed uniformly over large enough spaces). These requirements ensure that rewinding strategies work in extracting necessary secrets.

Theorem 2. *Let* HIBE *be a HIBE scheme,* $(\mathcal{P}, \mathcal{V})$ *a proof system for the language* $L_{\mathcal{R}}$ *(defined above), and* H *a random oracle. If* HIBE *is an* IND-ID-CCA*, (respectively* IND-ID-CPA*) HIBE scheme which is random identity hiding, the proof system* $(\mathcal{P}, \mathcal{V})$ *has high-entropy commitments and challenges, and satisfies special soundness and special zero-knowledge, then* $\mathsf{GS}(\mathsf{HIBE}, (\mathcal{P}, \mathcal{V}), \mathsf{H})$ *is a fully-anonymous (respectively fully-anonymous under CPA attacks) identity-based group signature scheme.*

Proof. We give the details of the proof in Appendix F. Here we only sketch its steps. Recall that at some point during his execution the adversary produces at a challenge $(\text{grpID}^*, \text{userID}_0^*, \text{userID}_1^*, m)$ which is a request for a signature of either one of the two users on behalf of the group grpID^*, on message m. It receives from the experiment for anonymity a signature of the form $(e, d, enc, \varSigma)$, where e is a key distilled from $(\text{grpID}, \text{userID}_b, m, \text{r}_{\text{ID}})$ for some random identity r_{ID}, d is a decryption key associated to e, *enc* is an encryption of userID_b and $\varSigma$ is a non-interactive proof (obtained via the Fiat–Shamir heuristic) that the signature is well formed.

The proof of security uses a standard game-hopping technique. We incrementally change the security game for encryption until it reaches a form in which the adversary cannot win. The transformations that we do are as follows. First, we replace the zero-knowledge proofs computed using Fiat–Shamir with proofs computed by the simulator associated to the proof system. The new way of computing the proofs requires programming of the random oracle. The standard collision problem (when an entry in the table maintained by the random oracle needs to be programmed in order to produce a valid simulated proof, but programming cannot be done due to an early query) is avoided by requiring that the commitment messages of the prover has high entropy.

In the next transformation the key e part of the challenge signature is obtained from a completely random identity. The idea here is that an adversary would be able to tell that e is not related to either userID_0 or userID_1, then this adversary breaks the random-identity hiding property of the underlying HIBE scheme.

In the next step we prohibit the adversary from making a certain kind of open queries. Specifically, whenever the adversary makes a query for which the encryption part is *enc* (i.e. the encryption part of the challenge query) then the experiment aborts. The intuition here is as follows. Suppose that the adversary indeed makes such a query. For the query to be valid, the adversary needs to append an appropriate zero-knowledge proof, which by the soundness property he can only do if he has knowledge of an appropriate witness. However, such a witness contains the plaintext encrypted in *enc*, which means the adversary manages to break the security of the underlying HIBE. The proof uses a rewinding technique, which requires that the challenge space of the underlying proof system to be large.

In the final step, we replace the identity encrypted in *enc* with a random identity. As a result, the challenge signature that is returned to the adversary is independent of the challenge bit of the experiment, and thus in this final game the adversary can only win with probability half. The argument that the adversary does not see a difference between the resulting experiment and the previous one is based on the intuition that if this were not true, then the adversary observes the change in the encryption part of the signature, that is it somehow breaks encryption.

Next we show that our scheme is fully-traceable. The intuition is that the signature produced by a coalition of signers needs to contain the encryption *enc* of some identity grpID. At the same time in a well-formed signature the distilled key e that is part of the signature has to be obtained from a hierarchical identity of the form $\text{grpID}, \text{userID}, m, \text{r}_{\text{ID}}$ for the same userID as encrypted in *enc*. However,the only way one can compute a key d associated to e is if one knows the secret key associated to some identity on the path from the root to $(\text{grpID}, \text{userID}, m, \text{r}_{\text{ID}})$.

Theorem 3. *Let* HIBE *be a HIBE,* $(\mathcal{P}, \mathcal{V})$ *a proof system for the language* $L_{\mathcal{R}}$ *and* H *a random oracle. If* HIBE *is an* $\mathsf{IND\text{-}ID\text{-}CCA}$ *secure HIBE, and* $(\mathcal{P}, \mathcal{V})$ *satisfies special soundness and has high-entropy challenges, then* $\mathsf{GS}(\mathsf{HIBE}, (\mathcal{P}, \mathcal{V}), \mathsf{H})$ *is fully-traceable.*

Proof. We show that if there exists an adversary that breaks the full-traceability property of our construction, then we can construct an adversary that breaks the security of the underlying HIBE scheme. Consider a signature $(e, d, \mathsf{enc}, \varSigma)$ on message $\mathfrak{m}$ on behalf of group grpID which cannot be traced to a corrupt user. The intuition is that if the proof system used to produce $\varSigma$ is sound, then $(e, \mathsf{enc}, \text{grpID}, m) \in L_{\mathcal{R}}$, which in particular means that $e = \mathsf{Distill}(\text{grpID}, \text{userID}, m, \text{r}_{\text{ID}})$ for some userID and r_{ID}, and such that userID is encrypted in enc. Rewinding the adversary that produced this forgery, and using the special soundness of the proof system in use we can obtain $\text{userID}, \text{r}_{\text{ID}}$. Since the forgery was valid, we know that d is a decryption key for the hierarchical identity $(\text{grpID}, \text{userID}, m, \text{r}_{\text{ID}})$, which immediately results in a security break for HIBE.

Let $\mathcal{A}$ be an adversary for $\mathbf{Exp}^{\mathrm{trace}}_{\mathsf{GS},\mathcal{A}}(k)$. We construct the following adversary $\mathcal{B}$ for the game $\mathbf{Exp}^{\mathsf{IND\text{-}ID\text{-}CCA}}_{\mathsf{HIBE},\mathcal{B}}(k)$. Adversary $\mathcal{B}$ is given as input the public key mpk of a 4-level HIBE scheme HIBE and has access to a decryption oracle and an extraction oracle, both keyed with secret key msk (See Definition 1). Adversary $\mathcal{B}$ runs a simulation of the experiment $\mathbf{Exp}^{\mathrm{trace}}_{\mathsf{GS},\mathcal{A}}(k)$ for $\mathcal{A}$. In particular it maintains the list L of the random oracle H to which $\mathcal{A}$ also has access. Adversary $\mathcal{B}$ also maintains the variables **corrgrpIDs**, **grpIDs**, **userIDs** as in the experiment. To setup a new group grpID (i.e. to answer the queries that adversary $\mathcal{A}$ makes to its $\mathsf{GrpSetUp}$ oracle) $\mathcal{B}$ uses its access to the extraction oracle to obtain $d_{\mathrm{grpID}} \leftarrow \mathsf{Extract}(\mathrm{msk}, \mathrm{grpID})$. To answer add user userID to group grpID (i.e. to answer the queries that adversary $\mathcal{A}$ makes to its Join oracle) $\mathcal{B}$ obtains the associated secret key associated with hierarchical identity $(\mathrm{grpID}, \mathrm{userID})$ from its $\mathsf{Extract}$ oracle. Notice that $\mathcal{B}$ can easily answer all of the signing queries of $\mathcal{A}$ since $\mathcal{B}$ possesses all of the secret signing keys of the users in the system. Finally, to answer an Open query of the form $(\mathrm{grpID}, m, \sigma)$, adversary $\mathcal{B}$ parses the signature as (e, d, enc, Σ), checks the validity of the proof and submits (grpID, enc) to its decryption oracle. It returns the answer to the adversary. At some point adversary $\mathcal{A}$ outputs its tentative forgery: $(m, (e, d, enc, (r, c, s)), \mathrm{grpID})$. Since the simulation that $\mathcal{B}$ provides is perfect, this event occurs with probability $\mathbf{Adv}^{\mathrm{trace}}_{\mathsf{GS},\mathcal{A}}(k)$. Recall that if the forgery is valid, then (r, c, s) is an accepting transcript for $(e, enc, \mathrm{grpID}, m)$. Also, $c = \mathsf{H}(r||(e, enc, \mathrm{grpID}, m)||m)$. At this point adversary $\mathcal{B}$ rewinds the execution of $\mathcal{A}$ up to the point where it made the query $r||(e, enc, \mathrm{grpID}, m)||m$ to the random oracle and provides as answer $c' \leftarrow \mathsf{ChallSpace}$ with $c \neq c'$. Notice that the simulation of the experiment that $\mathcal{B}$ provides is perfect. It then follows by a standard rewinding argument that adversary $\mathcal{A}$ will produce a new valid forgery $(m, (e, d', enc, (r, c', s')))$ (i.e for the same message, and with the same query to the random oracle) with probability at least $\left(\frac{\mathbf{Adv}^{\mathrm{trace}}_{\mathsf{GS},\mathcal{A}}(k)}{q_H} - \frac{1}{\mathsf{ChallSpace}}\right)$. At this point adversary $\mathcal{B}$ has (r, c, s) and (r, c', s') which are two valid transcripts for the statement $(e, enc, \mathrm{grpID}, m) \in L_{\mathcal{R}}$. It executes $(\mathrm{userID}, \mathrm{r}_{\mathrm{ID}}, r) \leftarrow \mathcal{E}((r, c, s), (r, c', s'))$ to obtain a witness $(\mathrm{userID}, \mathrm{r}_{\mathrm{ID}}, r)$ for the statement. Notice that since the signature is valid, it is the case that d is a valid decryption key for the hierarchical identity $(\mathrm{grpID}, \mathrm{userID}, m, \mathrm{r}_{\mathrm{ID}})$. With this knowledge one can trivially win the $\mathsf{IND\text{-}ID\text{-}CCA}$ game for challenge identity $(\mathrm{grpID}, \mathrm{userID}, m, \mathrm{r}_{\mathrm{ID}})$. From the above discussion we have that:

$$\mathbf{Adv}^{\mathsf{IND\text{-}ID\text{-}CCA}}_{\mathsf{HIBE},\mathcal{B}}(k) \geq \mathbf{Adv}^{\mathrm{trace}}_{\mathsf{GS},\mathcal{A}}(k) \cdot \left(\frac{\mathbf{Adv}^{\mathrm{trace}}_{\mathsf{GS},\mathcal{A}}}{q_H} - \frac{1}{\mathsf{ChallSpace}}\right)$$

from which:

$$\mathbf{Adv}^{\mathrm{trace}}_{\mathsf{GS},\mathcal{A}}(k) \leq \frac{q_H}{\mathsf{ChallSpace}} + \sqrt{\frac{q_H^2}{4 \cdot \mathsf{ChallSpace}^2} + q_H \cdot \mathbf{Adv}^{\mathsf{IND\text{-}ID\text{-}CCA}}_{\mathsf{HIBE},\mathcal{B}}(k)}$$

Since the underlying HIBE is $\mathsf{IND\text{-}ID\text{-}CCA}$ secure it follows that GS is fully-traceable. The proof for the weaker full-traceability under CPA attacks is essentially the one above where the adversary against GS does not have an Open oracle, and thus $\mathcal{B}$ does not need a decryption oracle.

7 Extensions: Standard and Hierarchical Groups Signatures from HIBE

We conclude with an application of the core idea of our paper to a couple of extensions. Recall that the structure that we use to setup identity-based group signature is as follows. We use a four-level HIBE scheme where on the first level we place group identities, on the second level we place user identities, on the third level messages to be signed, while the fourth level is reserved to a randomiser. The group manager of group grpID can add user userID to the group by handing over the secret key associated to hierarchical identity $(\mathrm{grpID}, \mathrm{userID})$. A signature by this user on a message m is then a pair of encryption, decryption keys that correspond to the hierarchical identity $(\mathrm{grpID}, \mathrm{userID}, m, \mathrm{r}_{\mathrm{ID}})$ together with extra information to allow for the recovery of the signer and that ensure the signature is well-formed.

Standard group signatures. The first observation that we make is that by eliminating the first layer, that of group identities, we obtain a standard group signature in a non PKI setting. More precisely, the public key of the group is the public key mpk of the underlying HIBE. The group manager who has the corresponding secret key msk adds users by extracting the secret keys associated to their identity. Signatures can then be formed as before, with the difference that the encryption *enc* is under mpk, as opposed to group identity. The resulting scheme shares with standard group signature schemes the idea of having a "standard" public-key, and with ID-based signature scheme, as defined in this paper (and as previously considered in the literature) the idea that parties are identified by unstructured identities. The intuition regarding the security of the resulting scheme follows the same lines as those of the construction we detailed in this paper.

Hierarchical group signatures. The second extension that we propose is to hierarchical group signatures. Here, we would like for groups of signers be organised in a hierarchy so that users at the lower level can sign on behalf of any of the groups to which they belong. For example, in a university `UniId`, one could have subgroups `faculty` and `admin`. The faculty could then be divided into research group `research1`, `research2,...`, where as the admin group could be on specialised departments `finance`, `undergraduate,...`. Finally, individual users `user1,user2,...` belong to one of these lower level subgroups. In a hierarchical identity based signature, we would like that managers of groups be permitted to add users to the group that it manages, or to any of his group's subgroups. Also, we would like for a user to be able to produce, anonymously, signatures for any of the groups to which he belongs. Finally, a group manager should be able to open signatures created by any of the users in the group that it manages, no matter on behalf of which of subgroups of the group the signature was produced.

Our construction can be easily extended to this more complex setting. Instead of working with a four-level HIBE, we work with a $k + 4$ level HIBE, where k is the maximal number of subgroups that a group can have (for $k = 0$ we fall on the setting of our main construction). The construction that we suggest is to place on the first k levels the group identities, in a way that reflects the desired hierarchy. Creating new groups, and adding group members is then done as before: the group managers extracts a key for the appropriate hierarchical identity. For example, the manager of the group `UniId` creates the group `faculty` by extracting the key associated to the hierarchical identity (`UniId,faculty`). The key of a user would be the key associated to (`level1,level2,...,levelk,user`). Signatures in this construction generalise ours, with one exception. The user can choose for which of the groups to which it belongs produces the signature, and in particular, under which of the subgroup identities it encrypts his own identity. There is flexibility also who can open a signature: any group manager that is a parent identity to the one under which the user encrypts his identity can identify the signer. For our example, a faculty members that belongs to the group `research` can sign on behalf of that group, on behalf of the whole group `faculty`, or on behalf of the university `UniId`. Furthermore, only the manager of the group for which the signature is produced (or a parent of the manger) can identify the signer. The security of this construction relies on the same basic idea as that of our main construction of this paper.

References

1. M. Abdalla, A.W. Dent, J. Malone-Lee, G. Neven, D.H. Phan and N.P. Smart, Identity-based traitor tracing, In *Public Key Cryptography – PKC 2007*, Springer-Verlag LNCS 4450, 361–376, 2007.
2. P.S.L.M. Barreto, B. Libert, N. McCullagh and J.-J. Quisquater. Efficient and provably-secure identity-based signatures and signcryption from bilinear maps. In *Advances in Cryptology – AsiaCrypt 2005*, Springer-Verlag LNCS 3788, 515–532, 2005.
3. 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*, Springer-Verlag LNCS 2656, 614–629, 2003.
4. M.Bellare and G.Neven. Multi-signatures in the plain public-key model and a general forking lemma. In *ACM Conference on Computer and Communications Security – CCS 2006*, ACM Press, 390–399, 2006.
5. D. Boneh and X. Boyen. Efficient selective-ID secure identity based encryption without random oracles. In *Advances in Cryptology – EuroCrypt 2004*, Springer-Verlag LNCS 3027, 223–238, 2004.
6. D. Boneh, X. Boyen and E.-J. Goh. Hierarchical identity based encryption with constant size ciphertext. In *Advances in Cryptology – EuroCrypt 2005*, Springer-Verlag LNCS 3494, 440–456, 2005.
7. D. Boneh and M. Franklin. Identity based encryption from the Weil pairing. In *Advances in Cryptology - Crypto 2001*, Springer-Verlag LNCS 2139, 213–229, 2001.

8. X. Boyen and B. Waters. Compact group signatures. In *Advances in Cryptology – EuroCrypt 2006*, Springer-Verlag LNCS 4004, 427–444, 2006.
9. R. Canetti, S. Halevi and J. Katz. Chosen-ciphertext security from identity based encryption. In *Advances in Cryptology – EuroCrypt 2004*, Springer-Verlag LNCS 3027, 207–222, 2004.
10. J.C. Cha and J.H. Cheon. An identity-based signature from gap Diffie-Hellman groups. In *Public Key Cryptography – PKC 2003*, Springer-Verlag LNCS 2567, 18–30, 2003.
11. M. Chase and A. Lysyanskaya. On signatures of knowledge. In *Advances in Cryptology – Crypto 2006*, Springer-Verlag LNCS 4117, 78–96, 2006.
12. L. Chen, Z. Cheng, J. Malone-Lee and N.P. Smart. Efficient ID-KEM based on the Sakai-Kasahara key construction. *IEE Proceedings - Information Security*, **153**, 19–26, 2006.
13. X. Chen, F. Zhang and K. Kim. A new ID-based group signature scheme from bilinear pairings. IACR e-Print, `eprint.iacr.org/2003/116.pdf`, 2003.
14. C. Cocks. An identity-based encryption scheme based on quadratic residues. In *Proceedings of Cryptography and Coding 2001*, Springer-Verlag LNCS 2260, 360–363, 2001.
15. C. Gentry and A. Silverberg. Hierarchical ID-based cryptography. In *Advances in Cryptology – Asiacrypt 2002*, Springer-Verlag LNCS 2501, 548–566, 2002.
16. J. Groth and A. Sahai. Efficient non-interactive proof systems for bilinear groups. In *Advances in Cryptology – EuroCrypt 2008*, Springer-Verlag LNCS 4965, 415–432, 2008.
17. F. Hess. Efficient identity based signature schemes based on pairings. In *Selected Areas in Cryptography – SAC 2002*, Springer-Verlag LNCS 2595, 310–324, 2003.
18. S. Han, J. Wang and W. Liu. An efficient identity-based group signature scheme over elliptic curves. In *Universal Multiservice Networks*, Springer-Verlag LNCS 3262, 417–429, 2004.
19. S. Park, S. Kim and D. Won. ID-based group signature. *Electronics Letters*, **33**, 1616–1617, 1997.
20. C. Popescu. An efficient ID-based group signature scheme. Studia Univ. Babes-Bolyai Info., **47**, 29–36, 2002.
21. R. Sakai, K. Ohgishi and M. Kasahara. Cryptosystems based on pairing. In *The 2000 Symposium on Cryptography and Information Security*, Okinawa, Japan, January 2000.
22. R. Sakai, K. Ohgishi and M. Kasahara. Cryptosystems based on pairing over elliptic curve (in Japanese). In *The 2001 Symposium on Cryptography and Information Security*, Oiso, Japan, January 2001.
23. R. Sakai and M. Kasahara. ID based cryptosystems with pairing on elliptic curve. Cryptology ePrint Archive, Report 2003/054. 2003.
24. A. Shamir. Identity-based cryptosystems and signature schemes. In *Advances in Cryptology - Crypto '84*, Springer-Verlag LNCS 196, 47–53, 1985.
25. M. Trolin and D. Wikström. Hierarchical group signatures. In *Automata, Languages and Programming – ICALP 2005*, Springer-Verlag LNCS 3580, 446–458, 2005.
26. Y. Tseng and J. Jan. A novel ID-based group signature. *Int. Comp. Symp. on Crypto and Info. Sec.*, 159–164, 1998.
27. B.R. Waters. Efficient identity-based encryption without random oracles. In *Advanced in Cryptology – EuroCrypt 2005*, Springer-Verlag LNCS 3494, 114–127, 2005.
28. V.K. Wei, T.H. Yuen and F. Zhang. Group signature where group manager, members and open authority are identity-based. In *Information Security and Privacy – ACIPS 2005*, Springer-Verlag LNCS 3574, 468–480, 2005.

A Notation

Throughout the appendices we use the following notational conventions. Our explicit constructions are all based on an asymmetric pairing $\hat{t} : \mathbb{G} \times \hat{\mathbb{G}} \longrightarrow \mathbb{G}_T$, between three groups of prime order q. We assume that $\mathbb{G} = \langle g \rangle$ and $\hat{\mathbb{G}} = \langle \hat{g} \rangle$. Elements of $\mathbb{G}$ will be denoted by lower case letters a, b, c etc, elements of $\hat{\mathbb{G}}$ will be denoted by $\hat{a}, \hat{b}, \hat{c}$ etc, elements of $\mathbb{G}_T$ will be denoted by gothic letters $\mathfrak{a}, \mathfrak{b}, \mathfrak{c}$ etc.

B Special Zero-Knowledge for *Σ*-protocols

Let $(\mathcal{P}, \mathcal{V})$ be a Σ protocol for some NP-relation $\mathcal{R}$. In this paper we work with such protocols that are special zero-knowledge. That is, we require the existence of a simulator which for any $x \in L_{\mathcal{R}}$, given a challenge $c \in \mathsf{ChallSpace}$ outputs a (r, s) such that (r, c, s) is an accepting transcript for x. Moreover, we require that the distribution of the simulated transcripts is identical to that of real ones. More precisely we demand that for any $(x, w) \in \mathcal{R}$, for any (R, C, S) the probabilities

$$\Pr\left[(r, c, s) \leftarrow (\mathcal{P}(x, w), \mathcal{V}(x)) \ : \ (r, c, s) = (R, C, S)\right]$$

and

$$\Pr\left[c \leftarrow \mathsf{ChallSpace}, (r, s) \leftarrow \mathcal{S}(x, c) \ : \ (r, c, s) = (R, C, S)\right]$$

are equal.

Using the special zero-knowledge we show that in schemes where the proof system is used in its non-interactive version given by the Fiat–Shamir transform, the resulting proofs can be replaced with simulated proofs, if one can program the random oracle. This is true, if the commitments output by the prover have high entropy. Consider the games in Figure 2. The first game corresponds to the case where an adversary is given as input non-interactive proofs obtained by the Fiat–Shamir transform (using messages of adversary's choice), and the random oracle is not programmed. In the second experiment the adversary is given as input simulated proofs, where the random oracle is programmed.

```
$\mathbf{Exp}^{\mathsf{FS}\text{-}1}_{\mathcal{P},\mathcal{A}}(k)$
    $(St, m, (x, w)) \leftarrow A^{\mathsf{H}}(1^k)$
    $(st, r) \leftarrow \mathcal{P}(x, w)$
    $c \leftarrow \mathsf{H}(x||r||m)$
    $s \leftarrow \mathcal{P}(st, c)$
    $b \leftarrow \mathcal{A}^{\mathsf{H}}(St, (r, c, s))$

$\mathbf{Exp}^{\mathsf{FS}\text{-}0}_{\mathcal{P},\mathcal{A}}(k)$
    $(St, m, (x, w)) \leftarrow A^{\mathsf{H}}(1^k)$
    $c \leftarrow \mathsf{ChallSpace}$
    $(r, s) \leftarrow \mathcal{S}(x, c)$
    if $((r||x||m), c') \in L$ and $c \neq c'$
        output fail
    add $((r||x||m), c)$ to $L$
    $b \leftarrow \mathcal{A}^{\mathsf{H}}(St, (r, c, s)$

Oracle $\mathsf{H}$
    On input $x$
    Search $L$ for $(x, c)$.
    If found, Return $c$.
    Else $c \in \mathsf{ChallSpace}$
        Add $(x, c)$ to $L$.
    Return $c$.
```

Fig. 2. In the experiment on the left, the adversary is given a proof computed using the Fiat–Shamir transform. In the experiment in the middle, the proof given to the adversary is simulated. Here, the experiment can program the random oracle H to which the adversary has access in both experiments. The list L is the internal state of the random oracle to which the second experiment has access.

We claim that for any adversary $\mathcal{A}$, if $(\mathcal{P}, \mathcal{V})$ has high entropy commitments, then no adversary can distinguish between proofs obtained via the Fiat–Shamir transform (using a non-programmable random oracle) and proof simulated in a programmable random oracle mode. Formally, we argue that if the commitment space of the proof system is $\mathsf{CommitSpace}$, then

$$\mathbf{Adv}^{\mathsf{FS}\text{-}1}_{\mathcal{P},\mathcal{A}}(k) \leq \frac{q_{\mathsf{H}}}{\mathsf{CommitSpace}} \tag{1}$$

Indeed, the only way the adversary can observe a difference between the two experiments is if the second experiment aborts. This even only occurs if there is a collision in the list L maintained by oracle H. Let q_{H} be the number of hash queries made during the execution of $\mathcal{A}$. By the assumption that the commitments of the proof system have high-entropy i.e. for any $(x, w) \in \mathcal{R}$, for any $R \in \mathsf{CommitSpace}$

$$\Pr\left[(r, c, s) \leftarrow (\mathcal{P}(x, w), \mathcal{V}(x)) \ : \ r = R\right] \leq \frac{1}{\mathsf{CommitSpace}}.$$

By the assumption that the transcripts output by $\mathcal{S}$ are identically distributed with those of the original proof system, we have that

$$\Pr\left[c \leftarrow \mathsf{ChallSpace}; (r, s) \leftarrow \mathcal{S}(x, c) \ : \ r = R\right] \leq \frac{1}{\mathsf{CommitSpace}}$$

for any R. The desired result follows by a simple union bound.

C The Boneh-Boyen-Goh HIBE

The structural assumptions that we make on the encryption and extraction algorithms are without loss of generality in the context of currently know efficient HIBE constructions. In particular, we show that the algorithms of the Boneh-Boyen-Goh HIBE [6] satisfies our requirements, and describe it below in canonical form.

We present the scheme for a HIBE of length L, although in our construction we only use a HIBE of length 4. Again we present the scheme in the setting of asymmetric pairings, which means our notation is slightly different from that of [6].

Setup: The trusted authority chooses random values $\hat{g}_2, \hat{u}_0, \dots, \hat{u}_L \in \hat{\mathbb{G}}$ and a value $\alpha \in \mathbb{Z}_q$. The trusted authority then computes $h_1 \leftarrow g^{\alpha}$, $\hat{h}_2 \leftarrow \hat{g}_2^{\alpha}$, and sets $\mathrm{mpk} \leftarrow (g, \hat{g}_2, h_1, \hat{u}_0, \dots, \hat{u}_L)$ and $\mathrm{msk} \leftarrow \hat{h}_2$.

Extract: A user's identity is given by a vector $\mathsf{ID} = (\mathsf{id}_1, \dots, \mathsf{id}_l)$ with $l \leq L$. A random $r \leftarrow \mathbb{Z}_q$ is chosen and the private key is computed via

$$d_{\mathsf{ID}} = (\hat{a}_0, \hat{a}_{l+1}, \dots, \hat{a}_L, a_{L+1}) \leftarrow \left(\hat{h}_2 \left(\hat{u}_0 \cdot \prod_{i=1}^{l} \hat{u}_i^{\mathsf{id}_i} \right)^r, \hat{u}_{l+1}^r, \dots, \hat{u}_L^r, g^r \right).$$

The private key for the identity $\mathsf{ID} = (\mathsf{id}_1, \dots, \mathsf{id}_l)$ can be derived from the private key

$$(\hat{a}_0, \hat{a}_l, \dots, \hat{a}_L, a_{L+1})$$

of its parent identity $\mathsf{ID} = (\mathsf{id}_1, \dots, \mathsf{id}_{l-1})$ by selecting $r' \leftarrow \mathbb{Z}_q$ and computing

$$d_{\mathsf{ID}} \leftarrow \left(\hat{a}_0 \cdot \hat{a}_l^{\mathsf{id}_l} \left(\hat{u}_0 \prod_{i=1}^{l} \hat{u}_i^{\mathsf{id}_i} \right)^{r'}, \hat{a}_{l+1} \cdot \hat{u}_{l+1}^{r'}, \dots, \hat{a}_L \cdot \hat{u}_L^{r'}, a_{L+1} \cdot g^{r'} \right).$$

Distill: A user's identity is given by a vector $\mathsf{ID} = (\mathsf{id}_1, \dots, \mathsf{id}_l)$ with $l \leq L$. The output of this function is the value

$$\hat{e}_{\mathsf{ID}} = \prod_{i=1}^{l} \hat{u}_i^{\mathsf{id}_i}.$$

Encr: To encrypt a message $\mathfrak{m} \in \mathbb{G}_T$ to an identity $\mathsf{ID} = (\mathsf{id}_1, \dots, \mathsf{id}_l)$ encoding in $\hat{e}_{\mathsf{ID}}$, the sender selects $t \leftarrow \mathbb{Z}_q$ and outputs the ciphertext $(e_1, \hat{e}_2, \mathfrak{e}_3)$ where

$$e_1 \leftarrow g^t, \quad \hat{e}_2 \leftarrow (\hat{u}_0 \cdot \hat{e}_{\mathsf{ID}})^t, \quad \mathfrak{e}_3 \leftarrow \mathfrak{m} \cdot \hat{t}(h_1, \hat{g}_2)^t.$$

Decrypt: A receiver secret key $(\hat{a}_0, \hat{a}_{l+1}, \dots, \hat{a}_L, a_{L+1})$ decrypts a ciphertext $(e_1, \hat{e}_2, \mathfrak{e}_3)$ as follows

$$\begin{aligned} \mathfrak{m} &\leftarrow \mathfrak{e}_3 \cdot \frac{\hat{t}(a_{L+1}, \hat{e}_2)}{\hat{t}(e_1, \hat{a}_0)} \\ &= \mathfrak{e}_3 \cdot \frac{\hat{t}\left(g, \left(\hat{u}_0 \prod_{i=1}^{l} \hat{u}_i^{\mathsf{id}_i}\right)\right)^{rt}}{\hat{t}(g, \hat{h}_2) \cdot \hat{t}\left(g, \left(\hat{u}_0 \prod_{i=1}^{l} \hat{u}_i^{\mathsf{id}_i}\right)\right)^{rt}} \\ &= \mathfrak{m}, \end{aligned}$$

assuming his identity is $\mathsf{ID} = (\mathsf{id}_1, \dots, \mathsf{id}_l)$.

D The Boneh-Boyen-Goh HIBE is random identity hiding

Theorem 4. *The Boneh-Boyen-Goh HIBE is random identity hiding.*

Proof. This is immediate upon noticing that the tuple

$$(e_{\mathsf{ID}}, d_{\mathsf{ID}}) = \left(\prod_{i=1}^{l} \hat{u}_i^{\mathsf{id}_i}, \hat{h}_2 \left(\hat{u}_0 \cdot \prod_{i=1}^{l} \hat{u}_i^{\mathsf{id}_i} \right)^r, \hat{u}_{l+1}^r, \ldots, \hat{u}_L^r, g^r \right)$$

information-theoretically hides the values in the identities contained in ID, assuming at least one identity in ID is unknown. Since the game for random identity hiding requires the adversary to output a non-trivial pattern, this latter property holds.

E Instantiation using the BBG HIBE

In this appendix we detail how our generic construction applies to the BBG HIBE. A similar construction can be given for other HIBE constructions, by following the same basic principles.

The original Boneh-Boyen-Goh HIBE is proved secure in the selective ID setting, this is turned into full security via replacing the identities with calls to a hash function, which is then modelled as a random oracle. Hence, to obtain a group signature scheme which is anon-ID-CPA secure we introduce a hash function G to hash the identities to elements of $\mathbb{Z}_q$. We also require a hash function $H : \{0,1\}^* \to \mathbb{Z}_q$ for our proof of knowledge proof, which we also model as a random oracle.

In addition the following scheme is only secure in the sense of anon-ID-CPA, i.e. Open are not allowed in the adversary queries. A fully secure version is possible to construct, but we present the simpler version here for clarity.

In addition we have performed some elementary optimisations on the scheme which results from the generic construction. These do not affect security, but make use of the properties of the $\mathsf{Distill}$ function of the BBG HIBE. In particular the signature only contains the unknown part of the $\mathsf{Distill}$ function, since the other public part can be reconstructed by the verifier. This not only makes the presentation simpler, it also simplifies the proof of knowledge.

E.1 The Required Proof of Knowledge:

We present the proof of knowledge and its verification which are required in the Sign and Verify operations. To aid exposition we set

$$\hat{f} = (\hat{u}_0 \cdot \hat{u}_1^{G(\text{grpID})}) \text{ and } \mathfrak{g} = \hat{t}(h_1, \hat{g}_2).$$

Our proof of knowledge is then given by the underlying Σ protocol for the language

$$\mathcal{L} = \left\{ \hat{c}_6 = \hat{u}_2^x \cdot \hat{u}_4^y \ \wedge \ e_1 = g^z \ \wedge \ \hat{e}_2 = \hat{f}^z \ \wedge \ \mathfrak{e}_3 = \mathfrak{n}^x \cdot \mathfrak{g}^z \ : (x, y, z) \right\},$$

where all values bar x, y, z are public. The naming of the variables is to aid the reader in seeing how this proofs fits in with the variables in the ID-based group signature below.

Standard techniques provide the following construction of a non-interactive proof of knowledge, assuming H is modelled as a random oracle.

Prover's Algorithm: The prover generates $k_1, k_2, k_3 \in \mathbb{Z}_q$ at random and sets

$$\hat{r}_1 \leftarrow \hat{u}_2^{k_1} \cdot \hat{u}_4^{k_2}, \quad r_2 \leftarrow g^{k_3}, \quad \hat{r}_3 \leftarrow \hat{f}^{k_3}, \quad \mathfrak{r}_4 \leftarrow \mathfrak{n}^{k_1} \cdot \mathfrak{g}^{k_3}.$$

Then the prover computes

$$c \leftarrow H(\mathrm{grpID} \| \hat{u}_0 \| \hat{u}_1 \| \hat{u}_2 \| \hat{u}_4 \| g \| \hat{f} \| \mathfrak{n} \| \mathfrak{g} \| \hat{c}_6 \| e_1 \| \hat{e}_2 \| \mathfrak{e}_3 \| \hat{r}_1 \| r_2 \| \hat{r}_3 \| \mathfrak{r}_4).$$

Finally the prover computes

$$s_1 \leftarrow k_1 + c \cdot x, \quad s_2 \leftarrow k_2 + c \cdot y \text{ and } s_3 \leftarrow k_3 + c \cdot z.$$

The proof of knowledge is then given by (c, s_1, s_2, s_3).

Verifier's Algorithm: To verify the proof the verifier computes the values

$$\hat{r}_1' \leftarrow \hat{u}_2^{s_1} \cdot \hat{u}_4^{s_2} \cdot \hat{c}_6^{-c}, \quad r_2' \leftarrow g^{s_3} \cdot e_1^{-c}, \quad \hat{r}_3' \leftarrow \hat{f}^{s_3} \cdot \hat{e}_2^{-c}, \quad \mathfrak{r}_4' \leftarrow \mathfrak{n}^{s_1} \cdot \mathfrak{g}^{s_3} \cdot \mathfrak{e}_3^{-c},$$

and then checks whether

$$c = H(\mathrm{grpID} \| \hat{u}_0 \| \hat{u}_1 \| \hat{u}_2 \| \hat{u}_4 \| g \| \hat{f} \| \mathfrak{n} \| \mathfrak{g} \| \hat{c}_6 \| e_1 \| \hat{e}_2 \| \mathfrak{e}_3 \| \hat{r}_1' \| r_2' \| \hat{r}_3' \| \mathfrak{r}_4').$$

E.2 An ID-based group signature from the BBG HIBE

Setup(1^k): The trusted authority chooses random values $\hat{g}_2, \hat{u}_0, \hat{u}_1, \hat{u}_2, \hat{u}_3, \hat{u}_4 \in \hat{\mathbb{G}}$ and a value $\alpha \in \mathbb{Z}_q$. The trusted authority then computes $h_1 \leftarrow g^\alpha$, $\hat{h}_2 \leftarrow \hat{g}_2^\alpha$, generates an element $\mathfrak{n}$ at random from $\mathbb{G}_T$, and sets

$$\mathrm{mpk} \leftarrow (g, \hat{g}_2, h_1, \hat{u}_0, \hat{u}_1, \hat{u}_2, \hat{u}_3, \hat{u}_4, \mathfrak{n}) \text{ and } \mathrm{msk} \leftarrow \hat{h}_2.$$

GrpSetUp(grpID, msk): On input of a group identifier string grpID, the trust authority generates a random value $r_1 \in \mathbb{Z}_q$ and sets gsk $\leftarrow (\hat{a}_0, \hat{a}_2, \hat{a}_3, \hat{a}_4, a_5)$, where

$$\hat{a}_0 \leftarrow \hat{h}_2 \cdot \left(\hat{u}_0 \cdot \hat{u}_1^{G(\mathrm{grpID})}\right)^{r_1}, \quad \hat{a}_2 \leftarrow \hat{u}_2^{r_1}, \quad \hat{a}_3 \leftarrow \hat{u}_3^{r_1}, \quad \hat{a}_4 \leftarrow \hat{u}_4^{r_1}, \quad a_5 \leftarrow g^{r_1}.$$

Extract(userID, gsk): On input of a user identifier string userID the group manager takes its key gsk $= (\hat{a}_0, \hat{a}_2, \hat{a}_3, \hat{a}_4, a_5)$, generates a random value $r_2 \in \mathbb{Z}_q$, and computes the user secret key via usk $\leftarrow (\hat{b}_0, \hat{b}_3, \hat{b}_4, b_5)$ where

$$\begin{aligned}
\hat{b}_0 &\leftarrow \hat{a}_0 \cdot \hat{a}_2^{G(\mathrm{userID})} \cdot \left(\hat{u}_0 \cdot \hat{u}_1^{G(\mathrm{grpID})} \cdot \hat{u}_2^{G(\mathrm{userID})}\right)^{r_2} \\
&= \hat{h}_2 \cdot \left(\hat{u}_0 \cdot \hat{u}_1^{G(\mathrm{grpID})}\right)^{r_1} \cdot \hat{u}_2^{r_1 \cdot G(\mathrm{userID})} \cdot \left(\hat{u}_0 \cdot \hat{u}_1^{G(\mathrm{grpID})} \cdot \hat{u}_2^{G(\mathrm{userID})}\right)^{r_2} \\
&= \hat{h}_2 \cdot \left(\hat{u}_0 \cdot \hat{u}_1^{G(\mathrm{grpID})} \cdot \hat{u}_2^{G(\mathrm{userID})}\right)^{r_1+r_2}, \\
\hat{b}_3 &\leftarrow \hat{a}_3 \cdot \hat{u}_3^{r_2} = \hat{u}_3^{r_1+r_2}, \qquad \hat{b}_4 \leftarrow \hat{a}_4 \cdot \hat{u}_4^{r_2} = \hat{u}_4^{r_1+r_2}, \qquad b_5 \leftarrow a_5 \cdot g^{r_2} = g^{r_1+r_2}.
\end{aligned}$$

Sign(m, usk): To sign a message $m \in \mathbb{Z}_q$ using the secret key usk $= (\hat{b}_0, \hat{b}_3, \hat{b}_4,\ b_5)$ the user generates a random values $r_3 \in \mathbb{Z}_q$, and a random identity r_4. The value r_3 acts very much like the values r_1 and r_2 in the GrpSetUp and the Extract algorithms, whilst the value r_4 is used to create a blinding identity, so as to maintain user anonymity. In addition the signer picks an additional random values $k \in \mathbb{Z}_q$, so as to encrypt its identity to the group manager. A signature is given by

$$\sigma \leftarrow (\hat{c}_0, c_5, \hat{c}_6, e_1, \hat{e}_2, \mathfrak{e}_3, \Sigma)$$

where

$$\begin{aligned}
\hat{c}_0 &\leftarrow \hat{b}_0 \cdot \hat{b}_3^m \cdot \hat{a}_4^{G(r_4)} \cdot \left(\hat{u}_0 \cdot \hat{u}_1^{G(\text{grpID})} \cdot \hat{u}_2^{G(\text{userID})} \cdot \hat{u}_3^m \cdot \hat{u}_4^{G(r_4)}\right)^{r_3} \\
&= \hat{h}_2 \cdot \left(\hat{u}_0 \cdot \hat{u}_1^{G(\text{grpID})} \cdot \hat{u}_2^{G(\text{userID})}\right)^{r_1+r_2} \cdot \hat{u}_3^{m(r_1+r_2)} \cdot \hat{u}_4^{G(r_4)\cdot(r_1+r_2)} \cdot \\
&\qquad \left(\hat{u}_0 \cdot \hat{u}_1^{G(\text{grpID})} \cdot \hat{u}_2^{G(\text{userID})} \cdot \hat{u}_3^m \cdot \hat{u}_4^{G(r_4)}\right)^{r_3} \\
&= \hat{h}_2 \cdot \left(\hat{u}_0 \cdot \hat{u}_1^{G(\text{grpID})} \cdot \hat{u}_2^{G(\text{userID})} \cdot \hat{u}_3^m \cdot \hat{u}_4^{G(r_4)}\right)^{r_1+r_2+r_3}, \\
c_5 &\leftarrow b_5 \cdot g^{r_3} = g^{r_1+r_2+r_3}, \qquad \hat{c}_6 \leftarrow \hat{u}_2^{G(\text{userID})} \cdot \hat{u}_4^{G(r_4)}, \\
e_1 &\leftarrow g^k, \qquad \hat{e}_2 \leftarrow (\hat{u}_0 \cdot \hat{u}_1^{G(\text{grpID})})^k, \qquad \mathfrak{e}_3 \leftarrow \mathfrak{n}^{G(\text{userID})} \cdot \hat{t}(h_1, \hat{g}_2)^k, \\
\Sigma &\leftarrow \text{POK}\begin{pmatrix} \hat{c}_6 = \hat{u}_2^x \cdot \hat{u}_4^y \ \wedge\ e_1 = g^z \ \wedge\ \hat{e}_2 = \left(\hat{u}_0 \cdot \hat{u}_1^{G(\text{grpID})}\right)^z \ \wedge \\ \mathfrak{e}_3 = \mathfrak{n}^x \cdot \hat{t}(h_1, \hat{g}_2)^z \ :\ \quad (G(\text{userID}), G(r_4), k) \end{pmatrix}.
\end{aligned}$$

Note, that the value of $\hat{t}(h_1, \hat{g}_2)$ can be precomputed, we shall indeed denote this value by $\mathfrak{g}$ in what follows. Thus, signing requires no pairing computations.

Verify(m, σ, mpk, grpID): We verify the signature by essentially encrypting a random message under the underlying HIBE and then checking whether it decrypts to the correct value. On input of a signature $\sigma = (\hat{c}_0, c_5, \hat{c}_6, e_1, \hat{e}_2, \mathfrak{e}_3,\ \Sigma)$ on a message m, as issued by a member of the group grpID, the verifier generates the following random values $t \in \mathbb{Z}_q, \mathfrak{m} \in \mathbb{G}_T$ and computes

$$d_1 \leftarrow g^t, \quad \hat{d}_2 \leftarrow \left(\hat{u}_0 \cdot \hat{u}_1^{G(\text{grpID})} \cdot \hat{u}_3^m \cdot \hat{c}_6\right)^t, \quad \mathfrak{d}_3 \leftarrow \mathfrak{m} \cdot \hat{t}(h_1, \hat{g}_2)^t.$$

The verifier then checks whether

$$\mathfrak{m} = \mathfrak{d}_3 \cdot \frac{\hat{t}(c_5, \hat{d}_2)}{\hat{t}(d_1, \hat{c}_0)} \text{ and verifies the POK } \Sigma.$$

That a valid signature will verify follows from the following set of equations:

$$\begin{aligned}
\frac{\hat{t}(c_5, \hat{d}_2)}{\hat{t}(d_1, \hat{c}_0)} &= \frac{\hat{t}(g^r, \left(\hat{u}_0 \cdot \hat{u}_1^{G(\text{grpID})} \cdot \hat{u}_3^m \cdot \hat{c}_6\right)^t)}{\hat{t}\left(g^t, \hat{h}_2 \cdot \left(\hat{u}_0 \cdot \hat{u}_1^{G(\text{grpID})} \cdot \hat{u}_2^{G(\text{userID})} \cdot \hat{u}_3^m \cdot \hat{u}_4^{G(r_4)}\right)^r\right)} \\
&= \frac{\hat{t}(g^r, \left(\hat{u}_0 \cdot \hat{u}_1^{G(\text{grpID})} \cdot \hat{u}_2^{G(\text{userID})} \cdot \hat{u}_3^m \cdot \hat{u}_4^{G(r_4)}\right)^t)}{\hat{t}\left(g^t, \hat{h}_2 \cdot \left(\hat{u}_0 \cdot \hat{u}_1^{G(\text{grpID})} \cdot \hat{u}_2^{G(\text{userID})} \cdot \hat{u}_3^m \cdot \hat{u}_4^{G(r_4)}\right)^r\right)} \\
&= \frac{1}{\hat{t}(g^t, \hat{h}_2)} = \frac{1}{\hat{t}(g^t, \hat{g}_2^\alpha)} = \frac{1}{\hat{t}(g^\alpha, \hat{g}_2)^t} = \frac{1}{\hat{t}(h_1, \hat{g}_2)^t}.
\end{aligned}$$

where $r = r_1 + r_2 + r_3$.

Open(gsk, σ): On input of a valid signature $\sigma = (\hat{c}_0, c_5, \hat{c}_6, e_1, \hat{e}_2, \mathfrak{e}_3, \Sigma)$ the group manager computes

$$\begin{aligned}\mathfrak{t} \leftarrow \frac{\hat{t}(e_1, \hat{a}_0)}{\hat{t}(a_5, \hat{e}_2)} &= \frac{\hat{t}(g^k, \hat{h}_2 \cdot \left(\hat{u}_0 \cdot \hat{u}_1^{G(\text{grpID})}\right)^{r_1})}{\hat{t}(g^{r_1}, (\hat{u}_0 \cdot \hat{u}_1^{G(\text{grpID})})^k)} \\ &= \frac{\hat{t}(g^k, \hat{h}_2) \cdot \hat{t}(g^{r_1}, (\hat{u}_0 \cdot \hat{u}_1^{G(\text{grpID})})^k)}{\hat{t}(g^{r_1}, (\hat{u}_0 \cdot \hat{u}_1^{G(\text{grpID})})^k)} \\ &= \hat{t}(g^k, \hat{h}_2) = \hat{t}(h_1, \hat{g}_2)^k.\end{aligned}$$

The Group manager goes through all user identifiers userID issued to the group grpID and checks which one is satisfies the equation

$$\mathfrak{e}_3 = \mathfrak{n}^{G(\text{userID})} \cdot \mathfrak{t}.$$

F Proof of Anonymity

To simplify notation we write GS for our construction $\mathsf{GS}(\mathsf{HIBE}, (\mathcal{P}, \mathcal{V}), \mathsf{H})$. The proof uses the game hopping technique. We give series of games which involve some adversary $\mathcal{A}$ against GS such that the first game is the same as the one that defines anonymity, the last one is a game which $\mathcal{A}$ can only win with probability 1/2, and the behaviour of the adversary across the game doesn't changes, unless one of the assumptions on the primitives used in the construction of GS is insecure. To describe the games, recall how the challenge signature is computed for the scheme GS in experiment $\mathbf{Exp}^{\text{anon}}_{\mathsf{GS},\mathcal{A}}(k)$. When the adversary outputs $(\text{grpID}^*, \text{userID}^*_0, \text{userID}^*_1, m)$ as his target, the experiment uses the key $\text{gsk}^* = d_{\text{grpID}^*}$ associated to grpID^* and proceeds as follows. It computes the distilled key $e \leftarrow \mathsf{Distill}(\text{mpk}, (\text{grpID}^*, \text{userID}^*_b, m, \text{r}_{\text{ID}}))$ for some random basic identity r_{ID}, and the associated decryption key d. It then computes enc a ciphertext of userID_b under the group identity grpID using some randomness r. It then computes $\Sigma = \mathsf{FS}^{\mathsf{H}}_{\mathcal{P}}((\text{userID}_b, \text{r}_{\text{ID}}, r), (e, enc, \text{grpID}, m))$. The signature is (e, d, enc, Σ). We define a sequence of experiments $\mathbf{Exp}^b_i$ for $i \in \{0, 1, 2, 3, 4\}$ which differ in the way in which this challenge signature is computed. Be convention we set experiment $\mathbf{Exp}^b_0(k)$ to be $\mathbf{Exp}^{\text{anon-}b}_{\mathsf{GS},\mathcal{A}}(k)$. We also define $\mathbf{Adv}_i(k)$ for $\Pr\left[\mathbf{Exp}^1_i(k) = 1\right] - \Pr\left[\mathbf{Exp}^0_i(k) = 1\right]$, s in particular $\mathbf{Adv}^{\text{anon}}_{\mathsf{GS},\mathcal{A}}(k) = \mathbf{Adv}_0(k)$.

Replacing true proofs with simulated proofs. We first modify experiment $\mathbf{Exp}^b_0(k)$ by changing how the zero knowledge proof in the challenge signature is computed. Specifically, instead of using the prover (and the Fiat–Shamir transform), we use the simulator $\mathcal{S}$ associated to $\mathcal{P}$. Furthermore, in the transformed game the random oracle is programmed: we maintain the list L of pairs (m, h) that signify that how $\mathsf{H}(m) = h$. Initially this list is empty. More specifically upon receiving the query $(\text{grpID}^*, \text{userID}^*_0, \text{userID}^*_1, m)$ the modified experiment, which we denote by $\mathbf{Exp}^b_1(k)$ produces the challenge signature as follows. It calculates e, d and enc as in experiment $\mathbf{Exp}^b_0(k)$. The proof Σ is computed as follows. A challenge $c \in \mathsf{ChallSpace}$ is selected at random, the experiment sets $x = (e, enc, \text{grpID}, m)$ and computes $(r, s) \leftarrow \mathcal{S}(x, c)$ to obtain $\Sigma = (r, c, s)$. If an element of the form $((r||x||m), h)$ with $h \neq c$ already occurs in L then the experiment aborts, otherwise $((r||x||m||), c)$ is added to L. We write $\mathbf{Exp}^b_1(k)$ for the resulting experiment. The idea is that if an adversary behaves differently under experiments $\mathbf{Exp}^b_0$ and $\mathbf{Exp}^b_1$, then the adversary can be used to distinguish between proofs obtained via the Fiat–Shamir transform and those obtained using the simulator for the proof system and programming the random oracle. Formally, we construct adversary $\mathcal{D}$ which simulates the environment of $\mathcal{A}$ perfectly, but producing the system parameters (mpk, msk) and using these to answer all of adversaries queries (as in $\mathbf{Exp}^{\text{anon-}b}_{\mathsf{GS},\mathcal{A}}(k)$), except for the challenge query. When $\mathcal{A}$ outputs $(\text{grpID}^*, \text{userID}^*_0, \text{userID}^*_1, \mathfrak{m})$ adversary $\mathcal{D}$ selects a random identity r_{ID} and random coins r for the encryption algorithm of the underlying HIBE. It then flips a bit b and computes $e \leftarrow \mathsf{Distill}(\text{mpk}, (\text{grpID}^*, \text{userID}^*_b, \mathfrak{m}, \text{r}_{\text{ID}}))$, computes the associated decryption key d using msk, and computes the encryption $enc \leftarrow \mathsf{Encrypt}(\text{grpID}^*, \text{userID}^*_b, \mathcal{R})$. It then outputs $(e, \mathsf{enc}, \text{grpID}^*, \mathfrak{m}), (\text{userID}_b, \text{r}_{\text{ID}}, r)$ to its environment. It obtains in return a proof Σ that indeed $(e, \mathsf{enc}, \text{grpID}^*, \mathfrak{m}), (\text{userID}_b, \text{r}_{\text{ID}}, r) \in \mathcal{R}$. Then $\mathcal{D}$ passes (e, d, enc, Σ) to the adversary and continues the simulation as before. When the adversary $\mathcal{A}$ stops and outputs its guess bit d, adversary $\mathcal{D}$ outputs 1 if $d = b$ and 0 otherwise.

Notice that the proof obtained from the environment of $\mathcal{D}$ is obtained via the Fiat–Shamir heuristic, if $\mathcal{D}$ is under experiment $\mathbf{Exp}^{\mathsf{FS}\text{-}1}(k)$ and is obtained using the simulator if $\mathcal{D}$ is under experiment $\mathbf{Exp}^{\mathsf{FS}\text{-}0}(k)$. It follows that the environment that $\mathcal{D}$ simulates for $\mathcal{A}$ is that of $\mathbf{Exp}_0^b$ if the proofs are obtained by the Fiat–Shamir heuristic and that of $\mathbf{Exp}_0^b$ if the proofs are simulated. Here the bit b is the one selected at random by $\mathcal{D}$. Formally, we obtain that:

$$\begin{aligned}
\Pr \Big[\, &\mathbf{Exp}^{\mathsf{FS}\text{-}1}_{\mathcal{P},\mathcal{D}}(k) = 1\Big] \\
&= \frac{1}{2} \cdot \Pr\left[\mathbf{Exp}^{\mathsf{FS}\text{-}1}_{\mathcal{P},\mathcal{D}}(k) = 1 \mid b = 1\right] + \frac{1}{2} \cdot \Pr\left[\mathbf{Exp}^{\mathsf{FS}\text{-}1}_{\mathcal{P},\mathcal{D}}(k) = 1 \mid b = 0\right] \\
&= \frac{1}{2} \cdot \Pr\left[\mathbf{Exp}_0^1(k) = 1\right] + \frac{1}{2} \cdot \Pr\left[\mathbf{Exp}_0^0(k) = 0\right] \\
&= \frac{1}{2} \cdot \Pr\left[\mathbf{Exp}_0^1(k) = 1\right] + \frac{1}{2}\left(1 - \mathbf{Exp}_0^0(k) = 1\right) \\
&= \frac{1}{2}\left(1 + \mathbf{Adv}_0(k)\right)
\end{aligned}$$

We also have that

$$\begin{aligned}
\Pr \Big[\, &\mathbf{Exp}^{\mathsf{FS}\text{-}0}_{\mathcal{P},\mathcal{D}}(k) = 1\Big] \\
&= \frac{1}{2} \cdot \Pr\left[\mathbf{Exp}^{\mathsf{FS}\text{-}0}_{\mathcal{P},\mathcal{D}}(k) = 1 \mid b = 1\right] + \frac{1}{2} \cdot \Pr\left[\mathbf{Exp}^{\mathsf{FS}\text{-}0}_{\mathcal{P},\mathcal{D}}(k) = 1 \mid b = 0\right] \\
&= \frac{1}{2} \cdot \Pr\left[\mathbf{Exp}_1^1(k) = 1\right] + \frac{1}{2} \cdot \Pr\left[\mathbf{Exp}_1^0(k) = 0\right] \\
&= \frac{1}{2} \cdot \Pr\left[\mathbf{Exp}_1^1(k) = 1\right] + \frac{1}{2}\left(1 - \mathbf{Exp}_1^0(k) = 1\right) \\
&= \frac{1}{2}\left(1 + \mathbf{Adv}_1(k)\right)
\end{aligned}$$

By subtracting and rearranging terms we have:

$$\mathbf{Adv}_0(k) = \mathbf{Adv}_1(k) + 2 \cdot \mathbf{Adv}^{\mathsf{FS}}_{\mathcal{P},\mathcal{D}}(k) \tag{2}$$

Since we assume that $\mathcal{P}$ has high-entropy commitments, the second term on the right hand side of the above equation is upper bounded by $\frac{q_{\mathsf{H}}}{\mathsf{CommitSpace}}$, where $\mathsf{CommitSpace}$ is the space of commitments of the Σ-protocol, and q_H is the number of hash oracle queries made during the execution of the experiment $\mathbf{Exp}_0^b$. We therefore have that:

$$\mathbf{Adv}_0(k) = \mathbf{Adv}_1(k) + \frac{2 \cdot q_{\mathsf{H}}}{\mathsf{CommitSpace}} \tag{3}$$

Replacing real distilled keys with fake ones. In the next step, we further modify the challenge signature so that the first part of the signature, i.e. (e, d) is obtained from a completely random identity as opposed to the identity $(\mathrm{grpID}^*, \mathrm{userID}_b^*, m, \mathrm{r_{ID}})$. More precisely, the signature is computed as follows. The experiment selects at random an identity $\mathsf{ID} \leftarrow \mathsf{IdSp}^4$ computes $e \leftarrow \mathsf{Distill}(\mathsf{ID})$, uses msk to obtain the key d associated to e. Next, it encrypts the identity userID_b^* under grpID^* to produce ciphertext enc. The proof Σ is computed as before. We write $\mathbf{Exp}_2^b(k)$ for the resulting experiment. The idea is that if an adversary behaves significantly differently between the games $\mathbf{Exp}_1^b$ and $\mathbf{Exp}_2^b$, then we can use that adversary to break the random recipient hiding property of the underlying HIBE. Indeed, for any adversary $\mathcal{A}$ against GS we construct the following adversary $\mathcal{B}$ against the HIBE.

Adversary $\mathcal{B}$ receives mpk, msk from the experiment in which it is running and uses msk to set up an identity based group signature scheme, per our construction. Notice that knowledge of msk allows $\mathcal{B}$ to answer all of the queries that $\mathcal{A}$ makes. When $\mathcal{A}$ outputs his challenge target $(\mathrm{grpID}^*, \mathrm{userID}_0^*, \mathrm{userID}_1^*, m)$, adversary $\mathcal{B}$ proceeds as follows. It selects a random bit $b \in \{0,1\}$, and returns to his environment the pattern $(\mathrm{grpID}^*, \mathrm{userID}_b^*, m, \star)$. It receives in return a pair (e,d) such that e is either the key distilled from $(\mathrm{grpID}^*, \mathrm{userID}_b^*, m, \mathrm{r}_{\mathrm{ID}})$ from some random identity, or it is distilled from some completely unrelated identity ID. It then encrypts the identity userID_b^* under grpID^* to produce *enc* and computes Σ as before. This challenge signature is passed to the adversary $\mathcal{A}$. When $\mathcal{A}$ returns his answer bit b' if $b = b'$ then $\mathcal{B}$ returns 1 as answer, otherwise it returns 0.

Notice that when adversary $\mathcal{B}$ is in experiment $\mathbf{Exp}^{\mathsf{RIdH}-1}_{\mathsf{HIBE},\mathcal{B}}(k)$, the environment that is simulated for $\mathcal{A}$ is that of $\mathbf{Exp}_1^b$ (where b is the bit selected at random by adversary $\mathcal{B}$). Also, when adversary $\mathcal{B}$ is in experiment $\mathbf{Exp}^{\mathsf{RIdH}-0}_{\mathsf{HIBE},\mathcal{B}}(k)$, then the environment that is simulated for $\mathcal{A}$ is that of $\mathbf{Exp}_2^b$ (where b is the bit selected at random by adversary $\mathcal{B}$). We therefore obtain that:

$$\begin{aligned}
&\Pr\left[\ \mathbf{Exp}^{\mathsf{RIdH}-1}_{\mathsf{HIBE},\mathcal{B}}(k) = 1\right] \\
&\quad = \frac{1}{2}\cdot \Pr\left[\mathbf{Exp}^{\mathsf{RIdH}-1}_{\mathsf{HIBE},\mathcal{B}}(k) = 1 \mid b = 1\right] + \frac{1}{2}\cdot \Pr\left[\mathbf{Exp}^{\mathsf{RIdH}-1}_{\mathsf{HIBE},\mathcal{B}}(k) = 1 \mid b = 0\right] \\
&\quad = \frac{1}{2}\cdot \Pr\left[\mathbf{Exp}_1^1(k) = 1\right] + \frac{1}{2}\cdot \Pr\left[\mathbf{Exp}_1^0(k) = 0\right] \\
&\quad = \frac{1}{2}\cdot \Pr\left[\mathbf{Exp}_1^1(k) = 1\right] + \frac{1}{2}\left(1 - \mathbf{Exp}_1^0(k) = 1\right) \\
&\quad = \frac{1}{2}\left(1 + \mathbf{Adv}_1(k)\right)
\end{aligned}$$

We also have that

$$\begin{aligned}
&\Pr\left[\ \mathbf{Exp}^{\mathsf{RIdH}-0}_{\mathsf{HIBE},\mathcal{B}}(k) = 1\right] \\
&\quad = \frac{1}{2}\cdot \Pr\left[\mathbf{Exp}^{\mathsf{RIdH}-0}_{\mathsf{HIBE},\mathcal{B}}(k) = 1 \mid b = 1\right] + \frac{1}{2}\cdot \Pr\left[\mathbf{Exp}^{\mathsf{RIdH}-0}_{\mathsf{HIBE},\mathcal{B}}(k) = 1 \mid b = 0\right] \\
&\quad = \frac{1}{2}\cdot \Pr\left[\mathbf{Exp}_2^1(k) = 1\right] + \frac{1}{2}\cdot \Pr\left[\mathbf{Exp}_1^0(k) = 0\right] \\
&\quad = \frac{1}{2}\cdot \Pr\left[\mathbf{Exp}_2^1(k) = 1\right] + \frac{1}{2}\left(1 - \mathbf{Exp}_2^0(k) = 1\right) \\
&\quad = \frac{1}{2}\left(1 + \mathbf{Adv}_2(k)\right)
\end{aligned}$$

By subtracting and rearranging terms we have that:

$$\mathbf{Adv}_1(k) = \mathbf{Adv}_2(k) + 2\cdot \mathbf{Adv}^{\mathsf{RIdH}}_{\mathsf{HIBE},\mathcal{B}}(k) \tag{4}$$

Preventing queries to the decryption oracle. In the next step, we preclude the adversary to ask certain open queries. More precisely, let Ask be the event that during the execution of experiment $\mathbf{Exp}_2^b(k)$ the adversary sends a valid opening query of the form $(\mathrm{grpID}^*, \mathrm{userID}_b^*, \sigma^*)$, where σ^* is of the form $(e, d, enc^*, (r, c, s))$ with enc^* the encryption contained in the challenge signature received by $\mathcal{A}$. (Clearly, the probability of Ask does not depend on b). Intuitively, since the proof system used for producing the proofs of consistency is sound, if the adversary is able to produce a proof, then the adversary should know a corresponding witness. However, a witness contains the identity encrypted in enc^*, which means that the adversary was able to break the ciphertext enc^* that it received as challenge. Let $\mathbf{Exp}_3^b$ be the same as experiment $\mathbf{Exp}_2^b$, except that when event Ask occurs, the experiment aborts. If we write $\overline{\mathsf{Ask}}$ for the event complementary to Ask, we have:

$$\begin{aligned}
&\mathbf{Adv}_2(k) = \\
&= \Pr\left[\mathbf{Exp}_2^1(k) = 1\right] - \Pr\left[\mathbf{Exp}_2^0(k) = 1\right] \\
&= \left(\Pr[\mathsf{Ask}] \cdot \Pr\left[\mathbf{Exp}_2^1(k) = 1 \mid \mathsf{Ask}\right] + \Pr[\overline{\mathsf{Ask}}] \cdot \Pr\left[\mathbf{Exp}_2^1(k) = 1 \mid \overline{\mathsf{Ask}}\right]\right) - \\
&\quad \left(\Pr[\mathsf{Ask}] \cdot \Pr\left[\mathbf{Exp}_2^0(k) = 1 \mid \mathsf{Ask}\right] + \Pr[\overline{\mathsf{Ask}}] \cdot \Pr\left[\mathbf{Exp}_2^0(k) = 1 \mid \overline{\mathsf{Ask}}\right]\right) \\
&= \Pr[\mathsf{Ask}] \cdot \left(\Pr\left[\mathbf{Exp}_2^1(k) = 1 \mid \mathsf{Ask}\right] - \Pr\left[\mathbf{Exp}_2^0(k) = 1 \mid \mathsf{Ask}\right]\right) + \\
&\quad \Pr[\overline{\mathsf{Ask}}] \cdot \left(\Pr\left[\mathbf{Exp}_2^1(k) = 1 \mid \overline{\mathsf{Ask}}\right] - \Pr\left[\mathbf{Exp}_2^0(k) = 1 \mid \overline{\mathsf{Ask}}\right]\right)
\end{aligned}$$

Using standard bounds we therefore have:

$$\mathbf{Adv}_2(k) \leq \Pr[\mathsf{Ask}] + \mathbf{Adv}_3(k) \tag{5}$$

We next bound the probability that event Ask occurs.

The idea here is that if Ask occurs, then the proof that is contained by the signature shows knowledge of the identity userID^b encrypted in enc, and therefore the adversary breaks the security of the HIBE scheme that underlies our construction.

The above idea sits behind the following adversary $\mathcal{C}$. The adversary is intended for experiment $\overline{\mathbf{Exp}}_{\mathsf{HIBE},\mathcal{C}}^{\mathsf{IND\text{-}ID\text{-}CCA\text{-}}b}(k)$. Here, we write $\overline{\mathbf{Exp}}_{\mathsf{HIBE},\mathcal{C}}^{\mathsf{IND\text{-}ID\text{-}CCA\text{-}}b}(k)$ for a variant of the experiment in Definition 1. In this variant, the adversary outputs $(\mathsf{ID}*, \mathfrak{m}_0^*, \mathfrak{m}_1^*)$ (i.e. two identities, as opposed to a single one). The experiment returns an encryption $enc \leftarrow \mathsf{Encrypt}(\mathsf{ID}^*, \mathfrak{m}_b^*)$ of $\mathfrak{m}_b^*$. The rest of the experiment remains the same.

Adversary $\mathcal{C}$ receives as input the public key mpk of a 4-level instance of HIBE, has access to an $\mathsf{Extract}$ oracle keyed with the key msk and to a decryption oracle $\mathsf{Decrypt}$, keyed with the same key. Adversary $\mathcal{C}$ uses his capabilities to setup a simulation of the experiment $\mathbf{Exp}_2$. Clearly, $\mathcal{C}$ can answer the queries that $\mathcal{A}$ makes to his $\mathsf{GrpSetUp}$, Join and $Sign$ using his access to the $\mathsf{Extract}$ oracle. Furthermore, it can answer Open queries using his decryption oracle to decrypt the ciphertext part of signatures. At some point, adversary $\mathcal{A}$ outputs $(\mathrm{grpID}^*, \mathrm{userID}_0^*, \mathrm{userID}_1^*, \mathfrak{m})$ as his challenge. At this point, adversary $\mathcal{C}$ flips a bit d, and outputs $(\mathrm{grpID}^*, \mathrm{userID}_0^*, \mathrm{userID}_1^*)$ as his own challenge. Adversary $\mathcal{C}$ then constructs a signature by producing $e \leftarrow \mathsf{Distill}(\mathsf{ID})$ and computing the associated d by querying e to the $\mathsf{Extract}$ oracle. Next, it computes a proof $\Sigma = (r, c, s)$ as in $\mathbf{Exp}_2$ (i.e. using the programmable random oracle). It returns (e, d, enc, Σ) to adversary $\mathcal{A}$. It then continues the simulation as before until event Ask occurs. If the simulation finishes and event Ask did not occur, then $\mathcal{C}$ outputs a randomly selected bit. If event Ask occurs, then adversary makes a query $(\mathrm{grpID}^*, m, (e', d', enc, \Sigma' = (r', c', s')))$ to its decryption oracle. Notice that if this query is indeed valid, then it must be the case that $\mathcal{A}$ had previously made a query of the form $(r'||(e', enc, \mathrm{grpID}, m)||m)$ to his random oracle and obtained as answer c'. Adversary $\mathcal{C}$ then rewinds the execution of $\mathcal{A}$ up to the moment $\mathcal{A}$ makes this particular query, and answers with $c'' \leftarrow \mathsf{ChallSpace}$. The execution then continues until event Ask occurs again. If Ask does not occur, then $\mathcal{C}$ outputs a randomly chosen bit. By a standard argument (See for example [4]) Ask occurs again for precisely the same query to the random oracle and $c' \neq c''$, with probability at least

$$\frac{\Pr[\mathsf{Ask}]}{q_H} - \frac{1}{\mathsf{ChallSpace}}.$$

In this case, adversary $\mathcal{C}$ will have to convincing transcripts (r', c', s') and (r', c'', s'') for the statement that $(e', enc, \mathrm{grpID}, m) \in L_{\mathcal{R}}$. Adversary $\mathcal{C}$ then computes a witness $(\mathrm{userID}, \mathrm{r_{ID}}, r) \leftarrow \mathcal{E}((r', c', s'), (r', c'', s''))$. If $\mathrm{userID} = \mathrm{userID}_d$ then $\mathcal{C}$ returns d.

To analyse the success of adversary $\mathcal{C}$, notice that for each experiment where $\overline{\mathbf{Exp}}_{\mathsf{HIBE},\mathcal{C}}^{\mathsf{IND\text{-}ID\text{-}CCA\text{-}}d}(k) = 1$ we have three situations. 1) Event Ask does not occur in the execution of $\mathcal{A}$, in which case $\mathcal{C}$ outputs bit d with probability $\frac{1}{2} \cdot \Pr\left[\overline{\mathsf{Ask}}\right]$. 2) Event Ask occurs, but the rewinding is not successful in which case $\mathcal{C}$ outputs bit d with probability at least $\frac{1}{2}\Pr\left[\mathsf{Ask}\right]\left(1 - \left(\frac{\Pr[\mathsf{Ask}]}{q_H} - \frac{1}{\mathsf{ChallSpace}}\right)\right)$. 3) Finally, event Ask occurs, and the rewinding is successful in which case $\mathcal{C}$ outputs bit d with probability $\Pr\left[\mathsf{Ask}\right] \cdot \left(\frac{\Pr[\mathsf{Ask}]}{q_H} - \frac{1}{\mathsf{ChallSpace}}\right)$.

If we let $x = \Pr[\mathsf{Ask}]$ we have:

$$\begin{aligned}
\mathbf{Adv}^{\mathsf{IND\text{-}ID\text{-}CCA}}_{\mathsf{HIBE},\mathcal{C}}(k) &= \Pr\left[\overline{\mathbf{Exp}}^{\mathsf{IND\text{-}ID\text{-}CCA\text{-}1}}_{\mathsf{HIBE},\mathcal{C}}(k) = 1\right] \\
&\quad + \Pr\left[\overline{\mathbf{Exp}}^{\mathsf{IND\text{-}ID\text{-}CCA\text{-}0}}_{\mathsf{HIBE},\mathcal{C}}(k) = 0\right] - 1 \\
&\geq (1-x) + x \cdot \left(1 - \frac{x}{q_H} + \frac{1}{\mathsf{ChallSpace}}\right) \\
&\quad + 2 \cdot x \cdot \left(\frac{x}{q_H} - \frac{1}{\mathsf{ChallSpace}}\right) - 1 \\
&= \frac{x^2}{q_H} - \frac{x}{\mathsf{ChallSpace}}
\end{aligned}$$

We can then upper bound $\Pr[\mathsf{Ask}] = x$ by:

$$\Pr[\mathsf{Ask}] \leq \frac{q_H}{2 \cdot \mathsf{ChallSpace}} + \sqrt{\frac{q_H^2}{4 \cdot \mathsf{ChallSpace}^2} + q_H \cdot \mathbf{Adv}^{\mathsf{IND\text{-}ID\text{-}CCA}}_{\mathsf{HIBE},\mathcal{C}}(k)}$$

From the above and Equation 5 we have that:

$$\mathbf{Adv}_2(k) \leq \frac{q_H}{2 \cdot \mathsf{ChallSpace}} + \sqrt{\frac{q_H^2}{4 \cdot \mathsf{ChallSpace}^2} + q_H \cdot \mathbf{Adv}^{\mathsf{IND\text{-}ID\text{-}CCA}}_{\mathsf{HIBE},\mathcal{C}}(k)} + \mathbf{Adv}_3(k) \tag{6}$$

Eliminating the identity of the signer out of the encryption. The final transformation that we perform is to create the ciphertext that is part of the signature by encrypting the same random identity that had been used to obtain the distilled key e part of the output signature. The behaviour of the new experiment $\mathbf{Exp}_4^b$ is identical to that of $\mathbf{Exp}_3^b$, with the following modification. When adversary outputs challenge $(\mathrm{grpID}^*, \mathrm{userID}_0^*, \mathrm{userID}_1^*, m)$, the challenge signature is created as in experiment $\mathbf{Exp}_3^b$, except that enc is computed by $enc \leftarrow \mathsf{Encrypt}(\mathrm{grpID}^*, 0^{|\mathrm{userID}_b^*|}, r)$ The idea here is that if an adversary would see a difference between $\mathbf{Exp}_3^b$ and $\mathbf{Exp}_4^b$, that adversary has somehow obtained information about the identity encrypted in enc. We formalise this idea using an adversary D against the $\mathsf{IND\text{-}ID\text{-}CCA}$ security of the underlying HIBE. Adversary $\mathcal{D}$ receives as input the public key of the HIBE mpk, has access to an extraction oracle keyed with the associated msk, and to a decryption oracle. Adversary $\mathcal{D}$ uses access to msk to set-up an ID-based group signature scheme for which (mpk, msk) are the master public and secret keys. Clearly, these keys are sufficient to simulated perfectly the $\mathsf{GrpSetUp}$, Join, and Sign queries that adversary $\mathcal{A}$ makes. To answer the queries that $\mathcal{A}$ makes to the Open oracle, adversary $\mathcal{D}$, after verifying the validity of the zero-knowledge proof, uses its decryption oracle to decrypt the ciphertext contained in the signature and recover the identity of the signer. The only query to the opening oracle that is treated differently is one of the form $(\mathrm{grpID}^*, (e, d, enc^*, \Sigma)$ which causes adversary $\mathcal{D}$ to fail. It only needs to be checked that adversary $\mathcal{D}$ defined as above is a valid $\mathsf{IND\text{-}ID\text{-}CCA}$ adversary, that is that 1) $\mathcal{D}$ does not submit grpID^* to its $\mathsf{Extract}$ oracle, and that 2) it does not submit the ciphertext enc^* to its decryption oracle. During its simulation $\mathcal{D}$ only needs to submit grpID^* to the extraction oracle only when $\mathcal{A}$ requests to setup group grpID^*. Since such a request would render $\mathcal{A}$ invalid we conclude that this query does not occur. To see why the second condition is satisfied by $\mathcal{D}$ notice that the only case when $\mathcal{D}$ needs to submit enc to its decryption oracle would be when $\mathcal{A}$ makes a valid query of the form $(\mathrm{grpID}^*, (e, d, enc, \Sigma))$ to his opening oracle. However, in this case algorithm $\mathcal{D}$ fails, just as do both experiments $\mathbf{Exp}_3^b$ and $\mathbf{Exp}_4^b$. We conclude that:

$$\begin{aligned}
\Pr\left[\mathbf{Exp}^{\mathsf{IND\text{-}ID\text{-}CCA\text{-}1}}_{\mathsf{HIBE},\mathcal{D}}(k)\right] &= \frac{1}{2} \cdot \Pr\left[\mathbf{Exp}_3^1(k) = 1\right] + \frac{1}{2} \cdot \Pr\left[\mathbf{Exp}_3^0(k) = 0\right] \\
&= \frac{1}{2}\left(1 + \mathbf{Adv}_3(k)\right)
\end{aligned}$$

and that

$$\begin{aligned}\Pr\left[\mathbf{Exp}_{\mathsf{HIBE},\mathcal{D}}^{\mathsf{IND\text{-}ID\text{-}CCA\text{-}0}}(k)\right] &= \frac{1}{2}\cdot\Pr\left[\mathbf{Exp}_4^1(k)=1\right]+\frac{1}{2}\cdot\Pr\left[\mathbf{Exp}_3^0(k)=0\right]\\ &= \frac{1}{2}\left(1+\mathbf{Adv}_4(k)\right)\end{aligned}$$

By subtracting and rearranging terms we get:

$$\mathbf{Adv}_3(k) = \mathbf{Adv}_4(k) + 2\cdot\mathbf{Adv}_{\mathsf{HIBE},\mathcal{D}}^{\mathsf{IND\text{-}ID\text{-}CCA}}(k) \tag{7}$$

Putting it all together. Clearly, the answer that the adversary $\mathcal{A}$ receives in $\mathbf{Exp}_4^b$ is independent of the bit b, and therefore $\mathbf{Adv}_4(k) \leq \frac{1}{2}$. Together with Equations (3),(4), and (6) we have that:

$$\begin{aligned}\mathbf{Adv}_{\mathsf{GS},\mathcal{A}}^{\mathrm{anon}}(k) \leq &\frac{1}{2}+\frac{2\cdot q_H}{\mathsf{CommitSpace}}+2\cdot\mathbf{Adv}_{\mathsf{HIBE},\mathcal{B}}^{\mathsf{RIdH}}(k)+\\ &\sqrt{\frac{q_H^2}{4\cdot\mathsf{ChallSpace}^2}+q_H\cdot\mathbf{Adv}_{\mathsf{HIBE},\mathcal{C}}^{\mathsf{IND\text{-}ID\text{-}CCA}}(k)}+2\cdot\mathbf{Adv}_{\mathsf{HIBE},\mathcal{D}}^{\mathsf{IND\text{-}ID\text{-}CCA}}(k)\end{aligned}$$