Hierarchical Group Signatures

Mårten Trolin and Douglas Wikström*

March 22, 2005

Abstract

We introduce the notion of *hierarchical group signatures.* This is a proper generalization of group signatures, which allows multiple group managers organized in a tree with the signers as leaves. For a signer that is a leaf of the subtree of a group manager, the group manager learns which of its children that (perhaps indirectly) manages the signer.

We provide definitions for the new notion and construct a scheme that is provably secure given the existence of a family of trapdoor permutations.

We also present a construction which is relatively practical, and prove its security in the random oracle model under the strong RSA assumption and the DDH assumption.

1 Introduction

Consider the notion of group signatures introduced by Chaum and van Heyst [21]. A group member can compute a signature that reveals nothing about the signer's identity except that he is a member of the group. On the other hand the group manager can always reveal the identity of the signer.

An application for group signatures is anonymous credit cards. The cardholder wishes to preserve his privacy when he pays a merchant for goods, i.e., he is interested in unlinkability of payments. The bank must obviously be able to extract the identity of a cardholder from a payment or at least an identifier for an account, to be able to debit the account. To avoid fraud, the bank, the merchant, and the cardholder all require that a cardholder cannot pay for goods without holding a valid card. To solve the problem using group signatures we let the bank be the group manager and the cardholders be signers. A cardholder signs a transaction and hands it to the merchant. The merchant then hands the signed transaction to the bank, which debits the cardholder and credits the merchant. Since signatures are unlinkable, the merchant learns nothing about the cardholder's identity. The bank on the other hand can always extract the cardholder's identity from a valid signature and debit the correct account.

The above scenario is somewhat simplified since normally there are many banks that issue cards of the same brand and which are processed through the same payment network. The payment network normally works as an administrator and

*Royal Institute of Technology (KTH), Stockholm, Sweden, {marten,dog}@nada.kth.se

routes transactions to several independent banks. Thus, the merchant hands a payment to the payment network which hands the payment to the issuing bank. We could apply group signatures here as well by making the payment network act as the group manager. The network would then send the extracted identity to the issuing bank. Another option is to set up several independent group signatures schemes, one for each issuer. In the first approach, the payment network learns the identity of the customer, and in the second approach the merchant learns which bank issued the customer's card. A better solution would reveal nothing except what is absolutely necessary to each party. The merchant needs to be convinced that the credit card is valid, the payment network must be able to route the payment to the correct card issuer and the issuer must be able to determine the identity of the cardholder.

A solution that comes to mind is to use ordinary group signatures with the modification that the customer encrypts his identity with his bank's public key. Then we have the problem of showing to the merchant that this encryption contains valid information. However, the customer cannot reveal the public key of the bank to the merchant, making such a proof far from trivial.

In this paper we introduce and investigate the notion of *hierarchical group signatures*. These can be employed to solve the above problem. When using a hierarchical group signature scheme there is not one single group manager. Instead there are several group managers organized in a tree, i.e., each group manager either manages a group of signers or a group of group managers. In the original notion the group manager can always identify the signer of a message, but nobody else can distinguish between signatures by different signers. The corresponding property for hierarchical group signatures is more complicated. If a manager directly manages a group of signers, it can identify all the signers that it manages, but the signatures of all other signers are indistinguishable to it. This corresponds directly to the original notion. If a manager manages a group of managers, it cannot identify the signer, but it can identify the manager directly below it which (perhaps indirectly) manages the signer. Thus, a manager that does not manage signers directly get only partial information on the identity of the signer.

When we use hierarchical group signatures to construct anonymous credit cards for the more realistic setting we let the payment network be the root manager that manages a set of group managers, i.e., the issuing banks, and we let the cardholders be signers. The credit card application also demonstrates what kind of responsibility model is likely to be used with a hierarchical group signature scheme. With a valid signature on a transaction, the merchant has a valid demand on the payment network. If the payment network has a signature that can be shown to belong to a certain bank, the network has a valid demand on that bank. Thus, it is in the network's interest to open the signatures it receives from merchants, and it is in the issuing banks' interest to open the signatures they receive from the network.

1.1 Previous Work

The concept of group signatures was first introduced by Chaum and van Heyst [21] in 1991. This and the group signature schemes that followed [22, 13] all had the property that the complexity of the scheme grows with the number of participants. In [18] Camenisch and Stadler presented a system where the key does not grow with the number of participants. This system, however, relies on a non-standard number-theoretic assumption. The assumption was actually found to be incorrect and modified in [3]. An efficient system whose security rests on the strong RSA assumption and the Diffie-Hellman decision assumption was presented by Camenisch and Michels in 1998 [16]. This system was improved in [2]. The currently most efficient scheme that is secure under standard assumptions is [14]. More efficient schemes do exist [10, 15], but they are based on bilinear maps and thus relies on less well-studied assumptions for security.

A related notion is *traceable signatures* introduced by Kiayias et al. [33], where signatures belonging to a member can be opened, or traced, in a distributed way without revealing the group secret.

In [7] Bellare et al. presented a scheme for group signatures based on general methods for static groups, i.e., when the set of members cannot be changed after the initial setup. This paper also contains a definitional framework for group signature schemes in this setting. In [8] Bellare et al. extend the definitions to handle also dynamic groups, and present a scheme that is secure under general assumptions. In [34] Kiayias and Yung have made a modification of [2] that is provably secure for dynamic groups under security definitions similar to those in [8]. Our scheme based on general assumptions can be seen as a generalization of [7].

In [3] the concepts of *multi-group signatures* and *subgroup signatures* are described, and in [35] a system for hierarchical multi-groups is given. It may be worthwhile to consider the differences between these concepts and hierarchical signatures introduced here. Multi-group signature schemes allow a signer who is a member of two groups to produce a signature that shows membership of either both groups or just one of them. In hierarchical multi-groups a signer who is a member of a supergroup with subgroups can produce a signature that reveals membership either of the supergroup or of a subgroup of his choice. However, the opening procedure is not hierarchical, e.g., there are no group managers for the subgroups.

Subgroup signatures make it possible for an arbitrary number i of signers to produce a joint signature which can be verified to stem from i distinct group members. None of these extensions contain the hierarchical property.

The connection between group signatures and anonymous payment systems is quite natural and has been studied before. In [36] a system for electronic cash based on group signatures is given by Lysyanskaya and Ramzan.

Group signatures, and especially hierarchical group signatures, should not be confused with zero-knowledge sets as described in [39]. Zero-knowledge sets enables a prover to commit to a set S. Given x he can then prove $x \in S$ or $x \notin S$ (whichever is true) without disclosing anything else about S. For zero-knowledge sets the prover has the necessary information to produce a proof of membership for any element in the set. With group signatures on the other hand the set of members may be public, and the signer proves that it belongs to this set.

1.2 Notation

We write $[a, b]$ to denote the set $\{x \in \mathbb{Z} \mid a \leq x \leq b\}$. We say that an element is chosen "randomly" instead of the more cumbersome "independently and uniformly at random". If T is a tree we denote by $\mathcal{L}(T)$ its set of leaves. We let ϕ denote Euler's ϕ function. By $r \in_R S$ we mean that r is chosen randomly in S. Throughout the paper, κ denotes the security parameter. A function $f : \mathbb{N} \rightarrow [0, 1]$ is said to be negligible if for each $c > 0$ there exists a $\kappa_0 \in \mathbb{N}$ such that $f(\kappa) < \kappa^{-c}$ for $\kappa_0 < \kappa \in \mathbb{N}$. We say that a function $f : \mathbb{N} \rightarrow [0, 1]$ is non-negligible whenever it is not negligible. When we say that a number is k-bit, we implicitly mean that it has a leading one (i.e., that it is in the interval $[2^{k-1}, 2^k - 1]$).

We sometimes do not explicitly state how a group given as input to an algorithm is described, e.g., we write $\mathsf{CHPg}(G_q)$ to denote that the algorithm CHPg is given a description of a group G_q of prime order q as input. Whenever we do that, G_q is assumed to be the unique subgroup of order q of $\mathbb{Z}_p^*$ for a prime $p = 2q + 1$, so the obvious description of G_q is p. In Section 5 we consider the issue of existence of such primes. We use QR_N to denote the subgroup of squares in $\mathbb{Z}_N^*$, i.e., the quadratic residues. We write $\emptyset$ to denote both the empty set and the empty string.

We say that a distribution ensemble $\mathcal{D} = \{D_\kappa\}$ is efficiently sampleable if there exists a polynomial time Turing machine $T_\mathcal{D}$ that on input 1^κ outputs a random sample distributed according to D_κ.

All adversaries in this paper are modeled as polynomial time Turing machines with *non-uniform* auxiliary advice string. We denote the set of such adversaries by PPT^*.

A public-key cryptosystem is said to be *CCA2-secure* if it is infeasible for an attacker to determine which one of two messages of his choice that a given cryptotext is the encryption of, even if the attacker has access to a decryption oracle both before the choice is made and after the cryptotext is received [44]. The following formalizes this property.

Let $\mathcal{CS} = (\mathsf{Kg}, E, D)$ be a public key cryptosystem. Consider the following experiment.

Experiment 1.1 (CCA2, $\mathbf{Exp}^{\mathsf{cca2}-b}_{\mathcal{CS},A}(\kappa)$)**.**

$$
\begin{aligned}
&(\mathrm{pk}, \mathrm{sk}) \leftarrow \mathsf{Kg}(1^\kappa) \\
&(m_0, m_1, \mathrm{state}) \leftarrow A^{D_{\mathrm{sk}}(\cdot)}(\mathsf{choose}, \mathrm{pk}) \\
&c \leftarrow \mathsf{Enc}_{\mathrm{sk}}(m_b) \\
&d \leftarrow A^{D_{\mathrm{sk}}(\cdot)}(\mathsf{guess}, \mathrm{state}, \mathrm{pk})
\end{aligned}
$$

The experiment returns 0 if the encryption oracle was queried on c, and d otherwise. The advantage of an adversary is defined as

$$\mathbf{Adv}^{\mathsf{cca2}}_{\mathcal{CS},A}(\kappa) = |\Pr[\mathbf{Exp}^{\mathsf{cca2}-0}_{\mathcal{CS},A}(\kappa) = 1] - \Pr[\mathbf{Exp}^{\mathsf{cca2}-1}_{\mathcal{CS},A}(\kappa) = 1]| \ .$$

Definition 1.2 (CCA2-secure)**.** *The cryptosystem* $\mathcal{CS}$ *is said to be* CCA2-secure *if* $\mathbf{Adv}^{\mathsf{cca2}}_{\mathcal{CS},A}(\kappa)$ *is negligible for any* $A \in \mathrm{PPT}^*$*.*

A signature scheme is said to be *CMA-secure* if it infeasible for an attacker to output a message-signature pair even if given a signing oracle [32]. Formally CMA-security is defined using the following experiment, where $\mathcal{SS} = (\mathsf{Kg}, \mathsf{Sig}, \mathsf{Vf})$ is a signature scheme

Experiment 1.3 (CMA, $\mathbf{Exp}^{\mathsf{cma}}_{\mathcal{SS},A}(\kappa)$)**.**

$$\begin{aligned}
&(\mathrm{pk}, \mathrm{sk}) \leftarrow \mathsf{Kg}(1^\kappa) \\
&(m, s) \leftarrow A^{\mathsf{Sig}_{\mathrm{sk}}(\cdot)}(\mathsf{guess}, \mathrm{pk})
\end{aligned}$$

If $\mathsf{Vf}_{\mathrm{pk}}(m, s) = 1$ and A's oracle was never queried on m return 1, else return 0. The advantage of an adversary A is defined as $\mathbf{Adv}^{\mathsf{cma}}_{\mathcal{SS},A}(\kappa) = \Pr[\mathbf{Exp}^{\mathsf{cma}}_{\mathcal{SS},A}(\kappa) = 1]$.

Definition 1.4 (CMA-secure)**.** *The signature scheme* $\mathcal{SS}$ *is said to be* CMA-secure *if* $\mathbf{Adv}^{\mathsf{cma}}_{\mathcal{SS},A}(\kappa)$ *is negligible for any* $A \in \mathrm{PPT}^*$*.*

Two ensembles $\{X_n\}_{n\in\mathbb{N}}$ and $\{Y_n\}_{n\in\mathbb{N}}$ are *statistically close* if $\sum_\alpha |\Pr[X_n = \alpha] - \Pr[Y_n = \alpha]|$ is negligible.

Definition 1.5 (Trapdoor Permutation Family)**.** *A* trapdoor permutation family *is a tuple of probabilistic polynomial time Turing machines* $\mathcal{F} = (\mathsf{Gen}, \mathsf{Eval}, \mathsf{Invert})$ *such that:*

1. $\mathsf{Gen}(1^\kappa)$ *outputs a pair* (f, f^{-1}) *such that* f *is a permutation of* $\{0,1\}^\kappa$*.*
2. $\mathsf{Eval}(1^\kappa, f, x)$ *is a deterministic algorithm which on input* f*, where* $(f, f^{-1}) \in \mathsf{Gen}(1^\kappa)$*, and* $x \in \{0,1\}^\kappa$ *outputs* $y = f(x)$*.*
3. $\mathsf{Invert}(1^\kappa, f^{-1}, y)$ *is a deterministic algorithm which on input* f^{-1}*, where* $(f, f^{-1}) \in \mathsf{Gen}(1^\kappa)$*, and* $y \in \{0,1\}^\kappa$ *outputs some* $x = f^{-1}(y)$*.*
4. *For all* κ*,* $(f, f^{-1}) \in \mathsf{Gen}(1^\kappa)$*, and* $x \in \{0,1\}^\kappa$ *we have* $f^{-1}f(x) = x$*.*
5. *For all adversaries* $A \in \mathrm{PPT}^*$*, the following is negligible*

$$\Pr[(f, f^{-1}) \leftarrow \mathsf{Gen}(1^\kappa), \quad x \leftarrow \{0,1\}^\kappa, \quad A(f, f(x)) = f^{-1}(y)] \ .$$

Definition 1.6 (Hard-Core Bit)**.** *Let* $\mathcal{B} = \{B_\kappa : \{0,1\}^\kappa \to \{0,1\}\}$ *be a collection of functions such that there exists a polynomial time Turing machine that outputs* $B_\kappa(x)$ *on input* $(1^\kappa, x)$*, where* $x \in \{0,1\}^\kappa$*. Let* $\mathcal{F} = (\mathsf{Gen}, \mathsf{Eval}, \mathsf{Invert})$ *be a trapdoor permutation family.* $\mathcal{B}$ *is a* hard-core bit *for* $\mathcal{F}$ *if the following is negligible for all adversaries* $A \in \mathrm{PPT}^*$

$$\left|\Pr[(f, f^{-1}) \leftarrow \mathsf{Gen}(1^\kappa), \quad x \leftarrow \{0,1\}^\kappa, \quad A(f, f(x)) = B(x)] - \frac{1}{2}\right| \ .$$

1.3 Outline of Paper

In Section 2 we formalize the notion of hierarchical group signatures and give definitions of security. We also briefly discuss why it is not trivial to transform a non-hierarchical group signature scheme into a hierarchical scheme. In Section 3 we discuss the concept of anonymous encryption schemes, which we use in both the general construction and the explicit construction. Our construction under general assumptions is presented in Section 4 and in Section 5 we give the explicit construction. The zero-knowledge proofs used in Section 5 can be found in Section 6. Finally in Sections 7 and 8 we discuss possible modifications and extensions of the current scheme.

1.4 Contributions

We introduce and formalize the notion of *hierarchical group signatures*. We give a construction that is provably secure under the existence of a trapdoor permutation family. Then we consider how a practical hierarchical group signature scheme can be constructed under specific complexity assumptions. We show that by a careful selection of primitives one can construct a relatively practical hierarchical group signature scheme that is provably secure under the DDH assumption and the strong RSA assumption in the random oracle model. For reasonable security parameters a few hundred exponentiations are required to produce a signature.

As part of the construction we show how to prove efficiently in zero-knowledge that a committed value is a signature of an encrypted message. This technique may be useful for other applications.

2 Hierarchical Group Signatures

In this section we discuss the notion of hierarchical group signatures. We begin by describing the parties of a hierarchical group signature system. Then we proceed by giving formal definitions.

2.1 Parties

There are two types of parties: signers denoted S_α for α in some index set $\mathcal{I}$, and group managers denoted M_α for indices α described below. The parties form a tree T, where the signers are leaves and the group managers are inner nodes. The indices of the group managers are formed as follows. If a group manager manages a set of signers $\{\alpha \mid \alpha \in \beta \subset \mathcal{I}\}$ we denote it by M_β. This corresponds to M_β having S_α for $\alpha \in \beta$ as children. If a group manager M_γ manages a set of group managers $\{M_{\beta_1}, \ldots, M_{\beta_l}\}$ we denote it by M_γ where $\gamma = \{\beta_1, \ldots, \beta_l\}$. This corresponds to M_γ having M_{β_i} for $i = 1, \ldots, l$ as children. Let M_ρ denote the root group manager. We assume that the root group manager is at depth 0 and that all leaves in the tree are at the same depth. When there is no risk of confusion we write α instead of M_α or S_α.

Note that standard group signatures correspond to having a single group manager $M_{[1,l]}$ that manages all signers $S_1, \ldots, S_l$.

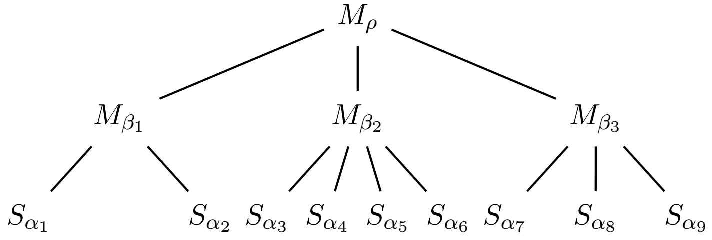

Figure 1: A tree of group managers and signers, where $\rho = \{\beta_1, \ldots, \beta_3\}$, $\beta_1 = \{\alpha_1, \alpha_2\}$, $\beta_2 = \{\alpha_3, \alpha_4, \alpha_5, \alpha_6\}$, and $\beta_3 = \{\alpha_7, \alpha_8, \alpha_9\}$.

2.2 Definition of Security

Bellare et al. [7] give a definition of a group signature scheme, but more importantly they argue that two properties of group signatures, full anonymity and full traceability, imply any reasonable security requirements one can expect from a group signature scheme.

We follow their definitional approach closely and develop definitions that are proper generalizations of the original.

The idea is that the managers and signers are organized in a tree T, and we wish to associate with each node (or leaf) α a public value $\mathrm{hpk}(\alpha)$ and a private value $\mathrm{hsk}(\alpha)$.

Definition 2.1 (Hierarchical Group Signature)**.** *A* hierarchical group signature scheme $\mathcal{HGS} = (\mathsf{HKg}, \mathsf{HSig}, \mathsf{HVf}, \mathsf{HOpen})$ *consists of the following four polynomial-time algorithms*

1. *The randomized* key generation algorithm HKg *takes as input* $(1^\kappa, T)$*, where* T *is a tree of size polynomially bounded in* κ *with all leaves at the same depth, and outputs a pair of maps* $\mathrm{hpk}, \mathrm{hsk} : T \to \{0,1\}^*$.

2. *The randomized* signature algorithm HSig *takes as input a message* m*, a tree* T*, a public map* hpk*, and a secret signing key* $\mathrm{hsk}(\alpha)$*, and returns a signature of* m.

3. *The deterministic* signature verification algorithm HVf *takes as input a tree* T*, a public map* hpk*, a message* m *and a candidate signature* σ *of* m *and returns either* 1 *or* 0.

4. *The deterministic* opening algorithm HOpen *takes as input a tree* T*, a public map* hpk*, a secret opening key* $\mathrm{hsk}(\beta)$*, a message* m*, and a candidate signature* σ*. It outputs an index* $\alpha \in \beta$ *or* $\perp$.

In the definition of HSig above, it is assumed that it is possible to verify in polynomial time given the public tree hpk, a secret key $\mathrm{hsk}(\alpha)$ and an index α', if $\alpha = \alpha'$. This is the case for the construction in [7]. We assume that hpk and hsk map any input that is not a node of T to $\perp$ and that $\mathsf{HOpen}(\cdot, \cdot, \perp, \cdot, \cdot) = \perp$.

We need to define what we mean when we say that a hierarchical group signature scheme is secure. Here we generalize the definitions of [7]. We begin with anonymity. Assume a message has been signed by either $\alpha^{(0)}$ or $\alpha^{(1)}$. Then any group manager on the path leading from $\alpha^{(0)}$ or $\alpha^{(1)}$ to the their first common ancestor can determine who the signer is. In Figure 2 those group managers are marked with black. In the definition of anonymity we capture the property that unless the adversary corrupts one of these group managers, it cannot determine whether $\alpha^{(0)}$ or $\alpha^{(1)}$ signed the message, even if the adversary is given the private keys of all signers and is allowed to select $\alpha^{(0)}$, $\alpha^{(1)}$ and the message itself.

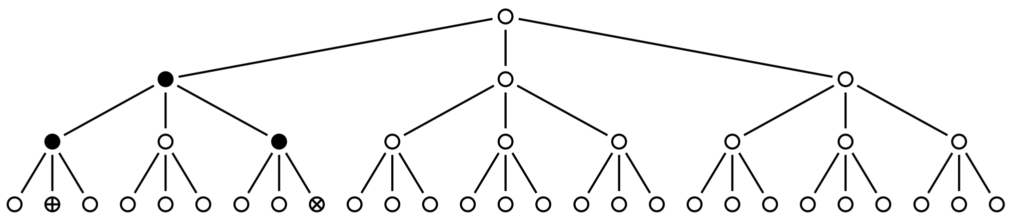

Figure 2: Nodes in black represent group managers able to distinguish between $\alpha^{(0)}$ and $\alpha^{(1)}$ (the two marked leaves).

We define Experiment 2.2 to formalize these ideas. Throughout the experiment the adversary has access to an $\mathsf{HOpen}(T, \mathrm{hpk}, \mathrm{hsk}(\cdot), \cdot, \cdot)$ oracle. At the start of the experiment the adversary is given the public keys of all parties and the private keys of all signers. Then it can adaptively ask for the private keys of the group managers. At some point it outputs the indices $\alpha^{(0)}$ and $\alpha^{(1)}$ of two leaves and a message m. The $\mathsf{HSig}(\cdot, T, \mathrm{hpk}, \mathrm{hsk}(\cdot))$ oracle computes the signature of m using the private key $\mathrm{hsk}(\alpha^{(b)})$ and hands it to the adversary. The adversary finally outputs a guess d of the value of b. If the scheme is anonymous the probability that $b = d$ should be negligibly close to $1/2$ when b is a randomly chosen bit. The labels $\mathsf{corrupt}$, choose and guess below distinguish between the phases of the experiment.

Experiment 2.2 (Hierarchical Anonymity, $\mathbf{Exp}^{\mathsf{anon}-b}_{\mathcal{HGS},A}(\kappa, T)$)**.**

$(\mathrm{hpk}, \mathrm{hsk}) \leftarrow \mathsf{HKg}(1^\kappa, T)$; $\quad s_{\mathrm{state}} \leftarrow (\mathrm{hpk}, \mathrm{hsk}(\mathcal{L}(T)))$; $\quad \mathcal{C} \leftarrow \emptyset$; $\quad \alpha \leftarrow \emptyset$;

While $\quad (\alpha \neq \perp) \quad$ do

$\quad (s_{\mathrm{state}}, \alpha) \leftarrow A^{\mathsf{HOpen}(T, \mathrm{hpk}, \mathrm{hsk}(\cdot), \cdot, \cdot)}(\mathsf{corrupt}, s_{\mathrm{state}}, \mathrm{hsk}(\alpha))$

$\quad \mathcal{C} \leftarrow \mathcal{C} \cup \{\alpha\}$

Done

$(s_{\mathrm{state}}, \alpha^{(0)}, \alpha^{(1)}, m) \leftarrow A^{\mathsf{HOpen}(T, \mathrm{hpk}, \mathrm{hsk}(\cdot), \cdot, \cdot)}(\mathsf{choose}, s_{\mathrm{state}})$

$\sigma \leftarrow \mathsf{HSig}(m, T, \mathrm{hpk}, \mathrm{hsk}(\alpha^{(b)}))$

$d \leftarrow A^{\mathsf{HOpen}(T, \mathrm{hpk}, \mathrm{hsk}(\cdot), \cdot, \cdot)}(\mathsf{guess}, s_{\mathrm{state}}, \sigma)$

Let B be the set of nodes on paths from $\alpha^{(0)}$ and $\alpha^{(1)}$ up to their first common ancestor α_t excluding $\alpha^{(0)}$ and $\alpha^{(1)}$ but including α_t, i.e., the set of nodes $\alpha_l^{(0)}$, $\alpha_l^{(1)}$, $l = t, \ldots, \delta - 1$, such that

$$\alpha^{(0)} \in \alpha^{(0)}_{\delta-1} \in \alpha^{(0)}_{\delta-2} \in \ldots \in \alpha^{(0)}_{t+1} \in \alpha_t \ni \alpha^{(1)}_{t+1} \ni \ldots \ni \alpha^{(1)}_{\delta-2} \ni \alpha^{(1)}_{\delta-1} \ni \alpha^{(1)} .$$

If $B \cap \mathcal{C} \neq \emptyset$ or if A asked its $\mathsf{HOpen}(T, \mathrm{hpk}, \mathrm{hsk}(\cdot), \cdot, \cdot)$ oracle a question $(\alpha_l^{(0)}, m, \sigma)$ or $(\alpha_l^{(1)}, m, \sigma)$ return 0. Otherwise return d.

It can be noted that no generality is lost by having a $\mathsf{corrupt}$ phase only before σ is computed. The reason for this is that before A receives σ, it has decided on $\alpha^{(0)}$ and $\alpha^{(1)}$ and can corrupt any group manager not on the path from $\alpha^{(0)}$ $\alpha^{(1)}$. Therefore nothing would be gained by having an additional $\mathsf{corrupt}$ phase after σ is received.

Consider the above experiment with a depth one tree T with root ρ. In that case we may assume that $\mathrm{hsk}(\rho)$ is never handed to the adversary, since the adversary fails in that case anyway. Similarly the $\mathsf{HOpen}(T, \mathrm{hpk}, \mathrm{hsk}(\cdot), \cdot, \cdot)$ oracle reduces to the Open oracle in [7]. Thus, our experiment reduces to the experiment for full anonymity given in [7] where the adversary gets the secret keys of all signers, but only the public key of the group manager.

Next we consider how the notion of full traceability can be defined in our setting. Full traceability as defined in [7] is similar to security against chosen message attacks (CMA-security) as defined by Goldwasser, Micali and Rivest [32] for signatures. The only essential difference is that the group manager must always be able to open a signature and identify the signer. In our setting this amounts to the following. Given a signature deemed valid by the HVf algorithm, the root should always be able to identify the child directly below it of which the signer is a descendent. The child should have the same ability for the subtree of which it is a root and so on until the child itself is a signer.

Again we define an experiment consisting of two phases. The adversary is given the secret keys of all group managers. Then the adversary adaptively chooses a set of signers to corrupt. Then in a second phase the adversary guesses a message and signature pair. If the guess amounts to a signature deemed valid by HVf and the signer cannot be traced, or if the signature is traced to a non-corrupted signer, the adversary has succeeded and the experiment outputs 1. Otherwise it outputs 0. Thus, the distribution of the experiment should be negligibly close to 0 for all adversaries if the scheme is secure.

Experiment 2.3 (Hierarchical Traceability, $\mathbf{Exp}^{\mathsf{trace}}_{\mathcal{HGS},A}(\kappa, T)$)**.**

$(\mathrm{hpk}, \mathrm{hsk}) \leftarrow \mathsf{HKg}(1^\kappa, T)$; $\quad s_{\mathrm{state}} \leftarrow (\mathrm{hpk}, \mathrm{hsk}(T \backslash \mathcal{L}(T))$; $\quad \mathcal{C} \leftarrow \emptyset$; $\quad \alpha \leftarrow \emptyset$;

While $\quad (\alpha \neq \bot) \quad$ do

$\quad (s_{\mathrm{state}}, \alpha) \leftarrow A^{\mathsf{HSig}(\cdot, T, \mathrm{hpk}, \mathrm{hsk}(\cdot))}(\mathsf{choose}, s_{\mathrm{state}}, \mathrm{hsk}(\alpha))$

$\quad \mathcal{C} \leftarrow \mathcal{C} \cup \{\alpha\}$

Done

$(m, \sigma) \leftarrow A^{\mathsf{HSig}(\cdot, T, \mathrm{hpk}, \mathrm{hsk}(\cdot))}(\mathsf{guess}, s_{\mathrm{state}})$

If $\mathsf{HVf}(T, \mathrm{hpk}, m, \sigma) = 0$ return 0. Define $\alpha_0 = \rho$ and $\alpha_l = \mathsf{HOpen}(T, \mathrm{hpk}, \mathrm{hsk}(\alpha_{l-1}), m, \sigma)$ for $l = 1, \ldots, \delta$. If $\alpha_l = \bot$ for some $0 < l \leq \delta$ return 1. If $\alpha_\delta \notin \mathcal{C}$ and the $\mathsf{HSig}(\cdot, T, \mathrm{hpk}, \mathrm{hsk}(\cdot))$ oracle did not get a question (m, α_δ) return 1. Otherwise return 0.

Consider the experiment above with a depth one tree. This corresponds to giving the adversary the secret key of the group manager, and letting it adaptively choose additional signing keys. Furthermore, the $\mathsf{HSig}(\cdot, T, \mathrm{hpk}, \mathrm{hsk}(\cdot))$ oracle reduces to the GSig oracle in [7]. This is precisely the setting of [7].

The advantage of the adversary is defined in the natural way by

$$\mathbf{Adv}^{\mathsf{anon}}_{\mathcal{HGS},A}(\kappa, T) = |\Pr[\mathbf{Exp}^{\mathsf{anon}-0}_{\mathcal{HGS},A}(\kappa, T) = 1] - \Pr[\mathbf{Exp}^{\mathsf{anon}-1}_{\mathcal{HGS},A}(\kappa, T) = 1]|$$

and

$$\mathbf{Adv}^{\mathsf{trace}}_{\mathcal{HGS},A}(\kappa, T) = \mathbf{Exp}^{\mathsf{trace}}_{\mathcal{HGS},A}(\kappa, T) \ .$$

Definition 2.4 (Security of Hierarchical Group Signatures)**.** *A hierarchical group signature scheme* $\mathcal{HGS} = (\mathsf{HKg}, \mathsf{HSig}, \mathsf{HVf}, \mathsf{HOpen})$ *is* secure *if for all trees T of polynomial size in κ with all leaves at the same depth, and all $A \in \mathrm{PPT}^*$, $\mathbf{Adv}^{\mathsf{trace}}_{\mathcal{HGS},A}(\kappa, T) + \mathbf{Adv}^{\mathsf{anon}}_{\mathcal{HGS},A}(\kappa, T)$ is negligible.*

Remark 2.5. The reason the adversary is not given access to both the HOpen and the HSig oracle in the experiments is that it is given sufficient private keys to simulate the missing oracle by itself.

Remark 2.6. An ordinary signature scheme $\mathcal{SS} = (\mathsf{Kg}, \mathsf{Sig}, \mathsf{Vf})$, with key generator Kg, signature algorithm Sig, and verification algorithm Vf, can be viewed as a hierarchical group signature scheme $(\mathsf{Kg}, \mathsf{Sig}, \mathsf{Vf}, \mathsf{HOpen})$ of depth 0 by defining $\mathsf{HOpen}(\sigma) = \perp$. Then $\mathbf{Adv}^{\mathsf{anon}}_{\mathcal{HGS},A}(\kappa, T) = 0$ and Definition 2.3 reduces to the definition of security against chosen message attacks as defined by Goldwasser, Micali, and Rivest [32].

2.3 Alternative Definitions

Above we define a hierarchical group signature scheme such that the group managers are organized in a tree where all leaves are at the same depth. Furthermore, a group manager can by looking at a signature decide whether the signer belongs to it or not without any interaction with other group managers. Several other variants are possible. Below we discuss some of these variants informally.

Trees with leaves on different depths could be considered. Any such tree can clearly be replaced by a tree with all leaves at the same depth by inserting dummy group managers in between signers and their immediate parents until all signers are at the same depth.

We could let group managers sign on behalf of its group. If this is needed a signer that correspond to the group manager is added. Depending on if the parent of the group manager should be able to distinguish between a signature of the group manager itself and its children or not, the signer is added to the group manager's parent or itself.

We could consider a forest of trees, i.e. there could be several roots. Such a scheme can be simulated in our definition by first joining the trees into a single tree by adding a root and then disposing of the private root key.

The group managers could be organized in a directed acyclic graph (DAG), e.g. two trees could share a common subtree. This would give alternative paths to some signers. There may be situations where this is advantageous, but the semantics of such a scheme are complex and involves many subtle issues, e.g. should all group managers (indirect and direct) of a signer get information on its identity, or should the signer decide on a path from a root and only reveal information to group managers along this path? Although we believe that the techniques we use for our construction would be useful also for this type of scheme we do not investigate such schemes further.

2.4 On Constructing Hierarchical Group Signatures

All known group signatures are based on the idea that the signer encrypts a secret of some sort using the group manager's public key, and then proves that the resulting cryptotext is on this special form. The security of the cryptosystem used implies anonymity, since no adversary can distinguish cryptotexts of two distinct messages if they are encrypted using the *same* public key. We generalize this approach.

First we consider the problem of forwarding partial information on the identity of the signer to group managers without leaking information. Each group manager M_β is given a secret key sk_β and a public key pk_β of a cryptosystem. We also give each signer S_α a public key pk_α that is used to identify the signer. Each signer is associated in the natural way with the path $\alpha_0, \alpha_1, \ldots, \alpha_\delta$ from the root $\rho = \alpha_0$ to the leaf $\alpha = \alpha_\delta$ in the tree T of group managers and signers. To compute a signature, the signer computes as part of the signature a chain

$$(C_0, C_1, \ldots, C_{\delta-1}) = \left(E_{\mathrm{pk}_{\alpha_0}}(\mathrm{pk}_{\alpha_1}), E_{\mathrm{pk}_{\alpha_1}}(\mathrm{pk}_{\alpha_2}), \ldots, E_{\mathrm{pk}_{\alpha_{\delta-1}}}(\mathrm{pk}_{\alpha_\delta})\right) \enspace .$$

Note that each cryptotext C_l in the list encrypts the public key $\mathrm{pk}_{\alpha_{l+1}}$ used to form the following cryptotext. The particular structure of the chain and the fact that all leaves are on the same depth in the tree ensures that a group manager M_β on depth l can try to open a signature by decrypting C_l, i.e. it computes $\mathrm{pk} = D_{\mathrm{sk}_\beta}(C_l)$.

If $\alpha_l = \beta$, then $\mathrm{pk} = \mathrm{pk}_{\alpha_{l+1}}$. Thus, if M_β manages signers, it learns the identity of the signer S_α, and if it manages other group managers it learns the identity of the group manager below it in the tree which (perhaps indirectly) manages the signer S_α.

Now suppose that $\alpha_l \neq \beta$, so $\mathrm{pk} \neq \mathrm{pk}_{\alpha_{l+1}}$. What does M_β, or indeed any outsider, learn about the identity of the signer S_α? It clearly does not learn anything from a cryptotext C_l about the encrypted cleartext, as long as the cryptosystem is semantically secure. However, if the cryptotext C_{l+1} somehow indicate which public key was used to form it, M_β, or any outsider, can simply look at C_{l+1} and recover the cleartext of C_l. This means that it can look at the chain of cryptotexts and extract information on the identity of the signer. We conclude that using the approach above, we need a cryptosystem which not only hides the cleartext, but also hides the public key used to form the cryptotext. Such a cryptosystem is called an *anonymous* cryptosystem [5].

Next we consider the problem of ensuring hierarchical traceability. This problem consists of two parts. We must ensure chosen message security to avoid that an illegitimate signer is able compute a valid signature at all. However, the more difficult problem is to ensure that the signer S_α not only formed $(C_0, \ldots, C_{\delta-1})$ as described above for some public keys $\mathrm{pk}_{\alpha_0}, \ldots, \mathrm{pk}_{\alpha_\delta}$, but also that the public keys used correspond to the unique path $\alpha_0, \alpha_1, \ldots, \alpha_\delta$ from the root $\rho = \alpha_0$ to the leaf $\alpha = \alpha_\delta$ corresponding to the signer S_α. This is the main obstacle to construct an efficient hierarchical group signature scheme.

3 Anonymous Cryptosystems

It turns out that the cryptosystem we use must not only be indistinguishable (semantically secure), but it must also have an incomparable security property which is called *anonymity* or *key-privacy*. This property was discussed in [1] and studied extensively by Bellare et al. in [5]. We formalize anonymity in Definition 3.4 below, but first we recall the definition of indistinguishability of a cryptosystem $\mathcal{CS} = (\mathsf{Kg}, E, D)$, with key generator Kg, encryption algorithm E, and decryption algorithm D, as defined by Goldwasser and Micali [30].

Experiment 3.1 (Indistinguishability, $\mathbf{Exp}^{\mathsf{ind}-b}_{\mathcal{CS},A}(\kappa)$ (cf. [30])).

$$\begin{aligned} (\mathrm{pk}, \mathrm{sk}) &\leftarrow \mathsf{Kg}(1^\kappa) \\ (m_0, m_1, s_{\mathrm{state}}) &\leftarrow A(\mathrm{pk}) \\ d &\leftarrow A(E_{\mathrm{pk}}(m_b), s_{\mathrm{state}}) \end{aligned}$$

We let $\mathbf{Adv}^{\mathsf{ind}}_{\mathcal{CS},A}(\kappa) = |\Pr[\mathbf{Exp}^{\mathsf{ind}-0}_{\mathcal{CS},A}(\kappa) = 1] - \Pr[\mathbf{Exp}^{\mathsf{ind}-1}_{\mathcal{CS},A}(\kappa) = 1]|$.

Definition 3.2. *Let $\mathcal{CS}$ be a cryptosystem. We say that $\mathcal{CS}$ is* indistinguishable *if for all $A \in \mathrm{PPT}^*$, $\mathbf{Adv}^{\mathsf{ind}}_{\mathcal{CS},A}(\kappa)$ is negligible.*

Informally, anonymity boils down to the property that the adversary cannot distinguish cryptotexts encrypted with distinct public keys.

Experiment 3.3 (Anonymity, $\mathbf{Exp}^{\mathsf{key\text{-}ind}-b}_{\mathcal{CS},A}(\kappa)$).

$$\begin{aligned} (\mathrm{pk}_0, \mathrm{sk}_0) &\leftarrow \mathsf{Kg}(1^\kappa), \quad (\mathrm{pk}_1, \mathrm{sk}_1) \leftarrow \mathsf{Kg}(1^\kappa) \\ (m, s_{\mathrm{state}}) &\leftarrow A(\mathrm{pk}_0, \mathrm{pk}_1) \\ d &\leftarrow A(E_{\mathrm{pk}_b}(m), s_{\mathrm{state}}) \end{aligned}$$

Note that compared to the standard definition of indistinguishability of cryptotexts, the roles played by public keys and messages are reversed. One could consider a variant definition which captures both types of indistinguishabilities, but we think it is more natural to think of anonymity as an additional property. We let $\mathbf{Adv}^{\mathsf{key\text{-}ind}}_{\mathcal{CS},A}(\kappa) = |\Pr[\mathbf{Exp}^{\mathsf{key\text{-}ind}-0}_{\mathcal{CS},A}(\kappa) = 1] - \Pr[\mathbf{Exp}^{\mathsf{key\text{-}ind}-1}_{\mathcal{CS},A}(\kappa) = 1]|$.

Definition 3.4. *Let $\mathcal{CS}$ be a cryptosystem. We say that $\mathcal{CS}$ is* anonymous *if for all $A \in \mathrm{PPT}^*$, $\mathbf{Adv}^{\mathsf{key\text{-}ind}}_{\mathcal{CS},A}(\kappa)$ is negligible.*

The property of anonymity is clearly useless if the cryptosystem is not indistinguishable, since it allows the encryption function to be the identity map. Thus, anonymity does not imply indistinguishability. To see that the other implication does not hold, note that if $\mathcal{CS}$ is an indistinguishable cryptosystem, then so is the cryptosystem where the encryption and decryption functions $c = E_{\mathrm{pk}}(m)$ and $D_{\mathrm{sk}}(c) = m$ are replaced by $(c, c') = E'_{\mathrm{pk}}(m) = (E_{\mathrm{pk}}(m), \mathrm{pk})$ and $D'_{\mathrm{sk}}(c, c') = D_{\mathrm{sk}}(c) = m$ respectively.

The lemma below characterizes the set of cryptosystems which are both indistinguishable and anonymous. Denote by $\mathbf{Exp}^{\mathsf{ind}-\mathcal{D}_{\mathrm{ind}}}_{\mathcal{CS},A}(\kappa)$ Experiment 3.1, but with the input $E_{\mathrm{pk}_b}(m)$ replaced by an element distributed according to a distribution D_κ, where $\mathcal{D}_{\mathrm{ind}} = \{D_\kappa\}$, and correspondingly for $\mathbf{Exp}^{\mathsf{anon}-\mathcal{D}_{\mathrm{ind}}}_{\mathcal{CS},A}(\kappa)$. We use $T_{\mathcal{D}}$ to denote the Turing machine that on input 1^κ returns a sample distributed according to D_κ.

Experiment 3.5 ($\mathcal{D}_{\mathrm{ind}}$-Indistinguishability, $\mathbf{Exp}^{\mathsf{ind}-\mathcal{D}_{\mathrm{ind}}}_{\mathcal{CS},A}(\kappa)$)**.**

$$\begin{aligned}(\mathrm{pk}, \mathrm{sk}) &\leftarrow \mathsf{Kg}(1^\kappa)\\ (m_0, m_1, s_{\mathrm{state}}) &\leftarrow A(\mathrm{pk})\\ d &\leftarrow A(T_{\mathcal{D}}(1^\kappa), s_{\mathrm{state}})\end{aligned}$$

Experiment 3.6 ($\mathcal{D}_{\mathrm{ind}}$-Anonymity, $\mathbf{Exp}^{\mathsf{key\text{-}ind}-\mathcal{D}_{\mathrm{ind}}}_{\mathcal{CS},A}(\kappa)$)**.**

$$\begin{aligned}(\mathrm{pk}_0, \mathrm{sk}_0) &\leftarrow \mathsf{Kg}(1^\kappa), \quad (\mathrm{pk}_1, \mathrm{sk}_1) \leftarrow \mathsf{Kg}(1^\kappa)\\ (m, s_{\mathrm{state}}) &\leftarrow A(\mathrm{pk}_0, \mathrm{pk}_1)\\ d &\leftarrow A(T_{\mathcal{D}}(1^\kappa), s_{\mathrm{state}})\end{aligned}$$

Lemma 3.7. *Let $\mathcal{CS}$ be a cryptosystem which is both indistinguishable and anonymous. Then there exists an efficiently sampleable distribution $\mathcal{D}_{\mathrm{ind}}$ such that for all $A \in \mathrm{PPT}^*$*

$$|\Pr[\mathbf{Exp}^{\mathsf{ind}-b}_{\mathcal{CS},A}(\kappa) = 1] - \Pr[\mathbf{Exp}^{\mathsf{ind}-\mathcal{D}_{\mathrm{ind}}(\kappa)}_{\mathcal{CS},A}(\kappa) = 1]| \enspace ,$$

is negligible for $b \in \{0, 1\}$. The reverse implication holds as well.

The intuition behind this lemma is that if a cryptosystem that is both indistinguishable and anonymous, then the cryptotexts are indistinguishable from a random distribution which is independent of *both* the key *and* the plaintext.

Proof. Suppose that a distribution $\mathcal{D}_{\mathrm{ind}}$ as in the lemma exists. The indistinguishability of $\mathcal{CS}$ then follows by a trivial hybrid argument. Suppose that $\mathcal{CS}$ is not anonymous. Then there exists an adversary $A \in \mathrm{PPT}^*$ such that

$$|\Pr[\mathbf{Exp}^{\mathsf{key\text{-}ind}-0}_{\mathcal{CS},A}(\kappa) = 1] - \Pr[\mathbf{Exp}^{\mathsf{key\text{-}ind}-1}_{\mathcal{CS},A}(\kappa) = 1]|$$

is non-negligible which by a trivial hybrid argument implies that

$$|\Pr[\mathbf{Exp}^{\mathsf{key\text{-}ind}-b}_{\mathcal{CS},A}(\kappa) = 1] - \Pr[\mathbf{Exp}^{\mathsf{key\text{-}ind}-\mathcal{D}_{\mathrm{ind}}}_{\mathcal{CS},A}(\kappa) = 1]|$$

is non-negligible for a fixed $b \in \{0, 1\}$, which we without loss assume to be 0. Let A' be the adversary in Experiment 3.1 defined as follows. On input pk it sets $\mathrm{pk}_0 = \mathrm{pk}$ generates $(\mathrm{pk}_1, \mathrm{sk}_1) = \mathsf{Kg}(1^\kappa)$ and hands $(\mathrm{pk}_0, \mathrm{pk}_1)$ to A, which returns (m, s_{state}).

Then A' returns (m, m, s_{state}). When handed (c, s_{state}) from the experiment, where c is either is $E_{\text{pk}_0}(m)$ or a sample from D_κ, it returns the output of $A(c, s_{\text{state}})$. By construction $\mathbf{Exp}^{\mathsf{ind}-0}_{\mathcal{CS},A'}(\kappa)$ is identically distributed to $\mathbf{Exp}^{\mathsf{key\text{-}ind}-0}_{\mathcal{CS},A}(\kappa)$, and $\mathbf{Exp}^{\mathsf{ind}-\mathcal{D}_{\text{ind}}}_{\mathcal{CS},A'}(\kappa)$ is identically distributed to $\mathbf{Exp}^{\mathsf{key\text{-}ind}-\mathcal{D}_{\text{ind}}}_{\mathcal{CS},A}(\kappa)$. This is a contradiction, since it implies that

$$|\Pr[\mathbf{Exp}^{\mathsf{ind}-0}_{\mathcal{CS},A'}(\kappa) = 1] - \Pr[\mathbf{Exp}^{\mathsf{ind}-\mathcal{D}_{\text{ind}}}_{\mathcal{CS},A'}(\kappa) = 1]|$$

is non-negligible.

Suppose next that $\mathcal{CS}$ is indistinguishable and anonymous. We define our prospective distribution $\mathcal{D}_{\text{ind}}$ as follows. To generate a sample from $\mathcal{D}_{\text{ind}}$, generate a key pair $(\text{pk}', \text{sk}') = \mathsf{Kg}(1^\kappa)$ and output an encryption $E_{\text{pk}'}(m')$, where $m' = 0^\kappa$. This implies that $\mathcal{D}_{\text{ind}}$ is efficiently sampleable. Assume that

$$|\Pr[\mathbf{Exp}^{\mathsf{ind}-b}_{\mathcal{CS},A}(\kappa) = 1] - \Pr[\mathbf{Exp}^{\mathsf{ind}-\mathcal{D}_{\text{ind}}}_{\mathcal{CS},A}(\kappa) = 1]|$$

is non-negligible for $b = 0$ (then it is also non-negligible for $b = 1$, since $\mathcal{CS}$ is indistinguishable). Let A'_0 be the adversary in Experiment 3.3 that does the following. On input $(\text{pk}_0, \text{pk}_1)$ it hands pk_0 to A which returns (m_0, m_1). Then A'_0 returns m_0, and is given $E_{pk_b}(m_0)$ for a randomly chosen $b \in \{0, 1\}$ by the experiment. It hands $E_{pk_b}(m_0)$ to A and returns the output of A. A'_1 is identical to A'_0 except that it hands m' to the experiment instead of m_0. From the construction follows that $\mathbf{Exp}^{\mathsf{ind}-0}_{\mathcal{CS},A}(\kappa)$ and $\mathbf{Exp}^{\mathsf{ind}-\mathcal{D}_{\text{ind}}}_{\mathcal{CS},A}(\kappa)$ are identically distributed to $\mathbf{Exp}^{\mathsf{key\text{-}ind}-0}_{\mathcal{CS},A'_0}(\kappa)$ and $\mathbf{Exp}^{\mathsf{key\text{-}ind}-1}_{\mathcal{CS},A'_1}(\kappa)$ respectively. Thus

$$|\Pr[\mathbf{Exp}^{\mathsf{key\text{-}ind}-0}_{\mathcal{CS},A'_0}(\kappa) = 1] - \Pr[\mathbf{Exp}^{\mathsf{key\text{-}ind}-1}_{\mathcal{CS},A'_1}(\kappa) = 1]|$$

is non-negligible. From the anonymity of $\mathcal{CS}$ we have that

$$|\Pr[\mathbf{Exp}^{\mathsf{key\text{-}ind}-0}_{\mathcal{CS},A'_b}(\kappa) = 1] - \Pr[\mathbf{Exp}^{\mathsf{key\text{-}ind}-1}_{\mathcal{CS},A'_b}(\kappa) = 1]|$$

is negligible for $b \in \{0, 1\}$. A hybrid argument implies that

$$|\Pr[\mathbf{Exp}^{\mathsf{key\text{-}ind}-b}_{\mathcal{CS},A'_0}(\kappa) = 1] - \Pr[\mathbf{Exp}^{\mathsf{key\text{-}ind}-b}_{\mathcal{CS},A'_1}(\kappa) = 1]|$$

is non-negligible for some $b \in \{0, 1\}$. Without loss we assume $b = 0$. Denote by A'' the adversary in Experiment 3.1 defined as follows. Given input pk it hands (pk) to A. When A returns (m_0, m_1), it outputs (m_0, m'), and receives either $E_{pk}(m_0)$ or $E_{pk}(m')$, which it forwards to A. Finally, it returns the output of A. Since, $\mathbf{Exp}^{\mathsf{ind}-0}_{\mathcal{CS},A''}(\kappa)$ is identically distributed to $\mathbf{Exp}^{\mathsf{key\text{-}ind}-0}_{\mathcal{CS},A'_0}(\kappa)$ and $\mathbf{Exp}^{\mathsf{ind}-1}_{\mathcal{CS},A''}(\kappa)$ is identically distributed to $\mathbf{Exp}^{\mathsf{key\text{-}ind}-0}_{\mathcal{CS},A'_1}(\kappa)$, this contradicts the indistinguishability of $\mathcal{CS}$. □

Note that $\mathcal{D}_{\text{ind}}$ depends on $\mathcal{CS}$ but is independent of all stochastic variables in the experiment. Below we show that the probabilistic cryptosystem of Goldwasser and Micali [30] is anonymous.

Remark 3.8. Several standard probabilistic cryptosystems can be made anonymous by minor modifications. E.g. it is not hard to see that the ElGamal [26] cryptosystem is anonymous if the group in which it is employed is fixed for each value of the security parameter.

4 A Construction under General Assumptions

In this section we show how hierarchical group signatures can be constructed under general assumptions. Our focus is on feasibility and conceptual simplicity. We prove the following theorem.

Theorem 4.1. *If there exists a family of trapdoor permutations, then there exists a secure hierarchical group signature scheme.*

To prove the theorem we construct a hierarchical group signature scheme by augmenting the group signature scheme of [7] with additional cryptotexts and a non-interactive zero-knowledge proof.

4.1 Assumptions and Primitives Used

Before we give our construction we review some constructions and results on which our construction is based.

4.1.1 Group Signature Scheme

The first building block we need is a group signature scheme secure under the assumption that trapdoor permutations exists. As shown by Bellare et al. such a scheme exists.

Theorem 4.2 (cf. [7])**.** *If there exists a family of trapdoor permutations, then there exists a secure group signature scheme* $\mathcal{GS} = (\mathsf{GKg}, \mathsf{GSig}, \mathsf{GVf}, \mathsf{Open})$.

4.1.2 Public Key Cryptosystem

The probabilistic cryptosystem of Goldwasser and Micali [30] is indistinguishable. We prove that their construction is also anonymous, but first we recall their construction.

Their construction is based on the existence of non-approximable trapdoor predicates. This concept can be captured in modern terminology as follows. A family of trapdoor permutations is a triple of polynomial time algorithms $\mathcal{F} = (\mathsf{Gen}, \mathsf{Eval}, \mathsf{Invert})$. The instance generator $\mathsf{Gen}(1^\kappa)$ outputs a description f of a permutation of $\{0,1\}^\kappa$ and a trapdoor f^{-1}. The evaluation algorithm $\mathsf{Eval}(1^\kappa, f, x)$ evaluates the permutation on input $x \in \{0,1\}^\kappa$. The the corresponding inversion algorithm $\mathsf{Invert}(1^\kappa, f^{-1}, y)$ evaluates the inverse permutation on input $y \in \{0,1\}^\kappa$. We abuse notation and write $f(x)$ and $f^{-1}(y)$ for the evaluation of the permutation and inverse permutation as described above. The last requirement on the family of trapdoor permutations is that it must be infeasible for any $A \in \mathrm{PPT}^*$ given f and $y = f(x)$, where $x \in \{0,1\}^\kappa$, to compute $x = f^{-1}(y)$. A hard-core bit for $\mathcal{F}$ is a family of functions $\mathcal{B} = \{B_\kappa : \{0,1\}^\kappa \rightarrow \{0,1\}\}$ such that it is infeasible to compute $B_k(x)$, given only f and $f(x)$ for a random $x \in \{0,1\}^\kappa$. Goldreich and Levin [29] show how to construct a family of trapdoor permutations $\mathcal{F}$ with a hard-core bit $\mathcal{B}$ from any family of trapdoor permutations.

The cryptosystem $\mathcal{GM} = (\mathsf{GMKg}, E, D)$ of Goldwasser and Micali [30] using $\mathcal{F}$ and $\mathcal{B}$ can be defined as follows (using modern terminology). The key generator $\mathsf{GMKg}(1^\kappa)$ simply outputs $(\mathrm{pk}, \mathrm{sk}) = (f, f^{-1}) = \mathsf{Gen}(1^\kappa)$. To compute a cryptotext $E_{\mathrm{pk}}(m)$ of a bit $m \in \{0,1\}$, choose $r \in \{0,1\}^\kappa$, and output $(f(r), \mathcal{B}(r) \oplus m)$. To decrypt a cryptotext (c, c'), compute $D_{\mathrm{sk}}(c, c') = \mathcal{B}(f^{-1}(c)) \oplus c'$. To encrypt a bit-string the encryption function is invoked with a fresh randomly chosen r for each bit in the natural way. Goldwasser and Micali essentially show the following theorem.

Theorem 4.3. *If $\mathcal{F}$ is a trapdoor permutation family with hard-core bit $\mathcal{B}$, then $\mathcal{GM}$ is indistinguishable.*

We show that the $\mathcal{GM}$ cryptosystem is also anonymous.

Lemma 4.4. *If $\mathcal{F}$ is a trapdoor permutation family with hard-core bit $\mathcal{B}$, then $\mathcal{GM}$ is anonymous.*

Proof. Suppose that $\mathcal{GM}$ is not anonymous. Let $U_{\kappa+1}$ be the uniform and independent distribution over $\{0,1\}^{\kappa+1}$. Then for some adversary $A \in \mathrm{PPT}^*$,

$$|\Pr[\mathbf{Exp}^{\mathsf{ind}-b}_{\mathcal{CS},A}(\kappa) = 1] - \Pr[\mathbf{Exp}^{\mathsf{ind}-U_{\kappa+1}}_{\mathcal{CS},A}(\kappa) = 1]|$$

is non-negligible for a fixed $b \in \{0,1\}$. Without loss we assume $b = 0$. Since $\mathcal{GM}$ is a bitwise cryptosystem, we may without loss assume that $m_0 = 0$ and $m_1 = 1$. Let $m \in \{0,1\}$ be randomly distributed, then a cryptotext $E_{\mathrm{pk}}(m) = (f(r), \mathcal{B}(r) \oplus m)$ is distributed according to $U_{\kappa+1}$, since f is a permutation and $\mathcal{B}(r) \oplus m$ is uniformly and independently distributed. A trivial average argument implies that $|\Pr[\mathbf{Exp}^{\mathsf{ind}-b}_{\mathcal{CS},A}(\kappa) = 1] - \Pr[\mathbf{Exp}^{\mathsf{ind}-1}_{\mathcal{CS},A}(\kappa) = 1]|$ is non-negligible which is a contradiction. □

4.1.3 Non-Interactive Zero-Knowledge Proofs

Non-interactive zero-knowledge proofs (NIZK) were introduced by Blum, Feldman, and Micali [9]. Several works have since refined and extended the notion in various ways. Following [7] we employ the definition of adaptive zero-knowledge for NIZK introduced by Feige, Lapidot, and Shamir [27] and we use the notion of simulation soundness introduced by Sahai [46]. The notion of simulation soundness is strengthened by De Santis et al. [47]. In contrast to [7], the NIZK we use must be adaptive zero-knowledge for polynomially many statements, and not only for a single statement. The requirement on simulation soundness is in fact unchanged compared with [7], i.e. single statement simulation soundness suffices.

Definition 4.5 (NIPS)**.** *A triple $(p(\kappa), P, V)$ is an* efficient adaptive non-interactive proof system *(NIPS) for a language $L \in \mathrm{NP}$ with witness relation R if $p(\kappa)$ is a polynomial and P and V are probabilistic polynomial time machines such that*

1. *Completeness.* $(x, w) \in R$ *and* $\xi \in \{0,1\}^{p(\kappa)}$ *implies* $V(x, P(x, w, \xi), \xi) = 1$.
2. *Soundness. For all computable functions A,* $\Pr_{\xi \in \{0,1\}^{p(\kappa)}}[A(\xi) = (x, \pi) \wedge x \notin L \wedge V(x, \pi, \xi) = 1]$ *is negligible in κ.*

We suppress p in our notation of a NIPS and simply write (P, V).

Loosely speaking a non-interactive zero-knowledge proof system is a NIPS, which is also zero-knowledge, but there are several flavors of zero-knowledge. We need a NIZK which is adaptive zero-knowledge (for a single statement) in the sense of Feige, Lapidot, and Shamir [27].

Experiment 4.6 (Adaptive Indistinguishability, $\mathbf{Exp}^{\mathsf{adind}-b}_{(P,V,S),A}(\kappa)$ (cf. [27]))**.**

$$\begin{array}{ll} \xi \leftarrow \{0,1\}^{f(\kappa)} & \text{if } b = 0 \\ (\xi, s_{\text{simstate}}) \leftarrow S(1^\kappa) & \text{if } b = 1 \\ \quad t \leftarrow \emptyset & \end{array}$$

While $\quad (t \neq \perp) \quad$ do

$$(s_{\text{state}}, t, w) \leftarrow \begin{cases} A(\mathsf{choose}, P(t, w, \xi)) & \text{if } (t, w) \in R \text{ and } b = 0 \\ A(\mathsf{choose}, S(t, \xi, s_{\text{simstate}})) & \text{if } (t, w) \in R \text{ and } b = 1 \\ A(\mathsf{choose}, \perp) & \text{otherwise} \end{cases}$$

Done

$d \leftarrow A(s_{\text{state}})$

The advantage in the experiment is defined

$$\mathbf{Adv}^{\mathsf{adind}}_{(P,V,S),A}(\kappa) = |\Pr[\mathbf{Exp}^{\mathsf{adind}-0}_{(P,V,S),A}(\kappa) = 1] - \Pr[\mathbf{Exp}^{\mathsf{adind}-1}_{(P,V,S),A}(\kappa) = 1]|$$

and the notion of adaptive zero-knowledge is given below.

Definition 4.7 (Adaptive Zero-Knowledge (cf. [27]))**.** *A NIPS (P, V) is adaptive zero-knowledge (NIZK) if there exists a polynomial time Turing machine S such that $\mathbf{Adv}^{\mathsf{adind}}_{(P,V,S),A}(\kappa)$ is negligible for all $A \in \mathrm{PPT}^*$.*

In cryptographic proofs one often performs hypothetic experiments where the adversary is run with simulated NIZKs. If the experiment simulates NIZKs to the adversary, the adversary could potentially gain the power to compute valid proofs of false statements. For a simulation sound NIZK this is not possible.

Experiment 4.8 (Simulation Soundness, $\mathbf{Exp}^{\mathsf{sims}}_{(P,V,S),A}(\kappa)$ (cf. [47]))**.**

$$\begin{array}{l} (\xi, s_{\text{simstate}}) \leftarrow S(1^\kappa) \\ (t, \pi) = A^{S(\cdot, \xi, s_{\text{simstate}})}(\xi) \end{array}$$

Let Q be the set of proofs returned by the $S(\cdot, \xi, s_{\text{simstate}})$ oracle. Return 1 if $\pi \notin Q$, $t \notin L$, and $V(t, \pi, \xi) = 1$, and 0 otherwise.

Definition 4.9 (Simulation Soundness (cf. [46, 47]))**.** *A NIZK (P, V) with polynomial time simulator S for a language L is unbounded simulation sound if*

$$\mathbf{Adv}^{\mathsf{sims}}_{(P,V,S),A}(\kappa) = \mathbf{Exp}^{\mathsf{sims}}_{(P,V,S),A}(\kappa)$$

is negligible for all $A \in \mathrm{PPT}^*$.

De Santis et al. [47] extend the results in [27] and [46] and prove the following result.

Theorem 4.10. *If there exists a family of trapdoor permutations, then there exists a simulation sound NIZK for any language in* NP.

In the the rest of this paper we abbreviate "efficient non-interactive adaptive zero-knowledge unbounded simulation sound proof" by NIZK.

4.2 Our Construction

We now describe our hierarchical group signature scheme $\mathcal{HGS} = (\mathsf{HKg}, \mathsf{HSig}, \mathsf{HVf}, \mathsf{HOpen})$. We let $\mathcal{F} = (\mathsf{Gen}, \mathsf{Eval}, \mathsf{Invert})$ denote a family of trapdoor permutations with a hard-core bit $\mathcal{B}$, and assume that a Goldwasser-Micali cryptosystem $\mathcal{GM}$ has been constructed from this. We denote by $\mathcal{GS} = (\mathsf{GKg}, \mathsf{GSig}, \mathsf{GVf}, \mathsf{Open})$ the group signature scheme of Bellare et al. also constructed from $\mathcal{F}$. However, we view this as a hierarchical group signature scheme of depth 1 and use the corresponding notation, i.e. the public and secret keys of the group manager M_ρ are denoted by $\mathrm{hpk}(\rho)$ and $\mathrm{hsk}(\rho)$ (not by gpk and gmsk etc. as in [7]). Below we also use $\mathcal{F}$ to construct a NIZK for a language L_{HGS}.

First keys to the $\mathcal{GS}$ group signature scheme are generated, where the signers correspond to the signers in the hierarchical group signature scheme we are constructing. However, the root group manager is not given its usual secret opening key $\mathrm{gsk}(\rho)$. Instead, each group manager (including the root) is given a key pair $(\mathrm{pk}_\beta, \mathrm{sk}_\beta)$ of the $\mathcal{GM}$ cryptosystem. When a signer S_α signs a message m it first forms a group signature σ of the message m. Suppose that the signer corresponds to the path $\alpha_0, \ldots, \alpha_\delta$ in the tree, i.e. $\alpha_0 = \rho$ and $\alpha_\delta = \alpha$. Then the signer forms a chain of cryptotexts $C = (E_{\mathrm{pk}_{\alpha_0}}(\mathrm{pk}_{\alpha_1}), \ldots, E_{\mathrm{pk}_{\alpha_{\delta-1}}}(\mathrm{pk}_{\alpha_\delta}))$. Finally, it forms a NIZK π that the chain of cryptotexts C is formed in this way, and that the encrypted path corresponds to the identity of the signer hidden in the group signature σ. The hierarchical group signature consists of the triple (σ, C, π). Verification of a signature corresponds to verifying the NIZK. Opening a signature using the secret opening key of a group manager at depth l corresponds to decrypting the lth cryptotext.

Algorithm 4.11 (Key Generation, $\mathsf{HKg}(1^\kappa, T)$)**.** The key generation is defined as follows.

1. Generate a random string $\xi \in \{0,1\}^*$ sufficiently long for a NIZK based on $\mathcal{F}$ of the language L_{HGS} defined below.

2. For each node α in T, compute $(\mathrm{pk}_\alpha, \mathrm{sk}_\alpha) = \mathsf{GMKg}(1^\kappa)$.

3. Let I be the bijection mapping each list $(\mathrm{pk}_{\alpha_0}, \ldots, \mathrm{pk}_{\alpha_\delta})$ such that $\alpha_0, \ldots, \alpha_\delta$ is a path in T, where $\alpha_0 = \rho$ and $\alpha_\delta \in \mathcal{L}(T)$ to α_δ. Define I to map anything else to $\perp$. Denote by $T_{\mathcal{GS}}$ the tree with root ρ and leaves $\mathcal{L}(T)$.

4. Run $(\mathrm{gpk}, \mathrm{gsk}) = \mathsf{GKg}(1^\kappa, T_{\mathcal{GS}})$, and set $(\mathrm{hpk}(\alpha), \mathrm{hsk}(\alpha)) = ((\mathrm{pk}_\alpha, \mathrm{gpk}(\alpha)), (\mathrm{sk}_\alpha, \mathrm{gsk}(\alpha)))$ for $\alpha \in \mathcal{L}(T)$.

5. Set $(\mathrm{hpk}(\rho), \mathrm{hsk}(\rho)) = ((\xi, \mathrm{pk}_\rho, \mathrm{gpk}(\rho)), \mathrm{sk}_\rho)$ and set $(\mathrm{hpk}(\beta), \mathrm{hsk}(\beta)) = (\mathrm{pk}_\beta, \mathrm{sk}_\beta)$ for $\beta \notin \mathcal{L}(T)$, $\beta \neq \rho$ (note that $\mathrm{hsk}(\rho)$ does not contain $\mathrm{gsk}(\rho)$).

6. Output $(\mathrm{hpk}, \mathrm{hsk})$.

The result of running the above algorithm is illustrated in Figure 3.

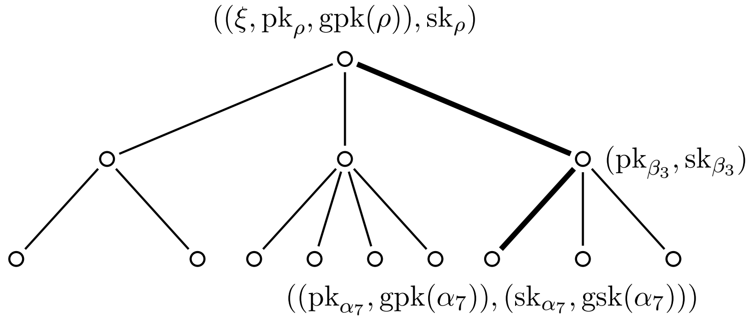

Figure 3: The figure illustrates the public and secret keys along a path (the thick edges) in the tree of keys corresponding to Figure 1. Each node contains a pair of public and secret keys.

Algorithm 4.12 (Signing, $\mathsf{HSig}(m, T, \mathrm{hpk}, \mathrm{hsk}(\alpha))$)**.** Let $\alpha_0, \ldots, \alpha_\delta$ be the path to the signer S_α, i.e. $\rho = \alpha_0$ and $\alpha_\delta = \alpha$.

1. Compute
$$\sigma = \mathsf{GSig}(m, T_{\mathcal{GS}}, \mathrm{gpk}, \mathrm{gsk}(\alpha))$$
and
$$C = (C_0, \ldots, C_{\delta-1}) = (E_{\mathrm{pk}_{\alpha_0}}(\mathrm{pk}_{\alpha_1}), \ldots, E_{\mathrm{pk}_{\alpha_{\delta-1}}}(\mathrm{pk}_{\alpha_\delta})) \ .$$

2. Compute a NIZK π of the language L_{HGS}
$$\left\{ (T, \mathrm{hpk}, m, \sigma, C) \;\middle|\; \begin{array}{l} \exists \mathrm{pk}_0, \ldots, \mathrm{pk}_{\delta-1} : \quad \alpha_0 = \rho \ , \\ C_l = E_{\mathrm{pk}_l}(\mathrm{pk}_{l+1}) \quad \text{for } l = 0, \ldots, \delta-1 \ , \\ I(\mathrm{pk}_0, \ldots, \mathrm{pk}_{\delta-1}) = \alpha \ , \quad \text{and} \\ \sigma = \mathsf{GSig}(m, T_{\mathcal{GS}}, \mathrm{gpk}, \mathrm{gsk}(\alpha)) \end{array} \right\} \ .$$

3. Output (σ, C, π).

Remark 4.13. Above the complete tree of public keys hpk is given to the signing algorithm, despite that only the public keys $\mathrm{hpk}(\alpha_0), \ldots, \mathrm{hpk}(\alpha_\delta)$ along the path are needed. This is for notational convenience.

Algorithm 4.14 (Verification, $\mathsf{HVf}(T, \mathrm{hpk}, m, (\sigma, C, \pi))$)**.** On input a signature (σ, C, π) invoke the NIZK verifier V on input $((T, \mathrm{hpk}, m, \sigma, C,), \pi)$ and return the result.

Algorithm 4.15 (Opening, $\mathsf{HOpen}(T, \mathrm{gpk}, \mathrm{gsk}(\beta), m, (\sigma, C, \pi))$)**.** If $\mathsf{HVf}(T, \mathrm{hpk}, m, (\sigma, C, \pi)) = 0$, then return $\perp$. Otherwise compute $\mathrm{pk}_\alpha = D_{\mathrm{sk}_\beta}(C_l)$. If $\alpha \in \beta$ return α and otherwise $\perp$.

Consider the construction $\mathcal{HGS}$ above, where $\mathcal{GM}$ is replaced by any indistinguishable cryptosystem $\mathcal{CS}$. Is the result secure? The answer is no. The problem is that the security of the cryptosystem $\mathcal{CS}$ does not imply that a cryptotext does not reveal the public key used for encryption. An adversary could possibly identify which key was used for encryption simply by looking at a cryptotext, and thereby extract partial information on the identity of the signer. Fortunately, we have shown that $\mathcal{GM}$ is anonymous which solves the problem.

Remark 4.16. In Section 7 we describe an alternative construction which seems better suited if we try to eliminate the trusted key generator, but which is harder to analyze.

Remark 4.17. It is an interesting question whether we can instantiate the Goldwasser-Micali scheme using RSA in our setting. The problem is that for a given security parameter the RSA permutations are defined for different moduli. This can be solved as follows. We modify the Goldwasser-Micali encryption algorithm such that it repeatedly chooses r until $f(r) < 2^\kappa$. This implies that $f(r) < N$ for all κ-bit moduli N. The probability that r has this property is at least $1/4$. Given that we put a polynomial bound on the number of tried r, the encryption process fails with negligible probability. The security of the modified scheme follows from the security of the original since the original scheme uses an r with $f(r) < 2^\kappa$ with probability at least $1/4$. Other examples of RSA-based trapdoor permutations with the same property can be found [5].

4.3 Security Analysis

We prove the following lemma on the security of our construction, from which Theorem 4.1 follows immediately.

Lemma 4.18. *If $\mathcal{F}$ is a family of trapdoor permutations, then $\mathcal{HGS}$ is secure.*

The proof of hierarchical anonymity is similar in structure to the proof of full anonymity for the one level case given in [7], e.g. we need only single-statement simulation soundness. In the proof of hierarchical traceability we cannot proceed as in the proof of full traceability [7], since we cannot simulate answers of the signature oracle without invoking the simulator of the NIZK a polynomial number of times. This is why we must assume that the NIZK is adaptive zero-knowledge.

Proof. We prove the hierarchical anonymity and the hierarchical traceability of $\mathcal{HGS}$ separately.

Proof of Hierarchical Anonymity. Suppose to the contrary that the adversary A breaks hierarchical anonymity. Then we have $\mathbf{Adv}^{\mathsf{anon}}_{\mathcal{HGS},A}(\kappa, T) \geq 1/\kappa^c$ for some polynomial size tree T, constant $c > 0$ and κ in an infinite index set $\mathcal{N}$. We construct a machine A' that runs A as a blackbox and breaks the hierarchical anonymity (i.e. full anonymity [7]) of $\mathcal{GS}$.

Definition of A'.

The adversary A' simulates the hierarchical anonymity experiment, Experiment 2.2, with $\mathcal{HGS}$ to A. It also plays the role of adversary in Experiment 2.2 with $\mathcal{GS}$.

The key generation is simulated as follows. First the NIZK simulator S is invoked to compute a reference string with a trapdoor $(\xi, s_{\text{simstate}})$. The string ξ is used instead of a random string. Recall that $T_{\mathcal{GS}}$ denotes the tree having ρ (the root of T) as root, and children $\mathcal{L}(T)$. A' first waits until it receives gpk and $(\text{gsk}(\alpha))_{\alpha \in \mathcal{L}(T)}$. Then it simulates the remaining part of the key generation honestly except that it uses these values, and it does not define $\text{gsk}(\rho)$ at all. Thus, the keys of all intermediate group managers are generated by A'.

In each iteration in the simulated experiment A may request $\text{gsk}(\alpha)$ for some group manager M_α. The only such request A' cannot answer honestly and correctly is a request for $\text{gsk}(\rho)$ which it answers by $\perp$, but this is not a problem since the experiment outputs 0 in this case anyway.

Queries to the $\mathsf{HOpen}(T, \text{hpk}, \text{hsk}(\cdot), \cdot, \cdot)$ oracle are simulated in the following way. Given a query on the form $(\beta, m, (\sigma, C, \pi))$, A' first checks that $\beta \in T$ and

$$\mathsf{HVf}(T, \text{hpk}, m, (\sigma, C, \pi)) = 1 \ .$$

If not it returns $\perp$. If so it asks its $\mathsf{Open}(T_{\mathcal{GS}}, \text{gpk}, \text{gsk}(\cdot), \cdot, \cdot)$ oracle the question (β, m, σ), which replies by $\alpha \in \mathcal{L}(T)$. Let $\alpha_0, \ldots, \alpha_\delta$ be its corresponding path, i.e. $\alpha_0 = \rho$ and $\alpha_\delta = \alpha$. Let β be on depth l. Then A' instructs the $\mathsf{HOpen}(T, \text{hpk}, \text{hsk}(\cdot), \cdot, \cdot)$ oracle to return α_{l+1} to A. Note that the answers computed in this way are not necessarily correct.

When A outputs $(\alpha^{(0)}, \alpha^{(1)}, m)$, A' outputs this as well. When A' is given a signature σ from its experiment, it computes δ samples $C = (C_0, \ldots, C_{\delta-1})$ distributed according to the distribution $\mathcal{D}_{\text{ind}}$ guaranteed to exist by Lemma 3.7. Then it invokes the simulator S on $((T, \text{hpk}, m, \sigma, C), s_{\text{simstate}})$ to form a proof π, and hands (σ, C, π) to A.

Eventually A outputs a bit d, which A' then returns as output.

Analysis of A'. We divide our analysis into three claims. Denote by $A^b_{\text{c,o,p}}$ the machine that on input κ simply simulates Experiment 2.2 with $\mathcal{HGS}$ to A and outputs the result. Then clearly $A^b_{\text{c,o,p}}(\kappa)$ is identically distributed to $\mathbf{Exp}^{\mathsf{anon}-b}_{\mathcal{HGS},A}(\kappa)$ for $b \in \{0,1\}$. Denote by $A^b_{\text{c,o}}$ the machine which is identical to $A^b_{\text{c,o,p}}$ except for the following two changes. Firstly, instead of generating ξ as a random string, it invokes the NIZK simulator S, which returns $(\xi, s_{\text{simstate}})$. Secondly, to form the NIZK π, it invokes the NIZK simulator S on input $((T, \text{hpk}, m, \sigma, C), s_{\text{simstate}})$. Thus, except from the fact that the proof π is simulated, $A^b_{\text{c,o}}$ simulates Experiment 2.2 with $\mathcal{HGS}$ to A. We also define A^b_{c} to be identical to $A^b_{\text{c,o}}$ except that it simulates the $\mathsf{HOpen}(T, \text{hpk}, \text{hsk}(\cdot), \cdot, \cdot)$ oracle to A precisely as A' does. Finally, we define A^b to be identical to A^b_{c} except that the $C_0, \ldots, C_\delta$ in the challenge signature are generated precisely as A' does.

Thus, by construction A^b is identically distributed to $\mathbf{Exp}^{\mathsf{anon}-b}_{\mathcal{GS},A'}(\kappa)$. This gives us a chain of distributions $A^b_{\text{c,o,p}}, A^b_{\text{c,o}}, A^b_{\text{c}}, A^b$ starting with $\mathbf{Exp}^{\mathsf{anon}-b}_{\mathcal{HGS},A}(\kappa)$ and ending with $\mathbf{Exp}^{\mathsf{anon}-b}_{\mathcal{GS},A'}(\kappa)$. In the following claims we show that the distance between each pair of distributions is negligible.

Claim 2. There is a negligible function $\epsilon_1(\kappa)$ in κ such that

$$|\Pr[A^b_{\mathrm{c,o,p}}(\kappa) = 1] - \Pr[A^b_{\mathrm{c,o}}(\kappa) = 1]| < \epsilon_1(\kappa) \ .$$

Proof. The proof follows from the adaptive zero-knowledge of the NIZK (P, V, S).

Consider the adaptive zero-knowledge adversary A_{adzk} in Experiment 4.6 which we define as follows. It waits for ξ from the experiment. Then starts the simulation of $A_{\mathrm{c,o}}$ except that it uses the ξ received from the experiment. Then it continues the simulation of $A_{\mathrm{c,o}}$ until it is about to generate the NIZK π. Instead of generating the NIZK, it requests a NIZK π of the statement $(T, \mathrm{hpk}, m, \sigma, C)$ from its experiment. It must also hand the experiment a witness of this statement, but this is easy since the statement was generated honestly. Finally, it continues the simulation of $A_{\mathrm{c,o}}$ until it halts.

It follows that $A^b_{\mathrm{c,o,p}}(\kappa)$ and $A^b_{\mathrm{c,o}}(\kappa)$ are identically distributed to the outcome of the experiments $\mathbf{Exp}^{\mathsf{adind}-0}_{(P,V,S),A_{\mathrm{adzk}}}(\kappa)$ and $\mathbf{Exp}^{\mathsf{adind}-1}_{(P,V,S),A_{\mathrm{adzk}}}(\kappa)$ respectively. The reader should note that if π is a simulated proof, then the "proved" statement is always true. Thus, simulation soundness plays no role here. From the adaptive zero-knowledge of the NIZK we have that there exists a negligible function $\epsilon_1(\kappa)$ such that

$$|\mathbf{Exp}^{\mathsf{adind}-0}_{(P,V,S),A_{\mathrm{adzk}}}(\kappa) - \mathbf{Exp}^{\mathsf{adind}-1}_{(P,V,S),A_{\mathrm{adzk}}}(\kappa)| < \epsilon_1(\kappa) \ ,$$

and the claim follows. □

Claim 3. There is a negligible function $\epsilon_2(\kappa)$ such that

$$|\Pr[A^b_{\mathrm{c,o}}(\kappa) = 1] - \Pr[A^b_{\mathrm{c}}(\kappa) = 1]| < \epsilon_2(\kappa) \ .$$

Proof. The proof of this claim is similar to the proof in [46] and follows from the simulation soundness of the NIZK.

A query $(\beta, m, (\sigma, C, \pi))$ from A to the simulated $\mathsf{Open}(T, \mathrm{hpk}, \mathrm{hsk}(\cdot), \cdot, \cdot)$ is answered incorrectly precisely when π is a valid proof, i.e., $V((T, \mathrm{hpk}, m, \sigma, C),$ $\pi, \xi) = 1$, but $(T, \mathrm{hpk}, m, \sigma, C) \notin L_{\mathrm{HGS}}$. Denote by $E_{\mathrm{bq}}(A^b_{\mathrm{c}})$ the event that A asks such a query in the simulation by A^b_{c}, and correspondingly for $A^b_{\mathrm{c,o}}$.

We construct an adversary A_{sims} against simulation soundness, i.e. Experiment 4.8, as follows. It simulates A^b_{c} (or $A^b_{\mathrm{c,o}}$). Whenever A asks a query $(\beta, m, (\sigma, C, \pi))$, A_{sims} interrupts the simulation of A^b_{c} (or $A^b_{\mathrm{c,o}}$) and checks whether the query is such that $(T, \mathrm{hpk}, m, \sigma, C) \in L_{\mathrm{HGS}}$. This is easily done using the keys to the cryptosystems and the group signature scheme. If $(T, \mathrm{hpk}, m, \sigma, C) \notin L_{\mathrm{HGS}}$, then A_{sims} outputs $((T, \mathrm{hpk}, m, \sigma, C),\ \pi)$. From the simulation soundness we conclude that there exists a negligible function $\epsilon(\kappa)$ such that

$$\Pr[E_{\mathrm{bq}}(A^b_{\mathrm{c}})] = \Pr[E_{\mathrm{bq}}(A^b_{\mathrm{c,o}})] = \mathbf{Exp}^{\mathsf{sims}}_{(P,V,S),A_{\mathrm{sims}}}(\kappa) < \epsilon(\kappa) \ .$$

By construction $(A^b_{\mathrm{c}}(\kappa) \mid \overline{E_{\mathrm{bq}}(A^b_{\mathrm{c}})})$ and $(A^b_{\mathrm{c,o}}(\kappa) \mid \overline{E_{\mathrm{bq}}(A^b_{\mathrm{c,o}})})$ are identically distributed. The claim follows. □

Claim 4. There is a negligible function $\epsilon_3(\kappa)$ in κ such that

$$|\Pr[A^b_{\mathrm{c}}(\kappa) = 1] - \Pr[A^b(\kappa) = 1]| < \epsilon_3(\kappa) \ .$$

Proof. The claim follows from the indistinguishability and anonymity of $\mathcal{GM}$ using Theorem 4.3, Lemma 4.4, and Lemma 3.7 by use of a standard hybrid argument.

We define a sequence of hybrid machines $A_{\mathrm{ind},l}$ for $l = 0, \ldots, \delta - 1$ as follows. $A_{\mathrm{ind},l}$ simulates A^b_{c} until it has computed $(C_0, \ldots, C_{\delta-1})$. Then it computes l samples $(C'_0, \ldots, C'_l)$ distributed according to the $\mathcal{D}_{\mathrm{ind}}$ distribution guaranteed by Lemma 3.7. Finally, it replaces $(C_0, \ldots, C_{\delta-1})$ in its simulation by $(C'_0, \ldots, C'_l, C_{l+1}, \ldots, C_{\delta-1})$, and continues the simulation of A^b_{c}. By construction, $A_{\mathrm{ind},-1}(\kappa)$ and $A_{\mathrm{ind},\delta-1}(\kappa)$ are identically distributed to $A^b_{\mathrm{c}}(\kappa)$ and $A^b(\kappa)$ respectively.

Suppose that the claim is false, i.e. there exists a constant c and an infinite index set $\mathcal{N}'$ such that

$$|\Pr[A_{\mathrm{ind},-1} = 1] - \Pr[A_{\mathrm{ind},\delta-1} = 1]| \geq \kappa^{-c}$$

for $\kappa \in \mathcal{N}'$. A hybrid argument implies that there exists a fixed $0 \leq l < \delta - 1$ such that

$$|\Pr[A_{\mathrm{ind},l-1} = 1] - \Pr[A_{\mathrm{ind},l} = 1]| \geq \kappa^{-c}/\delta \ .$$

Consider the adversary A_{ind} for the indistinguishability experiment (Experiment 3.1) run with $\mathcal{GM}$ defined as follows. It chooses $\beta^{(b)}_\delta$ randomly from $\mathcal{L}(T)$. Let $\beta_0, \ldots, \beta_\delta$ be the corresponding path, i.e. $\rho = \beta_0$ and $\beta_\delta = \beta^{(b)}_\delta$. Then it simulates $A_{\mathrm{ind},l-1}$ except that the keys $(\mathrm{pk}_{\beta_l}, \mathrm{sk}_{\beta_l})$ are not generated. Instead it defines pk_{β_l} to be the public key it received from Experiment 3.1. Then it hands (β_l, β_l) to its experiment. The experiment returns a sample C'_l.

If A requests the secret key sk_{β_l}, the simulation can not be continued and A_{ind} outputs 0. Similarly, if A outputs $(\alpha^{(0)}, \alpha^{(1)})$, where $\alpha^{(b)} \neq \beta^{(b)}$, then A_{ind} outputs 0. Since $\beta^{(b)}$ is randomly chosen, we have $\Pr[\alpha^{(b)} = \beta^{(b)}] = 1/|\mathcal{L}(T)|$.

If neither of the two events above occur, A_{ind} continues the simulation until the list of elements $(C'_0, \ldots, C'_{l-1}, C_l, \ldots, C_{\delta-1})$ has been computed. It interrupts the simulation and replaces C_l by the challenge C'_l it received from Experiment 3.1. Then it continues the simulation. We have that

$$\begin{aligned}
&|\Pr[\mathbf{Exp}^{\mathsf{ind}-b}_{\mathcal{GM},A_{\mathrm{ind}}}(\kappa) = 1] - \Pr[\mathbf{Exp}^{\mathsf{ind}-\mathcal{D}_{\mathrm{ind}}}_{\mathcal{GM},A_{\mathrm{ind}}}(\kappa) = 1]| \\
&\geq |\Pr[A_{\mathrm{ind},l-1} = 1 \wedge \alpha^{(b)} = \beta^{(b)}] - \Pr[A_{\mathrm{ind},l} = 1 \wedge \alpha^{(b)} = \beta^{(b)}]| \\
&= (1/|\mathcal{L}(T)|)|\Pr[A_{\mathrm{ind},l-1} = 1] - \Pr[A_{\mathrm{ind},l} = 1]| \geq 1/(|\mathcal{L}(T)|\delta\kappa^c) \ .
\end{aligned}$$

The inequality follows by construction, the equality follows by independence of the events $A_{\mathrm{ind},l} = 1$ and $\alpha^{(b)} = \beta^{(b)}$. In view of Theorem 4.3, Lemma 4.4, and Lemma 3.7 this contradicts either the indistinguishability or the anonymity of $\mathcal{GM}$. Thus, the claim is true. □

Claim 5. The hierarchical anonymity of $\mathcal{GS}$ is broken.

Proof. From Claim 2, Claim 3, and Claim 4 follows that

$$|\Pr[A^b_{\mathrm{c,o,p}}(\kappa) = 1] - \Pr[A^b(\kappa) = 1]| < \epsilon_1(\kappa) + \epsilon_2(\kappa) + \epsilon_3(\kappa) \ ,$$

which gives

$$
\begin{aligned}
&|\Pr[\mathbf{Exp}^{\mathsf{anon}-0}_{\mathcal{GS},A'}(\kappa) = 1] - \Pr[\mathbf{Exp}^{\mathsf{anon}-1}_{\mathcal{GS},A'}(\kappa) = 1]| \\
&\geq |\Pr[\mathbf{Exp}^{\mathsf{anon}-0}_{\mathcal{HGS},A}(\kappa) = 1] - \Pr[\mathbf{Exp}^{\mathsf{anon}-1}_{\mathcal{HGS},A}(\kappa) = 1]| - \\
&\quad 2(\epsilon_1(\kappa) + \epsilon_2(\kappa) + \epsilon_3(\kappa)) \ .
\end{aligned}
$$

The assumption about A implies that the hierarchical anonymity is broken. □

Proof of Hierarchical Traceability. Suppose to the contrary that A breaks the hierarchical traceability of $\mathcal{HGS}$. Then $\mathbf{Adv}^{\mathsf{trace}}_{\mathcal{HGS},A}(\kappa, T) \geq 1/\kappa^c$ for some polynomial size tree T, constant $c > 0$ and κ in an infinite index set $\mathcal{N}$. We construct a machine A' that runs A as a blackbox and breaks the hierarchical traceability (i.e. full traceability [7]) of $\mathcal{GS}$.

Definition of A'. The adversary A' simulates the hierarchical traceability experiment, Experiment 2.3, with $\mathcal{HGS}$ to A. It also plays the role of the adversary in Experiment 2.3 with $\mathcal{GS}$.

The key generation is simulated as follows. First the NIZK simulator is invoked to compute a reference string with a trapdoor $(\xi, s_{\mathrm{simstate}})$. The string ξ is used instead of a random string. Recall that $T_{\mathcal{GS}}$ denotes the tree having ρ (the root of T) as root, and children $\mathcal{L}(T)$. A' first waits until it receives the keys $(\mathrm{gpk}(\rho), \mathrm{gsk}(\rho))$ of the root, and the public keys of all signers $(\mathrm{gpk}(\alpha))_{\alpha \in \mathcal{L}(T)}$ from its experiment. Then it simulates the key generation honestly except that it uses the received values, and it does not define $\mathrm{gsk}(\alpha)$ for any $\alpha \in \mathcal{L}(T)$ at all. Thus, the keys of all intermediate group managers are generated by A'.

In each iteration in the experiment simulated to A, it may request $\mathrm{gsk}(\alpha)$ for some signer S_α. When this happens A' requests $\mathrm{gsk}(\alpha)$ from its experiment, and hands $(\mathrm{sk}_\alpha, \mathrm{gsk}(\alpha))$ to A.

When A queries its $\mathsf{HSig}(\cdot, T, \mathrm{hpk}, \mathrm{hsk}(\cdot))$ oracle on (m, α) the reply is computed as follows. First A' queries its $\mathsf{GSig}(\cdot, T_{\mathcal{GS}}, \mathrm{gpk}, \mathrm{gsk}(\cdot))$ oracle on (m, α). The answer is a $\mathcal{GS}$ signature σ. Then A' computes $C = (C_0, \ldots, C_{\delta-1})$ as defined by HSig. Finally, it invokes the NIZK simulator S on input $((T, \mathrm{hpk}, m, \sigma, C), \xi, s_{\mathrm{simstate}})$ to get a simulated proof π, and hands (σ, C, π) to A.

At some point A outputs a message signature pair $(m, (\sigma, C, \pi))$. Then A' outputs (m, σ). This concludes the definition of A'.

Analysis of A'. We divide our analysis into several claims. Denote by $A_{\pi,\mathrm{p}}$ the machine that simulates Experiment 2.3 with $\mathcal{HGS}$ to A and outputs 1 if the experiment outputs 1 and the output $(m, (\sigma, C, \pi))$ of A satisfies $(T, \mathrm{hpk}, m, \sigma, C) \in L_{\mathrm{HGS}}$.

Denote by A_π the machine that is identical to $A_{\pi,\mathrm{p}}$ except that it simulates the answers from the $\mathsf{HSig}(\cdot, T, \mathrm{hpk}, \mathrm{hsk}(\cdot))$ oracle to A precisely as A' does.

Claim 6. There is a negligible function $\epsilon_1(\kappa)$ in κ such that

$$\Pr[\mathbf{Exp}^{\mathsf{trace}}_{\mathcal{HGS},A}(\kappa) = 1] \leq \Pr[A_{\pi,\mathrm{p}}(\kappa) = 1] + \epsilon_1(\kappa) \ .$$

Proof. The claim follows from the soundness of the NIZK. Denote by $E_{\pi,p}$ the event that the output $(m, (\sigma, C, \pi))$ of A in the experiment satisfies (T, hpk, m, σ, C) $\in L_{\mathrm{HGS}}$. From the soundness of the NIZK follows that $\Pr[\overline{E_{\pi,\mathrm{p}}}]$ is negligible. By definition we have that $\Pr[A_{\pi,\mathrm{p}}(\kappa) = 1] = \Pr[\mathbf{Exp}^{\mathsf{trace}}_{\mathcal{HGS},A}(\kappa) = 1 \wedge E_{\pi,\mathrm{p}}]$. The claim follows. □

Claim 7. There is a negligible function $\epsilon_2(\kappa)$ in κ such that

$$|\Pr[A_\pi(\kappa) = 1] - \Pr[A_{\pi,\mathrm{p}}(\kappa) = 1]| \quad < \quad \epsilon_2(\kappa) \ .$$

Proof. The claim follows from the adaptive zero-knowledge of the NIZK. We construct an adversary A_{adzk} against the adaptive zero-knowledge, Experiment 4.6, as follows.

It simulates A_π except for the following two modifications. Firstly, it uses the random string ξ from the experiment instead of generating its own. Secondly, instead of invoking the simulator S on input $((T, \mathrm{hpk}, m, \sigma, C),\ \xi,\ s_{\mathrm{simstate}})$ to produce a NIZK π it requests a NIZK of (T, hpk, m, σ, C) from its experiment. To do this it must also hand a witness to the experiment, but this is not a problem since it has generated all keys.

It follows that

$$\begin{aligned}&|\Pr[A_{\pi,\mathrm{p}}(\kappa) = 1] - \Pr[A_\pi(\kappa) = 1]|\\ &= |\Pr[\mathbf{Exp}^{\mathsf{adzk}-0}_{(P,V,S),A_{\mathrm{adzk}}}(\kappa) = 1] - \Pr[\mathbf{Exp}^{\mathsf{adzk}-1}_{(P,V,S),A_{\mathrm{adzk}}}(\kappa) = 1]| < \epsilon_2(\kappa) \ ,\end{aligned}$$

for some negligible function $\epsilon_2(\kappa)$. □

Claim 8. $\Pr[A_\pi(\kappa) = 1] \leq \Pr[\mathbf{Exp}^{\mathsf{trace}}_{\mathcal{GS},A'}(\kappa) = 1]$.

Proof. All inputs to A in the simulation of A_π are identically distributed to the corresponding inputs in Experiment 2.3. The only difference in how the output of A_π and the experiment are defined is that A_π outputs 1 if the output $(m, (\sigma, C, \pi))$ of A satisfies $(T, \mathrm{hpk}, m, \sigma, C) \in L_{\mathrm{HGS}}$, and the experiment outputs 1 if $\mathsf{GVf}(\emptyset, \mathrm{gpk}, m, \sigma) = 1$, but the former requirement implies the latter. Thus, the claim follows. □

Claim 9. The hierarchical traceability of $\mathcal{GS}$ is broken.

Proof. From Claim 6 and Claim 7 Claim 8 follows that

$$\begin{aligned}&\Pr[\mathbf{Exp}^{\mathsf{trace}}_{\mathcal{HGS},A}(\kappa) = 1] \leq \Pr[A_{\pi,\mathrm{p}}(\kappa) = 1] + \epsilon_1(\kappa) \leq\\ &Pr[A_\pi(\kappa) = 1] + \epsilon_1(\kappa) + \epsilon_2(\kappa) \leq \Pr[\mathbf{Exp}^{\mathsf{trace}}_{\mathcal{GS},A'}(\kappa) = 1] + \epsilon_1(\kappa) + \epsilon_2(\kappa) \ .\end{aligned}$$

The claim now follows from the assumption that $\mathcal{HGS}$ is broken by A. □

Conclusion of Proof. If $\mathcal{HGS}$ is not hierarchically anonymous, then by Claim 5 neither is $\mathcal{GS}$. If $\mathcal{HGS}$ is not hierarchically traceable, then by Claim 9 neither is $\mathcal{GS}$. This concludes the proof. □

5 A Construction under the DDH Assumption and the Strong RSA Assumption

In this section we show how to construct hierarchical group signatures under the DDH assumption and the strong RSA assumption. In contrast to the previous section our focus here is on practicality. We give an explicit construction where the details of all subprotocols are completely specified. Then we prove the security of our construction in the random oracle model. By now it is known that the random oracle hypothesis is not literally true [19]. However, all known counter-examples are contrived, and for practical protocols a security proof in the random oracle model is often considered to be enough.

5.1 Review of Some Notions and Primitives

Before we give our construction we need to review some notions and primitives on which our construction is based. Readers familiar with these primitives can safely browse this section, but should observe that our notation differs slightly from the standard at some points because of name collisions.

5.1.1 Cunningham Chains

Let q_0 and q_1 be primes such that $q_0 = 2q_1 + 1$. Then there is a unique subgroup $G_{q_1} \subset \mathbb{Z}_{q_0}^*$ of order q_1. Let g_1 be a generator of G_{q_1}. The discrete logarithm of an element $z \in G_{q_1}$ in the basis g_1 is defined as the (unique) $r \in \mathbb{Z}_{q_1}$ such that $z = g_1^r$. This is usually written $\log_{g_1} z = r$.

The primes q_0 and q_1 above are clearly of a special form. In fact q_1 is called a Sophie Germain prime. One can demand that q_1 is of the same form as q_0 and so on. This leads to the following definition.

Definition 5.1 (Cunningham Chain). *A sequence $q_0, \ldots, q_{k-1}$ of primes is called a* Cunningham Chain[1] *of length k if $q_i = 2q_{i+1} + 1$ for $i = 0, \ldots, k-2$.*

Throughout this paper $q_0, \ldots, q_3$ are chosen to form a Cunningham Chain of length 4 and we denote by G_{q_i} the unique subgroup of order q_i of $\mathbb{Z}_{q_{i-1}}^*$. We also pick a generator g_i of G_{q_i}. Thus, the groups are set up such that $\mathbb{Z}_{q_l}^* \approx G_{q_{l+1}} \times \{-1, 1\}$, $\langle g_{l+1} \rangle = G_{q_{l+1}} \approx \mathbb{Z}_{q_{l+1}}$.

Before we start using Cunningham chains for cryptography we are obliged to ask if they exist at all. Unfortunately, there exists no proof that there are infinitely many Cunningham chains of any length, not even of length 2 which correspond to the Sophie Germain primes. However, one can apply a heuristic argument and assume that a randomly chosen integer n is prime with “probability” $1/\log n$. If we also assume that the event that $(n-1)/2$ is prime is independent of the event that n is prime for every prime we should find a Cunningham chain of length k with probability close to $1/\log^k n$. We stress that one must be careful with this type of heuristic argument since there exist counter examples [37], but the argument seems reasonable in our setting and agrees with computational experiments. In practice

[1] This is a chain of the second kind. A chain of the first kind satisfies $q_i = 2q_{i+1} - 1$.

it is not hard to find Cunningham chains of length 4 for primes of the size used in current cryptography (cf. [42], [43]). Young and Yung [49] have also published some heuristic tricks for finding length-3 Cunningham chains. We let $\mathsf{CunnGen}_k$ denote the algorithm that on input 1^κ outputs a κ-bit Cunningham chain of length k. Note that the existence of 2-Cunningham chains is implicitly assumed in many papers, e.g. [25].

Although we describe our scheme for Cunningham chains because they are well-known, the scheme also works for a sequence of primes q_0, q_1, q_2, q_3 such that $q_i = a_i q_{i+1} + 1$ for positive numbers a_i. Chains of this type have the advantage that they are easier to generate. The existence of such chains follows from the following formal assumption:

Assumption 5.2. *There exists a constant c and a probabilistic polynomial Turing machine that given a κ-bit number n as input with overwhelming probability outputs a $\log^c \kappa$-bit number a such that $an+1$ is prime.*

We generalize the notation G_n to denote the cyclic subgroup of order n of $\mathbb{Z}^*_{an+1}$, where a is the smallest number such that $an+1$ is prime.

5.1.2 Decision Diffie-Hellman Problem

The Decision Diffie-Hellman problem in a group G_n is defined as the problem of distinguishing the distributions $(g^\alpha, g^\beta, g^\gamma)$ and $(g^\alpha, g^\beta, g^{\alpha\beta})$, where $\alpha, \beta, \gamma \in \mathbb{Z}_n$ are randomly distributed.

Experiment 5.3 (Decision Diffie-Hellman, $\mathbf{Exp}^{\mathsf{ddh}-b}_{G_n,A}(\kappa)$)**.**

$$g \leftarrow G_n, \quad \alpha, \beta, \gamma \leftarrow \mathbb{Z}_n, \quad (D_1, D_2, D_3) \leftarrow (g^\alpha, g^\beta, g^{(b\gamma + (1-b)\alpha\beta)})$$
$$d \leftarrow A(g, D_1, D_2, D_3)$$

The advantage of the adversary is $\mathbf{Adv}^{\mathsf{ddh}}_{G_n,A}(\kappa) = |\Pr[\mathbf{Exp}^{\mathsf{ddh}-0}_{G_n,A}(\kappa) = 1] - \Pr[\mathbf{Exp}^{\mathsf{ddh}-1}_{G_n,A}(\kappa) = 1]|$.

Assumption 5.4 (Decision Diffie-Hellman Assumption over the group G_n)**.** *For all $A \in \mathrm{PPT}^*$ the advantage $A = \{A_\kappa\}$, $\mathbf{Adv}^{\mathsf{ddh}}_{G_n,A}(\kappa)$ is negligible.*

5.1.3 The ElGamal cryptosystem

We review the ElGamal [26] cryptosystem employed in G_q. We write $\mathsf{ElgKg}(G_q, g)$ for the algorithm that generates a random private key $x \in \mathbb{Z}_q$, computes a public key (g, y), where $y = g^x$, and outputs $((g, y), x)$. Encryption of a message $m \in G_q$ using the public key (g, y) is given by $E_{(g,y)}(m, r) = (g^r, y^r m)$ for $r \in_R \mathbb{Z}_q$, and decryption of a cryptotext on the form $(u, v) = (g^r, y^r m)$ using the private key x is given by $D_x(u, v) = u^{-x} v = m$.

5.1.4 The RSA cryptosystem

The key generation for the RSA cryptosystem [45] consists of generating primes p and q of the same size and computing $N = pq$. The parameters e and d are chosen so that $\gcd(e, d) = 1 \pmod{\phi(n)}$, where $\phi(n) = (p-1)(q-1)$. The public key is (e, n) and the private key is d. If we choose p and q so that $p = 2p' + 1$ and $q = 2q' + 1$ where p', q' are primes, we ensure that the order of the group QR_N, the quadratic residues modulo N, is $p'q'$. Throughout the paper we write members of QR_N using bold font (e.g., $\mathbf{g}, \mathbf{y}$).

5.1.5 The Strong RSA Assumption

The *Strong RSA Assumption* y the assumption says that it is hard to compute any non-trivial root in $\mathbb{Z}_N$ where N is an RSA modulus, even if allowed to select which root to compute. This differs from the standard RSA assumption, where the root to compute is predetermined. Like the standard RSA assumption the strong RSA assumption implies that factoring is hard.

Assumption 5.5 (Strong RSA Assumption)**.** *For any adversary $A \in \mathrm{PPT}^*$ the following holds: $\forall c > 0$, $\exists \kappa_0$, such that for $\kappa > \kappa_0$ we have:*

$$\Pr[(P, Q) \leftarrow \mathsf{csRSA}(1^\kappa), \boldsymbol{\sigma} \leftarrow \mathbb{Z}^*_{PQ}, (\mathbf{m}, e) \leftarrow A(PQ, \boldsymbol{\sigma}), \mathbf{m}^e = \boldsymbol{\sigma}, e > 1] < \frac{1}{\kappa^c} \ .$$

5.1.6 The Chaum van Heijst Pfitzmann Hash Function

We write $\mathsf{CHPg}(G_q, \delta)$ for the algorithm that takes as input the representation of a group G_q and $\delta \in \mathbb{N}$ and outputs a list $(h_1, \ldots, h_\delta) \in G_q^\delta$ of randomly chosen elements. We employ the Chaum van Heijst Pfitzmann hash function [20] defined as $H^{\mathsf{CHP}} : \mathbb{Z}_q^\delta \to G_q$, $H^{\mathsf{CHP}} : (z_1, \ldots, z_\delta) \mapsto \prod_{l=1}^{\delta} h_l^{z_l}$. They prove that this map is one-way and collision-free if the discrete logarithm problem is hard in G_q. We abuse notation and use H^{CHP} to denote both the map and its representation $(h_1, \ldots, h_\delta)$.

Lemma 5.6 (cf. [20])**.** *The hash function H^{CHP} is one-way and collision-free if the discrete logarithm problem is hard in G_q.*

5.1.7 The Shamir Hash Function

We write $H^{\mathsf{Sh}}_{(N,\mathbf{g})}(x)$ for the algorithm that computes $\mathbf{g}^x \bmod N$, where N is a composite number, $\mathbf{g} \in \mathrm{QR}_N$ and $x \in \mathbb{Z}$. The idea to use this as a hash function was proposed by Shamir. When $(N, \mathbf{g})$ are clear from the context we may leave them out. We let H^{Sh} denote both the map and the pair $(N, \mathbf{g})$. We have the following security result for H^{Sh}:

Lemma 5.7. *The hash function H^{Sh} is collision-free if the factoring problem is hard.*

Proof. Assume that H^{Sh} is not collision-free, and that $H^{\mathsf{Sh}}_{(N,\mathbf{g})}(x_1) = H^{\mathsf{Sh}}_{(N,\mathbf{g})}(x_2)$ where $x_1 \neq x_2$. Then $\mathbf{g}^{x_1 - x_2} = 1$, meaning that $x_1 - x_2$ is a multiple of the order of QR_N. This information is enough to factor N as shown in [40]. $\square$

5.1.8 The Cramer-Shoup Cryptosystem

We review the Cramer-Shoup cryptosystem [24] over G_q employed with a collision-free one-way function H. We denote their cryptosystem by $\mathcal{CS}^{\mathsf{cs}}_H = (\mathsf{CSKg}^{\mathsf{cs}}, E^{\mathsf{cs}}, D^{\mathsf{cs}})$.

The key generation algorithm $\mathsf{CSKg}^{\mathsf{cs}}(G_q, q)$ generates random $\bar{g}_1, \bar{g}_2 \in G_q$ and $\bar{x}_1, \bar{x}_2, \bar{y}_1, \bar{y}_2, \bar{z} \in \mathbb{Z}_q$, computes $\bar{c} = \bar{g}_1^{\bar{x}_1}\bar{g}_2^{\bar{x}_2}$, $\bar{d} = \bar{g}_1^{\bar{y}_1}\bar{g}_2^{\bar{y}_2}$, and $\bar{h} = \bar{g}_1^{\bar{z}}$ and outputs $(\bar{g}_1, \bar{g}_2, \bar{c}, \bar{d}, \bar{h}, \bar{x}_1, \bar{x}_2, \bar{y}_1, \bar{y}_2, \bar{z})$. Encryption of a message $m \in G_q$ using the public key $Y = (\bar{g}_1, \bar{g}_2, \bar{c}, \bar{d}, \bar{h})$ is given by

$$E^{\mathsf{cs}}_Y(m, r) = (u, \mu, v, \nu) = (\bar{g}_1^r, \bar{g}_2^r, \bar{h}^r m, \bar{c}^r \bar{d}^{rH(u,\mu,v)})$$

for a randomly chosen $r \in \mathbb{Z}_q$. Note that (u, v) is an ElGamal encryption of the message m, so decryption of a cryptotext (u, μ, v, ν) using the private key $X = (\bar{x}_1, \bar{x}_2, \bar{y}_1, \bar{y}_2, \bar{z})$ is given by $D^{\mathsf{cs}}_X(u, \mu, v, \nu) = D_{\bar{z}}(u, v) = m$ for valid cryptotexts. A cryptotext is considered valid if the predicate

$$T^{\mathsf{cs}}_X(u, \mu, v, \nu) = (u^{\bar{x}_1+\bar{x}_2 H(u,\mu,v)}\mu^{\bar{y}_1+\bar{y}_2 H(u,\mu,v)} = \nu)$$

is satisfied. We let $T^{\mathsf{cs}}_X(u, \mu, v, \nu) = 1$ if it is satisfied and 0 otherwise. An invalid cryptotext decrypts to $\perp$.

Cramer and Shoup [24] prove that their cryptosystem is CCA2-secure under standard assumptions.

Lemma 5.8 (cf. [24])**.** *The cryptosystem $\mathcal{CS}^{\mathsf{cs}}_H$ is CCA2-secure under the DDH assumption in G_q if H is collision-free.*

5.1.9 Cramer-Shoup RSA Signatures

We review the signature scheme by Cramer and Shoup [25]. We denote their construction by $\mathcal{SS}^{\mathsf{cs}}_{H_1,H_2} = (\mathsf{SSKg}^{\mathsf{cs}}, \mathsf{Sig}^{\mathsf{cs}}, \mathsf{Vf}^{\mathsf{cs}})$, and review the algorithms it consists of below. Here H_1 and H_2 are hash functions.

We write csRSA for the algorithm that on input 1^κ generates two random $\kappa/2$-bit primes $P = 2P' + 1$ and $Q = 2Q' + 1$, where P' and Q' are also prime, and returns (P, Q). Thus, csRSA generates a κ-bit RSA modulus $N = PQ$.

Algorithm 5.9 (Key Generation, $\mathsf{SSKg}^{\mathsf{cs}}(1^\kappa)$)**.**

1. Run $(P, Q) = \mathsf{csRSA}(1^\kappa)$ and set $P' = (P - 1)/2$, $Q' = (Q - 1)/2$ and $N = PQ$.
2. Choose $\mathbf{h}, \mathbf{z} \in \mathrm{QR}_N$ randomly.
3. Choose a random $(\kappa + 1)$-bit prime e' such that $e' \equiv 1 \pmod 4$.
4. Output $(N, e', \mathbf{h}, \mathbf{z}, P'Q')$.

Algorithm 5.10 (Signing, $\mathsf{Sig}^{\mathsf{cs}}(m, H_1, H_2, N, \mathbf{h}, \mathbf{z}, e', P'Q')$)**.**

1. Choose a random $(\kappa + 1)$-bit prime e such that $e \equiv 3 \bmod 4$ and a random $\boldsymbol{\sigma}' \in \mathrm{QR}_N$.

2. Compute $\mathbf{z}' = (\boldsymbol{\sigma}')^{e'} \mathbf{h}^{-H_1(m)}$, $\boldsymbol{\sigma} = \left(\mathbf{z}\mathbf{h}^{H_2(\mathbf{z}')}\right)^{1/e}$ and output $(e, \boldsymbol{\sigma}, \boldsymbol{\sigma}')$.

Algorithm 5.11 (Verification, $\mathsf{Vf}^{\mathsf{cs}}(H_1, H_2, N, \mathbf{h}, \mathbf{z}, e', m, (e, \boldsymbol{\sigma}, \boldsymbol{\sigma}'))$.)**.**

1. Verify that e is an odd number of length between $(\kappa + 1)$ bits and $\left(\frac{3}{2}\kappa - 4\right)$ bits that is distinct from e'.

2. Compute $\mathbf{z}' = (\boldsymbol{\sigma}')^{e'} \mathbf{h}^{-H_1(m)}$ and verify that $\mathbf{z} = \boldsymbol{\sigma}^e \mathbf{h}^{-H_2(\mathbf{z}')}$. If so output 1, otherwise output 0.

We have modified the scheme slightly by making e' always equal to 1 modulo 4 and the primes generated at signing, e, equal to 3 modulo 4. This makes it easier to prove later in zero-knowledge that $e \neq e'$. In the original scheme H_1 and H_2 are equal, but in our setting they will be different. This does not affect the security proof in any way.

Also in our description the exponent e is longer than the modulus, but in the original description e is shorter than P' and Q'. Below we argue why the security proof still holds.

Lemma 5.12. *The signature scheme $\mathcal{SS}^{\mathsf{cs}}_{H_1,H_2}$ is CMA-secure under the strong RSA assumption if H_1 and H_2 are collision-free one-way functions.*

Proof. We assume familiarity with [25]. When the length of the exponent is between $\kappa + 1$ bits and $\frac{3}{2}\kappa - 4$ bits, the proof of [25] holds except for how a Type III forger is used to break the strong RSA assumption, where a Type III forger is defined as a forger that outputs a signature (using our notation) $(e, \boldsymbol{\sigma}, \boldsymbol{\sigma}')$ such that $e \neq e_i$ for all signatures e_i it has seen previously. The output from the Type III forger is used to form the equation $\boldsymbol{\sigma}^{e'} = \mathbf{z}^t$ where $\mathbf{z}$ is the number we wish to compute a root of and t is known. Then the fact that $\gcd(e, t, 2P'Q') = 1$ is used, which in the original setting holds because e is odd and smaller than P' and Q'. In our setting it may or may not hold. If the forger outputs an e such that it does hold, then the original proof goes throgh. If it does not hold, then $\gcd(e, 2P'Q')$ is either P', Q', or $P'Q'$ since e is odd. In all of these cases we have enough information to factor N. □

5.1.10 Proofs of Knowledge

We use a relatively complex proof of knowledge in our construction, but we postpone a careful description of the properties we need for Section 6. We define some notation used in the description of our construction.

Given a three-move public coin interactive proof (P, V) for a language L, we can use the Fiat-Shamir heuristic to construct a non-interactive variant in the random oracle model, by simply replacing the message sent by the verifier by the output of the random oracle. We write $\pi = P^{\mathsf{O}(m,\cdot)}(x, w)$ to denote the transcript of such a protocol execution, where $x \in L$ is the common input, w is a witness for x, and P interacts with the random oracle $\mathsf{O}(m, \cdot)$. We write $V^{\mathsf{O}(m,\cdot)}(x, \pi)$ for the verification.

5.2 Our Construction

We are now ready to describe our construction. The basic idea is similar to the construction under general assumptions. A signer encrypts a path from the root to its leaf using the public keys along the path. Then it proves that it knows a $\mathcal{SS}^{\mathsf{cs}}_{H^{\mathsf{CHP}},H^{\mathsf{Sh}}}$ signature on the list of public keys it used. Thus, a private key of a signer is simply a $\mathcal{SS}^{\mathsf{cs}}_{H^{\mathsf{CHP}},H^{\mathsf{Sh}}}$ signature on the public keys along its path. We denote our construction by $\mathcal{HGS} = (\mathsf{HKg}, \mathsf{HSig}, \mathsf{HVf}, \mathsf{HOpen})$, and define algorithms HKg, HSig, HVf, and HOpen below.

5.2.1 Key Generation

The key generation phase proceeds as follows. Each group manager is given an ElGamal key pair, and each signer is given an RSA signature of the public keys of the group managers on the path from the root to the signer.

Algorithm 5.13 (Key Generation, $\mathsf{HKg}(1^\kappa, T)$)**.**

1. Run $(q_0, \ldots, q_3) = \mathsf{CunnGen}_4(1^\kappa)$ to generate a Cunningham chain, and choose $g_i \in G_{q_i}$ randomly for $i = 1, 2, 3$.

2. Let δ be the depth of the tree T, and run
$$H^{\mathsf{CHP}} = (h_1, \ldots, h_\delta) = \mathsf{CHPg}(G_{q_2}, \delta)$$
to generate a collision-free one-way function.

3. Run $(X, Y) = \mathsf{CSKg}^{\mathsf{cs}}(G_{q_3}, q_3)$ to generate keys for a Cramer-Shoup cryptosystem over G_{q_3}.

4. Run $(N, e', \mathbf{h}, \mathbf{z}, t) = \mathsf{SSKg}^{\mathsf{cs}}(1^\kappa)$ and randomly choose $\mathbf{g} \in \mathrm{QR}_N$ to generate keys for a Cramer-Shoup RSA signature scheme employed with the hash functions H^{CHP} and $H^{\mathsf{Sh}}_{(N,\mathbf{g})}$.

5. Choose a prime on the form $aN + 1$ and let G_N be the unique subgroup of order N of $\mathbb{Z}^*_{aN+1}$. Choose g_N, y_N randomly in G_N.

6. For each node β in T, generate keys
$$(\mathrm{hpk}(\beta), \mathrm{hsk}(\beta)) = (y_\beta, x_\beta) = \mathsf{ElgKg}(G_{q_3}, g_3)$$
for an ElGamal cryptosystem over G_{q_3}.

7. For each leaf α let $\alpha_0, \ldots, \alpha_\delta$ with $\alpha_0 = \rho$ and $\alpha_\delta = \alpha$ be the path from the root to α, compute
$$(e_\alpha, \boldsymbol{\sigma}_\alpha, \boldsymbol{\sigma}'_\alpha) = \mathsf{Sig}^{\mathsf{cs}}((y_{\alpha_1}, \ldots, y_{\alpha_\delta}), H^{\mathsf{CHP}}, H^{\mathsf{Sh}}, N, \mathbf{h}, \mathbf{z}, e', t)$$
and redefine $\mathrm{hsk}(\alpha) = (e_\alpha, \boldsymbol{\sigma}_\alpha, \boldsymbol{\sigma}'_\alpha)$.

8. Let ρ be the root of T. Choose $\mathbf{y} \in \mathrm{QR}_N$ randomly. Choose $x_i \in \mathbb{Z}_{q_i}$ randomly and compute $y_i = g_i^{x_i}$ for $i = 1, 2, 3$. Set the three security parameters $\kappa_1, \kappa_2, \kappa_3 = \Theta(\kappa)$. Redefine the public key $\mathrm{hpk}(\rho)$ of the root ρ to be $(\mathrm{hpk}(\rho), q_0, H^{\mathsf{CHP}}, N, e, Y, g_1, y_1, g_2, y_2, g_3, y_3, \mathbf{g}, \mathbf{y}, g_N, y_N, \kappa_1, \kappa_2, \kappa_3)$ and output hpk, hsk.

Remark 5.14. The three security parameters κ_1, κ_2 and κ_3 are used in the proof of knowledge. For details, see section 6.

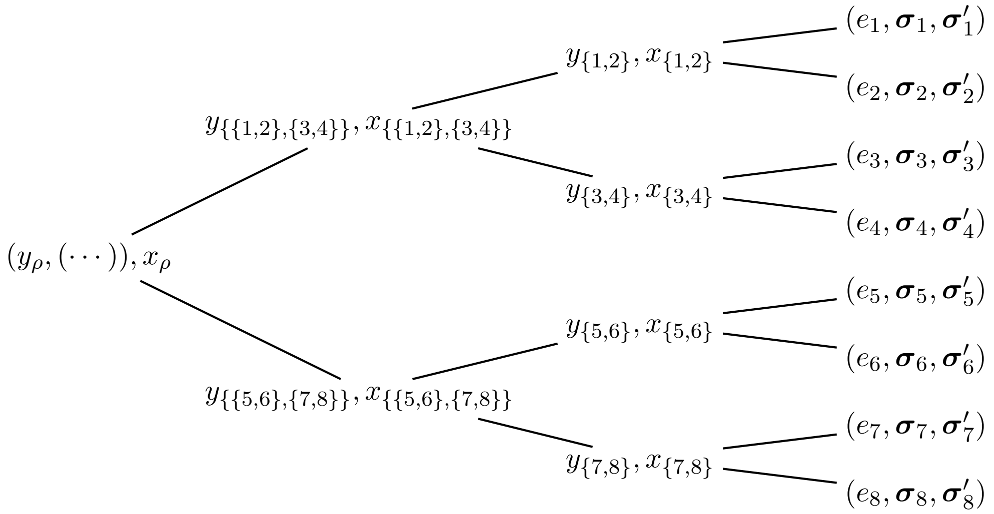

Figure 4: The output of HKg for a four-level tree. The common group parameters (key size, generators etc.) have been abbreviated by $(\cdots)$.

5.2.2 Computing, Verifying and Opening a Signature

Any leaf α can be associated with a path $\alpha_0, \ldots, \alpha_\delta$ where $\rho = \alpha_0$ and $\alpha_\delta = \alpha$ from the root to the leaf in the natural way. The signer encrypts its path using the public keys of the group managers along this path, i.e., the signer computes a list $(E_{y_{\alpha_0}}(y_{\alpha_1}), \ldots, E_{y_{\alpha_{\delta-1}}}(y_{\alpha_\delta}))$. Note the particular way in which the public keys are chained. It also commits to the $\mathcal{SS}^{\mathsf{cs}}_{H^{\mathsf{CHP}}, H^{\mathsf{Sh}}}$ signature of the message $(y_{\alpha_1}, \ldots, y_{\alpha_\delta})$ it was given by the key generator. Then it computes a Schnorr-like "proof of knowledge" in the random oracle model that the cryptotexts and commitments are indeed formed as described. The message to be signed is included in the input to the hash function (the random oracle) similarly to Schnorr signatures. Thus, the signer actually proves that it possesses an $\mathcal{SS}^{\mathsf{cs}}_{H^{\mathsf{CHP}}, H^{\mathsf{Sh}}}$ signature of the list of public keys it uses to encrypt the public keys along its path. Hierarchical anonymity follows since only the holders of the secret keys corresponding to y_{α_l} can open any part of the signature. Hierarchical traceability follows since the signer cannot forge a $\mathcal{SS}^{\mathsf{cs}}_{H^{\mathsf{CHP}}, H^{\mathsf{Sh}}}$ signature corresponding to a path different from its own.

Algorithm 5.15 (Signing, $\mathsf{HSig}(m, T, \mathrm{hpk}, \mathrm{hsk}(\alpha))$)**.**
Let $\alpha_0, \ldots, \alpha_\delta$ with $\rho = \alpha_0$ and $\alpha_\delta = \alpha$ be the path to the signer S_α.

1. Choose $r_0, \ldots, r_\delta \in \mathbb{Z}_{q_3}$ randomly and compute $(u_l, v_l) = E_{(y_{\alpha_l}, g_3)}(y_{\alpha_{l+1}}, r_l)$, for $l = 0, \ldots, \delta - 1$, and $C_\delta = E^{\mathsf{cs}}_Y(y_{\alpha_\delta}, r_\delta)$. This is the list of cryptotexts.

2. Choose $r, s, r', s', t \in [0, 2^{\kappa_3} N - 1]$ randomly and set $(\mathbf{u}, \mathbf{v}) = (\mathbf{g}^s \mathbf{y}^r, \mathbf{g}^r \boldsymbol{\sigma}_\alpha)$, $(\mathbf{u}', \mathbf{v}') = (\mathbf{g}^{s'} \mathbf{y}^{r'}, \mathbf{g}^{r'} \boldsymbol{\sigma}'_\alpha)$, and $\mathbf{C} = \mathbf{g}^{e_\alpha} \mathbf{y}^t$. This is a commitment of the signature $(e_\alpha, \boldsymbol{\sigma}_\alpha, \boldsymbol{\sigma}'_\alpha)$.

3. Denote by R_{HGS} the binary relation consisting of pairs (X, W), where

$$\begin{aligned} X &= \big((u_l, v_l)_{l=0}^{\delta-1}, C_\delta, (\mathbf{u}, \mathbf{v}), (\mathbf{u}', \mathbf{v}'), \mathbf{C}\big) \in \\ &\qquad (G_{q_3} \times G_{q_3})^\delta \times G_{q_3}^4 \times \mathrm{QR}_N^2 \times \mathrm{QR}_N^2 \times \mathrm{QR}_N \\ W &= \big((\tau_0, \ldots, \tau_{\delta-1}), (\tau, \zeta, \tau', \zeta', \psi, \varepsilon)\big) \in \\ &\qquad \mathbb{Z}_{q_3}^\delta \times [0, 2^{\kappa_3} N - 1]^5 \times [2^\kappa, 2^{\kappa+1} - 1] \end{aligned}$$

   such that

$$\begin{array}{l} \gamma_0 = y_{\alpha_0} \\ \big((u_l, v_l) = E_{(\gamma_l, g)}(\gamma_{l+1}, \tau_l)\big)_{l=0}^{\delta-1} \\ C_\delta = E^{\mathsf{cs}}_Y(\gamma_\delta, \tau_\delta) \end{array} \quad \text{and} \quad \begin{array}{l} \mathbf{u} = \mathbf{g}^\zeta \mathbf{y}^\tau, \quad \mathbf{u}' = \mathbf{g}^{\zeta'} \mathbf{y}^\tau, \\ \mathbf{C} = \mathbf{g}^\varepsilon \mathbf{y}^\psi \\ \mathsf{Vf}^{\mathsf{cs}}(H^{\mathsf{CHP}}, H^{\mathsf{Sh}}, N, \mathbf{h}, \mathbf{z}, e', \\ \qquad (\gamma_1, \ldots, \gamma_\delta), \\ \qquad (\varepsilon, \mathbf{v}/\mathbf{y}^{\boldsymbol{\tau}}, \mathbf{v}'/\mathbf{y}^{\boldsymbol{\tau}'})) = 1 \end{array} .$$

   In Section 6 we construct an honest verifier public coin zero-knowledge proof of knowledge (P, V) for this relation. The signer computes a non-interactive proof $\pi = P^{\mathsf{O}(m,\cdot)}(X, W)$ in the random oracle model.

4. Output the signature $\big((u_l, v_l)_{l=0}^{\delta-1}, C_\delta, (\mathbf{u}, \mathbf{v}), (\mathbf{u}', \mathbf{v}'), \mathbf{C}, \pi\big)$.

Remark 5.16. Note that we switch the order of the components in the ElGamal cryptosystem so that, for example, $D_{-x_\rho}(u_0, v_0) = y_{\alpha_1}$ in order to simplify the construction of the proof of knowledge.

The construction of the proof of knowledge is involved and postponed until Section 6. The verification algorithm consists simply of verifying the proof of knowledge contained in a signature.

Algorithm 5.17 (Verification, $\mathsf{HVf}(T, \mathrm{hpk}, m, \sigma)$)**.**
On input a candidate signature $\sigma = (c, \pi) = \big(\big((u_l, v_l)_{l=0}^{\delta-1},\ C_\delta,\ (\mathbf{u}, \mathbf{v}),\ (\mathbf{u}', \mathbf{v}'),\ \mathbf{C}\big), \pi\big)$ return the result of $V^{\mathsf{O}(m,\cdot)}(c, \pi)$.

To open a signature a group manager on depth l simply decrypts the lth component of the chain of cryptotexts contained in the signature.

Algorithm 5.18 (Open, $\mathsf{HOpen}(T, \mathrm{hpk}, \mathrm{hsk}(\beta), m, \sigma)$)**.**
On input $\sigma = \big((u_l, v_l)_{l=0}^{\delta-1}, C_\delta, (\mathbf{u}, \mathbf{v}), (\mathbf{u}', \mathbf{v}'), \mathbf{C}, \pi\big)$ proceed as follows. If $\mathsf{HVf}(T,$ hpk,m, $\sigma) = \perp$, return $\perp$. Otherwise compute $y_\alpha = D_{-x_\beta}(u_l, v_l)$. If $\alpha \in \beta$ return α and otherwise $\perp$.

In our construction we require that cryptotexts encrypted with *distinct* public keys are indistinguishable, since otherwise the cryptotexts themselves leak information on the identity of the signer. This property, which is called anonymity, is formalized and characterized in Section 3. Note that semantic security does not imply anonymity. It is not hard to see that ElGamal is anonymous if we fix a group for each security parameter such that all cryptotexts for a security parameter are formed over this group.

It may seem that we have picked arbitrary primitives and then deferred the problem of forming the proof of knowledge needed. This is *not* the case. The primitives are carefully selected and slightly modified to allow a reasonably practical proof of knowledge.

5.2.3 Efficiency Analysis

We now analyze the number of exponentiations needed for some typical parameters. Since the bulk of the computations stem from the cut-and-choose protocols, Protocols 6.10 and 6.31 described in Section 6, we focus on these. The important parameters are the key length κ, κ_2 determining the soundness of the protocols, and the depth of the tree δ. Protocol 6.10 is called $\delta + 1$ times and Protocol 6.31 twice. Some exponentiations are done for pairs of generators. Using simultaneous exponentiation techniques the time needed for such an exponentiation is only slightly more than for a single exponentiation [38], so we count these as one exponentiation. In each execution, the prover performs $4\kappa_2$ exponentiations of κ-bit numbers. This gives a total of $4\kappa_2(\delta+3)$ κ-bit exponentiations for both the prover and the verifier.

All the exponentiations are with fixed bases, which makes it possible to use precomputation techniques for speed up the computations [38]. Using these tricks, one κ-bit exponentiation costs about a fifth of a general κ-bit exponentiation. The amount of precomputation is small in comparison, and it can be reused for several signatures. A realistic example might be $\delta = 3$, $\kappa = 1024$ and $\kappa_2 = 100$. In this case about 2400 κ-bit exponentiations are necessary to create a signature, which takes about as much time as 500 general exponentiations. If we add the time for the other protocols, we still end up well below 1000 general exponentiations. Verification is about the same.

We now look at the size of a signature. Also here the cut-and-choose protocols contribute the most. For each protocol execution, $4\kappa_2$ values need to be stored, each of which has κ bits. This gives a total of $4\kappa_2(\delta+3)$ κ-bit numbers. With the same parameters as above this gives a signature size of about 700 kb.

In the above computations we have ignored the fact that some computations involve exponents slightly larger than κ bits, and also all non-cut-and-choose protocols, but the numbers still gives a good picture of the amount of computation necessary. One can conclude that on and ordinary PC a signature can be created in about half a minute.

5.3 Security Analysis

We analyze the security of our construction in the random oracle model, and prove that its security follows from the DDH assumption and the strong RSA assumption.

Theorem 5.19. *The hierarchical signature scheme $\mathcal{HGS}$ is secure under the DDH assumption and the strong RSA assumption in the random oracle model.*

Proof. The proof proceeds by contradiction. Assume that the signature scheme $\mathcal{SS}^{\mathsf{cs}}_{H^{\mathsf{CHP}},H^{\mathsf{Sh}}}$ is secure. Then we show that if there exists an adversary A which breaks $\mathcal{HGS}$, there exists another adversary which executes A as a blackbox and breaks either the Cunningham chain DDH assumption, the security of $\mathcal{CS}^{\mathsf{cs}}_{H^{\mathsf{CHP}}}$ or the security of $\mathcal{SS}^{\mathsf{cs}}_{H^{\mathsf{CHP}},H^{\mathsf{Sh}}}$. In view of Corollary 5.12 and Lemma 5.8 this gives a contradiction.

Breaking $\mathcal{HGS}$ means either breaking the hierarchical anonymity, i.e., succeeding in Experiment 2.2, or hierarchical traceability, i.e., succeeding in Experiment 2.3. The adversary A' simulates to A that it participates in one of these experiments, i.e., it simulates the random oracle O, the HKg oracle, the HOpen oracle, and the HSig oracle. We consider the details of the simulation of each experiment below, starting with the case where A breaks hierarchical anonymity.

Hierarchical Anonymity. Suppose that the attacker A breaks hierarchical anonymity. Then we have $\mathbf{Adv}^{\mathsf{anon}}_{\mathcal{HGS},A}(\kappa, T) \geq 1/\kappa^c$ for some polynomial size tree T, constant $c > 0$ and sufficiently large κ.

Definition of A'. As an intermediate step we define a machine A' that runs A as a blackbox. A' is the basis of the adversaries we construct. A' takes as input a single bit b and outputs a single bit. However, A' itself also plays the role of the adversary in a DDH experiment, a CCA2 experiment, and a CMA experiment, i.e., it plays the adversary in Experiment 5.3 over G_{q_3}, Experiment 1.1 with $\mathcal{CS}^{\mathsf{cs}}_{H^{\mathsf{CHP}}}$, and Experiment 1.3 with $\mathcal{SS}^{\mathsf{cs}}_{H^{\mathsf{CHP}},H^{\mathsf{Sh}}}$. We reach a contradiction by proving that A' is too successful in at least one of these experiments.

We now describe how A' simulates the oracles to A. The $\mathsf{HKg}(\cdot)$ oracle is simulated as follows. A' first waits until it receives (g_3, D_1, D_2, D_3) in the DDH experiment. Step 1 is simulated honestly, except that A' uses the value of g_3 received from its oracle instead of generating it randomly. Step 2 is simulated honestly. Then A' waits until it receives a $\mathcal{CS}^{\mathsf{cs}}_{H^{\mathsf{CHP}}}$ public key Y in the CCA2 experiment. In Step 3 it takes Y to be the public key, and X is never defined. Then A' waits until it receives a $\mathcal{SS}^{\mathsf{cs}}_{H^{\mathsf{CHP}},H^{\mathsf{Sh}}}$ public key $(N, \mathbf{h}, \mathbf{z}, e')$ in the CMA experiment. In Step 4 it uses these values, and t is never defined. Step 5 is simulated honestly. Next A' chooses two leaves $\beta^{(0)}_\delta$ and $\beta^{(1)}_\delta$ randomly. Intuitively, this is the two leaves A' guess that A will later ask to be challenged on. Let $\beta^{(0)}_\delta$, $\ldots$, $\beta^{(0)}_t$ and $\beta^{(1)}_\delta, \ldots, \beta^{(1)}_t$ be the paths to their common ancestor $\beta^{(0)}_t = \beta^{(1)}_t$. Step 6 is then simulated honestly except that for $\beta^{(0)}_l, \beta^{(1)}_l$, for $l = t, \ldots, \delta$, the public keys are instead defined using (D_1, D_2, D_3) as follows

$$y_{\beta^{(0)}_l} = D_1^{x_{\beta^{(0)}_l}}, \quad y_{\beta^{(1)}_l} = D_1^{x_{\beta^{(1)}_l}} .$$

A' simulates Step 7 by requesting the $\mathsf{Sig}^{\mathsf{cs}}(\cdot, (H, N, \mathbf{h}, \mathbf{z}, e'), t)$ oracle in the CMA experiment to sign the message $(y_{\alpha_1}, \ldots, y_{\alpha_\delta})$ for each α. Then A' uses the answer, a $\mathcal{SS}^{\mathsf{cs}}$ signature $(e_\alpha, \boldsymbol{\sigma}_\alpha, \boldsymbol{\sigma}'_\alpha)$, to define $\mathrm{hsk}(\alpha)$ properly. Step 8 is simulated honestly.

In each iteration of Experiment 2.2 A may ask for the private key of any inner node α. If $\alpha = \beta_l^{(0)}$ or $\beta_l^{(1)}$ for some $l = t, \ldots, \delta$, then A' is unable to satisfy the request of A properly, since it does not know $\log_{g_3} y_\alpha$. Instead of continuing the simulation, A' outputs a random bit $d \in \{0, 1\}$ and halts. Otherwise it hands x_α to A.

The $\mathsf{HOpen}(T, \mathrm{hpk}, \mathrm{hsk}(\cdot), \cdot, \cdot)$ oracle is simulated as follows given a query (α, m, σ), where α is on depth l. If $\mathsf{HVf}(T, \mathrm{hpk}, m, \sigma) = 0$, return $\perp$. Otherwise assume that the signature is on the form $\sigma = \big((u_l, v_l)_{l=0}^{\delta-1}, C_\delta, (\mathbf{u}, \mathbf{v}), (\mathbf{u}', \mathbf{v}'), \mathbf{C}, \pi\big)$. A' hands C_δ to its $D_X^{\mathsf{cs}}(\cdot)$ oracle in the CCA2 experiment. If the answer is not on the form y_{α_δ} for some $\alpha_\delta \in \mathcal{L}(T)$, then the $\mathsf{HOpen}(T, \mathrm{hpk}, \mathrm{hsk}(\cdot), \cdot, \cdot)$ oracle is instructed to return $\perp$. This is probably an incorrect reply, which reveals to A that it is simulated. Intuitively this can only happen if A somehow is able to manufacture a $\mathcal{SS}^{\mathsf{cs}}$ signature. Suppose now that y_{α_δ} is on the expected form. Then there is a path $\rho = \alpha_0, \ldots, \alpha_\delta$ corresponding to α_δ. If $\alpha = \alpha_l$ for some $0 \le l \le \delta$, then the $\mathsf{HOpen}(T, \mathrm{hpk}, \mathrm{hsk}(\cdot), \cdot, \cdot)$ oracle is instructed to return α_{l+1} to A. Otherwise it returns $\perp$. Also in this case the reply may be incorrect if A can manufacture a $\mathcal{SS}^{\mathsf{cs}}$ signature. In the analysis below we show that if incorrect replies are a non-negligible event we reach a contradiction.

At some point A outputs $(s_{\mathrm{state}}, \alpha^{(0)}, \alpha^{(1)}, m)$. If $\alpha^{(0)} \neq \beta_\delta^{(0)}$ or $\alpha^{(1)} \neq \beta_\delta^{(1)}$, then A' guessed incorrectly the challenge indices chosen by A. It outputs a random bit $d \in \{0, 1\}$ and halts in this case. Thus, we assume from this point on that $\alpha^{(0)} = \beta_\delta^{(0)}$ and $\alpha^{(1)} = \beta_\delta^{(1)}$.

The single query m to the $\mathsf{HSig}(T, \mathrm{hpk}, \mathrm{hsk}(\alpha^{(b)}, \cdot)$ oracle is simulated as follows. To simplify the exposition we write α_l instead of $\alpha_l^{(b)}$ as in Experiment 2.2. A' chooses $r_l, r'_l \in \mathbb{Z}_{q_3}$ and $\boldsymbol{\tau}, \boldsymbol{\zeta}, \boldsymbol{\tau}', \boldsymbol{\zeta}', \boldsymbol{\psi} \in \mathbb{Z}_N$ randomly and computes

$$
\begin{aligned}
&(u_l, v_l) = (y_{\alpha_l}^{r_l} D_3^{x_{\alpha_l} r'_l}, y_{\alpha_{l+1}} g_3^{r_l} D_2^{r'_l}), \quad \text{for } l = 0, \ldots, \delta - 1 \ , \\
&C_\delta = E_Y^{\mathsf{cs}}(y_{\alpha_\delta}, r) \ , \quad \text{and} \\
&(\mathbf{u}, \mathbf{v}) = (\mathbf{g}^{\boldsymbol{\zeta}}, \mathbf{g}^{\boldsymbol{\tau}}), \quad (\mathbf{u}', \mathbf{v}') = (\mathbf{g}^{\boldsymbol{\zeta}'}, \mathbf{g}^{\boldsymbol{\tau}'}), \quad \mathbf{C} = \mathbf{g}^{\boldsymbol{\psi}} \ .
\end{aligned}
$$

To construct the proof π, A' simply invokes the simulator for the proof of knowledge.

If (D_1, D_2, D_3) is a DDH triple we have $(D_1^{x_{\alpha_l}}, D_2, D_3^{x_{\alpha_l}}) = (y_{\alpha_l}, g_3^f, y_{\alpha_l}^f)$ for some f. If on the other hand (D_1, D_2, D_3) is not a DDH triple, then we have that $(D_1^{x_{\alpha_l}}, D_2, D_3^{x_{\alpha_l}}) = (y_{\alpha_l}, g_3^f, g_3^{f'} y_{\alpha_l}^f)$ for some random $f' \in \mathbb{Z}_{q_3}$. It follows that if (D_1, D_2, D_3) is a DDH triple, the ElGamal cryptotexts are identically distributed to the corresponding parts of a signature returned by the $\mathsf{Sig}^{\mathsf{cs}}(T, \mathrm{hpk}, \mathrm{hsk}(\alpha^{(b)}), \cdot)$ oracle in Experiment 2.2, and otherwise they are encryptions of random elements. Note that when (D_1, D_2, D_3) is not a DDH triple there are no a priori guarantees for how the output of the simulator of the proof of knowledge is distributed.

A' simulates A until it halts with output $d \in \{0,1\}$. Then A' halts with output d. This concludes the definition of the adversary A'.

Analysis of the behavior of A'. We must now analyze the output of A' and reach a contradiction. We divide our analysis into a number of claims.

Denote by d'_b the output of A' on input b. Let E_{noask} denote the event that A never asks for $x_{\alpha_l^{(0)}}$ or $x_{\alpha_l^{(1)}}$. Let E_{guess} denote the event that $\alpha^{(0)} = \beta_\delta^{(0)}$, $\alpha^{(1)} = \beta_\delta^{(1)}$, i.e., that A' guesses the challenge leaves correctly. Let E_{ddh} denote the event that (D_1, D_2, D_3) is a DDH triple.

Claim 2. $\Pr[\mathbf{Exp}^{\mathsf{anon}-b}_{\mathcal{HGS},A}(\kappa, T) = 1 \mid E_{\text{noask}} \wedge E_{\text{guess}}] = \Pr[\mathbf{Exp}^{\mathsf{anon}-b}_{\mathcal{HGS},A}(\kappa, T) = 1]$.

Proof. The variables $\beta^{(0)}$ and $\beta^{(1)}$ are independent from $\mathbf{Exp}^{\mathsf{anon}-b}_{\mathcal{HGS},A}(\kappa, T)$. Thus,

$$\Pr[\mathbf{Exp}^{\mathsf{anon}-b}_{\mathcal{HGS},A}(\kappa, T) = 1 \mid E_{\text{guess}}] = \Pr[\mathbf{Exp}^{\mathsf{anon}-b}_{\mathcal{HGS},A}(\kappa, T) = 1] \ .$$

Experiment 2.2 is defined such that it returns 0 if the event $\overline{E_{\text{noask}}}$ occurs. This implies that

$$\Pr[\mathbf{Exp}^{\mathsf{anon}-b}_{\mathcal{HGS},A}(\kappa, T) = 1 \mid \overline{E_{\text{noask}}}] = 0 \ .$$

Thus

$$\Pr[\mathbf{Exp}^{\mathsf{anon}-b}_{\mathcal{HGS},A}(\kappa, T) = 1 \mid E_{\text{noask}}] = \Pr[\mathbf{Exp}^{\mathsf{anon}-b}_{\mathcal{HGS},A}(\kappa, T) = 1] \ .$$

The claim follows. □

Claim 3. There exists a negligible function $\epsilon(\kappa)$, such that for $b \in \{0,1\}$

$$\begin{aligned} |\Pr[d'_b = 1 \mid E_{\text{noask}} \wedge E_{\text{guess}} \wedge E_{\text{ddh}}] \\ - \Pr[\mathbf{Exp}^{\mathsf{anon}-b}_{\mathcal{HGS},A}(\kappa, T) = 1 \mid E_{\text{noask}} \wedge E_{\text{guess}}]| < \epsilon(\kappa) \ . \end{aligned}$$

Proof. The only part of how A' simulates the experiment to A that is not identically distributed to the corresponding part in Experiment 2.2 is the simulation of the $\mathsf{HOpen}(T, \text{hpk}, \text{hsk}(\cdot), \cdot, \cdot)$ oracle. Differently phrased, the only way that A can distinguish between being simulated by A' and actually run in Experiment 2.2 is by asking the $\mathsf{HOpen}(T, \text{hpk}, \text{hsk}(\cdot), \cdot, \cdot)$ oracle a question that it fails to answer correctly. We show that the oracle answers all questions correctly with overwhelming probability.

Let $p(\kappa)$ denote the running time of A. Then it follows that A asks the simulated $\mathsf{HOpen}(T, \text{hpk}, \text{hsk}(\cdot), \cdot, \cdot)$ oracle at most $p(\kappa)$ queries (α, m, σ) such that $\mathsf{HVf}(T, \text{hpk}, m, \sigma) = 1$. Denote by T_l the machine that simulates A' until $l-1$ queries have been answered by the simulated $\mathsf{HOpen}(T, \text{hpk}, \text{hsk}(\cdot), \cdot, \cdot)$ oracle, and then halts outputting the lth query. We say that a query is *difficult* if it is answered incorrectly by A'.

We must show for $l = 0, \ldots, p(k)$ that T_l outputs a difficult query with negligible probability.

The statement is clearly true for T_0, since its output is empty. Suppose now that the statement is true for T_l for $l < s$, but false for T_s. Then the distribution of the output of T_s is indistinguishable from the distribution of the sth query A asks its $\mathsf{HOpen}(T, \mathrm{hpk}, \mathrm{hsk}(\cdot), \cdot, \cdot)$ oracle. This follows from the union bound, since the lth question is incorrectly answered with negligible probability for $l < s$. Thus, T_s outputs a difficult query (α, m, σ) with probability κ^{-c_1} for some constant c_1 (and large enough κ).

Intuitively, it is clear that we can extract a signature from T_s. Formally, we invoke Lemma 6.6 (in some sense a weak "Forking Lemma" [41]) and conclude that there exists a polynomial Turing machine T_s' that runs T_s as a black-box (simulating the random oracle) and outputs with probability $1/(k^{c_1}p(\kappa))^3$ a message $(\gamma_1, \ldots, \gamma_\delta)$ and a $\mathcal{SS}^{\mathsf{cs}}$ signature $(\varepsilon, \boldsymbol{\sigma}, \boldsymbol{\sigma}')$ such that

$$\mathsf{Vf}^{\mathsf{cs}}(H, N, \mathbf{h}, \mathbf{z}, e', (\gamma_1, \ldots, \gamma_\delta), (\varepsilon, \boldsymbol{\sigma}, \boldsymbol{\sigma}')) = 1 \ ,$$

with the added property that the query was difficult.

A query is difficult precisely when A' has not asked its $\mathsf{Sig}^{\mathsf{cs}}(\cdot, (H, N, \mathbf{h}, \mathbf{z}, e'), t)$ oracle the query $(\gamma_1, \ldots, \gamma_\delta)$. This implies that T_s' breaks the CMA-security of $\mathcal{SS}^{\mathsf{cs}}$, which is a contradiction.

Thus, we conclude that A asks a difficult query with negligible probability. □

Claim 4. There exists a negligible function $\epsilon(\kappa)'$ such that

$$\begin{aligned} &|\Pr[d_0' = 1 \mid E_{\text{guess}} \wedge E_{\text{noask}} \wedge \overline{E_{\text{ddh}}}] \\ &\quad - \Pr[d_1' = 1 \mid E_{\text{guess}} \wedge E_{\text{noask}} \wedge \overline{E_{\text{ddh}}}]| < \epsilon'(\kappa) \ . \end{aligned}$$

Proof. Suppose that the claim is false, i.e., that the left side of the equation is greater or equal to κ^{-c_2} for some constant c_2. The only part of the simulation of A' that differs between the two cases is how C_δ is formed. This implies that we can break the security of $\mathcal{CS}^{\mathsf{cs}}_{H^{\mathsf{CHP}}}$ as follows.

Let A'' be the adversary that is identical to A' except for the following modifications. It receives no input, it simulates the DDH experiment by handing itself a random tuple $(g_3, D_1, D_2, D_3) \in G_{q_3}$, and it simulates the CMA experiment to itself honestly. The CCA2 experiment is not simulated, i.e., A'' plays the role of the adversary in a CCA2 experiment. When A' would construct C_δ as described above, A'' instead requests an encryption of $y_{\alpha_\delta^{(0)}}$ or $y_{\alpha_\delta^{(1)}}$ from the CCA2 experiment. The experiment hands back a $\mathcal{CS}^{\mathsf{cs}}_{H^{\mathsf{CHP}}}$ cryptotext $C_\delta' = E^{\mathsf{cs}}_Y(y_{\alpha_\delta^{(b)}})$ for a randomly chosen $b \in \{0, 1\}$. Instead of generating C_δ by itself, A'' continues with $C_\delta = C_\delta'$. Let $d_b'' = \mathbf{Exp}^{\mathsf{cca}-b}_{\mathcal{CS}^{\mathsf{cs}}_{H^{\mathsf{CHP}}}, A''}(\kappa)$. By construction d_b'' is identically distributed to the conditioned variable $(d_b' \mid \overline{E_{\text{ddh}}})$. Thus

$$\begin{aligned} &|\Pr[\mathbf{Exp}^{\mathsf{cca}-0}_{\mathcal{CS}^{\mathsf{cs}}_{H^{\mathsf{CHP}}}, A''}(\kappa) = 1] - \Pr[\mathbf{Exp}^{\mathsf{cca}-1}_{\mathcal{CS}^{\mathsf{cs}}_{H^{\mathsf{CHP}}}, A''}(\kappa) = 1]| \\ &= |\Pr[d_0'' = 1 \mid E_{\text{guess}} \wedge E_{\text{noask}}] + \Pr[d_0'' = 1 \mid \overline{E_{\text{guess}}} \vee \overline{E_{\text{noask}}}] \\ &\quad - (\Pr[d_1'' = 1 \mid E_{\text{guess}} \wedge E_{\text{noask}}] + \Pr[d_1'' = 1 \mid \overline{E_{\text{guess}}} \vee \overline{E_{\text{noask}}}])| \\ &= |\Pr[d_0'' = 1 \mid E_{\text{guess}} \vee E_{\text{noask}}] - \Pr[d_1'' = 1 \mid E_{\text{guess}} \vee E_{\text{noask}}]| \\ &= |\Pr[d_0' = 1 \mid E_{\text{guess}} \wedge E_{\text{noask}} \wedge \overline{E_{\text{ddh}}}] - \Pr[d_1' = 1 \mid E_{\text{guess}} \wedge E_{\text{noask}} \wedge \overline{E_{\text{ddh}}}]| \\ &\geq \kappa^{-c_2} \end{aligned}$$

where the next to last equality follows from the fact that A' outputs a random bit when the event $\overline{E_{\text{guess}}} \vee \overline{E_{\text{noask}}}$ takes place. The resulting inequality contradicts Lemma 5.8 and Lemma 5.6. □

Claim 5. There exists a $b \in \{0,1\}$ and a constant $c_3 > 0$ such that

$$\kappa^{-c_3} \leq |\Pr[d'_b = 1 \mid E_{\text{ddh}}] - \Pr[d'_b = 1 \mid \overline{E_{\text{ddh}}}]| \ .$$

Proof. Write $p_{b_0,b_1} = \Pr[d'_{b_0} = 1 \mid E_{\text{noask}} \wedge E_{\text{guess}} \wedge E^{b_1}_{\text{ddh}}]$, where we understand $E^{b_1}_{\text{ddh}}$ to be $\overline{E_{\text{ddh}}}$ or E_{ddh} depending on if $b_1 = 0$ or 1. Without loss we assume that $\epsilon(\kappa) = \epsilon'(\kappa)$. Then we have

$$\begin{aligned}
\kappa^{-c} &\leq |\Pr[\mathbf{Exp}^{\mathsf{anon}-0}_{\mathcal{HGS},A}(\kappa,T) = 1] - \Pr[\mathbf{Exp}^{\mathsf{anon}-1}_{\mathcal{HGS},A}(\kappa,T) = 1]| \\
&= |\Pr[\mathbf{Exp}^{\mathsf{anon}-0}_{\mathcal{HGS},A}(\kappa,T) = 1 \mid E_{\text{noask}} \wedge E_{\text{guess}}] \\
&\quad - \Pr[\mathbf{Exp}^{\mathsf{anon}-1}_{\mathcal{HGS},A}(\kappa,T) = 1 \mid E_{\text{noask}} \wedge E_{\text{guess}}]| \\
&\leq |\Pr[d'_0 = 1 \mid E_{\text{noask}} \wedge E_{\text{guess}} \wedge E_{\text{ddh}}] \\
&\quad\quad - \Pr[d'_1 = 1 \mid E_{\text{noask}} \wedge E_{\text{guess}} \wedge E_{\text{ddh}}]| + 2\epsilon(\kappa) \\
&\leq |p_{0,1} - p_{1,1}| + 2\epsilon(\kappa) \leq |p_{0,1} - p_{0,0}| + |p_{0,0} - p_{1,1}| + 2\epsilon(\kappa) \\
&\leq |p_{0,1} - p_{0,0}| + |p_{1,0} - p_{1,1}| + 3\epsilon(\kappa)
\end{aligned}$$

from which it follows that there exists a fixed $b_0 \in \{0,1\}$ and c_3 such that the claim holds. The first inequality holds by assumption, the equality follows from Claim 2, the second inequality follows from Claim 3, and the last inequality follows from Claim 4. □

Claim 6. The DDH assumption is broken.

Proof. Let A''' be identical to A' except from the following modifications. The input b is fixed to the value guaranteed to exist by Claim 5, so it takes no input. It simulates the CCA2 experiment and the CMA experiment to itself. It does not simulate the DDH experiment, i.e., A''' plays the role of the adversary in a DDH experiment. Then we have

$$|\mathbf{Exp}^{\mathsf{ddh}-0}_{G_{q_3},A'''}(\kappa) - \mathbf{Exp}^{\mathsf{ddh}-1}_{G_{q_3},A'''}(\kappa)| = |\Pr[d'_b = 1 \mid \overline{E_{\text{ddh}}}] - \Pr[d'_b = 1 \mid E_{\text{ddh}}]| \ .$$

Combined with Claim 5 this contradicts the DDH assumption over G_{q_3}. □

We now turn to the case when A breaks the hierarchical traceability and describe how A' is implemented in this case.

Hierarchical Traceability. Suppose that A breaks hierarchical traceability. Then

$$\mathbf{Adv}^{\mathsf{trace}}_{\mathcal{HGS},A}(\kappa,T) \geq 1/k^c$$

for some polynomial size tree T, constant $c > 0$ and sufficiently large k. We construct a machine A' that runs A as a blackbox and breaks the security of $\mathcal{SS}^{\mathsf{cs}}$.

Definition of A'. A' takes no input, but it plays the role of the adversary in the CMA experiment.

A' simulates the HKg oracle to A as follows. Steps 1, 2 and 3 are simulated honestly. Then it waits until it receives a public key $(N, \mathbf{h}, \mathbf{x}, e')$ in the CMA experiment. In Step 4, $(N, \mathbf{h}, \mathbf{x}, e')$ is used instead of generating a $\mathcal{SS}^{\mathsf{cs}}$ key pair, i.e., the private key t is never defined. Step 5 and Step 6 are simulated honestly. Step 7 is not performed at all. Step 8 is simulated honestly.

In each iteration in the loop of the experiment simulated to A, A may request $\mathrm{hsk}(\alpha)$. Given such a request A' requests from its own signature oracle $\mathsf{Sig}^{\mathsf{cs}}(\cdot, H, N, \mathbf{h}, \mathbf{z}, e', t)$ a signature $(e_\alpha, \boldsymbol{\sigma}_\alpha, \boldsymbol{\sigma}'_\alpha)$ of the message $(y_{\alpha_1}, \ldots, y_{\alpha_\delta})$, where $\rho = \alpha_0, \alpha_1, \ldots, \alpha_\delta = \alpha$ is the path corresponding to α.

Queries to the HSig oracle are simulated as follows. The first step is simulated honestly. In Step 2 $(\mathbf{u}, \mathbf{v})$, $(\mathbf{u}', \mathbf{v}')$ and $\mathbf{C}$ are replaced by $(\mathbf{g}^{\boldsymbol{\zeta}}, \mathbf{g}^{\boldsymbol{\tau}})$, $(\mathbf{g}^{\boldsymbol{\zeta}'}, \mathbf{g}^{\boldsymbol{\tau}'})$ and $\boldsymbol{g}^{\boldsymbol{\psi}}$ respectively. In Step 3, the simulator of the proof of knowledge is invoked to construct π. The resulting signature is identically distributed to the reply of the $\mathsf{Sig}^{\mathsf{cs}}$ oracle in Experiment 2.3.

At some point A halts with output (m, σ), where the signature is given by $\sigma = \big((u_l, v_l)_{l=0}^{\delta-1},\, C_\delta,\, (\mathbf{u}, \mathbf{v}),\, (\mathbf{u}', \mathbf{v}'),\, \mathbf{C},\, \pi\big)$. Then A' computes $\mathsf{HVf}(T, \mathrm{hpk}, m, \sigma)$. If the result is 0, it outputs $\perp$.

Suppose that the running time of A' is bounded by $p(\kappa)$ for a polynomial κ. We invoke Lemma 6.6 (in some sense a weak "Forking Lemma" [41]) and conclude that there exists a polynomial Turing machine T' that runs A' as a black-box (simulating the random oracle) and outputs with probability $1/(k^c p(\kappa))^3$ a message $(\gamma_1, \ldots, \gamma_\delta)$ and a $\mathcal{SS}^{\mathsf{cs}}$ signature $(\varepsilon, \boldsymbol{\sigma}, \boldsymbol{\sigma}')$ such that

$$\mathsf{Vf}^{\mathsf{cs}}(H, N, \mathbf{h}, \mathbf{z}, e', (\gamma_1, \ldots, \gamma_\delta), (\varepsilon, \boldsymbol{\sigma}, \boldsymbol{\sigma}')) = 1 \ .$$

Claim 7. The security of $\mathcal{SS}^{\mathsf{cs}}$ is broken.

Proof. Consider the list $\gamma = (\gamma_1, \ldots, \gamma_\delta)$. A succeeds in Experiment 2.3 whenever γ does not correspond to a path in T, or γ corresponds to a path, but $\gamma_\delta \notin \mathcal{C}$. In the first case A' clearly succeeds as well since it never asks its $\mathsf{Sig}^{\mathsf{cs}}$ oracle to sign any message that is not a list of public keys corresponding to a path in T. In the second case A' succeeds since by construction, if $\gamma_\delta \notin \mathcal{C}$ it never asked its $\mathsf{Sig}^{\mathsf{cs}}$ oracle to sign the message γ. Thus, A' succeeds whenever A succeeds, and we have $\mathbf{Exp}^{\mathsf{trace}}_{\mathcal{SS}^{\mathsf{cs}}}(\kappa, \emptyset) \geq \mathbf{Exp}^{\mathsf{trace}}_{\mathcal{HGS},A}(\kappa, T) \geq 1/\kappa^c$ which contradicts Lemma 5.12 and Lemma 5.6. □

Conclusion of Proof. If $\mathbf{Adv}^{\mathsf{anon}}_{\mathcal{HGS},A}(\kappa, T)$ is not negligible, Claim 6 gives a contradiction, and if $\mathbf{Adv}^{\mathsf{trace}}_{\mathcal{HGS},A}(\kappa, T)$ is not negligible Claim 7 gives a contradiction. Thus, the theorem is true. □

As noted above, hierarchical anonymity as defined here is a proper generalization of full anonymity as defined in [7], and our scheme can be used as an ordinary (non-hierarchical) group signature scheme by setting the depth of the tree equal to two. Thus our scheme is fully anonymous.

The definition of full anonymity is stronger than previously considered anonymity definitions, since the adversary is allowed to use an Open oracle during the attack. In our case we can handle it since we use a CCA2-secure encryption scheme, and thus reach a contradiction if the adversary is able to form a signature on his own that we cannot answer.

It is shown in [7] that it is necessary to use a CCA2-secure cryptosystem to form a group signature scheme. Still we only use a CCA2-secure cryptosystem for the leaves. This apparent contradiction is resolved by noting that since the public keys y_α are distinct, and we may identify the leaves with their paths in the tree, any query to the $\mathsf{HOpen}(T, \mathrm{hpk}, \mathrm{hsk}(\cdot), \cdot, \cdot)$ oracle for intermediate levels of the tree can be answered using a single query to the decryption oracle for the cryptosystem used to encrypt leaves.

6 Construction of the Proof of Knowledge

We give honest verifier zero-knowledge public coin proofs of knowledge for a number of subprotocols which combined gives the proof of knowledge we need to apply the Fiat-Shamir heuristic to get a signature scheme in the random oracle model. Our protocols are based on a variety of proof techniques including, e.g., proofs of knowledge of exponents, double decker exponentiation, equality of exponents over distinct groups, interval proofs, and equality of exponents over an RSA modulus.

We divide the exposition into a number of subsections. First we describe proofs of knowledge over the groups G_{q_i}. Then we give a proof of equality of exponents over distinct groups. This is followed by the proofs over an RSA modulus $\mathbb{Z}_N$. Finally, the combined protocol is described.

The protocol can be seen as consisting of three steps. In the first step a value ν is shown to be a commitment to the hash of the encrypted public keys along the chain. In the second step it is shown that this value (which is in G_{q_3}) is equal (over $\mathbb{Z}$) to a value in $\mathbb{Z}_N$ hidden in a commitment $\mathbf{C}'$. Finally in the third step the Cramer-Shoup signature the prover has committed to is shown to be a valid signature of the value committed to in $\mathbf{C}'$. This is shown schematically in Figure 5.

Although we focus on efficiency, in some cases we have chosen to divide the protocol into subprotocols for clarity, thus sacrificing some efficiency. Since the by far most time-consuming part of the protocol is the proofs of exponential relations, where to our knowledge the best current method is based on cut-and-choose, saving a few exponentiations in other parts of the protocol yields little in terms of overall performance.

Before we start we recall some definitions.

6.1 Review of Some Notions

We assume familiarity with the notion of interactive proofs (IP) introduced by Goldwasser, Micali and Rackoff [31], and public-coin IPs called Arthur-Merlin games introduced by Babai [4]. The notion of zero-knowledge was introduced by

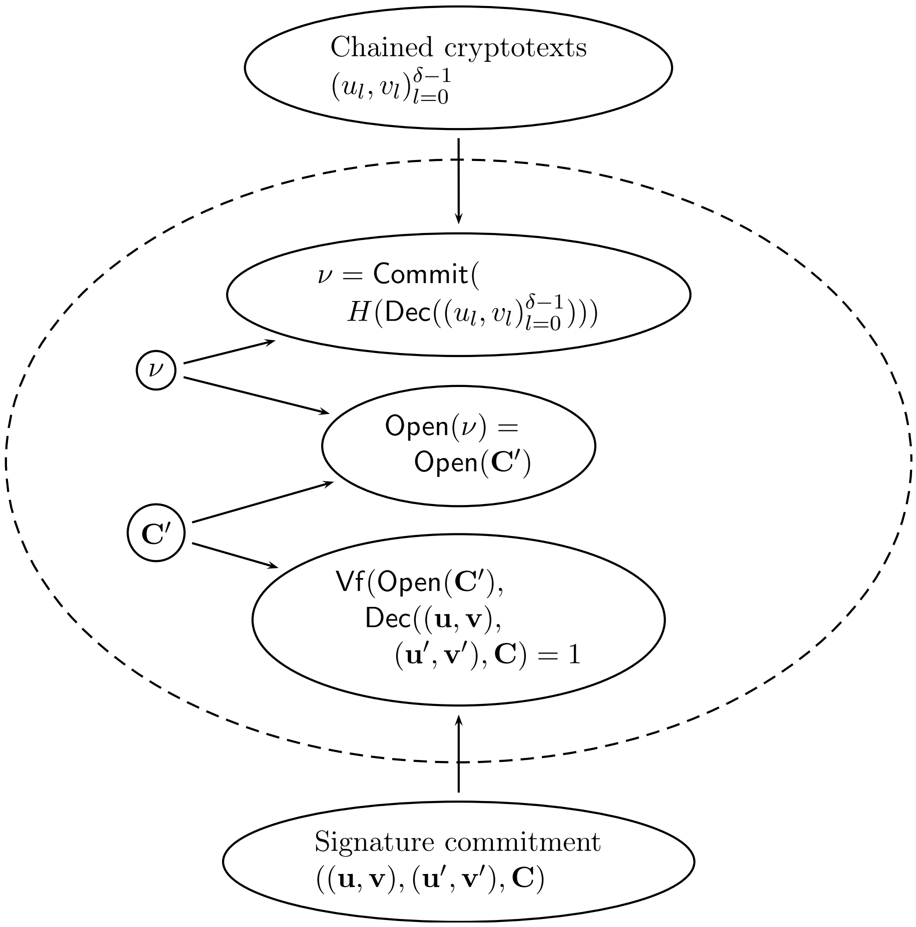

Figure 5: Schematic overview of the proof of knowledge. The functions Commit and Open represent creation and opening of commitments.

Goldwasser, Micali and Rackoff [31]. Statistical special zero-knowledge and special soundness is defined below.

Let $\text{view}_P^V(x)$ denote the view of the verifier V (including its random string) when interacting with the prover P on common input x. Note that the view of the verifier V in a three-move protocol can be written (r, x, α, c, d), where r is the random input of V, x is the common input, α is the first message sent by P, c is the random challenge from a set C sent by V, and d is final message sent by P.

Definition 6.1 (Statistical Special Honest Verifier Zero-Knowledge)**.** *Let (P, V) be a three-move IP for a language L. We say that (P, V) is* statistical special honest verifier zero-knowledge *proof (HVZKP) if there exists a probabilistic polynomial time algorithm S such that the ensembles $\{S(x, c)\}_{x \in L, c \in C}$ and $\{\text{view}_P^V(x)\}_{x \in L, c \in C}$ are statistically close as functions of $|x|$.*

The term *special* is used since the simulator S is not allowed to pick the challenge c itself, but must be able to compute a valid view when given c together with

x as input.

Suppose the challenge $c = (c_1, \dots, c_k)$ is randomly chosen from a product set $C_1 \times C_2 \times \cdots \times C_k$ for some constant k and that $|C_i| \geq 2^\kappa$ for $i = 1, \dots, k$ where κ is the security parameter. Then the following slight generalization of special soundness makes sense. We get the standard definition of special-sound if $k = 1$.

Definition 6.2 (Special Soundness)**.** *A three-move IP (P, V) for a binary relation R is $C_1 \times C_2 \times \cdots \times C_k$-*special-sound *if there exists a polynomial time algorithm that given two accepting outcomes of the view (r, x, α, c, d) and (r, x, α, c', d') with $c_i \neq c'_i$ for all $i = 1, \dots, k$, outputs a witness w such that $(x, w) \in R$.*

We use a generalized definition of Σ-protocol along the lines suggested by Cramer, Damgård, and Schoenmakers [23].

Definition 6.3 (Σ-Protocol)**.** *A $C_1 \times C_2 \times \cdots \times C_k$-$\Sigma$-*protocol *is a three-move interactive proof (P, V) that is both statistical special honest verifier zero-knowledge and $C_1 \times C_2 \times \cdots \times C_k$-special-sound for some product set.*

Observation 6.4. Let (P_l, V_l) be a $C_1 \times C_2 \times \cdots \times C_k$-$\Sigma$-protocol for a language L_l for $l = 1, \dots, k(\kappa)$, where $k(\kappa)$ is polynomial. Then the parallel composition (P, V) of the protocols where a *single* challenge in C is used for *all* protocols is a C-Σ-protocol.

Observation 6.5. Let (P_l, V_l) be a C_l-Σ-protocol for a language L_l for $l = 1, \dots,$ k, where k is constant. Then the parallel composition (P, V) of the protocols is a $C_1 \times C_2 \times \cdots \times C_k$-$\Sigma$-protocol.

It essentially follows from the "Forking Lemma" of Pointcheval and Stern [41] that a Σ-protocol is a proof of knowledge in the sense of [6]. One must only generalize the lemma slightly to hold also for the generalized special soundness as we define it above. However, we only need the following lemma in the proof of Theorem 5.19.

Lemma 6.6. *Let (P, V) be a $C_1 \times C_2 \times \cdots \times C_k$-$\Sigma$-protocol for a language L. Let $L' \subset L$. If A^{O} is a probabilistic random oracle machine running in polynomial time T such that*

$$\Pr[A^{\mathsf{O}} = (m, \alpha, c, d) \wedge \alpha \in L' \wedge V^{\mathsf{O}(m,\cdot)}(\alpha, c, d) = 1] \geq p$$

where the probability is taken over the random choices of A^{O} and the random oracle, and $1/p$ is polynomial, then there exists a polynomial time machine A' running A as a blackbox (and simulating the random oracle), that outputs $(\alpha, w) \in R_L$ such that $\alpha \in L'$ with probability at least $(p/4T)^3 - \epsilon(\kappa)$, where $\epsilon(\kappa)$ is a negligible function.

Proof. Let $C = C_1 \times C_2 \times \cdots \times C_k$. First note that $V^{\mathsf{O}(m,\cdot)}(\alpha, c, d) = 1$ implies that $\mathsf{O}(m, \alpha) = c$. The machine A asks at most T queries to O. Thus, we may identify O with a list of random answers $(r_1, \dots, r_T)$, where $r_i \in C$. Set $r = (r_0, \dots, r_T)$ and write $A_r = (m_r, \alpha_r, c_r, d_r)$ to denote the output of A on internal randomness

$r_0 \in \{0,1\}^T$ and using the oracle O defined by $(r_1,\ldots,r_T)$. Without loss we assume that A has queried O on (m_r,α_r) at some point.

We define A'_j as follows, where j is defined below. It chooses $r \in \{0,1\}^T \times C^T$ and $r' \in \{0,1\}^T \times C^T$ randomly under the restriction $(r_0,r_1,\ldots,r_{j-1}) = (r'_0,r'_1,\ldots,r'_{j-1})$ and executes A twice, the first time with r as random oracle and the second time with r' as random oracle. Then it invokes the extractor guaranteed by the Σ-protocol on (α_r,c_r,d_r) and $(\alpha_{r'},c_{r'},d_{r'})$ and outputs the result. Note that the extractor outputs a correct result only if $m_r = m_{r'}$, $\alpha_r = \alpha_{r'}$ and $c_r \neq c_{r'}$. We now show that this happens with non-negligible probability.

Denote by E_r the event $\alpha_r \in L' \wedge V^{\mathsf{O}(m_r,\cdot)}(\alpha_r,c_r,d_r) = 1$. Then we have $\Pr[E_r] = \sum_{l=1}^{T} \Pr[E_r \wedge c_r = r_l]$, which implies that there exists a j (used in the algorithm above) such that $\Pr[E_r \wedge c_r = r_j] \geq p/T$.

Define

$$F = \{(t_0,\ldots,t_{j-1}) \mid \Pr[E_r \wedge c_r = r_j \mid (r_0,\ldots,r_{j-1}) = (t_0,\ldots,t_{j-1})] \geq p/(2T)\}$$

and let $r^j = (r_0,r_1,\ldots,r_{j-1})$. Then $\Pr[r^j \in F] \geq p/(2T)$ by an average argument, and we have from independence that

$$\begin{aligned}
& \Pr[(E_r \wedge c_r = r_j) \wedge (E_{r'} \wedge c_{r'} = r'_j)] \\
\geq\ & \Pr[(E_r \wedge c_r = r_j) \wedge (E_{r'} \wedge c_{r'} = r'_j) \mid r^j \in F] \Pr[r^j \in F] \\
=\ & \Pr[E_r \wedge c_r = r_j \mid r \in F] \Pr[E_{r'} \wedge c_{r'} = r'_j) \mid r^j \in F] \Pr[r^j \in F] \\
\geq\ & \left(\frac{p}{4T}\right)^3
\end{aligned}$$

Denote by E the event $(E_r \wedge c_r = r_j) \wedge (E_{r'} \wedge c_{r'} = r'_j)$. Denote by E_s the event that $\forall l \in [1,k] : r_{j,l} \neq r'_{j,l}$. This corresponds to fulfilling the hypothesis of the extractor guaranteed by the special soundness. Denote by E_u the event that all r_l and r'_l for $l = 1,\ldots,T$ are unique. The event E_u ensures that answer to the j'th query is used as challenge c, which in turn means that M must have decided on a m and α at that point. Given that the event E occurs this implies $(m_r,\alpha_r) = (m_{r'},\alpha_{r'})$. We clearly have $\Pr[r_{j,l} = r'_{j,l}] = 1/|C_l|$, thus $\Pr[\overline{E_s}] \leq \sum_{l=1}^{k} 1/|C_l|$ by the union bound. We also have $\Pr[r_l = r'_l] = 1/|C|$ and the union bound gives $\Pr[\overline{E_u}] \leq T^2/|C|$. A final application of the union bound gives $\Pr[\overline{E_s} \vee \overline{E_u}] \leq \sum_{l=1}^{k} 1/|C_l| + T^2/|C|$.

We have

$$\begin{aligned}
\Pr[E] &= \Pr[E \wedge E_s \wedge E_u] + \Pr\left[E \wedge \left(\overline{E_s} \vee \overline{E_u}\right)\right] \\
&\leq \Pr[E \wedge E_s \wedge E_u] + \sum_{l=1}^{k} 1/|C_l| + T^2/|C| \\
&\leq \Pr[E \wedge E_s \wedge E_u] + k/2^{\kappa} + T^2/2^{k\kappa} \ .
\end{aligned}$$

It now follows that

$$\Pr[E \wedge E_s \wedge E_u] \geq \Pr[E] - \left(k/2^{\kappa} + T^2/2^{k\kappa}\right) \geq (p/4T)^3 - \left(k/2^{\kappa} + T^2/2^{k\kappa}\right)$$

which concludes the proof, since if the event $E \wedge E_s \wedge E_u$ occurs the extractor is guaranteed to succeed. $\square$

6.2 Proofs of Knowledge in Groups of Known Prime Order

The goal of this section is to provide subprotocols that can be used to prove knowledge of $\gamma_1, \ldots, \gamma_\delta$ and $\tau_0, \ldots, \tau_{\delta-1}$ satisfying the relations that are defined exclusively over G_{q_1}, G_{q_2}, and G_{q_3} in Step 3 in Algorithm 5.15. Relations involving elements over $\mathbb{Z}_N$ are handled in Section 6.3 and Section 6.4. Most of the ideas we use in this section have appeared in various forms in the literature.

In some protocols we use the security parameters κ_1, κ_2 and κ_3. Where used the completeness depends on κ_1, the soundness depends on κ_2 and the amount of information disclosed depends on κ_3.

We begin our program by considering a problem related to that of proving that a list of cryptotexts is chained properly. To simplify the exposition the DDH assumption and strong RSA assumption are implicitly assumed in the formulation of the lemmas, and the bases in the common input are assumed to be chosen at random. In the application of the protocols this is the case.

Protocol 6.7 (Chained Cryptotexts)**.**
COMMON INPUT: $y_0, g, y \in G_q$ and $\left((u_l, v_l), (\mu_l, \nu_l) \right)_{l=0}^{\delta-1} \in G_q^{4\delta}$
PRIVATE INPUT: $r_l, s_l, t_l \in \mathbb{Z}_q$ for $l = 0, \ldots, \delta-1$ and $y_l \in G_q$ for $l = 1, \ldots, \delta$ such that $(u_l, v_l) = E_{(y_l, g)}(y_{l+1}, r_l) = \left(y_l^{r_l}, g^{r_l} y_{l+1}\right)$ and $(\mu_l, \nu_l) = \left(g^{s_l} y^{t_l}, y^{s_l} y_{l+1}\right)$.

1. The prover chooses $a_l \in \mathbb{Z}_q$ randomly and computes
$$A_{1,l} = g^{a_l} \mu_l^{r_l}, \quad A_{2,l} = y^{a_l} \nu_l^{r_l} \tag{1}$$
for $l = 0, \ldots, \delta - 1$.

2. The prover chooses $b_l, e_l, f_l, h_l, i_l, j_l \in \mathbb{Z}_q$ randomly and computes
$$B_0 = y_0^{e_0} \ , \tag{2}$$
for $l = 0, \ldots, \delta - 1$
$$B_{1,l} = g^{b_l} \mu_l^{e_l}, \quad B_{2,l} = y^{b_l} \nu_l^{e_l} \tag{3}$$
$$B_{3,l} = g^{e_l} y^{i_l}, \quad B_{4,l} = g^{i_l} y^{j_l} \ , \tag{4}$$
and for $l = 0, \ldots, \delta - 2$
$$B_{5,l} = g^{f_l} y^{h_l}, \quad B_{6,l} = y^{f_l} \tag{5}$$
and hands $B_0, \left(A_{1,l}, A_{2,l}, B_{1,l}, B_{2,l}, B_{3,l}, B_{4,l}\right)_{l=0}^{\delta-1}, \left(B_{5,l}, B_{6,l}\right)_{l=0}^{\delta-2}$ to the verifier.

3. The verifier chooses $c \in \mathbb{Z}_q$ randomly and hands c to the prover.

4. The prover computes
$$d_{1,l} = c a_l + b_l, \quad d_{2,l} = c r_l + e_l, \tag{6}$$
$$d_{3,l} = -c s_l + i_l, \quad d_{4,l} = -c t_l + j_l \tag{7}$$
for $l = 0, \ldots, \delta - 1$ and for $l = 0, \ldots, \delta - 2$
$$d_{5,l} = c\left(a_l + s_l r_l\right) + f_l, \quad d_{6,l} = c t_l r_l + h_l \tag{8}$$
and hands $(d_{1,l}, d_{2,l}, d_{3,l}, d_{4,l})_{l=0}^{\delta-1}, (d_{5,l}, d_{6,l})_{l=0}^{\delta-2}$ to the verifier.

5. The verifier checks that
$$u_0^c B_0 = y_0^{d_{2,0}} \quad , \tag{9}$$
for $l = 0, \ldots, \delta - 1$ that
$$A_{1,l}^c B_{1,l} = g^{d_{1,l}} \mu_l^{d_{2,l}}, \quad A_{2,l}^c B_{2,l} = y^{d_{1,l}} \nu_l^{d_{2,l}} \tag{10}$$
$$(v_l/\nu_l)^c B_{3,l} = g^{d_{2,l}} y^{d_{3,l}}, \quad B_{4,l} = \mu_l^c g^{d_{3,l}} y^{d_{4,l}} \quad . \tag{11}$$
and finally for $l = 0, \ldots, \delta - 2$ that
$$A_{1,l}^c B_{5,l} = g^{d_{5,l}} y^{d_{6,l}}, \quad (A_{2,l}/u_{l+1})^c B_{6,l} = y^{d_{5,l}} \quad , \tag{12}$$

Intuitively the proof works by first showing that (u_l, v_l) encrypt the key that is committed to in (μ_l, ν_l) and then by showing that key in the commitment (μ_l, ν_l) is the encryption key used for the encryption in (u_{l+1}, v_{l+1}). This concept is depicted in Figure 6.

$$\begin{tikzcd}
(u_0, v_0) \arrow[d, leftrightarrow] & (u_1, v_1) \arrow[d, leftrightarrow] & \cdots & (u_{\delta-1}, v_{\delta-1}) \arrow[d, leftrightarrow] \\
(\mu_0, \nu_0) \arrow[ur, leftrightarrow] & (\mu_1, \nu_1) \arrow[ur, leftrightarrow] & \cdots \arrow[ur, leftrightarrow] & (\mu_{\delta-1}, \nu_{\delta-1})
\end{tikzcd}$$

Figure 6: Overview of Protocol 6.7.

The idea here is that the prover first computes two Pedersen commitments in the bases g, μ_l and y, ν_l respectively. Then $B_{1,l}$ and $B_{2,l}$ are used to show that the prover can open the commitments. With $B_{3,l}$ and $B_{4,l}$ the prover shows that v_l encrypts what is hidden in the commitment ν_l and that the exponent of g in μ_l is the same as the exponent of y in ν_l.With $B_{5,l}$ the prover shows that it can open $A_{1,l}$ also in the base g, y. With $B_{6,l}$ it finally shows that it can open $A_{2,l}/u_l$ as a power of y, which implies that $A_{2,l}$ and u_l contain y_l to the same power and hence that y_l is the key used in the encryption (u_l, v_l). The detailed proof follows.

Lemma 6.8. *Protocol 6.7 is a $\mathbb{Z}_{q_3}$-Σ-protocol.*

Proof. We prove special soundness first. Suppose we have a list $\big(A_{1,l},\, A_{2,l},\, B_{1,l},$ $B_{2,l},\, B_{3,l},\, B_{4,l}\big)_{l=0}^{\delta-1}$, $\big(B_{5,l},\, B_{6,l}\big)_{l=0}^{\delta-2}$, c and $(d_{1,l},\, d_{2,l},\, d_{3,l},\, d_{4,l})_{l=0}^{\delta-1}$, $(d_{5,l},\, d_{6,l})_{l=0}^{\delta-2}$ that satisfy the Equations (1)-(5), and $c' \neq c$ and $(d'_{1,l},\, d'_{2,l},\, d'_{3,l},\, d'_{4,l})_{l=0}^{\delta-2}$, $(d'_{5,l},\, d'_{6,l})_{l=0}^{\delta-1}$ that satisfies the same equations. We solve the equation systems corresponding to Equations (6)-(8) to extract λ_l, α_l, ρ_l, ω_l, ζ_l, and τ_l such that
$$\begin{aligned}
&u_0 = y_0^{\rho_0} \\
&A_{1,l} = g^{\alpha_l} \mu_l^{\rho_l}, \quad A_{2,l} = y^{\alpha_l} \nu_l^{\rho_l}, \\
&A_{1,l} = g^{\omega_l} y^{\lambda_l}, \quad A_{2,l}/u_{l+1} = y^{\omega_l} \\
&v_l/\nu_l = g^{\rho_l} y^{-\zeta_l}, \quad \mu_l = g^{\zeta_l} y^{\tau_l} \quad .
\end{aligned}$$

From this we can compute $\zeta_l^* = (\omega_l - \alpha_l)/\rho_l$ and $\tau_l^* = \lambda_l/\rho_l$ such that $\mu_l = g^{\zeta_l^*} y^{\tau_l^*}$ since

$$\left(g^{(w_l-\alpha_l)} y^{\lambda_l}\right)^{1/\rho_l} = \left(\frac{A_{1,l}}{g^{\alpha_l}}\right)^{1/\rho_l} = \mu_l \ .$$

We have $\nu_l = y^{\zeta_l^*}\gamma_{l+1}$ for some γ_{l+1}, i.e., $(\mu_l, \nu_l) = (y^{\tau_l^*} g^{\zeta_l^*}, y^{\zeta_l^*}\gamma_{l+1})$. This implies that $u_{l+1} = A_{2,l} y^{-\omega_l} = y^{\alpha_l}\nu_l^{\rho_l} y^{-\omega_l} = y^{\alpha_l} y^{\zeta_l^*\rho_l}\gamma_{l+1}^{\rho_l} = y^{\alpha_l+\zeta_l^*\rho_l-\omega_l}\gamma_{l+1}^{\rho_l} = \gamma_{l+1}^{\rho_l}$. Define γ_{l+1}^* by $v_l = g^{\rho_l}\gamma_{l+1}^*$, i.e., $(u_l, v_l) = E_{(\gamma_l,g)}(\gamma_{l+1}^*, \rho_l)$. What remains is to argue that $\zeta_l^* = \zeta_l$, $\tau_l^* = \tau_l$, and $\gamma_{l+1}^* = \gamma_{l+1}$ to connect the "links in the chain". The first two equalities follow from $g^{\zeta_l} y^{\tau_l} = \mu_l = g^{\zeta_l^*} y^{\tau_l^*}$, since otherwise we could use the extractor to extract the discrete logarithm $(\zeta_l - \zeta_l^*)/(\tau_l - \tau_l^*) = \log_g y$, which is a contradiction. The last equality follows from $u_l = g^{\rho_l}$, $v_l/\nu_l = g^{\rho_l} y^{-\zeta_l}$, and $\mu_l = g^{\zeta_l} y^{\tau_l}$. Thus, the protocol is special-sound.

Simulation is straightforward. Choose $a_l \in \mathbb{Z}_q$ randomly and set $(A_{1,l}, A_{2,l}) = E_{(g,y)}(u_l, a_l)$. Then choose $(d_{1,l}, d_{2,l}, d_{3,l}, d_{4,l}, d_{5,l}, d_{6,l})$, where $d_{i,l} \in \mathbb{Z}_q$, and $c \in \mathbb{Z}_q$ randomly and define B_0, $\left(A_{1,l},\ A_{2,l},\ B_{1,l},\ B_{2,l},\ B_{3,l},\ B_{4,l},\ B_{5,l},\ B_{6,l}\right)$ by solving Equations (9)-(11). It is easy to see that the resulting simulation is perfectly distributed. Thus, the protocol is special honest verifier perfect zero-knowledge. □

Next we consider the problem of proving that the values $y_\alpha \in G_{q_3}$ and $g_2^{y_\alpha} \in G_{q_2}$ committed to in two commitments $(\mu, \nu) = (y_3^t g_3^s,\ y_3^s y_\alpha)$ and $(\mu', \nu') = (y_2^{t'} g_2^{s'},\ y_2^{s'} g_2^{y_\alpha})$ respectively satisfy an exponential relation. Stadler [48] studied a simpler problem, namely, given $E_{y_3}(m)$ and g_2^m, prove that an exponential relation holds between the cleartext and the exponent. Although we consider a more complex problem, our protocol is based on his ideas. Note that proving that our relation holds is equivalent to proving knowledge of $s, t \in \mathbb{Z}_{q_2}$ and $s', t' \in \mathbb{Z}_{q_3}$ such that $(\theta, \omega, \phi) = ((\mu')^{\nu^{-1}}, (\nu')^{\nu^{-1}}, \mu^{-1})$ is on the form $(y_2^{t'} g_2^{s'}, y_2^{s'} g_2^{y_3^s}, y_3^t g_3^s)$. For clarity we state this observation as a protocol.

Protocol 6.9 (Exponential Relation Between Committed Values)**.**
Common Input: $g_2, y_2, \mu', \nu' \in G_{q_2}$ and $g_3, y_3, \mu, \nu \in G_{q_3}$.
Private Input: $t', s' \in \mathbb{Z}_{q_2}$ such that $(\mu', \nu') = (y_2^{t'} g_2^{s'}, y_2^{s'} g_2^{y_\alpha})$ and $t, s \in \mathbb{Z}_{q_3}$ such that $(\mu, \nu) = (y_3^t g_3^s, y_3^s y_\alpha)$.

1. Invoke Protocol 6.10 on common input $g_2, y_2, \theta, \omega \in G_{q_2}$ and $g_3, y_3, \phi \in G_{q_3}$, where $(\theta, \omega, \phi) = \left((\mu')^{\nu^{-1}}, (\nu')^{\nu^{-1}}, \mu^{-1}\right)$, and private input $-t, -s \in \mathbb{Z}_{q_3}$ and $t'\nu^{-1}, s'\nu^{-1} \in \mathbb{Z}_{q_2}$. .

We now give the double-decker exponentiation protocol called from within the protocol above. Here κ_2 is a security parameter that determines the soundness of the protocol.

Protocol 6.10 (Double-Decker Exponentiation)**.**
Common Input: $g_2, y_2, \theta, \omega \in G_{q_2}$ and $g_3, y_3, \phi \in G_{q_3}$.
Private Input: $t', s' \in \mathbb{Z}_{q_2}$ and $t, s \in \mathbb{Z}_{q_3}$ such that $(\theta, \omega, \phi) = (y_2^{t'} g_2^{s'},\ y_2^{s'} g_2^{y_3^s},\ y_3^t g_3^s)$.

1. The prover chooses $e_l, f_l \in \mathbb{Z}_{q_3}$ and $e'_l, f'_l \in \mathbb{Z}_{q_2}$ randomly for $l = 1, \ldots, \kappa_2$, computes $F_{1,l} = y_2^{e'_l} g_2^{f'_l}$, $F_{2,l} = y_2^{f'_l} g_2^{y_3^{f_l}}$, and $A_l = y_3^{e_l} g_3^{f_l}$. Then it hands $(F_{1,l}, F_{2,l}, A_l)_{l=1}^{\kappa_2}$ to the verifier.
2. The verifier chooses $b = (b_1, \ldots, b_{\kappa_2}) \in \{0,1\}^{\kappa_2}$ randomly and hands b to the prover.
3. The prover computes $d_{1,l} = e_l - b_l t$, $d_{2,l} = f_l - b_l s$, $d_{3,l} = f'_l - b_l y_3^{d_{2,l}} s'$, and $d_{4,l} = e'_l - b_l y_3^{d_{2,l}} t'$, and hands $(d_{1,l}, d_{2,l}, d_{3,l}, d_{4,l})_{l=1}^{\kappa_2}$ to the verifier.
4. The verifier checks for $l = 1, \ldots, \kappa_2$ that

$$\theta^{b_l y_3^{d_{2,l}}} y_2^{d_{4,l}} g_2^{d_{3,l}} = F_{1,l}, \quad y_2^{d_{3,l}} (\omega^{b_l} g_2^{(1-b_l)})^{y_3^{d_{2,l}}} = F_{2,l}, \tag{13}$$

$$\phi^{b_l} y_3^{d_{1,l}} g_3^{d_{2,l}} = A_l \ . \tag{14}$$

Since Protocol 6.9 only calls Protocol 6.10, we only need to consider Protocol 6.10:

Lemma 6.11. *Protocol 6.10 is a $\{0,1\}^{\kappa_2}$-Σ-protocol with soundness $1 - 2^{\kappa_2}$.*

Proof. We prove special soundness first. Suppose that we are given the outputs from two executions $(F_{1,l}, F_{2,l}, A_l)_{l=1}^{\kappa_2}$, b, $(d_{1,l}, d_{2,l})_{l=1}^{\kappa_2}$ and b', $(d'_{1,l}, d'_{2,l})_{l=1}^{\kappa_2}$ with $b \neq b'$ that satisfy Equations (13)-(14). Thus, for some l, $b_l \neq b'_l$.

Let (ε, τ) and $(\psi, \zeta) \in \mathbb{Z}_{q_3}$ be solutions to the equation systems

$$\left\{ \begin{array}{l} d_{1,l} = e_l - b_l t \\ d'_{1,l} = e_l - b'_l t \end{array} \right\} \quad \text{and} \quad \left\{ \begin{array}{l} d_{2,l} = f_l - b_l s \\ d'_{2,l} = f_l - b'_l s \end{array} \right\} ,$$

This implies that $\phi = y^\tau g^\zeta$.

Consider next the equation system

$$\left\{ \begin{array}{l} d_{3,l} = f'_l - b_l y_3^{d_{2,l}} s' \\ d'_{3,l} = f'_l - b'_l y_3^{d'_{2,l}} s' \end{array} \right\} .$$

Note that $b_l y_3^{d_{2,l}}$ is zero if $b_l = 0$ and non-zero otherwise. Thus, the system is solvable. Let (ψ', ζ') be a solution and assume without loss that $b'_l = 0$. Then we have

$$F_{2,l} = y_2^{d_{3,l}} \omega^{y_3^{d_{2,l}}} = y_2^{\psi' - y_3^{d_{2,l}} \zeta'} \omega^{y_3^{d_{2,l}}} = y_2^{\psi' - y_3^{\psi-\zeta} \zeta'} \omega^{y_3^{\psi-\zeta}}$$

$$F_{2,l} = y_2^{d'_{3,l}} g_2^{y_3^{d'_{2,l}}} = y_2^{\psi'} g_2^{y_3^{d'_{2,l}}} = y_2^{\psi'} g_2^{y_3^{\psi}}$$

Solving for ω gives $\omega = y_2^{\zeta'} g_2^{y_3^{\zeta}}$. Finally, let (ε', τ') be the solution to

$$\left\{ \begin{array}{l} d_{4,l} = e'_l - b_l y_3^{d_{2,l}} t' \\ d'_{4,l} = e'_l - b'_l y_3^{d'_{2,l}} t' \end{array} \right\} .$$

Then we have

$$F_{1,l} = \theta^{y_3^{d_{2,l}}} y_2^{d_{4,l}} g_2^{d_{3,l}} = \theta^{y_3^{d_{2,l}}} y_2^{\varepsilon' - y_3^{d_{2,l}}\tau'} g_2^{\psi' - y_3^{d_{2,l}}\zeta'}$$
$$F_{1,l} = y_2^{d'_{4,l}} g_2^{d'_{3,l}} = y_2^{\varepsilon'} g_2^{\psi'}$$

Solving for θ gives $\theta = y_2^{\tau'} g_2^{\zeta'}$. We conclude that the protocol is special-sound.

The simulator is defined as follows. For $l = 1, \ldots, \kappa_2$ choose $b_l \in \{0, 1\}$ and $d_{1,l}, d_{2,l} \in \mathbb{Z}_{q_3}$ and $d_{3,l}, d_{4,l} \in \mathbb{Z}_{q_2}$ randomly and define $(F_{1,l}, F_{1,l}, A_l)$ by Equations (13)-(14). We conclude that the protocol is special honest verifier perfect zero-knowledge. □

Our next protocol shows that the cleartext of an ElGamal encryption is the value hidden in a commitment. Since the protocol is used in conjunction with Cramer-Shoup cryptotexts, we use a notation that is consistent with the Cramer-Shoup cryptosystem.

Protocol 6.12 (Equality of Committed and Encrypted Cleartexts)**.**
COMMON INPUT: $g_3, y_3, u', v', \bar{g}_1, \bar{h}, u, v \in G_{q_3}$.
PRIVATE INPUT: t', s', r such that $(u', v') = (g_3^{t'} y_3^{s'}, g_3^{s'} m)$ and $(u, v) = (\bar{g}_1^r, \bar{h}^r m)$.

1. The prover chooses $a, e, f \in \mathbb{Z}_{q_3}$ randomly, computes $A_1 = g_3^a y_3^e$, $A_2 = g_3^e \bar{h}^f$, $A_3 = \bar{g}_1^f$, and hands (A_1, A_2, A_3) to the verifier.

2. The verifier chooses $c \in \mathbb{Z}_{q_3}$ randomly and hands it to the verifier.

3. The prover computes $d_1 = ct' + a$, $d_2 = cs' + e$, $d_3 = -cr + f$ and hands (d_1, d_2, d_3) to the verifier.

4. The verifier checks that

$$(u')^c A_1 = g_3^{d_1} y_3^{d_2}, \quad (v'/v)^c A_2 = g_3^{d_2} \bar{h}^{d_3}, \quad u^c A_3 = \bar{g}_1^{d_3} \ . \tag{15}$$

Lemma 6.13. *Protocol 6.12 is a $\mathbb{Z}_{q_3}$-Σ-protocol.*

Proof. Consider special soundness. Given (A_1, A_2, A_3), (c, d_1, d_2, d_3), and (c', d'_1, d'_2, d'_3), with $c \neq c'$, that satisfy the check above, we can solve the corresponding equation systems to find $\tau', \zeta', \tau \in \mathbb{Z}_{q_3}$ such that

$$(u', v'/v, u) \quad = \quad (g_3^{\tau'} y_3^{\zeta'}, g_3^{\zeta'} \bar{h}^{\tau}, \bar{g}_1^{\tau}) \ .$$

This implies that the cryptotext and commitment holds the same value $v/\bar{h}^{\tau}$ as prescribed. Thus, thus the protocol is special-sound.

The simulator chooses $c, d_1, d_2, d_3 \in \mathbb{Z}_{q_3}$ and defines A_1, A_2, A_3 by Equation (15). It is easy to see that the protocol is special honest verifier perfect zero-knowledge. □

Our next protocol shows that a Cramer-Shoup cryptotext is valid. We define it for an arbitrary hash function, although we will later instantiate it with a H^{CHP} hash function.

Protocol 6.14 (Validity of Cramer-Shoup Cryptotext).
Common Input: $H : G_{q_3}^3 \rightarrow \mathbb{Z}_{q_3}$, $\bar{g}_1, \bar{g}_2, u, \mu, v, \nu \in G_{q_3}$, and $\bar{c}, \bar{d} \in G_{q_3}$.
Private Input: $r \in \mathbb{Z}_{q_3}$ such that $(u, \mu, v, \nu) = (\bar{g}_1^r, \bar{g}_2^r, v, \bar{c}^r \bar{d}^{rH(u,\mu,v)})$.

1. The prover randomly selects $a \in \mathbb{Z}_{q_3}$ and computes $B_1 = \bar{g}_1^a$, $B_2 = \bar{g}_2^a$, $B_3 = \left(\bar{c}\bar{d}^{H(u,\mu,v)}\right)^a$ and hands (B_1, B_2, B_3) to the verifier.
2. The verifier randomly selects $c \in \mathbb{Z}_{q_3}$ and hands it to the prover.
3. The prover computes $d = cr + a$ and hands d to the verifier.
4. The verifier checks that $u^c B_1 = \bar{g}_1^d$, $\mu^c B_2 = \bar{g}_2^d$ and $\nu^c B_3 = \left(\bar{c}\bar{d}^{H(u,\mu,v)}\right)^d$.

Lemma 6.15. *Protocol 6.14 is a $\mathbb{Z}_{q_3}$-Σ-protocol.*

Proof. It can easily be verified that the protocol never fails on valid input.

Assuming the output of two executions B_1, B_2, B_3, c, d and B_1, B_2, B_3, c', d' for $c \neq c'$ both satisfying the verification of Step 4, we can compute $\rho = (d-d')/(c-c')$ such that $(u, \mu, \nu) = (\bar{g}_1^\rho, \bar{g}_2^\rho, \bar{c}^\rho \bar{d}^{\rho H(u,\mu,v)})$. Thus, the protocol is special-sound.

The simulator chooses $c, d \in \mathbb{Z}_{q_3}$ randomly and defines B_1, B_2, and B_3 by the equations in Step 4. It follows that the protocol is special honest verifier perfect zero-knowledge. ☐

The next protocol combines the protocols above and provides a solution to the goal of this section, i.e., proving the relations in Step 3 in Algorithm 5.15 involving only elements from G_{q_1}, G_{q_2}, and G_{q_3}.

Protocol 6.16 (Commitment to Hash of Chained Keys).
Common Input: $g_3, y_3, y_{\alpha_0} \in G_{q_3}$, $g_2, y_2 \in G_{q_2}$, $g_1, y_1 \in G_{q_1}$, $H^{\mathsf{CHP}} = (h_1, \ldots, h_\delta) \in G_{q_2}^\delta$, $(u_l, v_l)_{l=0}^{\delta-1} \in G_{q_3}^{2\delta}$, $(\mu'', \nu'') \in G_{q_1}^2$, $\bar{g}_1, \bar{g}_2, \bar{c}, \bar{d}, \bar{h} \in G_{q_3}$, $C_\delta = (\bar{u}, \bar{\mu}, \bar{v}, \bar{\nu}) \in G_{q_3}^4$.
Private Input: $r_0, \ldots, r_\delta \in \mathbb{Z}_{q_3}$, $r, y_{\alpha_1}, \ldots, y_{\alpha_\delta} \in G_{q_3}$ satisfying the equations in Step 3 in Algorithm 5.15, and $s'', t'' \in \mathbb{Z}_{q_2}$ such that

$$(\mu'', \nu'') = (y_1^{t''} g_1^{s''}, y_1^{s''} g_1^{H^{\mathsf{CHP}}(y_{\alpha_1}, \ldots, y_{\alpha_\delta})}) \enspace .$$

1. The prover chooses $y_{\alpha_{\delta+1}} \in G_{q_3}$ and $r_\delta \in \mathbb{Z}_{q_3}$ randomly, computes $(u_\delta, v_\delta) = E_{y_{\alpha_\delta}}\left(y_{\alpha_{\delta+1}}, r_\delta\right)$, and hands (u_δ, v_δ) to the verifier.
2. The prover chooses $s_l, t_l \in \mathbb{Z}_{q_2}$, computes commitments
$$(\mu_l, \nu_l) = \left(g_3^{s_l} y_3^{t_l}, y_3^{s_l} y_{\alpha_{l+1}}\right)$$
for $l = 0, \ldots, \delta - 1$, and hands $(\mu_l, \nu_l)_{l=0}^{\delta-1}$ to the verifier.
3. The prover chooses $s'_l, t'_l \in \mathbb{Z}_{q_3}$ randomly, computes commitments $(\mu'_l, \nu'_l) = (y_2^{t'_l} g_2^{s'_l}, y_2^{s'_l} h_{l+1}^{y_{\alpha_{l+1}}})$ for $l = 0, \ldots, \delta - 1$, and hands $(\mu'_l, \nu'_l)_{l=1}^{\delta}$ to the verifier.
4. The prover and verifier computes $(\mu', \nu') = \left(\prod_{l=0}^{\delta-1} \mu'_l, \prod_{l=0}^{\delta-1} \nu'_l\right)$. The prover computes $s' = \sum_{l=0}^{\delta-1} s'_l$ and $t' = \sum_{l=0}^{\delta-1} t'_l$.

5. Invoke the following protocols in parallel:

   (a) Protocol 6.7 on public input y_{α_0}, g_3, y_3, $\big((u_l,v_l),(\mu_l,\nu_l)\big)_{l=0}^{\delta-1}$, and private input $(r_l,s_l,t_l)_{l=0}^{\delta-1}$ to show that the chain is a valid chain of encrypted keys.

   (b) Protocol 6.9 for $l=0,\ldots,\delta-1$ on public input $g_3,y_3,\mu_l,\nu_l \in G_{q_3}$ and $g_2,y_2,h_l,\mu_l',\nu_l' \in G_{q_2}$, and private input $s_l,t_l \in \mathbb{Z}_{q_3}$ and $s_l',t_l' \in \mathbb{Z}_{q_2}$.

   (c) Protocol 6.9 on public input $g_2,y_2,\mu',\nu' \in G_{q_2}$ and $g_1,y_1,g_1,\mu'',\nu'' \in G_{q_1}$, and private input $s',t' \in \mathbb{Z}_{q_2}$ and $s'',t'' \in \mathbb{Z}_{q_1}$. These two protocols show that (μ'',ν'') is a commitment of the hash value of the public keys in the commitments (μ_l,ν_l).

   (d) Protocol 6.12 on common input $g_3,y_3,\mu_\delta,\nu_\delta \in G_{q_3}$, $\bar{g}_1,\bar{h} \in G_{q_3}$, $\bar{u},\bar{v} \in G_{q_3}$, and private input $t_\delta,s_\delta,r \in \mathbb{Z}_{q_3}$ to show that the $\mathcal{CS}^{\mathsf{cs}}_{H^{\mathsf{CHP}}}$ encryption is an encryption of the value committed to in (μ_δ,ν_δ).

   (e) Protocol 6.14 on common input $\bar{g}_1,\bar{g}_2,\bar{c},\bar{d},\bar{h} \in G_{q_3}$, $C_\delta=(\bar{u},\bar{\mu},\bar{v},\bar{\nu}) \in G_{q_3}^4$, and private input $r \in \mathbb{Z}_{q_3}$ to show that the $\mathcal{CS}^{\mathsf{cs}}_{H^{\mathsf{CHP}}}$ encryption is correctly formed.

Lemma 6.17. *Protocol 6.16 is a* $\{0,1\}^{\kappa_2} \times \mathbb{Z}_{q_3}$*-*$\Sigma$*-protocol.*

Proof. From Lemma 6.8, Lemmas 6.11, 6.13, 6.15 and Observations 6.4 and 6.5 it follows that Step 5 may be considered a single combined $\{0,1\}^{\kappa_2} \times \mathbb{Z}_{q_3}$-$\Sigma$-protocol. However, we must show that the extracted values satisfy additional equations.

From the combined protocols we can extract $\tau_l,\zeta_l,\psi_l \in \mathbb{Z}_{q_3}$, $\gamma_l \in G_{q_3}$, ζ_l', ψ_l', ζ', $\psi' \in \mathbb{Z}_{q_2}$, ζ'', $\psi'' \in \mathbb{Z}_{q_1}$, ψ_δ^*, ζ_δ^*, $\tau \in \mathbb{Z}_{q_3}$, $\Gamma \in G_{q_2}$ such that for $l=0,\ldots,\delta-1$

$$\begin{aligned}
(u_l,v_l) &= E_{(\gamma_l,g_3)}(\gamma_{l+1},\tau_l) = (\gamma_l^{\tau_l}, g_3^{\tau_l}\gamma_{l+1}) \\
(\mu_l,\nu_l) &= (y_3^{\psi_l} g_3^{\zeta_l}, y_3^{\zeta_l}\gamma_l) \\
(\mu_l',\nu_l') &= (y_2^{\psi_l'} g_2^{\zeta_l'}, y_2^{\zeta_l'} h_l^{\gamma_l}) \\
(\mu',\nu') &= (y_2^{\psi'} g_2^{\zeta'}, y_2^{\zeta'}\Gamma) \\
(\mu'',\nu'') &= (y_1^{\psi''} g_1^{\zeta''}, y_1^{\zeta''} g_1^{\Gamma}) \\
(\bar{u},\bar{v}) &= (\bar{g}_1^{\tau}, \bar{h}^{\tau}\gamma_\delta^*) \ .
\end{aligned}$$

If $\prod_{l=1}^{\delta} h_l^{\gamma_l} \neq \Gamma$, then either $\psi' \neq \sum_{l=1}^{\delta}\psi_l'$ or $\zeta' \neq \sum_{l=1}^{\delta}\zeta_l'$. In either case this implies that we can extract $\log_{g_2} y_2$, which is a contradiction. Under Lemma 6.13 it must hold that $\gamma_\delta = \gamma_\delta^*$. Thus, the protocol is special-sound.

To simulate the proof the simulator chooses $u_\delta,v_\delta,\mu_l,\nu_l \in G_{q_3}$, $\mu_l',\nu_l' \in G_{q_2}$ randomly instead of as defined in the protocol. It is easy to see that these elements are identically distributed to the corresponding elements in an execution of the protocol. The simulator invokes the simulator for the combined proof of knowledge of Step 5. It follows that the protocol is special honest verifier perfect zero-knowledge. □

6.3 Proof of Equality of Exponents Over Distinct Groups

In several of our subprotocols, we need to prove relations over an RSA modulus. These proofs differ slightly from proofs over a group of prime order. When performing proofs over an RSA modulus the order of the multiplicative group is not known, and therefore we cannot reduce the exponents.

One way to get around this problem is illustrated in the example below, where we prove knowledge of how to open a Pedersen commitment. Here κ_1, κ_2 and κ_3 are three security parameters. The completeness depends on κ_1, the soundness depends on κ_2 and the amount of information disclosed depends on κ_3.

Zero-knowledge proofs over such moduli have been studied by Fujisaki and Okamoto [28]. Here we use techniques that are similar, although not identical to theirs. In [28] the equivalent to a Pedersen commitment over a composite modulus such as $\mathbb{Z}_N$ is studied. To commit to a number s the prover computes $\mathbf{g}^s\mathbf{y}^r$ where r is drawn from $[0, 2^{\kappa_3}N - 1]$ for a security parameter κ_3. The following two lemmas are proven.

Lemma 6.18 (cf. [28])**.** *There exists a polynomial-time algorithm that takes as input a composite number N and r_1, s_1, r_2, s_2, $r_1 \neq r_2$ and $s_1 \neq s_2$, such that $\mathbf{g}^{r_1}\mathbf{y}^{s_1} = \mathbf{g}^{r_2}\mathbf{y}^{s_2}$ that with high probability outputs the factorization of N.*

Lemma 6.19 (cf. [28])**.** *If $\kappa_3 = \Theta(\log N)$, then $\mathbf{g}^s\mathbf{y}^r$ statistically reveals no information about s.*

First consider the problem of proving equality of exponents over distinct groups. This is used as a bridge between the two parts of our protocol. Two Pedersen commitments are given: one over G_q denoted C and one over $\mathbb{Z}_N$ denoted $\mathbf{C}$. The task is to prove that the committed values are equal when interpreted over the integers. This problem has been studied by Boudot and Traoré [12] as well as by Camenisch and Michels [17] and we follow their example.

Protocol 6.20 (Equality of Exponents Over Distinct Groups)**.**
COMMON INPUT: $\mathbf{g}, \mathbf{y}, \mathbf{C} \in \mathrm{QR}_N$ and $g, y, C \in G_q$, where $q < N$.
PRIVATE INPUT: $e \in [0, q-1]$, $s \in [0, 2^{\kappa_3}N - 1]$, and $s' \in \mathbb{Z}_q$. such that $\mathbf{C} = \mathbf{g}^e\mathbf{y}^s$ and $C = g^e y^{s'}$.

1. The prover chooses $a \in [0, 2^{\kappa_1+\kappa_2+\kappa_3}q - 1], b \in [0, 2^{\kappa_1+\kappa_2+\kappa_3}N - 1]$ and $b' \in [0, 2^{\kappa_1+\kappa_2+\kappa_3}q - 1]$ randomly, computes $\mathbf{A} = \mathbf{g}^a\mathbf{y}^b$, $A = g^a y^{b'}$, and hands $(\mathbf{A}, A)$ to the verifier.

2. The verifier chooses $c \in [0, 2^{\kappa_2} - 1]$ and hands it to the prover.

3. The prover computes

$$\begin{aligned} d_1 &= a + ce \mod 2^{\kappa_1+\kappa_2+\kappa_3}q, & (16)\\ d_2 &= b + cs \mod 2^{\kappa_1+\kappa_2+\kappa_3}N, & (17)\\ d_3 &= b' + cs' \mod 2^{\kappa_1+\kappa_2+\kappa_3}q & (18)\end{aligned}$$

   and hands (d_1, d_2, d_3) to the verifier.

4. The verifier checks that $\mathbf{g}^{d_1}\mathbf{y}^{d_2} = \mathbf{C}^c\mathbf{A}$ (in $\mathbb{Z}_N$) and $g^{d_1}y^{d_3} = C^cA$ (in G_q).

Lemma 6.21. *Protocol 6.20 is a* $[0, 2^{\kappa_2} - 1]$*-*Σ*-protocol with completeness* $1 - 3 \cdot 2^{-\kappa_2}$*.*

Proof. If the prover is honest the verifier accepts if there is no reduction in the computation of d_1, d_2 or d_3. By the union bound this happens with probability not more than $3 \cdot 2^{-\kappa_1}$, which gives a completeness of $1 - 3 \cdot 2^{-\kappa_2}$.

To prove that the protocol is special-sound, assume we have $\mathbf{A}, A, c, d_1, d_2, d_3$ as well as $c' \neq c$, d_1', d_2', d_3', each list satisfying the equations of Step 4. Then by solving the equations system consisting of Equations (16) to (18) over $\mathbb{Z}$ we get $\varepsilon = \frac{d_1-d_1'}{c-c'}$, $\zeta = \frac{d_2-d_2'}{c-c'}$ and $\zeta' = \frac{d_3-d_3'}{c-c'}$.

We now show that ε and ζ' are integers. In [28] the following lemma is proven:

Lemma 6.22 (cf. [28])**.** *Let* P^* *be an oracle that on input* $N, \mathbf{g}, \mathbf{y}$ *outputs* $\mathbf{u}, \boldsymbol{\mu}$ *and two lists* (c, d_1, d_2)*,* (c', d_1', d_2') *satisfying the equations of Step 4 with* $(c - c') \nmid (d_1 - d_1')$ *or* $(c-c') \nmid (d_2 - d_2')$*. Then there exists a polynomial-time machine which on input* $\mathbf{C}, N$ *and access to* P^* *outputs* $\mathbf{z}, e$ *such that* $\mathbf{z}^e = \mathbf{C}$ *and* $e > 1$*.*

From the above Lemma it follows that ζ, τ are integers since otherwise a prover that is able to construct such proofs can easily be made into the oracle of Lemma 6.22 and hence used to break the strong RSA assumption.

A prover able to construct a proof such that the extracted value ζ' is a non-integer can also compute $\log_g y$, so we conclude that also β is an integer and hence $\mathbf{C} = \mathbf{g}^{\varepsilon}\mathbf{y}^{\zeta}$, $C = g^{\varepsilon}y^{\zeta'}$, which concludes the extraction.

The protocol can be simulated by choosing the challenge $c \in [0, 2^{\kappa_2} - 1]$, and the prover's response $d_1, d_2 \in [0, 2^{\kappa_1+\kappa_2+\kappa_3}q - 1]$, $d_3 \in [0, 2^{\kappa_1+\kappa_2+\kappa_3}N - 1]$. Then A and $\mathbf{A}$ are computed according to the equations of Step 4. This gives the same distribution as an execution of the protocol. □

6.4 Zero-knowledge proofs over an RSA modulus

Sometimes it is more convenient to keep the committed number in the base rather than in the exponent. In this case a commitment to a number $\mathbf{z}$ can be computed as $(\mathbf{g}^r\mathbf{y}^s, \mathbf{g}^r\mathbf{z})$ where r, s are chosen at random from $[0, 2^{\kappa_3}N - 1]$.

Protocol 6.23 (Commitment over $\mathbb{Z}_N$)**.**
COMMON INPUT: $\mathbf{g}, \mathbf{y}, \mathbf{u}, \mathbf{v} \in \mathrm{QR}_N$.
PRIVATE INPUT: $s, t \in [0, 2^{\kappa_3}N - 1]$, $\mathbf{r} \in \mathrm{QR}_N$ such that $(\mathbf{u}, \mathbf{v}) = (\mathbf{g}^s\mathbf{y}^t, \mathbf{g}^t\mathbf{r})$.

1. The prover randomly chooses $a, b \in [0, 2^{\kappa_1+\kappa_2+\kappa_3}N - 1]$, computes $\boldsymbol{\mu} = \mathbf{g}^a\mathbf{y}^b$, and hands $\boldsymbol{\mu}$ to the verifier.

2. The verifier chooses $c \in [0, 2^{\kappa_2} - 1]$ randomly and hands it to the prover.

3. The prover computes

$$\begin{aligned} d_1 &= cs + a \mod 2^{\kappa_1+\kappa_2+\kappa_3}N \ , \\ d_2 &= ct + b \mod 2^{\kappa_1+\kappa_2+\kappa_3}N \end{aligned}$$

   and hands (d_1, d_2) to the verifier.

4. The verifier checks that $\mathbf{u}^c\boldsymbol{\mu} = \mathbf{g}^{d_1}\mathbf{y}^{d_2}$.

Lemma 6.24. *Protocol 6.23 is a* $[0, 2^{\kappa_2} - 1]$*-*Σ*-protocol with completeness* $1 - 2 \cdot 2^{-\kappa_1}$.

Proof. It is easy to check that the verifier accepts when there is no reduction modulo $2^{\kappa_1+\kappa_2+\kappa_3}N$ in the computation of d_1 or d_2. Therefore we want to compute the probability of such a reduction to occur. There are at most $2^{\kappa_2+\kappa_3}N$ values of a (corresponding to $ct = 2^{\kappa_2+\kappa_3}N$) for which a reduction occurs for d_1. With $2^{\kappa_1+\kappa_2+\kappa_3}N$ possible values of a the probability for such a reduction to occur is bounded by $\frac{2^{\kappa_2+\kappa_3}N}{2^{\kappa_1+\kappa_2+\kappa_3}N} = 2^{-\kappa_1}$. Since the same reasoning holds for d_2 the union bound gives an upper bound of $2 \cdot 2^{-\kappa_1}$ for a reduction to occur in one of the two computations. Hence the completeness is at least $1 - 2 \cdot 2^{-\kappa_1}$.

For the extraction of s, t and $\mathbf{r}$ to prove special soundness, assume that we have two lists $(\boldsymbol{\mu}, c, d_1, d_2)$ and $(\boldsymbol{\mu}, c', d_1', d_2')$ where $c \neq c'$ which both satisfy the equations in Step 4. By solving the equations in Step 3 (over $\mathbb{Z}$) we get $\zeta = \frac{d_1'-d_1}{c'-c}$ and $\tau = \frac{d_2'-d_2}{c'-c}$. By Lemma 6.22 it follows that ζ and τ are integers. We now have that the value of $\mathbf{r}$ can now be computed as $\mathbf{v}\mathbf{g}^{-\tau}$.

The protocol can be simulated by choosing $c \in [0, 2^{\kappa_2} - 1]$ and $d_1, d_2 \in [0, 2^{\kappa_1+\kappa_2+\kappa_3}N - 1]$ at random and computing $\boldsymbol{\mu} = \mathbf{g}^{d_1}\mathbf{y}^{d_2}\mathbf{u}^{-c}$. This gives a distribution of $\boldsymbol{\mu}, c, d_1, d_2$ equal to that of an execution. □

It is possible to write the protocol without the reduction in the computations of d_1 and d_2. Then we get perfect completeness, but since d_1 and d_2 are not uniformly distributed our simulation will not yield the exact distribution of an execution. It seems that we are forced to choose between perfect zero-knowledge and perfect completeness. It our description we choose perfect zero-knowledge. By choosing κ_1 large enough, say $\kappa_1 = 64$, we can in practice ignore the risk of failure with only a minor increase in running time.

Next we give a protocol that shows that two committed values are equal. The idea of the proof is to show that their ratio is one. The protocol is not a proof of knowledge of the committed value nor of the exponents of the commitments, only of the difference between the exponents. Note that we parametrize the protocol on z to allow for different sizes of the exponents. The different sizes appear when the protocol is applied to products of commitments.

Protocol 6.25 (Equality of Committed Values over $\mathbb{Z}_N$).
Common Input: $\mathbf{g}, \mathbf{y}, \in \mathrm{QR}_N$ and $(\mathbf{u}, \mathbf{v}), (\mathbf{u}', \mathbf{v}') \in \mathrm{QR}_N^2$.
Private Input: $\mathbf{r} \in \mathrm{QR}_N$ such that $(\mathbf{u}, \mathbf{v}) = (\mathbf{g}^s\mathbf{y}^t, \mathbf{g}^t\mathbf{r})$, $(\mathbf{u}', \mathbf{v}') = (\mathbf{g}^{s'}\mathbf{y}^{t'}, \mathbf{g}^{t'}\mathbf{r})$ for some $s, t, s', t' \in [-2^{\kappa_3}z + 1, 2^{\kappa_3}z - 1]$.

1. The prover chooses a, b at random from $[0, 2^{\kappa_1+\kappa_2+\kappa_3}z - 1]$, computes $\boldsymbol{\mu} = (\mathbf{g}^a\mathbf{y}^b)$ and hands $\boldsymbol{\mu}$ to the verifier.

2. The verifier randomly selects $c \in [0, 2^{\kappa_2} - 1]$ and hands it to the prover.

3. The prover computes $d_1 = c(s-s')+a \mod 2^{\kappa_1+\kappa_2+\kappa_3}z$ and $d_2 = c(t-t')+b \mod 2^{\kappa_1+\kappa_2+\kappa_3}z$ and hands (d_1, d_2) to the verifier.

4. The verifier checks that $(\mathbf{u}/\mathbf{u}')^c\boldsymbol{\mu} = \mathbf{g}^{d_1}\mathbf{y}^{d_2}$.

Lemma 6.26. *Protocol 6.25 is a* $[0, 2^{\kappa_2} - 1]$*-*Σ*-protocol with completeness* $1 - 2 \cdot 2^{-\kappa_1}$.

Proof. An honest prover fails to convince the verifier if there is a reduction in the computation of d_1 or d_2. For d_1 this happens either if $c(s - s') + a > 2^{\kappa_1+\kappa_2}z - 1$ or $c(s - s') + a < 0$. The first case can happen only if $c(s - s') > 0$, and then with probability at most $2^{-\kappa_1}$. The second case can happen only if $c(s - s') < 0$, and also then with probability at most $2^{-\kappa_1}$. The same reasoning holds for d_2, giving by the union bound a probability of at most $2 \cdot 2^{-\kappa_1}$ for a reduction to happen. Therefore the completeness is $1 - 2 \cdot 2^{-\kappa_1}$.

To show special soundness assume that we have (a, b), c and (d_1, d_2) satisfying the equations of Step 4 as well as $c' \neq c$ and d_1', d_2' satisfying the same equations. By solving the equations of Step 3 we get ζ, τ satisfying the equations $\mathbf{u}/\mathbf{u}' = \mathbf{g}^\zeta\mathbf{y}^\tau$ and $\mathbf{v}/\mathbf{v}' = \mathbf{g}^\tau$. By Lemma 6.22 ζ and τ are integers. This shows that $(\mathbf{u}/\mathbf{u}', \mathbf{v}/\mathbf{v}')$ is a commitment of the value 1, and hence that $(\mathbf{u}, \mathbf{v})$ and $(\mathbf{u}', \mathbf{v}')$ commit to the same value.

For the simulation choose d_1, d_2 at random from $[0, 2^{\kappa_1+\kappa_2+\kappa_3}z - 1]$ and c from $[0, 2^{\kappa_2} - 1]$. Compute $\boldsymbol{\mu}, \boldsymbol{\nu}$ to satisfy the equations from Step 4. The distribution we get this way is equal to the distribution from an honest execution. □

The above protocol can also be used to prove that a pair $\mathbf{u}, \mathbf{v}$ is a commitment to a public value $\mathbf{w}$. For clarity we state this as a protocol, also this time parametrized on z:

Protocol 6.27 (Committed Value over $\mathbb{Z}_N$)**.**
COMMON INPUT: $\mathbf{g}, \mathbf{y}, \mathbf{u}, \mathbf{v}, \mathbf{w} \in \mathrm{QR}_N$.
PRIVATE INPUT: $s, t \in [-2^{\kappa_3}z + 1, 2^{\kappa_3}z - 1]$ such that $(\mathbf{u}, \mathbf{v}) = (\mathbf{g}^s\mathbf{y}^t, \mathbf{g}^t\mathbf{w})$.

1. Invoke protocol 6.25 on public input $\mathbf{g}, \mathbf{y}, (\mathbf{u}, \mathbf{v}), (\mathbf{1}, \mathbf{w})$ and private exponents $s, t, 1, 1$.

Lemma 6.28. *Protocol 6.27 is a* $[0, 2^{\kappa_2} - 1]$*-*Σ*-protocol with completeness* $1 - 2 \cdot 2^{-\kappa_1}$.

Proof. Follows directly from Lemma 6.26. □

In Protocol 6.9 we showed how to prove that two committed values have an exponential relation. We need to be able to do this also over $\mathbb{Z}_N$. Also in this case we use a protocol for double-decker expontial relations similar to Protocol 6.10 as base to construct the following protocol. Once again we use the fact that proving $(u, v) = (g_N^{s'}y_N^{t'}, y_N^{s'}g_N^{\mathbf{r}})$, $(\mathbf{u}, \mathbf{v}) = (\mathbf{g}^s\mathbf{y}^t, \mathbf{y}^s\mathbf{r})$ is equivalent to proving that $(\theta, \omega, \boldsymbol{\phi}) = (u^{\mathbf{v}^{-1}}, v^{\mathbf{v}^{-1}}, \mathbf{u}^{-1})$ is on the form $(y_N^{f'}g_N^{e'}, y_N^{e'}g_N^{\mathbf{y}^e}, \mathbf{y}^f\mathbf{g}^e)$.

Protocol 6.29 (Exponential Relations over $\mathbb{Z}_N$)**.**
COMMON INPUT: $\mathbf{g}, \mathbf{y}, \mathbf{h} \in \mathrm{QR}_N$, $(\mathbf{u}, \mathbf{v}), (\mathbf{u}', \mathbf{v}') \in \mathrm{QR}_N^2$, and $g_N, y_N \in G_N$.
PRIVATE INPUT: $s, t, s', t' \in [0, 2^{\kappa_3}N - 1]$ such that $(\mathbf{u}, \mathbf{v}) = (\mathbf{g}^s\mathbf{y}^t, \mathbf{g}^t\mathbf{r})$ and $(\mathbf{u}', \mathbf{v}') = (\mathbf{g}^{s'}\mathbf{y}^{t'}, \mathbf{y}^{s'}\mathbf{h}^{\mathbf{r}})$.

1. The prover generates $s'', t'' \in \mathbb{Z}_N$, computes $(u, v) = (g_N^{s''} y_N^{t''}, g_N^{t''} y_N^{\mathbf{r}})$ and hands (u, v) to the verifier.

2. The following two protocols are executed in parallel:

   (a) Protocol 6.31 on common input $\mathbf{g}, \mathbf{y}, \boldsymbol{\phi} \in \mathrm{QR}_N$ and $g_N, y_N, \theta, \omega \in G_N$ where $(\theta, \omega, \boldsymbol{\phi}) = (u^{\mathbf{v}^{-1}}, v^{\mathbf{v}^{-1}}, \mathbf{u}^{-1})$ and private input $t''\mathbf{v}^{-1}, s''\mathbf{v}^{-1} \in \mathbb{Z}_N$ and $-t, -s \in [0, 2^{\kappa_3}N - 1]$.

   (b) Protocol 6.20 on common input $\mathbf{y}, \mathbf{h}, \mathbf{v}' \in \mathrm{QR}_N$, $g_N, y_N, v \in G_N$ and private input $\mathbf{r}, t', t''$.

The idea behind the above protocol is that the value $\mathbf{r}$ first is "lifted" to G_N. The relation is then shown between an element in G_N and an element in $\mathbb{Z}_N$, as shown in Figure 7.

$$\begin{tikzcd} & G_N \arrow[d] \\ \mathbb{Z}_N \arrow[ur] & \mathbb{Z}_N \end{tikzcd}$$

Figure 7: Structure of Protocol 6.29

Lemma 6.30. *Protocol 6.29 is a* $\{0,1\}^{\kappa_2}$*-*Σ*-protocol with completeness* $1 - (2\kappa_2 + 3)2^{-\kappa_1}$.

Proof. Follows from Lemma 6.21 and Lemma 6.32 by using the union bound for the completeness. ☐

Protocol 6.31 (Double-Decker Exponentiation over $\mathbb{Z}_N$)**.**
COMMON INPUT: $\mathbf{g}, \mathbf{y}, \boldsymbol{\phi} \in \mathrm{QR}_N$ and $g_N, y_N, \theta, \omega \in G_N$.
PRIVATE INPUT: $t, s \in [0, 2^{\kappa_3}N - 1]$ and $t', s' \in \mathbb{Z}_N$ such that $(\theta, \omega, \boldsymbol{\phi}) = (y_N^{t'} g_N^{s'}, y_N^{s'} g_N^{\mathbf{y}^s}, \mathbf{y}^t \mathbf{g}^s)$.

1. The prover chooses $e_l, f_l \in [0, 2^{\kappa_1+\kappa_3}N - 1]$ and $e'_l, f'_l \in \mathbb{Z}_N$ randomly for $l = 1, \ldots, \kappa_2$. Then it computes $F_{1,l} = y_N^{e'_l} g_N^{f'_l}$, $F_{2,l} = y_N^{f'_l} g_N^{\mathbf{y}^{f_l}}$, $\mathbf{A}_l = \mathbf{y}^{e_l}\mathbf{g}^{f_l}$ and hands $(F_{1,l}, F_{2,l}, \mathbf{A}_l)_{l=1}^{\kappa_2}$ to the verifier.

2. The verifier randomly chooses $b = (b_1, \ldots, b_{\kappa_2}) \in \{0,1\}^{\kappa_2}$ and hands b to the prover.

3. The prover computes $d_{1,l} = e_l - b_l t \bmod 2^{\kappa_1+\kappa_3}N$, $d_{2,l} = f_l - b_l s \bmod 2^{\kappa_1+\kappa_3}N$, $d_{3,l} = f'_l - b_l \mathbf{y}^{d_{2,l}} s' \bmod N$, $d_{4,l} = e'_l - b_l \mathbf{y}^{d_{2,l}} t' \bmod N$ and hands $(d_{1,l}, d_{2,l}, d_{3,l}, d_{4,l})_{l=1}^{\kappa_2}$ to the verifier.

4. The verifier checks for $l = 1, \ldots, \kappa_2$ that

$$\begin{aligned} \theta^{b_l \mathbf{y}^{d_{2,l}}} y_N^{d_{4,l}} g_N^{d_{3,l}} &= F_{1,l} \ , \\ y_N^{d_{3,l}} (\omega^{b_l} g_N^{1-b_l})^{\mathbf{y}^{d_{2,l}}} &= F_{2,l} \ , \\ \boldsymbol{\phi}^{b_l} \mathbf{y}^{d_{1,l}} \mathbf{g}^{d_{2,l}} &= \mathbf{A}_l \ . \end{aligned}$$

Lemma 6.32. *Protocol 6.31 is a* $\{0,1\}^{\kappa_2}$*-*Σ*-protocol with completeness* $1-2\kappa_2 2^{-\kappa_1}$.

Proof. If there is no reduction in the computations of $d_{1,l}$ and $d_{2,l}$ the verifier will accept if the prover is honest. The probability of a reduction in one computation is $2^{-\kappa_1}$. By the union bound this gives that the probability for a reduction in one of the $2\kappa_2$ computation is at most $2\kappa_2 2^{-\kappa_1}$, giving a completeness of' $1 - 2\kappa_2 2^{-\kappa_1}$.

Now we prove special soundness by describing the extraction. For this we follow the proof of Lemma 6.11, taking into account that the multiplicative order of $\mathbb{Z}_N$ is unknown.

Suppose that we are given two outputs $(F_{1,l},\ F_{2,l},\ \mathbf{A}_l)_{l=1}^{\kappa_2}$, b, $(d_{1,l}, d_{2,l})_{l=1}^{\kappa_2}$ and b', $(d'_{1,l}, d'_{2,l})_{l=1}^{\kappa_2}$ with $b \neq b'$ that satisfy the equations of Step 4. Thus, for some l, $b_l \neq b'_l$.

Let (ε, τ) and (ψ, ζ) be solutions to the equation systems

$$\left\{ \begin{array}{l} d_{1,l} = e_l - b_l t \\ d'_{1,l} = e_l - b'_l t \end{array} \right\} \quad \text{and} \quad \left\{ \begin{array}{l} d_{2,l} = f_l - b_l s \\ d'_{2,l} = f_l - b'_l s \end{array} \right\} \ ,$$

i.e., $\tau = \frac{d_{1,l} - d'_{1,l}}{b_l - b'_l}$ and $\zeta = \frac{d_{2,l} - d'_{2,l}}{b_l - b'_l}$. Since $|b_l - b'_l| = 1$ this gives integral values of τ, ζ when the system is solved over $\mathbb{Z}$. We now have that $\boldsymbol{\phi} = \mathbf{y}^{\tau} \mathbf{g}^{\zeta}$.

Consider next the equation system

$$\left\{ \begin{array}{l} d_{3,l} = f'_l - b_l \mathbf{y}^{d_{2,l}} s' \\ d'_{3,l} = f'_l - b'_l \mathbf{y}^{d'_{2,l}} s' \end{array} \right\} \ .$$

Note that $b_l \mathbf{y}^{d_{2,l}}$ is zero if $b_l = 0$ and non-zero otherwise, and that the inverse of $\mathbf{y}$ over $\mathbb{Z}_N$ can be found in polynomial time. Thus, the system is solvable. Let (ψ', ζ') be a solution and assume without loss that $b'_l = 0$. Then we have

$$\begin{aligned} F_{2,l} &= y_N^{d_{3,l}} \omega^{\mathbf{y}^{d_{2,l}}} = y_N^{\psi' - \mathbf{y}^{d_{2,l}} \zeta'} \omega^{\mathbf{y}^{d_{2,l}}} = y_N^{\psi' - \mathbf{y}^{\psi - \zeta} \zeta'} \omega^{\mathbf{y}^{\psi - \zeta}} \\ F_{2,l} &= y_N^{d'_{3,l}} g_N^{\mathbf{y}^{d'_{2,l}}} = y_N^{\psi'} g_N^{\mathbf{y}^{d'_{2,l}}} = y_N^{\psi'} g_N^{\mathbf{y}^{\psi}} \end{aligned}$$

Solving for ω gives $\omega = y_N^{\zeta'} g_N^{\mathbf{y}^{\zeta}}$. Finally, let (ε', τ') be the solution to

$$\left\{ \begin{array}{l} d_{4,l} = e'_l - b_l \mathbf{y}^{d_{2,l}} t' \\ d'_{4,l} = e'_l - b'_l \mathbf{y}^{d'_{2,l}} t' \end{array} \right\} \ .$$

Then we have

$$\begin{aligned} F_{1,l} &= \theta^{\mathbf{y}^{d_{2,l}}} y_N^{d_{4,l}} g_N^{d_{3,l}} = \theta^{\mathbf{y}^{d_{2,l}}} y_N^{\varepsilon' - \mathbf{y}^{d_{2,l}} \tau'} g_N^{\psi' - \mathbf{y}^{d_{2,l}} \zeta'} \\ F_{1,l} &= y_N^{d'_{4,l}} g_N^{d'_{3,l}} = y_N^{\varepsilon'} g_N^{\psi'} \end{aligned}$$

Solving for θ gives $\theta = y_N^{\tau'} g_N^{\zeta'}$. We conclude that the protocol is special-sound. $\square$

Protocol 6.33 (Knowledge of a Root of a Committed Value over $\mathbb{Z}_N$).
COMMON INPUT: $\mathbf{g}, \mathbf{y}, \mathbf{u}, \mathbf{v}, \mathbf{u}', \mathbf{v}', \mathbf{C} \in \mathrm{QR}_N$.
PRIVATE INPUT: $s, t, s', t', s'', e \in [0, 2^{\kappa_3}N - 1]$ and $\mathbf{r} \in \mathrm{QR}_N$ such that $(\mathbf{u}, \mathbf{v}) = (\mathbf{g}^s\mathbf{y}^t, \mathbf{g}^t\mathbf{r})$, $(\mathbf{u}', \mathbf{v}') = (\mathbf{g}^{s'}\mathbf{y}^{t'}, \mathbf{g}^{t'}\mathbf{r}^e)$ and $\mathbf{C} = \mathbf{g}^{s''}\mathbf{y}^e$.

1. The prover chooses $a, b, f, h, i, j \in [0, 2^{\kappa_1+\kappa_2+\kappa_3}N - 1]$ randomly and computes

$$\begin{aligned} \mathbf{A_1} &= \mathbf{g}^a\mathbf{y}^b\mathbf{u}^e && (19)\\ \mathbf{A_2} &= \mathbf{g}^b\mathbf{v}^e && (20)\\ \mathbf{B_1} &= \mathbf{g}^f\mathbf{y}^h\mathbf{u}^i && (21)\\ \mathbf{B_2} &= \mathbf{g}^h\mathbf{v}^i && (22)\\ \mathbf{B_3} &= \mathbf{g}^j\mathbf{y}^i \ . && (23) \end{aligned}$$

   Then it hands $\mathbf{A_1}, \mathbf{A_2}, \mathbf{B_1}, \mathbf{B_2}, \mathbf{B_3}$ to the verifier. The following protocols are executed in parallel with the protocol below:

   (a) Protocol 6.25 parameterized with $z = (2^{\kappa_2}N)^2 + 2^{\kappa_1+\kappa_2+\kappa_3}N$ on public input $\mathbf{g}$, $\mathbf{y}$, $(\mathbf{A_1}, \mathbf{A_2})$, $(\mathbf{u}', \mathbf{v}')$ and private input $\mathbf{r}^e$ where the secret exponents are $(se + a, te + b)$ and (s', t').
   (b) Protocol 6.23 on public input $\mathbf{g}, \mathbf{y}, (\mathbf{u}, \mathbf{v})$ and private input $s, t, \mathbf{r}$.
   (c) Protocol 6.23 on public input $\mathbf{g}, \mathbf{y}, (\mathbf{u}', \mathbf{v}')$ and private input $s', t', \mathbf{r}^e$.

2. The verifier chooses $c \in [0, 2^{\kappa_2} - 1]$ randomly and hands it to the prover.

3. The prover computes

$$\begin{aligned} d_1 &= ca + f \mod 2^{\kappa_1+\kappa_2+\kappa_3}N && (24)\\ d_2 &= cb + h \mod 2^{\kappa_1+\kappa_2+\kappa_3}N && (25)\\ d_3 &= ce + i \mod 2^{\kappa_1+\kappa_2+\kappa_3}N && (26)\\ d_4 &= cs'' + j \mod 2^{\kappa_1+\kappa_2+\kappa_3}N \ . && (27) \end{aligned}$$

4. The verifier checks that

$$\begin{aligned} \mathbf{A_1}^c \cdot \mathbf{B_1} &= \mathbf{g}^{d_1}\mathbf{y}^{d_2}\mathbf{u}^{d_3} && (28)\\ \mathbf{A_2}^c \cdot \mathbf{B_2} &= \mathbf{g}^{d_1}\mathbf{y}^{d_2}\mathbf{v}^{d_3} && (29)\\ \mathbf{C}^c \cdot \mathbf{B_3} &= \mathbf{g}^{d_4}\mathbf{y}^{d_3} \ . && (30) \end{aligned}$$

Lemma 6.34. *Protocol 6.33 is a* $[0, 2^{\kappa_2} - 1]$*-*Σ*-protocol with completeness* $1 - 10 \cdot 2^{-\kappa_1}$.

Proof. The verifier rejects if one of the three subprotocols fail or there is an overflow in the computation of d_1, d_2, d_3 or d_4. Each of the subprotocols has a probability of failure of $2 \cdot 2^{-\kappa_1}$, and the probability for an overflow for each d_i is $2^{-\kappa_1}$. The union bound then gives a completeness of at least $1 - 10 \cdot 2^{-\kappa_1}$.

Extraction of $s, t, s', t', \mathbf{r}$ follows from Lemmas 6.24 and 6.26. Extraction of s'' and e, assuming two lists ($\mathbf{A_1}$, $\mathbf{A_2}$, $\mathbf{B_1}$, $\mathbf{B_2}$, $\mathbf{B_3}$, c, d_1, d_2, d_3, d_4) and ($\mathbf{A_1}$, $\mathbf{A_2}$, $\mathbf{B_1}$, $\mathbf{B_2}$, $\mathbf{B_3}$, c', d_1', d_2', d_3', d_4') satisfying equations in Step 4, $c \neq c'$, is by solving equations in Step 3 to get ζ, ε such that $\mathbf{C} = \mathbf{g}^{\zeta}\mathbf{y}^{\varepsilon}$. By Lemma 6.22. ζ and ϵ are integers. Thus the protocol is special-sound.

We now show that the protocol is zero-knowledge. The protocol can be simulated by choosing randomly $a, b, a', b' \in [0, 2^{\kappa_1+\kappa_2+\kappa_3}N]$ and setting $\mathbf{A_1} = \mathbf{g}^a\mathbf{y}^b$, $\mathbf{A_2} = \mathbf{g}^{a'}\mathbf{y}^{b'}$. Then we pick at random $d_1, d_2, d_3, d_4 \in [0, 2^{\kappa_1+\kappa_2+\kappa_3}N - 1]$ and $c \in [0, 2^{\kappa_2} - 1]$. $\mathbf{B_1}, \mathbf{B_2}, \mathbf{B_3}$ can then be computed from the equations in Step 4. This distribution is equal to the distribution from an honest execution of the protocols. The subprotocols can be simulated using the same c according to Lemmas 6.24 and 6.26. □

Protocol 6.35 (Equality of Exponents of Committed Values over $\mathbb{Z}_N$).
COMMON INPUT: $\mathbf{g}, \mathbf{y}, \mathbf{h}, \mathbf{u}, \mathbf{v}, \mathbf{C} \in \mathrm{QR}_N$
PRIVATE INPUT: $r, s, t, w \in [0, 2^{\kappa_3}N - 1]$ such that $(\mathbf{u}, \mathbf{v}) = (\mathbf{g}^r\mathbf{y}^s, \mathbf{g}^s\mathbf{h}^w)$ and $\mathbf{C} = \mathbf{g}^w\mathbf{y}^t$.

1. The prover chooses $a, b, e, f \in [0, 2^{\kappa_1+\kappa_2+\kappa_3}N - 1]$, sets $(\boldsymbol{\mu}, \boldsymbol{\nu}) = (\mathbf{g}^a\mathbf{y}^b, \mathbf{g}^b\mathbf{h}^e)$ and $\mathbf{B} = \mathbf{g}^e\mathbf{y}^f$ and hands $(\boldsymbol{\mu}, \boldsymbol{\nu}), \mathbf{B}$ to the verifier.

2. The verifier randomly chooses $c \in [0, 2^{\kappa_2} - 1]$ and hands it to the prover.

3. The prover computes

$$\begin{aligned} d_1 &= cr + a \mod 2^{\kappa_1+\kappa_2+\kappa_3}N , \\ d_2 &= cs + b \mod 2^{\kappa_1+\kappa_2+\kappa_3}N , \\ d_3 &= ct + e \mod 2^{\kappa_1+\kappa_2+\kappa_3}N , \\ d_4 &= cw + f \mod 2^{\kappa_1+\kappa_2+\kappa_3}N \end{aligned}$$

   and hands (d_1, d_2, d_3, d_4) to the verifier.

4. The verifier checks that $\mathbf{u}^c\boldsymbol{\mu} = \mathbf{g}^{d_1}\mathbf{y}^{d_2}$, $\mathbf{v}^c\boldsymbol{\nu} = \mathbf{g}^{d_2}\mathbf{h}^{d_4}$ and $\mathbf{C}^c\mathbf{B} = \mathbf{g}^{d_4}\mathbf{y}^{d_3}$

Lemma 6.36. *Protocol 6.35 is a* $[0, 2^{\kappa_2} - 1]$*-*Σ*-protocol with completeness* $1 - 4 \cdot 2^{-\kappa_1}$.

Proof. An honest verifier will convince the verifier except possibly when there is a reduction in the computation of d_1, d_2, d_3, or d_4. Since the probability of a reduction in the computation of any of these values is $2^{-\kappa_1}$, the union bound gives a completeness of at least $1 - 4 \cdot 2^{-\kappa_1}$.

Now we show that the protocol is special-sound. Assume we have two lists $(\boldsymbol{\mu}, \boldsymbol{\nu})$, $\mathbf{B}$, c, d_1, d_2, d_3, d_4 and $(\boldsymbol{\mu}, \boldsymbol{\nu})$, $\mathbf{B}$, c', d_1', d_2', d_3', d_4' with $c \neq c'$ both satisfying the equations of Step 4. Then we can compute $\rho, \zeta, \tau, \omega$ such that $(\mathbf{u}, \mathbf{v}) = (\mathbf{g}^{\rho}\mathbf{y}^{\zeta}, \mathbf{g}^{\zeta}\mathbf{h}^{\omega})$ and $\mathbf{C} = \mathbf{g}^{\omega}\mathbf{y}^{\tau}$. By Lemma 6.22 $\rho, \zeta, \tau, \omega$ are all integers.

The simulator first chooses d_1, d_2, d_3, d_4 from $[0, 2^{\kappa_1+\kappa_2+\kappa_3} - 1]$ and c from $[0, 2^{\kappa_2} - 1]$. Then $\boldsymbol{\mu}$, $\boldsymbol{\nu}$ and $\mathbf{C}$ are computed by solving the equations of Step 4. This gives a distribution equal to that of an honest execution. □

The following is a protocol (parameterized on k and l) to show that a committed value can be written as $ka + l$ for some a.

Protocol 6.37 (A Committed Value Can Be Written as $ka + l$ over $\mathbb{Z}_N$)**.**
COMMON INPUT: $\mathbf{g}, \mathbf{y}, \mathbf{C} \in \mathrm{QR}_N$.
PRIVATE INPUT: $a, t \in [0, 2^{\kappa_3} N - 1]$ such that $\mathbf{C} = \mathbf{g}^{ka+l}\mathbf{y}^t$.

1. The prover selects $e, f, h \in [0, 2^{\kappa_1+\kappa_2+\kappa_3} N - 1]$, $i \in [0, 2^{\kappa_1+\kappa_2+\kappa_3} kN - 1]$ at random, computes

$$\begin{aligned} \mathbf{A} &= \mathbf{g}^a \mathbf{y}^e &(31)\\ \mathbf{B_1} &= \mathbf{g}^f \mathbf{y}^h &(32)\\ \mathbf{B_2} &= \mathbf{y}^i &(33) \end{aligned}$$

   and hands $(\mathbf{A}, \mathbf{B_1}, \mathbf{B_2})$ to the verifier. Both prover and verifier computes $\mathbf{C}' = \mathbf{g}^l \mathbf{A}^k / \mathbf{C}$.

2. The verifier randomly chooses $c \in [0, 2^{\kappa_2} - 1]$ and hands it to the prover.

3. The prover computes

$$\begin{aligned} d_1 &= ca + f \mod 2^{\kappa_1+\kappa_2+\kappa_3} N &(34)\\ d_2 &= ce + h \mod 2^{\kappa_1+\kappa_2+\kappa_3} N &(35)\\ d_3 &= c(ek - t) + i \mod 2^{\kappa_1+\kappa_2+\kappa_3} kN &(36) \end{aligned}$$

   and hands (d_1, d_2, d_3) to the verifier.

4. The verifier checks that $\mathbf{A}^c \mathbf{B_1} = \mathbf{g}^{d_1} \mathbf{y}^{d_2}$ and $(\mathbf{C}')^c \mathbf{B_2} = y^{d_3}$.

Lemma 6.38. *Protocol 6.37 is an $[0, 2^{\kappa_2} - 1]$-Σ-protocol with completeness $1 - 3 \cdot 2^{-\kappa_2}$.*

Proof. The prover succeeds to convince the verifier unless there is an overflow in one of the computations of d_i. By the union bound, this probability is at most $3 \cdot 2^{\kappa_2}$, giving a completeness of $1 - 3 \cdot 2^{-\kappa_2}$.

For the extraction assume we have two lists $\mathbf{A}$, $\mathbf{B_1}$, $\mathbf{B_2}$, c, d_1, d_2, d_3 and $\mathbf{A}$, $\mathbf{B_1}$, $\mathbf{B_2}$, c', d_1', d_2', d_3', $c \neq c'$, satisfying the equations of Step 4. From this we can compute $\alpha, \varepsilon, \theta$ such that $\mathbf{A} = \mathbf{g}^\alpha \mathbf{y}^\varepsilon$ and $\mathbf{C}' = \mathbf{y}^\theta$. By Lemma 6.22 α, ε and θ are all integers. If we set $\tau = k\varepsilon - \theta$ it holds that $\mathbf{C} = \mathbf{g}^\varepsilon \mathbf{y}^\tau$. This concludes the extraction.

The verifier's view can be simulated by randomly choosing $v \in \mathbb{Z}_N$ and setting $\mathbf{A} = \mathbf{g}^v$. Then $c \in [0, 2^{\kappa_2} - 1]$ is chosen at random together with $d_1, d_2 \in [0, 2^{\kappa_1+\kappa_2} N - 1]$, $d_3 \in [0, 2^{\kappa_1+\kappa_2} N^2 - 1]$. Finally $\mathbf{B_1}, \mathbf{B_2}$ are computed from the equations in Step 4. This gives a distribution equal to that of an honest execution of the protocol. □

We also need the protocol that a committed value lies in an interval by Boudot. Instead of giving the complete protocol, we only give the interface and refer the reader to [11] for complete details.

Protocol Head 6.39 (A Committed Number Lies in an Interval).
COMMON INPUT: $\mathbf{g}, \mathbf{y}, \mathbf{C} \in \mathrm{QR}_N$ and $a, b \in \mathbb{Z}$.
PRIVATE INPUT: $s \in [a, b]$ and $r \in [0, 2^{\kappa_3} N - 1]$ such that $\mathbf{C} = \mathbf{g}^s \mathbf{y}^r$.

Lemma 6.40 (cf. [11]). *Protocol 6.39 is a $\{0,1\}^{\kappa_2}$-Σ-protocol with perfect completeness.*

From these building blocks we can now present the proof that a committed signature is valid.

Protocol 6.41 (Validity of Committed Signature from Hash).
COMMON INPUT: $\mathbf{g}, \mathbf{y}, \mathbf{h}, \mathbf{z}, \mathbf{u}, \mathbf{v}, \mathbf{u}', \mathbf{v}', \mathbf{C}, \mathbf{C}' \in \mathrm{QR}_N, N, H^{\mathsf{Sh}}, e' \in [2^\kappa, 2^{\kappa+1} - 1]$.
PRIVATE INPUT: $r, s, r', s', t, t' \in [0, 2^{\kappa_3} N - 1]$, $e \in [2^\kappa, 2^{\kappa+1} - 1]$, and $w_\alpha \in \mathbb{Z}_{q_2}$ such that

$$
\begin{aligned}
(\mathbf{u}, \mathbf{v}) &= (\mathbf{g}^s \mathbf{y}^r, \mathbf{g}^r \boldsymbol{\sigma}) \\
(\mathbf{u}', \mathbf{v}') &= (\mathbf{g}^{s'} \mathbf{y}^{r'}, \mathbf{g}^{r'} \boldsymbol{\sigma}') \\
\mathbf{C} &= \mathbf{y}^t \mathbf{g}^e \\
\mathbf{C}' &= \mathbf{y}^{t'} \mathbf{g}^{w_\alpha}
\end{aligned}
$$

and the signature is valid, i.e., $\mathsf{Vf}^{\mathsf{cs}}(\mathrm{id}, H^{\mathsf{Sh}}, N, \mathbf{h}, \mathbf{z}, e', w_\alpha, (e, \boldsymbol{\sigma}, \boldsymbol{\sigma}')) = 1$.

1. Let $\mathbf{z}'$ denote $(\boldsymbol{\sigma}')^{e'} \mathbf{h}^{-w_\alpha}$. The prover chooses ζ, τ, ζ', τ', ζ'', τ'', ζ''', τ''', ζ'''', $\tau'''' \in [0, 2^{\kappa_3} N - 1]$ and sets

$$
\begin{aligned}
(\boldsymbol{\mu}, \boldsymbol{\nu}) &= (\mathbf{g}^{\zeta} \mathbf{y}^{\tau}, \mathbf{g}^{\tau} \boldsymbol{h}^{-w_\alpha}) \ , \\
(\boldsymbol{\mu}', \boldsymbol{\nu}') &= (\mathbf{g}^{\zeta'} \mathbf{y}^{\tau'}, \mathbf{g}^{\tau'} \mathbf{z}') \ , \\
(\boldsymbol{\mu}'', \boldsymbol{\nu}'') &= (\mathbf{g}^{\zeta''} \mathbf{y}^{\tau''}, \mathbf{g}^{\tau''} \boldsymbol{\sigma}^{e}) \ , \\
(\boldsymbol{\mu}''', \boldsymbol{\nu}''') &= (\mathbf{g}^{\zeta'''} \mathbf{y}^{\tau'''}, \mathbf{y}^{\zeta'''} H^{\mathsf{Sh}}_{(N,\mathbf{g})}(\mathbf{z}')) \ , \\
(\boldsymbol{\mu}'''', \boldsymbol{\nu}'''') &= (\mathbf{g}^{\zeta''''} \mathbf{y}^{\tau''''}, \mathbf{y}^{\zeta''''} \mathbf{h}^{-H^{\mathsf{Sh}}_{(N,\mathbf{g})}(\mathbf{z}')}) \ .
\end{aligned}
$$

   Then it hands $(\boldsymbol{\mu}, \boldsymbol{\nu})$, $(\boldsymbol{\mu}', \boldsymbol{\nu}')$, $(\boldsymbol{\mu}'', \boldsymbol{\nu}'')$, $(\boldsymbol{\mu}''', \boldsymbol{\nu}''')$, and $(\boldsymbol{\mu}'''', \boldsymbol{\nu}'''')$ to the verifier.

2. The following protocols are run in parallel (using a common challenge $c \in [0, 2^{\kappa_2} - 1]$):

   (a) Protocol 6.23 on the public input $\mathbf{g}, \mathbf{y}, (\mathbf{u}, \mathbf{v})$ and private input $s, r, \boldsymbol{\sigma}$ to show that the prover knows how to open the commitment $(\mathbf{u}, \mathbf{v})$.

   (b) Protocol 6.23 on the public input $\mathbf{g}, \mathbf{y}, (\mathbf{u}', \mathbf{v}')$ and private input $s', r', \boldsymbol{\sigma}'$ to show that the prover knows how to open the commitment $(\mathbf{u}', \mathbf{v}')$.

   (c) Protocol 6.35 on public input $\mathbf{g}, \mathbf{y}, \mathbf{h}, (\boldsymbol{\mu}, \boldsymbol{\nu}), (\mathbf{C}')^{-1}$ and private input ζ, τ, t', w_α to show that $(\boldsymbol{\mu}, \boldsymbol{\nu})$ is a commitment of $\mathbf{h}^{-w_\alpha}$.

   (d) Protocol 6.25 with $z = N + N2^{\kappa+1}$ on public input $\mathbf{g}$, $\mathbf{y}$, $(\boldsymbol{\mu}', \boldsymbol{\nu}')$, $(\boldsymbol{\mu}(\mathbf{u}')^{e'}, \boldsymbol{\nu}(\mathbf{v}')^{e'})$ and private input ζ', τ', $\zeta + se'$, $\tau + re'$ to show that $(\boldsymbol{\mu}', \boldsymbol{\nu}')$ is a commitment of $\mathbf{z}'$.

(e) Protocol 6.33 on public input $\mathbf{g}, \mathbf{y}, (\mathbf{u}, \mathbf{v}), (\boldsymbol{\mu}'', \boldsymbol{\nu}''), \mathbf{C}$ and private exponents s, r, ζ'', τ'', t. This shows that $(\boldsymbol{\mu}'', \boldsymbol{\nu}'')$ hides the value hidden in $(\mathbf{u}, \mathbf{v})$ to the power of the value hidden in $\mathbf{C}$.

(f) Protocol 6.29 on public input $\mathbf{g}, \mathbf{y}, \mathbf{g}, (\boldsymbol{\mu}', \boldsymbol{\nu}'), (\boldsymbol{\mu}''', \boldsymbol{\nu}'''), g_N, y_N$ and ζ', τ', ζ''', τ''' as private input to show that $(\boldsymbol{\mu}''', \boldsymbol{\nu}''')$ is a commitment of a Shamir hash of $\mathbf{z}'$.

(g) Protocol 6.29 on public input $\mathbf{g}, \mathbf{y}, \mathbf{h}^{-1}, (\boldsymbol{\mu}''', \boldsymbol{\nu}'''), (\boldsymbol{\mu}'''', \boldsymbol{\nu}''''), g_N, y_N$ and private input ζ''', τ''', ζ'''', τ'''' to show that $(\boldsymbol{\mu}'''', \boldsymbol{\nu}'''')$ commits to $\mathbf{h}$ to the power of $H^{\mathsf{Sh}}(\mathbf{z}')$.

(h) Protocol 6.27 with $z = 2N$ on public input $\mathbf{g}, \mathbf{y}, (\boldsymbol{\mu}''\boldsymbol{\mu}'''', \boldsymbol{\nu}''\boldsymbol{\nu}''''), \mathbf{z}$ with private input $\zeta'' + \zeta'''', \tau'' + \tau''''$ to finally show that the signature is valid.

(i) Protocol 6.37 with $k = 4$ and $l = 3$ on public input $\mathbf{g}, \mathbf{y}, \mathbf{C}$ and private input e, t to prove that e is odd and different from e'.

(j) Protocol 6.39 on public input $\mathbf{g}, \mathbf{y}, \mathbf{C}, 2^{\kappa}, 2^{\kappa+1} - 1$. and private input e, t to prove that e belongs to the correct interval.

Lemma 6.42. *Protocol 6.41 is* $[0, 2^{\kappa_2} - 1]$-Σ*-protocol with completeness* $1 - (4\kappa_2 + 24)2^{-\kappa_2}$.

Proof. By using the union bound on the probability of failure of the subprotocols, we get a completeness of at least $1 - (4\kappa_2 + 24)2^{-\kappa_2}$.

We now describe the extraction to show that the protocol is special-sound. By Lemmas 6.24, 6.36, and 6.34 we can extract $\rho, \zeta, \rho', \zeta', \tau, \tau', \varepsilon$ such that $\mathbf{u} = \mathbf{g}^{\rho}\mathbf{y}^{\zeta}$, $\mathbf{u}' = \mathbf{g}^{\rho'}\mathbf{y}^{\zeta'}$, $\mathbf{C} = \mathbf{g}^{\tau}\mathbf{y}^{\varepsilon}$, and $\mathbf{C}' = \mathbf{g}^{\tau'}\mathbf{y}^{\omega}$ from Steps 2a, 2b, 2c, and 2e. Now also $\boldsymbol{\varsigma}$, $\boldsymbol{\varsigma}'$ such that $\mathbf{v} = \mathbf{g}^{\zeta}\boldsymbol{\varsigma}$ and $\mathbf{v}' = \mathbf{g}^{\zeta'}\boldsymbol{\varsigma}'$ can be computed.

It now remains to be shown that $(\varepsilon, \boldsymbol{\sigma}, \boldsymbol{\sigma}')$ is a valid signature of ω. We do that by checking that the two steps of Algorithm 5.11 holds. Steps 2i and 2j ensure that Step 1 of the verification algorithm for Cramer-Shoup signatures holds. From Steps 2c and 2d and Lemmas 6.36 and 6.26 it follows that $(\boldsymbol{\mu}', \boldsymbol{\nu}')$ is a commitment of $\mathbf{z}' = (\boldsymbol{\sigma}')^{e'}\mathbf{h}^{-\omega}$. Step 2e and Lemma 6.34 give that $(\boldsymbol{\mu}'', \boldsymbol{\nu}'')$ is a commitment of $\mathbf{h}^{\varepsilon}$, and Steps 2f, 2g with Lemma 6.30 shows that $(\boldsymbol{\mu}'''', \boldsymbol{\nu}'''')$ commits to $\mathbf{h}^{H^{\mathsf{Sh}}(\mathbf{z}')}$. Finally Step 2g shows that the equality of Step 2 of Algorithm 5.11 holds. Hence $(\varepsilon, \boldsymbol{\sigma}, \boldsymbol{\sigma}')$ is valid signature of ω.

Since all subprotocols are Σ-protocols, the constructed protocol can also be simulated. Also all protocols are either of type $[0, 2^{\kappa_2} - 1]$-Σ or $\{0,1\}^{\kappa_2}$-Σ. Since there is a natural bijection between $[0, 2^{\kappa_2} - 1]$ and $\{0,1\}^{\kappa_2}$, the resulting protocol is a $[0, 2^{\kappa_2} - 1]$-Σ-protocol.

□

6.5 Final Protocol

We are finally ready to give the complete proof of a correct signature corresponding to the proof in Step 3 of Algorithm 5.15. The common input consists of a chain of cryptotexts and commitments of a $\mathcal{SS}^{\mathsf{cs}}$ signature of the public keys corresponding to the path of the signer in the tree.

Protocol 6.43 (Valid $\mathcal{HGS}$ Signature)**.**
COMMON INPUT:

- $\mathbf{g}, \mathbf{y}, \mathbf{h}, \mathbf{z} \in \mathrm{QR}_N$
- $e' \in [2^\kappa, 2^{\kappa+1} - 1]$
- $(u_l, v_l)_{l=0}^{\delta-1} \in G_{q_3}^{2\delta}$
- $C_\delta \in G_{q_3}^4$
- $(\mathbf{u}, \mathbf{v}) \in \mathrm{QR}_N^2$
- $(\mathbf{u}', \mathbf{v}') \in \mathrm{QR}_N^2$
- $\mathbf{C} \in \mathrm{QR}_N$
- $H = (h_1, \ldots, h_\delta) \in G_{q_2}^\delta$
- $g_1, y_1 \in G_{q_1}$
- $g_2, y_2 \in G_{q_2}$
- $g_3, y_3 \in G_{q_3}$
- $y_{\alpha_0} \in G_{q_3}$
- $Y \in G_{q_3}^5$.

PRIVATE INPUT:

- $(\tau_0, \ldots, \tau_\delta) \in \mathbb{Z}_{q_3}^{\delta+1}$
- $(\gamma_1, \ldots, \gamma_\delta) \in G_{q_3}^\delta$
- $\varepsilon \in [2^{\kappa/2}, 2^{\kappa/2+1} - 1]$,
- $(\tau, \zeta, \tau', \zeta', \psi) \in [0, 2^{\kappa_3} N - 1]^6$

such that

$$\begin{aligned}
\gamma_0 &= y_{\alpha_0} \\
(u_l, v_l) &= E_{(\gamma_l, g_3)}(\gamma_{l+1}, \tau_l) \text{ for } l = 0, \ldots, \delta - 1 \\
C_\delta &= E_Y^{\mathsf{cs}}(\gamma_\delta, \tau_\delta) \\
\mathbf{u} &= \mathbf{g}^\zeta \mathbf{y}^\tau \\
\mathbf{u}' &= \mathbf{g}^{\zeta'} \mathbf{y}^{\tau'} \\
\mathbf{C} &= \mathbf{g}^\varepsilon \mathbf{y}^\psi
\end{aligned}$$

and $\mathsf{Vf}^{\mathsf{cs}}(H, H^{\mathsf{Sh}}_{(\mathbf{g},N)}, N, \mathbf{h}, \mathbf{z}, e, (\gamma_1, \ldots, \gamma_\delta), (\varepsilon, \mathbf{v}/\mathbf{y}^\tau, \mathbf{v}'/\mathbf{y}^{\tau'})) = 1$.

1. The prover randomly selects $s, t \in \mathbb{Z}_{q_2}$, $t' \in [0, 2^{\kappa_3}N - 1]$ and computes $(\mu, \nu) = (g_1^t y_1^s, y_1^t g_1^{H(\gamma_1,\ldots,\gamma_\delta)})$, $\mathbf{C}' = \mathbf{g}^{H(\gamma_1,\ldots,\gamma_\delta)}\mathbf{y}^{t'}$. The prover hands (μ, ν) and $\mathbf{C}'$ to the verifier.

2. The following protocols are executed in parallel

   (a) Protocol 6.16 on public input g_3, y_3, y_{α_0}, g_2, y_2, g_1, y_1, H, $(u_l, v_l)_{l=0}^{\delta-1}$, (μ, ν), Y, C_δ and private input $\tau_0, \ldots, \tau_l$, $\gamma_1, \ldots, \gamma_\delta$, s, t.

   (b) Protocol 6.20 on public input $\mathbf{g}, \mathbf{y}, \mathbf{C}', g_1, y_1, \nu$ and private input $H(\gamma_1, \ldots, \gamma_\delta)$, t, t'.

   (c) Protocol 6.41 on public input $\mathbf{g}, \mathbf{y}, (\mathbf{u}, \mathbf{v}), (\mathbf{u}', \mathbf{v}'), \mathbf{C}, \mathbf{C}'$ and private input $\tau, \zeta, \tau', \zeta', \psi, s, \varepsilon$.

Lemma 6.44. *Protocol 6.43 is a $[0, 2^{\kappa_2} - 1] \times \mathbb{Z}_{q_2}$-$\Sigma$-protocol with completeness $1 - (4\kappa_2 + 27)2^{-\kappa_2}$.*

Proof. Since the completeness of Steps 2a, 2b, 2c are 1, $1 - 3 \cdot 2^{-\kappa_2}$ and.$1 - (4\kappa_2 + 24)2^{-\kappa_2}$ respectively, the union bound gives a completeness of at least $1 - (4\kappa_2 + 27)2^{-\kappa_2}$ for the constructed protocol.

Extraction of $\tau_0, \ldots, \tau_\delta$ and $\gamma_1, \ldots, \gamma_\delta$ with the necessary properties follows from Lemma 6.17, from which is also follows that ν is a commitment of $H(\gamma_1, \ldots, \gamma_\delta)$. Extraction of $\tau, \zeta, \tau', \zeta', \varepsilon, \psi$ follows from Lemma 6.42. By Lemma 6.42 we can also extract ξ such that

$$\mathsf{Vf}^{\mathsf{cs}}(\mathrm{id}, H^{\mathsf{Sh}}_{(\mathbf{g},N)}, N, \mathbf{h}, \mathbf{z}, e, \xi, (\varepsilon, \mathbf{v}/\mathbf{y}^{\tau}, \mathbf{v}'/\mathbf{y}^{\tau'})) = 1$$

and $\mathbf{C}'$ is a commitment of ξ.

Finally from Step 2b and Lemma 6.21 it follows that $\mathbf{C}'$ and ν are commitments of the same number, i.e., $\xi = H(\gamma_1, \ldots, \gamma_\delta)$. This implies that

$$\mathsf{Vf}^{\mathsf{cs}}(H, H^{\mathsf{Sh}}_{(\mathbf{g},N)}, N, \mathbf{h}, \mathbf{z}, e, (\gamma_1, \ldots, \gamma_\delta), (\varepsilon, \mathbf{v}/\mathbf{y}^{\tau}, \mathbf{v}'/\mathbf{y}^{\tau'})) = 1 \ .$$

Therefore we can conclude that the protocol is special-sound.

The protocol can be simulated since it is constructed from subprotocols that can be simulated. □

7 An Alternative Construction

In this section we sketch an alternative provably secure construction. Let $\mathcal{SS} = (\mathsf{Kg}, \mathsf{Sig}, \mathsf{Vf})$ be a signature scheme. For each group manager M_α (or signer S_α), $(\mathrm{spk}_\alpha, \mathrm{ssk}_\alpha) \leftarrow \mathsf{Kg}(1^\kappa)$, and $(\mathrm{pk}_\alpha, \mathrm{sk}_\alpha) \leftarrow \mathsf{GMKg}(1^\kappa)$ are generated. Then for each child α of $\beta \in T$, $\sigma_\beta(\alpha) = \mathsf{Sig}_{\mathrm{ssk}_\beta}(\mathrm{pk}_\alpha, \mathrm{spk}_\alpha)$ is computed. Finally, for each $\alpha \in T\backslash\{\rho\}$ set $\mathrm{hpk}(\alpha) = (\mathrm{spk}_\alpha, \mathrm{pk}_\alpha, \sigma_\beta(\alpha))$, where $\alpha \in \beta$, and $\mathrm{hsk}(\alpha) = (\mathrm{sk}_\alpha)$. For the root ρ we set $\mathrm{hpk}(\rho) = (\mathrm{spk}_\rho, \mathrm{pk}_\rho)$ and $\mathrm{hsk}(\rho) = (\mathrm{ssk}_\rho, \mathrm{sk}_\rho)$.

Consider a signer S_α corresponding to a path $\alpha_0, \ldots, \alpha_\delta$, where $\alpha_0 = \rho$ and $\alpha_\delta = \alpha$. To sign a message m the signer computes

$$C = (C_0, \ldots, C_\delta) = (E_{\mathrm{pk}_0}(\sigma_{\alpha_0}(\alpha_1)), \ldots, E_{\mathrm{pk}_{\delta-1}}(\sigma_{\alpha_{\delta-1}}(\alpha_\delta)), E_{\mathrm{pk}_\delta}(\mathsf{Sig}_{\mathrm{ssk}_\alpha}(m))) \ ,$$

and provides a NIZK π that C is formed as above with $\mathrm{pk}_0 = \mathrm{pk}_\rho$ and $\alpha_0 = \rho$. The signature consists of the pair (C, π).

To verify a signature (C, π) the verifier simply checks the NIZK π. To open a signature, a group manager M_β on depth l first verifies the signature. If it is not valid, it returns $\perp$. Otherwise it computes $\sigma = D_{\mathrm{sk}_\beta}(C_l)$. If σ is equal to $\sigma_\beta(\alpha)$ for some $\alpha \in \beta$, then it returns α and otherwise $\perp$.

This construction is a strict generalization of the construction in [7] except that we require that the cryptosystem used is anonymous. The construction is provably secure under the existence of a family of trapdoor permutations. However, as part of the proof we must essentially redo the analysis of the CCA2-secure cryptosystem of Sahai [46], and the group signature scheme of Bellare et al. [7], which makes the proof more complex than the proof for the construction we detail in this paper.

A potential advantage of this scheme is that key generation need not be performed centrally. Each group manager M_β could also be given the secret signature key ssk_β which allows it to generate $(\mathrm{spk}_\alpha, \mathrm{pk}_\alpha)$ and $(\mathrm{ssk}_\alpha, \mathrm{sk}_\alpha)$ for a child M_α or S_α by itself. Thus, a group manager could issue keys without interacting with any other group manager. However, as we will see in the next section, it is far from obvious how to define the security of such a scheme.

8 Eliminating the Trusted Key Generator

We have defined hierarchical group signatures using a trusted key generator. This is a natural first step when trying to understand a new notion, but there are situations where one would like a (hierarchical) group signature scheme *without* a trusted party.

If there exists a set of parties of which the majority is trusted, general multi-party techniques can be used to replace the trusted key generator by the secure evaluation of a function. However, this introduces a trust hierarchy that is inconsistent with the hierarchy of the group managers and signers.

Consider now the security of the construction when there is no trusted key generator. In this case hierarchical anonymity and hierarchical traceability (full anonymity and full traceability) do not suffice to ensure security. The problem is that the experiments only consider the advantage of an adversary when all keys are generated honestly. Thus, all bets are off if this is not the case. It is however not clear what a definition of security for (hierarchical) group signatures without a trusted key generator should look like.

The adversary should probably have the power to choose its keys adaptively, based on the keys and signatures of honest parties. There are many subtle issues. For example, without a trusted key generator the default for hierarchical group signatures is that not only trees but general acyclic graphs of group managers are allowed. Is this what we want? If only trees are supposed to be allowed, certificates must embed additional information that restricts how a certificate chain may look and the NIZK must consider this as well. Other interesting questions are: Is there a well defined tree? Do all participants know what the tree look like? Who generates the common random string used by the NIZKs?

We believe that the alternative construction described above is well suited to a setting without a trusted key generator. However, without a rigorous definition of security we cannot claim anything, and currently there exists as far as we know not even a rigorous definition of security for group signatures without a trusted party, even less so for hierarchical group signatures.

9 Conclusion

We have introduced and formalized the notion of hierarchical group signatures and given two constructions. The first construction is provably secure under general assumptions, whereas the second is provably secure under the DDH assumption, the strong RSA assumption and the 4-Cunningham chain assumption in the random oracle model.

Although the latter construction is practical, i.e., it can be implemented and run on modern workstations, it is still relatively slow. Thus, an interesting open problem is to find more efficient constructions of hierarchical group signatures.

10 Acknowledgments

We thank Johan Håstad for excellent advice. We also thank Torsten Ekedahl for explaining the heuristic view on the existence of Cunningham chains. We thank Ronald Cramer for providing us with references. We had valuable discussions on anonymity of cryptosystems with Gustav Hast.

References

[1] M. Abadi and P. Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). In *Theoretical Computer Science, Exploring New Frontiers of Theoretical Informatics IFIP TCS 2000*, volume 1872 of *Lecture Notes in Computer Science*. Springer Verlag, 2000.

[2] G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A practical and provably secure coalition-resistant group signature scheme. In *Advances in Cryptology – CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 255–270. Springer Verlag, 2000.

[3] G. Ateniese and G. Tsudik. Some open issues and directions in group signatures. In *Financial Cryptography '99*, volume 1648 of *Lecture Notes in Computer Science*, pages 196–211. Springer Verlag, 1999.

[4] L. Babai. Trading group theory for randomness. In *17th ACM Symposium on the Theory of Computing (STOC)*, pages 421–429. ACM Press, 1985.

[5] M. Bellare, A. Boldyreva, A. Desai, and D. Pointcheval. Key-privacy in public-key encryption. In *Advances in Cryptology – ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*. Springer Verlag, 2001.

[6] M. Bellare and O. Goldreich. On defining proofs of knowledge. In *Advances in Cryptology – CRYPTO'92*, volume 740 of *Lecture Notes in Computer Science*, pages 390–420. Springer Verlag, 1992.

[7] 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*, volume 2656 of *Lecture Notes in Computer Science*, pages 614–629. Springer Verlag, 2003.

[8] M. Bellare, H. Shi, and C. Zhang. Foundations of group signatures: The case of dynamic groups. In *RSA Conference 2005, Cryptographers' Track 2005*, 2005. To be published. Available from `http://eprint.iacr.org/2004/077`.

[9] M. Blum, P. Feldman, and S. Micali. Non-interactive zero-knowledge and its applications. In *20th ACM Symposium on the Theory of Computing (STOC)*, pages 103–118. ACM Press, 1988.

[10] D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*. Springer Verlag, 2004.

[11] F. Boudot. Efficient proofs that a committed number lies in an interval. In *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 431–444. Springer Verlag, 2000.

[12] F. Boudot and J. Traoré. Efficient publicly veriable secret sharing schemes with fast or delayed recovery. In *2nd International Conference on Information and Communication Security (ICICS)*, volume 1726 of *Lecture Notes in Computer Science*, pages 87–102. Springer Verlag, 1999.

[13] J. Camenisch. Efficient and generalized group signature. In *Advances in Cryptology – EUROCRYPT'97*, volume 1233 of *Lecture Notes in Computer Science*, pages 465–479. Springer Verlag, 1997.

[14] J. Camenisch and J. Groth. Group signatures: Better efficiency and new theoretical aspects. In *Security in Communication Networks 2004*, volume 3352 of *Lecture Notes in Computer Science*. Springer Verlag, 2005.

[15] J. Camenisch and A. Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*. Springer Verlag, 2004.

[16] J. Camenisch and M. Michels. A group signature scheme with improved effiency. In *Advances in Cryptology – ASIACRYPT'98*, volume 1514 of *Lecture Notes in Computer Science*, pages 160–174. Springer Verlag, 1999.

[17] J. Camenisch and M. Michels. Separability and efficiency for generic group signature schemes. In *Advances in Cryptology – CRYPTO'99*, volume 1666 of *Lecture Notes in Computer Science*, pages 413–430. Springer Verlag, 1999.

[18] J. Camenisch and M. Stadler. Efficient group signature schemes for large groups. In *Advances in Cryptology – CRYPTO'97*, volume 1294 of *Lecture Notes in Computer Science*, pages 410–424. Springer Verlag, 1997.

[19] R. Canetti, O. Goldreich, and S. Halevi. The random oracle model revisited. In *30th ACM Symposium on the Theory of Computing (STOC)*, pages 209–218. ACM Press, 1998.

[20] D. Chaum, E. van Heijst, and B. Pfitzmann. Cryptographically strong undeniable signatures, unconditionally secure for the signer. In *Advances in Cryptology – CRYPTO'91*, volume 576 of *Lecture Notes in Computer Science*, pages 470–484. Springer Verlag, 1991.

[21] D. Chaum and E. van Heyst. Group signatures. In *Advances in Cryptology – EUROCRYPT'91*, volume 547 of *Lecture Notes in Computer Science*, pages 257–265. Springer Verlag, 1991.

[22] L. Chen and T.P. Pedersen. New group signature schemes. In *Advances in Cryptology – EUROCRYPT'94*, volume 950 of *Lecture Notes in Computer Science*, pages 171–181. Springer Verlag, 1994.

[23] R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Advances in Cryptology – CRYPTO'94*, volume 839 of *Lecture Notes in Computer Science*, pages 174–187. Springer Verlag, 1994.

[24] R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *Advances in Cryptology – CRYPTO'98*, volume 1462 of *Lecture Notes in Computer Science*, pages 13–25. Springer Verlag, 1998.

[25] R. Cramer and V. Shoup. Signature schemes based on the strong RSA assumption. In *6th ACM Conference on Computer and Communications Security (CCS)*, pages 46–51. ACM Press, 1999.

[26] T. ElGamal. A public key cryptosystem and a signiture scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.

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

[28] E. Fujisaki and T. Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In *Advances in Cryptology – CRYPTO'97*, volume 1294 of *Lecture Notes in Computer Science*, pages 16–30. Springer Verlag, 1997.

[29] O. Goldreich and L. Levin. A hard-core predicate for all one-way functions. In *21st ACM Symposium on the Theory of Computing (STOC)*, pages 25–32. ACM Press, 1989.

[30] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[31] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.

[32] S. Goldwasser, S. Micali, and R. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, 1988.

[33] A. Kiayias, Y. Tsiounis, and M. Yung. Traceable signatures. In *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*. Springer Verlag, 2004.

[34] A. Kiayias and M. Yung. Group signatures: Provable security, efficient constructions and anonymity from trapdoor-holders. Cryptology ePrint Archive, Report 2004/076, 2004. `http://eprint.iacr.org/2004/076`.

[35] S. Kim, S. Park, and D. Won. Group signatures for hierarchical multigroups. In *Information Security Workshop – ISW'97*, volume 1396 of *Lecture Notes in Computer Science*, pages 273–281. Springer Verlag, 1998.

[36] A. Lysyanskaya and Z. Ramzan. Group blind digital signatures: A scalable solution to electronic cash. In *Financial Cryptography'98*, volume 1465 of *Lecture Notes in Computer Science*, pages 184–197. Springer Verlag, 1998.

[37] H. Maier. Primes in short intervals. *Michigan Mathematical Journal*, 32(2):221–225, 1985.

[38] A. Menezes, P. Oorschot, and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

[39] S. Micali, M. Rabin, and J. Kilian. Zero-knowledge sets. In *44th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 80–91. IEEE Computer Society Press, 2003.

[40] G.L. Miller. Riemann's hypothesis and tests for primality. *Journal of Compter and System Sciences*, 13:300–317, 1976.

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

[42] The prime pages. `http://primes.utm.edu`, March 2004.

[43] The proth search page. `http://www.prothsearch.net`, March 2004.

[44] C. Rackoff and D. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In *Advances in Cryptology – CRYPTO'91*, volume 576 of *Lecture Notes in Computer Science*, pages 433–444. Springer Verlag, 1991.

[45] R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[46] A. Sahai. Non-malleable non-interactive zero-knowledge and adaptive chosen-ciphertext security. In *40th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 543–553. IEEE Computer Society Press, 1999.

[47] A. De Santis, G. Di Crescenzo, R. Ostrovsky, G. Persiano, and A. Sahai. Robust non-interactive zero knowledge. In *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 566–598. Springer Verlag, 2001.

[48] M. Stadler. Publicly verifiable secret sharing. In *Advances in Cryptology – EUROCRYPT'96*, volume 1070 of *Lecture Notes in Computer Science*, pages 190–199. Springer Verlag, 1996.

[49] A. Young and M. Yung. Finding length-3 positive Cunningham chains and their cryptographic significance. In *Algorithmic Number Theory – ANTS-III*, volume 1423 of *Lecture Notes in Computer Science*, pages 289–298. Springer Verlag, 1998.