# Commuting Signatures and Verifiable Encryption and an Application to Non-Interactively Delegatable Credentials

Georg Fuchsbauer

École normale supérieure, CNRS - INRIA, Paris, France
http://www.di.ens.fr/∼fuchsbau

**Abstract**

Verifiable encryption allows to encrypt a signature and prove that the plaintext is valid. We introduce a new primitive called *commuting signature* that extends verifiable encryption in multiple ways: a signer can encrypt both signature and message and prove validity; more importantly, given a ciphertext, a signer can create a verifiably encrypted signature on the encrypted message; thus signing and encrypting commute. We instantiate commuting signatures using the proof system by Groth and Sahai (EUROCRYPT '08) and the automorphic signatures by Fuchsbauer (ePrint report 2009/320). As an application, we give an instantiation of *delegatable anonymous credentials*, a powerful primitive introduced by Belenkiy et al. (CRYPTO '09). Our instantiation is arguably simpler than theirs and it is the first to provide *non-interactive* issuing and delegation, which is a standard requirement for non-anonymous credentials. Moreover, the size of our credentials and the cost of verification are less than half of those of the only previous construction, and efficiency of issuing and delegation is increased even more significantly. All our constructions are proved secure in the standard model.

## 1 Introduction

A verifiably-encrypted-signature scheme [BGLS03] enables a signer to make a digital signature on a message, encrypt the signature under a third party's encryption key, and produce a proof asserting that the ciphertext contains a valid signature. Suppose the message is only available as an encryption. The signer cannot make a signature on it, as this would contradict the security of the encryption scheme (given two messages and the encryption of one of them, a signature on the plaintext could be used to decide which message was encrypted). However, the following does not seem a priori impossible: given an encryption, instead of producing a signature on the plaintext, the signer produces a verifiably encrypted signature on it.

We show that—surprisingly—such a functionality is feasible and moreover give a practical instantiation of it. We then use this new primitive to build the first non-interactively delegatable anonymous credential scheme: given an encrypted public key, a delegator can make an encrypted certificate on the key together with a proof of validity.

**Delegatable Anonymous Credentials.** Access control that respects users' privacy concerns is a challenging problem in security. To gain access to resources, a participant must prove to possess the required credential issued by an authority. To increase manageability of the system, the authority does usually not issue credentials directly to each user, but relies on intermediate layers in the hierarchy. Belenkiy et al. [BCC+09] give the following example: a system administrator issues credentials for webmasters to use his server. The latter are entitled to create forums and delegate rights to moderators, who in turn can give posting privileges to users.

In the real world delegation of rights is realized by certifying the public key of the delegated user. Consecutive delegation leads to a credential chain, consisting of public keys and certificates linking them, starting with the *original issuer* of the credential and ending with a user, say Alice. To delegate her credential to Bob, Alice simply extends the length of the chain by one by appending a certificate on Bob's public key under hers.

*Anonymous* credentials [Cha85, Dam90, LRSW00, Bra99, CL01, CL02, CL04, BCKL08] aim to provide a similar functionality while at the same time not revealing information about the user's identity when obtaining or showing a credential. However, the goal of reconciling delegatability and anonymity remained elusive—until

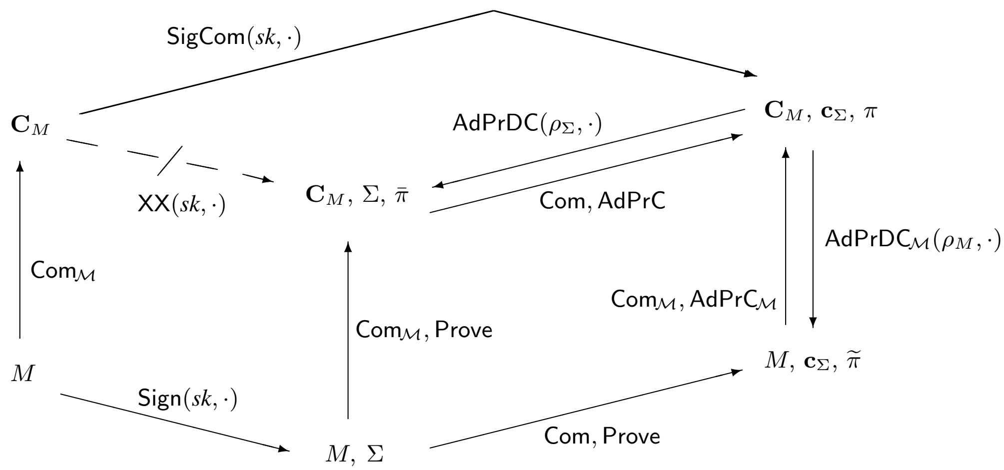


Figure 1: Commuting signatures

recently. Chase and Lysyanskaya [CL06] give delegatable anonymous credentials for which the size of a credential is *exponential* in its length (i.e., the number of delegations), which makes them impractical. In [BCC$^+$09] Belenkiy et al. introduce a new approach using a non-interactive zero-knowledge (NIZK) proof system [BFM88] with *randomizable* proofs: a credential is a non-interactive *proof of knowledge* of a certification chain that can be randomized before being re-delegated or shown; this guarantees anonymity and unlinkability.

The functionality of the system can be sketched as follows: each user holds a secret key which she uses to produce multiple pseudonyms *Nym*. A user $A$ can be known to user $O$ as $\mathit{Nym}_A^{(O)}$ and to $B$ as $\mathit{Nym}_A^{(B)}$. Given a credential issued by $O$ for $\mathit{Nym}_A^{(O)}$, $A$ can transform it into a credential for $\mathit{Nym}_A^{(B)}$ and show it to $B$. Moreover $A$ can delegate the credential to user $C$, known to $A$ as $\mathit{Nym}_C^{(A)}$. $C$ can then show a credential from $O$ for $\mathit{Nym}_C^{(D)}$ to user $D$ (without revealing neither $\mathit{Nym}_A^{(C)}$ nor $\mathit{Nym}_C^{(A)}$), or redelegate it, and so on.

Delegation preserves anonymity, i.e., delegator and delegatee learn nothing more about each other than their respective pseudonyms. In the instantiation of [BCC$^+$09] (BCCKLS), the delegation protocol is fairly complex and highly interactive—as opposed to (non-anonymous) credentials, where it suffices to know a user's public key in order to issue or delegate a credential to her. We correct this shortcoming by giving an instantiation of the BCCKLS model that enables *non-interactive* delegation: pseudonyms are *encryptions* of the public key; given a pseudonym *Nym*, the delegator can produce a credential for the holder of *Nym* without any interaction, since she can make a proof of knowledge of a signature on a public key given to her as an encryption *Nym*. We note that, as for the BCCKLS instantiation, abuse prevention mechanisms such as anonymity revocation [CL01] or limited show [CHK$^+$06] can be added to our construction.

**Commuting Signatures and Verifiable Encryption.** Our main building block to instantiate non-interactively delegatable anonymous credentials will be a new primitive we call *commuting signature* which we sketch in the following and formally define in Sect. 4. Assume we have a digital signature scheme and an encryption scheme combined with a proof system with the following properties: given a verification key, a message and a signature on it valid under the key, we can encrypt any subset of {key, message, signature}, and make a proof that the plaintexts constitute a triple of a key, a message and a valid signature. We also require that the proof does not leak any more information about the encrypted values besides validity.

For consistency with our instantiation using the Groth-Sahai methodology [GS08], we will say *commitment* instead of *encryption*. Note that the commitments we use are *extractable*, and therefore constitute an encryption scheme (see below). We denote committing to signatures by Com and committing to messages by $\mathsf{Com}_{\mathcal{M}}$. Besides allowing to prove validity of committed values, a commuting-signature scheme provides the following additional functionalities (sketched in Figure 1). Note that none of them requires the extraction (decryption) key.

SigCom. Given a commitment $\mathbf{C}_M$ to a message $M$ and a signing key *sk*, SigCom produces a commitment $\mathbf{c}_\Sigma$ to a signature $\Sigma$ on $M$ under *sk*, and a proof $\pi$ that the content of $\mathbf{c}_\Sigma$ is a valid signature on the content of $\mathbf{C}_M$.

$\mathsf{AdPrC}$ ("adapt proof when committing"). Given a commitment $\mathbf{C}_M$ to $M$, a signature $\Sigma$ on $M$ and a proof $\bar{\pi}$ of validity of $\Sigma$ on the content of $\mathbf{C}_M$, we can make a commitment $\mathbf{c}_\Sigma$ to $\Sigma$ using randomness $\rho_\Sigma$ and run algorithm $\mathsf{AdPrC}$ on $\mathbf{C}_M, \Sigma, \rho_\Sigma$ and $\bar{\pi}$. Its output is a proof $\pi$ that the content of $\mathbf{c}_\Sigma$ is a valid signature on the content of $\mathbf{C}_M$. $\mathsf{AdPrDC}$ ("adapt proof when decommitting") does the converse: given a committed message $\mathbf{C}_M$, a committed signature $\mathbf{c}_\Sigma$ together with the used randomness $\rho_\Sigma$, and a proof $\pi$, $\mathsf{AdPrDC}$ outputs a proof $\bar{\pi}$ of validity of the signature $\Sigma$ on the committed message.

$\mathsf{AdPrC}_\mathcal{M}$. Analogously we define algorithms for proof adaptation when committing and decommitting to the *message*. Given a message $M$, a commitment $\mathbf{c}_\Sigma$ to a signature on $M$ and a proof of validity $\widetilde{\pi}$, $\mathsf{AdPrC}_\mathcal{M}$ transforms the proof to the case when the message is committed as well. $\mathsf{AdPrDC}_\mathcal{M}$ is given commitments $\mathbf{C}_M$ and $\mathbf{c}_\Sigma$ to a signature and a message $M$, the randomness $\rho_M$ for $\mathbf{C}_M$ and a proof $\pi$. It adapts $\pi$ to a proof $\widetilde{\pi}$ that the content of $\mathbf{c}_\Sigma$ is a valid signature on $M$.

$\mathsf{AdPrC}_\mathcal{K}$. Finally, we can also adapt proofs when committing or decommitting to the *verification key*. Given commitments $\mathbf{C}_M$ and $\mathbf{c}_\Sigma$ to a message and a signature, a proof of validity $\pi$, the verification key *vk* and randomness $\rho_{vk}$, $\mathsf{AdPrC}_\mathcal{K}$ outputs a proof $\widehat{\pi}$ that the content of $\mathbf{c}_\Sigma$ is a signature on the content of $\mathbf{C}_M$ valid under the key *vk* given as a commitment $\mathbf{c}_{vk}$ with randomness $\rho_{vk}$. $\mathsf{AdPrDC}_\mathcal{K}$ is given $(vk, \rho_{vk}, \mathbf{C}_M, \mathbf{c}_\Sigma)$ and adapts a proof $\widehat{\pi}$ for $(\mathbf{c}_{vk}, \mathbf{C}_M, \mathbf{c}_\Sigma)$ to a proof for $(vk, \mathbf{C}_M, \mathbf{c}_\Sigma)$.

We require that committing, signing and the functionalities above *commute* with each other, that is, it does not matter in which order we execute them; e.g. signing a message, committing to the message and the signature and proving validity yields the same as committing to the message and then running $\mathsf{SigCom}$. Thus, the diagram in Fig. 1 commutes. Note that due to the argument given in the beginning there cannot exist a functionality $\mathsf{XX}$ that given a commitment $\mathbf{C}_M$ to a message $M$ and a secret key *sk* outputs a signature $\Sigma$ on $M$.

**Instantiating Commuting Signatures.** In [Fuc09], Fuchsbauer gave the first efficient implementation of blind signatures [Cha82] with *round-optimal* issuing [Fis06]: this means that the user who wants to obtain a blind signature sends one message to signer, who replies with one message from which the user can construct the blind signature. The scheme can be sketched as follows: the user *randomizes* the message by multiplying it with a random term, makes an (extractable) commitment to the message and the randomness and adds a witness-indistinguishable (WI) proof that the commitments contain the correct values. The signer therefore learns nothing about the message. Using the randomized message, the signer can fabricate a "pre-signature", from which, knowing the randomizer, the user can retrieve an actual signature. To prevent the signer from linking the resulting signature to the signing session, the actual blind signature is a proof of knowledge (PoK) of the signature. The PoK consists of extractable commitments to the signature components and a WI proof that the committed values satisfy the signature-verification equation on the message, in other words, a verifiably encrypted signature.

We observe that the values sent from the user to the signer can be seen as a commitment to (or an encryption of) the message. We show that this commitment can be used by the signer to *directly* construct a proof of knowledge of a signature on the committed message (that is, extractable commitments to the signature components and a proof that the committed values constitute a valid signature on the committed message). As in [Fuc09], the commitments and WI proofs are instantiated with the Groth-Sahai methodology for committing to elements from a bilinear group and constructing proofs that they satisfy *pairing-product equations*. These commitments are extractable, thus the extraction key acts as the decryption key and witness indistinguishability implies semantic security (cf. Sect. 3.1). We will use the notions *encryption* and *extractable commitment* interchangeably. An extractable commitment to a signature together with a proof of validity is a *verifiably encrypted signature* (VES) and can also be interpreted as a proof of knowledge of a signature (since by decryption, the signature can be extracted). Our instantiation of commuting signatures is given in Sect. 6.

**Instantiating Delegatable Anonymous Credentials.** Belenkiy et al. [BCC+09] show that Groth-Sahai proofs can be randomized and combine them with an authentication scheme for secret keys to construct delegatable credentials. A pseudonym *Nym* is a commitment to the user's secret key and a credential is a proof of knowledge of an authentication chain. Such a proof consists of commitments to secret keys, commitments to authenticators between the keys, and proofs of validity. To issue or delegate, the issuer and the user jointly compute a proof of knowledge of an authenticator on the content of the user's pseudonym. In the case of delegation, the issuer prepends her own credential, which she randomizes before. The authors note that secret keys cannot be extracted from the commitments, and that an adversary against the authentication scheme must be allowed to ask for authenticators *on* as well as *under* the attacked key. They therefore give an *F-unforgeable certification-secure* authentication scheme.

We avoid this notion and interactivity of delegation by replacing the authenticators on secret keys by signatures on public keys, in particular we use the automorphic signatures from [Fuc09]. (Automorphic signatures are Groth-Sahai compatible signatures whose verification keys lie in the message space.) A credential is then a chain of *public keys* and *certificates* (as in the non-anonymous case), which are all given as commitments completed with proofs of validity. Commuting signatures enable non-interactive delegation (and issuing, which is a special case of delegation): given a pseudonym (i.e., a commitment to the public key) of a user, the issuer can produce a commitment $\mathbf{c}_\Sigma$ to a signature on the committed user key and a proof $\pi$ of validity using $\mathsf{SigCom}$. If it is a delegation, the issuer then randomizes her own credential $\mathit{cred}_I$, yielding a credential $\mathit{cred}_I{}'$ on a new pseudonym $\mathit{Nym}_I{}'$ that is unlinkable to $\mathit{cred}_I$. Finally, running $\mathsf{AdPrC}_\mathcal{K}$, she adapts the proof $\pi$ to a proof $\widehat{\pi}$ of validity of the content of $\mathbf{c}_\Sigma$ on the content of the user pseudonym under the content of the issuer's new pseudonym $\mathit{Nym}_I{}'$. The credential for the user is then $\mathit{cred}_I{}' \,\|\, \mathit{Nym}_I{}' \,\|\, (\mathbf{c}_\Sigma, \widehat{\pi})$.

Replacing the authenticators from [BCC+09], which consist of 11 group elements and are verified by 8 pairing-product equations (PPE), with automorphic signatures (consisting of 5 group elements and satisfying 3 PPEs) more than doubles efficiency of the scheme. More importantly, our delegation (and issuing) protocol outperforms theirs significantly (see Sect. 5.3 for a more detailed comparison).

Automorphic signatures were combined with Groth-Sahai proofs in [Fuc09] to construct *anonymous proxy signatures* [FP08]. This primitive is related to anonymous credentials in that it considers proving rights in an anonymous way; but it does not achieve mutual anonymity between the delegator and the delegated user. Note that if in our credential scheme we give the extraction key for the commitments to a tracing authority, and if we define a *proxy signing algorithm* which works like delegation but produces a committed signature on a *clear* message rather than a committed user key, we get an instantiation of anonymous proxy signatures with mutually anonymous delegation, a feature not considered so far.

**Overview.** We start with giving an overview of our notation. In Sect. 3, we define extractable commitments and randomizable witness-indistinguishable proofs for them. We also define digital signatures and discuss how they can be combined with extractable commitments and proofs to verifiably encrypted signatures. In Sect. 4 we formally define commuting signatures and give some immediate black-box results, such as blind signatures. In Sect. 5 we recall the model for delegatable credentials from [BCC+09] and describe our instantiation providing non-interactive delegation. We prove security and conclude with a comparison to the BCCKLS instantiation in Sect. 5.3. In Sect. 6, we give the instantiations of the primitives defined in Sect. 3: Groth-Sahai proofs and automorphic signatures. In Sect. 7 we state and prove 5 lemmas about properties of Groth-Sahai proofs which are used in Sect. 8, where we instantiate our commuting-signature scheme. In Sect. 9 we give a variant of the automorphic signatures from [Fuc09] which enables a more efficient instantiation of delegatable credentials. In Sect. 10 we discuss some issues of simulatability of Groth-Sahai proofs that arise when they are used to instantiate delegatable credentials satisfying a simulation-based anonymity definition. The discussion concerns the BCCKLS instantation as well as ours. Finally, in Appendix C, we give some complementary results. In particular, we describe how to extend commuting signatures when several messages are to be signed at once.

## 2 Notation

Since we are going to combine quite a few concepts we give a guideline on notation. We tried to stick to our framework, but deviated sometimes for the sake of consistency with other work such as Groth and Sahai's.

- Capital Roman letters denote elements of a bilinear group. *Diffie-Hellman pairs* of group elements are mostly two consecutive letters of the alphabet.
- Lower-case Roman letters denote integers. Mostly they correspond to the logarithm of the corresponding capital letter in a common basis, e.g. $M = G^m$.

- Greek letters denote the randomness used in the commitments to a group element that is denoted by the corresponding Roman letter, e.g. $\mathbf{c}_M = \mathsf{Com}(ck, M, \mu)$.
- $\mathcal{M}$, $\mathcal{V}$, $\mathcal{R}$ and $\mathcal{C}$ denote spaces for messages, values, randomness and commitments, respectively; $\mathcal{E}$ denotes a class of equations, $\mathcal{H}$ a hash function, $\mathbb{G}$ denotes a group, $\mathbb{Z}$ denotes integers and $\mathbb{N}$ non-negative integers.
- Special cases:
  - $p$ denotes a prime, $e$ denotes a bilinear map (pairing) and E denotes a generic equation;
  - $\phi$ and $\theta$ denote the components of Groth-Sahai proofs, whereas $\pi$ denotes generic proofs;
  - $\vec{\mathbf{u}}$, $\vec{\mathbf{v}}$ and their components $u_{ij}$ and $v_{ij}$ are the keys for Groth-Sahai commitments;
  - $\mathbf{c}, \mathbf{d}$ and $\mathbf{C}$ denote commitments;
  - $\mathbf{t}_T$ denotes an element from the target group $\mathbb{G}_T$;
  - $\Gamma$ and $Z$ denote matrices in with entries $\gamma_{ij}$ and $z_{ij}$ in $\mathbb{Z}_p$, respectively.
- In a bilinear group, we denote the group operation by "$\cdot$". For vectors of group elements, "$\odot$" denotes applying the group operation componentwise.
- By ":=", we denote either a definition—with the definiens on the right-hand side (RHS) and the definiendum on the left-hand side (LHS)—or an assignment of the value on the RHS to the variable on the LHS.
- "$\leftarrow$" denotes a random assignment. If the RHS is a set it denotes choosing a value from it uniformly and assigning it to the LHS. If the RHS is a probabilistic algorithm it denotes choosing its random tape uniformly and assigning the outcome to the LHS.

# 3 Preliminaries

We recall the definitions and security requirements of a number of primitives from the literature, which we will combine to a system of commuting signatures and verifiable encryption in Sect. 4.

## 3.1 Commitments

A (non-interactive) *randomizable extractable commitment scheme* $\mathbf{Com}$ is composed of the algorithms Setup, Com, RdCom, ExSetup, Extr, and WISetup, which define $\mathcal{V}$, the space of "committable" values, $\mathcal{R}$, the randomness space and $\mathcal{C}$ the space of commitments. Setup outputs a *commitment key ck*, and Com, on inputs *ck*, a message $M \in \mathcal{V}$ and randomness $\rho \in \mathcal{R}$ outputs a commitment $\mathbf{c} \in \mathcal{C}$. ExSetup outputs $(ck, ek)$, where *ck* is distributed as the output of Setup, and *ek* is the *extraction key*. We require the following:

- The scheme is *perfectly binding*, i.e., for any commitment $\mathbf{c} \in \mathcal{C}$ there exists exactly one $M \in \mathcal{V}$ s.t. $\mathbf{c} = \mathsf{Com}(ck, M, \rho)$ for some $\rho$. Moreover, $\mathsf{Extr}(ek, \mathbf{c})$ extracts that value $M$ from $\mathbf{c}$.
- The scheme is *computationally hiding*, in particular, WISetup outputs keys $ck^*$ that are computationally indistinguishable from those output by Setup, and which generate perfectly hiding commitments, i.e., for every $\mathbf{c}$ and $M \in \mathcal{V}$ there exists a $\rho \in \mathcal{R}$ s.t. $\mathbf{c} = \mathsf{Com}(ck^*, M, \rho)$.
- The scheme is *randomizable*, i.e., RdCom takes as input a commitment $\mathbf{c}$ and fresh randomness $\rho' \leftarrow \mathcal{R}$ and outputs a *randomized* commitment $\mathbf{c}'$. If $\rho'$ is chosen uniformly from $\mathcal{R}$ then $\mathbf{c}'$ is distributed as $\mathsf{Com}(ck, M, \rho)$ where $\rho$ is picked uniformly from $\mathcal{R}$.

  (In particular we have $\mathsf{RdCom}(ck, \mathsf{Com}(ck, M, \rho), \rho') = \mathsf{Com}(ck, M, \rho + \rho')$.)

A commitment scheme with the above properties is actually a lossy encryption scheme [BHY09]; in particular, it satisfies the IND-CPA definition of semantic security.[1]

[1] Consider the security game for IND-CPA for encryption schemes. The challenger creates a key pair, gives the encryption key to the adversary, who outputs two messages, gets an encryption of one of them and has to guess which one. Replacing the key by a "lossy" encryption key output by WISetup is indistinguishable; and then the encryption is independent of the message.

We write Com also when we commit to a *vector* in $\mathcal{V}^n$: if $M = (M_1, \dots, M_n)$ and $\rho = (\rho_1, \dots, \rho_n)$ then $\mathsf{Com}(ck, M, \rho) := \big(\mathsf{Com}(ck, M_1, \rho_1), \dots, \mathsf{Com}(ck, M_n, \rho_n)\big)$. Likewise, we define $\mathsf{Extr}(ck, (\mathbf{c}_1, \dots, \mathbf{c}_n)) := \big(\mathsf{Extr}(ck, \mathbf{c}_1), \dots, \mathsf{Extr}(ck, \mathbf{c}_n)\big)$.

### 3.2 Proofs for Committed Values

A *randomizable witness-indistinguishable proof system* $\mathbf{Proof}$ for a commitment scheme $\mathbf{Com}$ for a class $\mathcal{E}$ of equations E consists of the algorithms Prove, Verify and RdProof. Given values $M_1, \dots, M_n \in \mathcal{V}$ satisfying an equation $\mathrm{E} \in \mathcal{E}$, the algorithm Prove, on input *ck*, $(M_1, \dots, M_n)$ and $\rho_1, \dots, \rho_n \in \mathcal{R}$, outputs a proof $\pi$. On inputs *ck*, E, $\mathbf{c}_1, \dots, \mathbf{c}_n$ and $\pi$, Verify outputs 0 or 1, indicating acceptance or rejection of a proof. We require that the system satisfies the following:

- *Completeness.* For all $(M_1, \dots, M_n)$ satisfying E, and $\rho_1, \dots, \rho_n \in \mathcal{R}$ we have

$$\mathsf{Verify}\big(ck, \mathrm{E}, \mathsf{Com}(ck, M_1, \rho_1), \dots, \mathsf{Com}(ck, M_n, \rho_n), \mathsf{Prove}(ck, \mathrm{E}, (M_1, \rho_1), \dots, (M_n, \rho_n))\big) = 1 \ .$$

- *Soundness.* Let $(ck, ek) \leftarrow \mathsf{ExSetup}$, $\mathrm{E} \in \mathcal{E}$, and $\mathbf{c}_1, \dots, \mathbf{c}_n \in \mathcal{C}$. If $\mathsf{Verify}(ck, \mathrm{E}, \mathbf{c}_1, \dots, \mathbf{c}_n, \pi) = 1$ for some $\pi$, then letting $M_i := \mathsf{Extr}(ek, \mathbf{c}_i)$, we have that $(M_1, \dots, M_n)$ satisfy E.
- *Witness indistinguishability.* Let $ck^* \leftarrow \mathsf{WISetup}$, and $M_1, \dots, M_n, M'_1, \dots, M'_n \in \mathcal{V}$ be such that both $(M_1, \dots, M_n)$ and $(M'_1, \dots, M'_n)$ satisfy an equation E. Let $\rho_1, \dots, \rho_n, \rho'_1, \dots, \rho'_n \in \mathcal{R}$ be such that for all $i$, $\mathsf{Com}(ck^*, M_i, \rho_i) = \mathsf{Com}(ck^*, M'_i, \rho'_i)$. Then the outputs of $\mathsf{Prove}(ck^*, \mathrm{E}, (M_1, \rho_1), \dots, (M_n, \rho_n))$ and $\mathsf{Prove}(ck^*, \mathrm{E}, (M'_1, \rho'_1), \dots, (M'_n, \rho'_n))$ are equally distributed.
- *Randomizability.* Given commitments $\mathbf{c}_1, \dots, \mathbf{c}_n$, a proof $\pi$ for $(\mathbf{c}_1, \dots, \mathbf{c}_n)$ and E, and $\rho'_1, \dots, \rho'_n \in \mathcal{R}$, algorithm RdProof outputs a proof $\pi'$ for the randomized commitments $\mathbf{c}'_i := \mathsf{RdCom}(ck, \mathbf{c}_i, \rho'_i)$. For all $i$, let $M_i$ be the value committed in $\mathbf{c}_i$ and let $\rho_i$ be such that $\mathbf{c}_i = \mathsf{Com}(ck, M_i, \rho_i)$. If $\rho'_1, \dots, \rho'_n$ are chosen uniformly, then $\pi'$ and $(\mathbf{c}'_i)_{i=1}^n$ are distributed as the output of $\mathsf{Prove}\big(ck, \mathrm{E}, (M_1, \hat{\rho}_1), \dots, (M_n, \hat{\rho}_n)\big)$ and $\big(\mathsf{Com}(ck, M_i, \hat{\rho}_i)\big)_{i=1}^n$ with $\hat{\rho}_i$ chosen uniformly from $\mathcal{R}$.

If E is the conjunction of equations $\mathrm{E}_1, \dots, \mathrm{E}_k$ over variables $M_1, \dots, M_n$ then for $M = (M_1, \dots, M_n)$ and $\rho = (\rho_1, \dots, \rho_n)$ we define $\mathsf{Prove}(ck, \mathrm{E}, (M, \rho)) := \big(\mathsf{Prove}(ck, \mathrm{E}_j, (M_i, \rho_i)_{i=1}^n)\big)_{j=1}^k$. For $\pi = (\pi_1, \dots, \pi_k)$ we define $\mathsf{Ver}(ck, \mathrm{E}, (\mathbf{c}_i)_{i=1}^n, \pi) := \bigwedge_{j=1}^k \mathsf{Ver}(ck, \mathrm{E}_j, (\mathbf{c}_i)_{i=1}^n, \pi_j)$.

### 3.3 Digital Signatures

A digital signature scheme $\mathbf{Sig}$ consists of the following algorithms: $\mathsf{Setup_S}$ outputs public parameters *pp* and defines a message space $\mathcal{M}$. On input *pp*, $\mathsf{KeyGen_S}$ outputs a pair $(vk, sk)$ of verification and signing key. $\mathsf{Sign}(sk, M)$, for $M \in \mathcal{M}$ outputs a signature $\Sigma$, which is verified by $\mathsf{Ver}(vk, M, \Sigma)$. We require that $\mathbf{Sig}$ satisfies the following:

- *Strong unforgeability (under chosen message attack).* No probabilistic polynomial-time (p.p.t.) adversary, given *vk* and an oracle for adaptive signing queries on messages of its choice can output a pair $(M, \Sigma)$, s.t. $\mathsf{Ver}(vk, M, \Sigma) = 1$ and $(M, \Sigma) \neq (M_i, \Sigma_i)$ for all $i$; where $M_i$ are the queried messages and $\Sigma_i$ the oracle responses.
- *Compatibility* with $\mathbf{Com}$ and $\mathbf{Proof}$. The verification keys and signatures are composed of values in $\mathcal{V}$, the value space of $\mathbf{Com}$, and signature verification consists of checking equations from $\mathcal{E}$, the class of equations for $\mathbf{Proof}$.

For our application in Sect. 5 we require furthermore that $\mathbf{Sig}$ is *automorphic*, that is, besides being compatible, its verification keys have to lie in $\mathcal{M}$.

### 3.4 Verifiably Encrypted Signatures

The triple $(\mathbf{Com}, \mathbf{Proof}, \mathbf{Sig})$ constitutes a *verifiable encryption scheme* satisfying the definitions of Rückert and Schröder [RS09], who revisited those of Boneh et al. [BGLS03]—if a proof that a committed signature satisfies $\mathsf{Ver}(vk, M, \cdot)$ can be *simulated* (see Sect. 10). This is the case for our instantiations[2], given in Sect. 6.

**Definition 1** (Verifiably encrypted signatures (VES))**.** *A verifiably encrypted signature scheme is defined as the tuple* $(\mathsf{Kg}, \mathsf{AdjKg}, \mathsf{Sig}, \mathsf{Vf}, \mathsf{Create}, \mathsf{VesVf})$*.* $\mathsf{Kg}$ *outputs a signature key pair* $(vk, sk)$*,* $\mathsf{Sig}$ *and* $\mathsf{Vf}$ *produce and verify signatures.* $\mathsf{AdjKg}$ *outputs a key pair* $(apk, ask)$ *for the* adjudicator*.* $\mathsf{Create}(sk, apk, M)$ *returns a VES* $\omega$ *which is verified by* $\mathsf{VesVf}(apk, vk, \omega, M)$ *and* $\mathsf{Adj}(ask, apk, vk, \omega, M)$ *returns a signature* $\sigma$ *on* $M$ *under vk. The security notions from [RS09] are the following:*

- Unforgeability *means that no adversary given the public keys and access to a* $\mathsf{Create}$ *and* $\mathsf{Adj}$ *oracle can output a VES on a message* $M$ *that it has never queried to its oracles.*
- Abuse freeness *states that no malicious adjudicator provided with a* $\mathsf{Create}$ *oracle can output a valid VES for a message it never queried.*
- Extractability *means that no malicious signer that can create its own vk and has access to a* $\mathsf{Adj}$ *oracle can output a valid VES from which cannot be extracted a valid signature.*
- Opacity *means that no adversary given vk and ck and access to oracles* $\mathsf{Create}$ *and* $\mathsf{Adj}$ *for messages of its choice, can output a valid pair* $(M, \Sigma)$ *if it has never queried* $\mathsf{Adj}$ *on* $M$*.*

**A Straightforward Instantiation.** Based on $(\mathbf{Com}, \mathbf{Proof}, \mathbf{Sig})$ we instantiate a VES scheme as follows: we define the signer's key generation, the adjudicator's key generation, signing and verification as $\mathsf{Kg} := \mathsf{KeyGen}_{\mathsf{S}}$, $\mathsf{AdjKg} := \mathsf{ExSetup}$, $\mathsf{Sig} := \mathsf{Sign}$, $\mathsf{Vf} := \mathsf{Ver}$. $\mathsf{Create}(sk, ck, M)$ creates a VES on $M$ by setting $\Sigma \leftarrow \mathsf{Sign}(sk, M)$, choosing $\rho \leftarrow \mathcal{R}$ and returning $\omega := \big(\mathbf{c}_\Sigma := \mathsf{Com}(ck, \Sigma, \rho),\, \widetilde{\pi} \leftarrow \mathsf{Prove}(ck, \mathrm{E}_{\mathsf{Ver}(vk,M,\cdot)}, (\Sigma, \rho))\big)$. Verification $\mathsf{VesVf}(ck, vk, (\mathbf{c}_\Sigma, \widetilde{\pi}), M)$ is defined as $\mathsf{Verify}(ck, \mathrm{E}_{\mathsf{Ver}(vk,M,\cdot)}, \mathbf{c}_\Sigma, \widetilde{\pi})$. Finally, $\mathsf{Adj}(ek, ck, vk, \omega, M)$ checks if $\omega = (\mathbf{c}_\Sigma, \widetilde{\pi})$ is valid and if so returns $\mathsf{Extr}(ek, \mathbf{c}_\Sigma)$.

The scheme $(\mathsf{Kg}, \mathsf{AdjKg}, \mathsf{Sig}, \mathsf{Vf}, \mathsf{Create}, \mathsf{VesVf}, \mathsf{Adj})$ satisfies the security notions from Def. 1 The first three notions are reduced to unforgeability of $\mathbf{Sig}$ and soundness of $\mathbf{Proof}$ in a straightforward manner; note in particular that Groth-Sahai proofs are *perfectly* sound, so no adversary even when given the extraction key can make a proof of a false statement.

Opacity can be proved by a reduction with a security loss that is exponential in the number of $\mathsf{Adj}$ calls:[3] Let Game 0 be the original game. In Game 1 we abort if the adversary makes an $\mathsf{Adj}$ query for a message it never queried $\mathsf{Create}$ for; or if it makes an $\mathsf{Adj}$ query for $(\mathbf{c}_\Sigma, \widetilde{\pi})$ such that the committed value $\Sigma$ was never used to answer one of the $\mathsf{Create}$ queries. By strong unforgeability of $\mathbf{Sig}$ and soundness of $\mathbf{Proof}$, the probability of aborting is negligible. In Game 2, when queried $\mathcal{O}_{\mathsf{Adj}(ek,ck,vk,\cdot,\cdot)}((\mathbf{c}_\Sigma, \widetilde{\pi}), M)$, instead of extracting the signature committed to in $\mathbf{c}_\Sigma$, we return the signature produced when answering $\mathcal{O}_{\mathsf{Create}(sk,ck,\cdot)}(M)$; if there have been more such calls then we guess randomly. (If, in the worst case, the adversary queries $\mathsf{Create}$ $q_C$ times and $\mathsf{Adj}$ $q_A$ times on the *same* message then the probability of correctly simulating Game 2 is $1/q_C^{q_A}$.) Game 3 is Game 2 but replacing $ck$ by $ck^*$ output by $\mathsf{WISetup}$. Game 3 can now be simulated by a challenger playing the unforgeability game against $\mathbf{Sig}$: Given the trapdoor for Groth-Sahai proofs in the WI setting, the challenger *simulates* the commitments and proofs to answer $\mathsf{Create}$ queries, i.e., without using signatures. When queried $\mathsf{Adj}$ on a message $M$, it asks its own $\mathsf{Sign}$ oracle for a signature on $M$ and returns it (or returns a signature it had already returned, depending on the guess). A successful adversary outputs a signature on $M$ for which it has never queried $\mathsf{Adj}$ (and thus never made the challenger query $\mathsf{Sign}$ for it) and can therefore be used to break strong unforgeability of $\mathbf{Sig}$.

**A Fully Secure Instantiation.** The security reduction for opacity can be made tight if outputs of $\mathsf{Create}$ are *non-malleable* (i.e., from an $\omega$ returned by $\mathsf{Create}$ on $M$, one cannot produce a different valid $\omega'$ for $M$): the

[2] Groth-Sahai proofs for pairing-product equations can be simulated if the equations only contain elements from $\mathbb{G}_1$ and $\mathbb{G}_2$ but not from $\mathbb{G}_T$. This is the case for the equations in (9), which constitute $\mathsf{Ver}$ of our instantiation.

[3] If the adversary is only allowed a *constant* number of $\mathsf{Adj}$ queries, this suffices (see also Remark 1).

adversary can then make $\mathsf{Adj}$ queries only for $\omega$'s received from a $\mathsf{Create}$ query, and in the reduction the challenger need not guess the correct signature for a certain message. Non-malleability can be achieved by replacing $\omega$ by $(\omega, \mathsf{Sign}(sk, \omega))$ in the definition of $\mathsf{Create}$ and adding a check of the second component to $\mathsf{VesVf}$. (This relies on the fact that $\mathbf{Sig}$ is strongly unforgeable.)

**Remark 1.** In our application to delegatable credentials, we require somewhat different properties from the triple $(\mathbf{Com}, \mathbf{Proof}, \mathbf{Sig})$. On the one hand, we do not require opacity, since no adversary can query *extraction* of committed values—the exponential reduction of the straightforward instantiation is thus irrelevant.

On the other hand, we require that the verification keys are in $\mathcal{M}$ (i.e., $\mathbf{Sig}$ is automorphic) and that we can commit to messages and verification keys (for which we will introduce a commitment scheme $\mathbf{Com}_\mathcal{M}$) and prove validity of an encrypted signature, possibly on an encrypted message or under an encrypted key; we also require that two verifiably encrypted signatures are *indistinguishable*. The last property is implied by witness-indistinguishability of $\mathbf{Proof}$, and is not required for VES.

# 4 Commuting Signatures and Verifiable Encryption

## 4.1 Definition

Recall the primitives introduced in Sect. 3. Let $\mathbf{Com} = (\mathsf{Setup}, \mathsf{Com}, \mathsf{RdCom}, \mathsf{ExSetup}, \mathsf{Extr}, \mathsf{WISetup})$ be an extractable commitment scheme with value space $\mathcal{V}$; let $\mathbf{Proof} = (\mathsf{Prove}, \mathsf{Verify}, \mathsf{RdProof})$ be a randomizable WI proof system for $\mathbf{Com}$; let $\mathbf{Sig} = (\mathsf{Setup}_\mathsf{S}, \mathsf{KeyGen}_\mathsf{S}, \mathsf{Sign}, \mathsf{Ver})$ be an strongly unforgeable signature scheme with message space $\mathcal{M}$ that is *compatible* with $(\mathbf{Com}, \mathbf{Proof})$. We extend $(\mathbf{Com}, \mathbf{Proof}, \mathbf{Sig})$ by the following functionalities presented in the introduction and formally defined in Def. 2:

- $\mathbf{Com}_\mathcal{M}$ is a randomizable extractable commitment scheme whose message space is that of $\mathbf{Sig}$.
- $\mathsf{AdPrC}$ and $\mathsf{AdPrC}_\mathcal{M}$ are algorithms that, given a message/signature pair of which one is verifiably encrypted, produce a proof of validity when both are encrypted.
- $\mathsf{AdPrDC}$ and $\mathsf{AdPrDC}_\mathcal{M}$ are algorithms that, given a verifiably encrypted message/signature pair and the randomness for one of them, return an adapted proof; in particular, $\mathsf{AdPrDC}$ returns a proof that a signature is valid on a committed message and $\mathsf{AdPrDC}_\mathcal{M}$ returns a proof that a committed signature is valid on a given message.
- $\mathsf{SigCom}$ takes a $\mathbf{Com}_\mathcal{M}$ commitment and a signing key, and produces a verifiably encrypted signature on the committed value.
- $\mathsf{AdPrC}_\mathcal{K}$ is given a proof of validity for a committed signature and a committed message and adapts it to a proof for when the verification key is also committed. $\mathsf{AdPrDC}_\mathcal{K}$, given the randomness of the committed verification key adapts a proof to when the key is given in the clear.

**Definition 2.** *A system of* **commuting signatures and verifiable encryption** *consists of an extractable commitment scheme* $\mathbf{Com}$*, a (randomizable) WI proof system* $\mathbf{Proof}$ *for* $\mathbf{Com}$*, a compatible signature scheme* $\mathbf{Sig}$ *and the functionalities* $\mathbf{Com}_\mathcal{M}$, $\mathsf{AdPrC}$, $\mathsf{AdPrDC}$, $\mathsf{AdPrC}_\mathcal{M}$, $\mathsf{AdPrDC}_\mathcal{M}$, $\mathsf{AdPrC}_\mathcal{K}$, $\mathsf{AdPrDC}_\mathcal{K}$, $\mathsf{SigCom}$*, and* $\mathsf{SmSigCom}$ *defined below.*

$\mathbf{Com}_\mathcal{M}$ *On input $pp = (ck, pp_\mathsf{S})$ returned by* $\mathsf{Setup}$ *and* $\mathsf{Setup}_\mathsf{S}$*, respectively, a message $M \in \mathcal{M}$ and $\mu \in \mathcal{R}_\mathcal{M}$, algorithm* $\mathsf{Com}_\mathcal{M}$ *outputs a commitment $\mathbf{C} \in \mathcal{C}_\mathcal{M}$, the space of commitments.* $\mathsf{RdCom}_\mathcal{M}$ *takes inputs $\mathbf{C}$, $\mu' \leftarrow \mathcal{R}_\mathcal{M}$ and outputs a randomized commitment $\mathbf{C}'$. On input ek output by* $\mathsf{ExSetup}$*, and $\mathbf{C}$,* $\mathsf{Extr}_\mathcal{M}$ *outputs the committed value $M$.*

*We require that $\mathbf{Com}_\mathcal{M} := (\mathsf{Setup}, \mathsf{Com}_\mathcal{M}, \mathsf{RdCom}_\mathcal{M}, \mathsf{ExSetup}, \mathsf{Extr}_\mathcal{M}, \mathsf{WISetup})$ is a randomizable extractable commitment scheme that is perfectly binding and computationally hiding as defined in Sect. 3.1. Moreover, it must be* compatible *with* $\mathbf{Proof}$*, i.e., $\mathcal{M} \subseteq \mathcal{V}$,* $\mathsf{Prove}$ *and* $\mathsf{RdProof}$ *accept inputs from $\mathcal{R}_\mathcal{M}$ and* $\mathsf{Verify}$ *accepts $\mathbf{Com}_\mathcal{M}$ commitments as inputs.*

*In the following we assume that ck $\leftarrow$* $\mathsf{Setup}$*, $pp_\mathsf{S} \leftarrow \mathsf{Setup}_\mathsf{S}$, $(vk, sk) \leftarrow \mathsf{KeyGen}_\mathsf{S}(pps)$, $M \in \mathcal{M}$ and $\mu \in \mathcal{R}_\mathcal{M}$.*

$\mathsf{AdPrC}(ck, vk, \mathbf{C}, (\Sigma, \rho), \bar{\pi})$. *If* $\mathsf{Verify}(ck, \mathrm{E}_{\mathsf{Ver}(vk,\cdot,\Sigma)}, \mathbf{C}, \bar{\pi}) = 1$ *then the algorithm outputs* $\pi$ *that is distributed as*

$$\left[\mathsf{Prove}(ck, \mathrm{E}_{\mathsf{Ver}(vk,\cdot,\cdot)}, (M, \mu), (\Sigma, \rho))\right] ,$$

*where* $M$ *and* $\mu$ *are such that* $\mathbf{C} = \mathsf{Com}_{\mathcal{M}}(ck, M, \mu)$.

$\mathsf{AdPrDC}(ck, vk, \mathbf{C}, (\Sigma, \rho), \pi)$. *If* $\mathsf{Verify}(ck, \mathrm{E}_{\mathsf{Ver}(vk,\cdot,\cdot)}, \mathbf{C}, \mathsf{Com}(ck, \Sigma, \rho), \pi) = 1$*, the algorithm outputs* $\bar{\pi}$ *which is distributed as*

$$\left[\mathsf{Prove}(ck, \mathrm{E}_{\mathsf{Ver}(vk,\cdot,\Sigma)}, (M, \mu))\right] ,$$

*where* $M$ *and* $\mu$ *are such that* $\mathbf{C} = \mathsf{Com}_{\mathcal{M}}(ck, M, \mu)$.

$\mathsf{AdPrC}_{\mathcal{M}}(ck, vk, (M, \mu), \mathbf{c}_{\Sigma}, \widetilde{\pi})$. *If* $\mathsf{Verify}(ck, \mathrm{E}_{\mathsf{Ver}(vk,M,\cdot)}, \mathbf{c}_{\Sigma}, \widetilde{\pi}) = 1$ *then it outputs* $\pi$ *which is distributed as*

$$\left[\mathsf{Prove}(ck, \mathrm{E}_{\mathsf{Ver}(vk,\cdot,\cdot)}, (M, \mu), (\Sigma, \rho))\right] ,$$

*where* $\Sigma$ *and* $\rho$ *are such that* $\mathbf{c}_{\Sigma} = \mathsf{Com}(ck, \Sigma, \rho)$.

$\mathsf{AdPrDC}_{\mathcal{M}}(ck, vk, (M, \mu), \mathbf{c}_{\Sigma}, \pi)$. *If* $\mathsf{Verify}(ck, \mathrm{E}_{\mathsf{Ver}(vk,\cdot,\cdot)}, \mathsf{Com}_{\mathcal{M}}(ck, M, \mu), \mathbf{c}_{\Sigma}, \pi) = 1$*, the algorithm outputs* $\widetilde{\pi}$ *which is distributed as*

$$\left[\mathsf{Prove}(ck, \mathrm{E}_{\mathsf{Ver}(vk,M,\cdot)}, (\Sigma, \rho))\right] ,$$

*where* $\Sigma$ *and* $\rho$ *are such that* $\mathbf{c}_{\Sigma} = \mathsf{Com}(ck, \Sigma, \rho)$.

$\mathsf{AdPrC}_{\mathcal{K}}(ck, (vk, \xi), \mathbf{C}, \mathbf{c}_{\Sigma}, \pi)$. *If* $\mathsf{Verify}(ck, \mathrm{E}_{\mathsf{Ver}(vk,\cdot,\cdot)}, \mathbf{C}, \mathbf{c}_{\Sigma}, \pi) = 1$*, the algorithm outputs* $\widehat{\pi}$ *which is distributed as*

$$\left[\mathsf{Prove}(ck, \mathrm{E}_{\mathsf{Ver}(\cdot,\cdot,\cdot)}, (vk, \xi), (M, \mu), (\Sigma, \rho)))\right] ,$$

*where* $M, \mu, \Sigma$ *and* $\rho$ *are such that* $\mathbf{C} = \mathsf{Com}_{\mathcal{M}}(ck, M, \mu)$ *and* $\mathbf{c}_{\Sigma} = \mathsf{Com}(ck, \Sigma, \rho)$.

$\mathsf{AdPrDC}_{\mathcal{K}}(ck, (vk, \xi), \mathbf{C}, \mathbf{c}_{\Sigma}, \widehat{\pi})$. *If* $\mathsf{Verify}(ck, \mathrm{E}_{\mathsf{Ver}(\cdot,\cdot,\cdot)}, \mathsf{Com}(ck, vk, \xi), \mathbf{C}, \mathbf{c}_{\Sigma}, \widehat{\pi}) = 1$*, the algorithm outputs* $\pi$ *which is distributed as*

$$\left[\mathsf{Prove}(ck, \mathrm{E}_{\mathsf{Ver}(vk,\cdot,\cdot)}, (M, \mu), (\Sigma, \rho)))\right] ,$$

*where* $M, \mu, \Sigma$ *and* $\rho$ *are such that* $\mathbf{C} = \mathsf{Com}_{\mathcal{M}}(ck, M, \mu)$ *and* $\mathbf{c}_{\Sigma} = \mathsf{Com}(ck, \Sigma, \rho)$.

$\mathsf{SigCom}(ck, sk, \mathbf{C})$. *If* $\mathbf{C} \in \mathcal{C}_{\mathcal{M}}$ *then the algorithm outputs a commitment to a signature and a proof of validity* $(\mathbf{c}_{\Sigma}, \pi)$ *which is distributed as*

$$\left[\Sigma \leftarrow \mathsf{Sign}(sk, M);\ \rho \leftarrow \mathcal{R}\ :\ \left(\mathsf{Com}(ck, \Sigma, \rho),\ \mathsf{Prove}(ck, \mathrm{E}_{\mathsf{Ver}(vk,\cdot,\cdot)}, (M, \mu), (\Sigma, \rho))\right)\right] ,$$

*where* $M$ *and* $\mu$ *are such that* $\mathbf{C} = \mathsf{Com}_{\mathcal{M}}(ck, M, \mu)$.

$\mathsf{SmSigCom}(ck, ek, vk, \mathbf{C}, \Sigma)$. *Assume* $(ck, ek) \leftarrow \mathsf{ExSetup}$. *If* $\mathsf{Ver}(vk, \mathsf{Extr}_{\mathcal{M}}(ek, \mathbf{C}), \Sigma) = 1$ *then the algorithm outputs* $(\mathbf{c}_{\Sigma}, \pi)$ *which is distributed as*

$$\left[\rho \leftarrow \mathcal{R}\ :\ \left(\mathsf{Com}(ck, \Sigma, \rho),\ \mathsf{Prove}(ck, \mathrm{E}_{\mathsf{Ver}(vk,\cdot,\cdot)}, (M, \mu), (\Sigma, \rho))\right)\right] ,$$

*where* $M$ *and* $\mu$ *are such that* $\mathbf{C} = \mathsf{Com}_{\mathcal{M}}(ck, M, \mu)$. *([4])*

**Remark 2.** When we verify a signature $\Sigma$ on a message $M$ running $\mathsf{Ver}(vk, M, \Sigma)$, we implicitly assume that Verify also checks whether $M \in \mathcal{M}$. Analogously, we assume that when verifying a proof of validity by running Verify on $\mathrm{E}_{\mathsf{Ver}}$ and $\mathbf{C}$, it checks whether $\mathbf{C} \in \mathcal{C}_{\mathcal{M}}$, too.

Def. 2 implies that running $\mathsf{Com}_{\mathcal{M}}$ on $M$ and then SigCom yields the same output as running $\Sigma \leftarrow \mathsf{Sign}(sk, M)$ and then $\mathsf{Com}_{\mathcal{M}}$ on $M$, Com on $\Sigma$ and Prove for $\mathrm{E}_{\mathsf{Ver}(vk,\cdot,\cdot)}$; or running Sign, then $\mathsf{Com}_{\mathcal{M}}$ on $M$ and Prove for $\mathrm{E}_{\mathsf{Ver}(vk,\cdot,\Sigma)}$, and then Com on $\Sigma$ and AdPrC; or running Sign, then Com on $\Sigma$ and Prove for $\mathrm{E}_{\mathsf{Ver}(vk,M,\cdot)}$, and then $\mathsf{Com}_{\mathcal{M}}$ on $M$ and $\mathsf{AdPrC}_{\mathcal{M}}$. And similar statements hold for sequences of algorithm executions including decommitments and proof adaptation. This means that the diagram in Fig. 1 *commutes*.

[4]Note that SmSigCom is not trivial: given *ek* it might recover the message $M$ but not the randomness $\mu$ used for $\mathbf{C}$. The difference to AdPrC is that SmSigCom does not get $\bar{\pi}$ as input but *ek* instead.

### 4.2 Black-Box Results

Security notions for commuting signatures follow from the security of the used building blocks and the fact that all algorithms perfectly commute.

**Unforgeability.** Extractability of $\mathbf{Com}$ and $\mathbf{Com}_{\mathcal{M}}$, perfect soundness of $\mathbf{Proof}$, strong unforgeability of $\mathbf{Sig}$ and commutativity of SigCom with signing and verifiably encrypting implies unforgeability, defined as the intractability for a p.p.t. adversary $\mathcal{A}$ of winning the following game:

> Run $(ck, ek) \leftarrow \mathsf{ExSetup}$ and $(vk, sk) \leftarrow \mathsf{KeyGen}_{\mathsf{S}}(\mathsf{Setup}_{\mathsf{S}})$; provide $\mathcal{A}$ with $(ck, ek, vk)$ and access to a SigCom oracle that on input $\mathbf{C}$, a commitment to a message, outputs $\mathsf{SigCom}(ck, sk, \mathbf{C})$. Let $\mathbf{C}_i$ be the value submitted in the $i$-th oracle call and $(\mathbf{c}_i, \pi_i)$ be the response; define $M_i := \mathsf{Extr}_{\mathcal{M}}(ek, \mathbf{C}_i)$ and $\Sigma_i := \mathsf{Extr}(ek, \mathbf{c}_i)$. Then $\mathcal{A}$ wins if it outputs $(\mathbf{C}^*, \mathbf{c}^*, \pi^*)$ such that $\mathsf{Verify}(ck, \mathrm{E}_{\mathsf{Ver}(vk,\cdot,\cdot)}, \mathbf{C}^*, \mathbf{c}^*, \pi^*) = 1$ and $\big(\mathsf{Extr}_{\mathcal{M}}(ek, \mathbf{C}^*), \mathsf{Extr}(ek, \mathbf{c}^*)\big) \notin \big\{(M_1, \Sigma_1), \ldots, (M_n, \Sigma_n)\big\}$.

This unforgeability notion is reduced to strong unforgeability of $\mathbf{Sig}$. On receiving $vk$, run $(ck, ek) \leftarrow \mathsf{ExSetup}$ and give $(ck, ek, vk)$ to the adversary. Answer a query for $\mathbf{C}$ as follows: using $ek$, extract $M$, query it to the signing oracle to receive $\Sigma$; then run $(\mathbf{c}_\Sigma, \pi) \leftarrow \mathsf{SmSigCom}(ck, ek, vk, \mathbf{C}, \Sigma)$ and return $(\mathbf{c}_\Sigma, \pi)$. Since by Def. 2, SmSigCom and SigCom both commute with $\mathsf{Com}_{\mathcal{M}}$, Com and Prove, this perfectly simulates the adversary's oracle. If the adversary wins the game, we return $\mathsf{Extr}_{\mathcal{M}}(ek, \mathbf{C}^*)$ and $\mathsf{Extr}(ek, \mathbf{c}^*)$ which yields a valid forgery $(M, \Sigma)$ by perfect soundness of $\mathbf{Proof}$.

**Indistinguishability.** The message in $\mathbf{C}$ remains hidden to a signer running SigCom, in a computational sense: replacing $ck$ by $ck^* \leftarrow \mathsf{WISetup}$ is computationally indistinguishable and results in perfectly hiding outputs of Com and $\mathsf{Com}_{\mathcal{M}}$.

**Blind Signatures.** Given a system of commuting signatures and verifiable encryption, we can easily build a blind-signature scheme in a black-box way. To get a signature on a message $M$, the user chooses $\mu \leftarrow \mathcal{R}_{\mathcal{M}}$ and sends the commitment $\mathbf{C} := \mathsf{Com}_{\mathcal{M}}(ck, M, \mu)$ to the signer. The latter uses SigCom to produce and send $(\mathbf{c}_\Sigma, \pi)$, a committed signature on $M$ and a proof of validity. The user can produce a proof $\widetilde{\pi} \leftarrow \mathsf{AdPrDC}_{\mathcal{M}}(ck, vk, (M, \mu), \mathbf{c}_\Sigma, \pi)$, which asserts validity of the committed signature on $M$. The blind signature is defined as $(\mathbf{c}_\Sigma, \widetilde{\pi})$ and is verified by $\mathsf{Verify}(ck, \mathrm{E}_{\mathsf{Ver}(vk,M,\cdot)}, \mathbf{c}_\Sigma, \widetilde{\pi})$.

This yields a generically more efficient construction than that from [Fis06], in which, besides $\mathbf{c}_\Sigma$ and $\pi$, the blind signature contains $\mathbf{C}$ and a proof that $\mathbf{C}$ opens to $M$.

## 5 Application: Non-Interactively Delegatable Anonymous Credentials.

Our main application of commuting signatures and verifiable encryption is a black-box construction of a delegatable anonymous credential scheme with a non-interactive delegation protocol. Our scheme borrows the idea of combining Groth-Sahai proofs and automorphic signatures from the instantiation of *anonymous proxy signatures* from [Fuc09]. This primitive is similar in that it enables to prove knowledge of a certification chain, but there is no mutual anonymity of the users in the (non-interactive) delegation protocol. Commuting signatures now allow to define a delegation protocol where both delegator and delegatee remain anonymous w.r.t. each other. Moreover, the protocol is *non-interactive*, that is, a user can publish a pseudonym, which can be used by the delegator to produce a credential for the user—as it would be in the non-anonymous case with public keys instead of pseudonyms.

We start by presenting the model for delegatable credentials defined in [BCC+09]. In Sect. 5.2 we give our instantiation of it, and compare it to that from [BCC+09] in the subsequent section.

### 5.1 The BCCKLS Model

The system parameters are set up by a trusted party. Every user holds a secret key *sk*, of which she can publish *pseudonyms Nym*. Any user can be an *originator* of a credential by publishing a pseudonym $Nym_O$ as the public key. If user $A$ was issued a credential for pseudonym $Nym_A$, she can transform it into a credential for any other pseudonym $Nym'_A$. Moreover, credentials can be *delegated* to other users. A (non-interactively) delegatable anonymous credential system consists of the following algorithms:[5]

$\mathsf{Setup}_{\mathsf{C}}(1^{\lambda})$ outputs the system parameters $pp$

$\mathsf{KeyGen}_{\mathsf{C}}(pp)$ creates a user secret key $sk$

$\mathsf{NymGen}(pp, sk)$ outputs a new pseudonym $Nym$ and auxiliary information $aux$ related to $Nym$

$\mathsf{Issue}(pp, Nym_O, sk_I, Nym_I, aux_I, cred, Nym_U, L):$ $sk_I, Nym_I$ and $aux_I$ are the issuer's secret key, pseudonym and auxiliary information. $cred$ is a level $L$ credential for the issuer rooted at $Nym_O$, and $Nym_U$ is the pseudonym of the delegated user. If $L = 0$ then $cred = \emptyset$. The algorithm outputs $credproof$.

$\mathsf{Obtain}(pp, Nym_O, sk_U, Nym_U, aux_U, Nym_I, L, credproof):$ $sk_U, Nym_U$ and $aux_U$ are the user's secret key, pseudonym and auxiliary information. $Nym_O$ and $Nym_I$ are the originator's and the issuer's pseudonym, and $credproof$ is the output of Issue. The algorithm outputs a credential $cred$.

$\mathsf{CredProve}(pp, Nym_O, cred, sk, Nym, aux, L)$ takes a level $L$ credential from $Nym_O$, and $sk, Nym$ and $aux$, and outputs a $credproof$ for $Nym$.

$\mathsf{CredVerify}(pp, Nym_O, credproof, Nym, L)$ verifies a level $L$ $credproof$ for a pseudonym $Nym$ rooted at $Nym_O$.

Security is defined by *correctness*, *anonymity* and *unforgeability*, which we sketch below. For a formal security definition we refer to Appendix A of the full version of [BCC+09].

**Correctness.** A credential is *proper* if for all user pseudonyms, CredProve outputs a proof that is accepted by CredVerify. Run honestly, Issue and Obtain must produce a proper credential.

**Anonymity.** There exists a simulator $(\mathsf{SimSetup}_{\mathsf{C}}, \mathsf{SimCredProve}, \mathsf{SimIssue}, \mathsf{SimObtain})$ with the following properties: $\mathsf{SimSetup}_{\mathsf{C}}$ outputs parameters that are indistinguishable from those produced by Setup and a trapdoor $sim$. Under theses parameters the outputs $Nym$ of NymGen are distributed independently of $sk$.

SimCredProve gets $sim$ instead of $cred$, $sk$ and $aux$ and outputs $credproof$ that is indistinguishable from outputs of CredProve. SimIssue has input $sim$ instead of $sk_I, aux_I$ and $cred$ and cannot be distinguished from Issue by an adversary interacting with it. SimObtain gets $sim$ instead of $sk_U$ and $aux_U$ and cannot be distinguished from Obtain by an adversary interacting with it.

Note that for the case of *non-interactive* delegation, this means: SimIssue produces $credproof$ that is indistinguishable from outputs of Issue. And SimObtain is obsolete, since the issuer does not *interact* with it.

**Unforgeability.** (I) There exists $\mathsf{ExSetup}_{\mathsf{C}}$ that outputs parameters $pp$ (distributed as those from $\mathsf{Setup}_{\mathsf{C}}$) and an extraction key $ek$. Under $pp$ pseudonyms are perfectly binding for $sk$ and $\mathsf{Extract}_{\mathsf{C}}$ using $ek$ outputs the chain of $L$ identities from a level $L$ $credproof$.

(II) No adversary $\mathcal{A}$ can output a valid credential from which can be extracted an *unauthorized* chain of identities. This means that $\mathcal{A}$ is given the parameters and has oracles to add honest users, request pseudonyms from them, request issuings between honest users, request proofs and it can run Issue and Obtain with the simulator. When $\mathcal{A}$ requests Issue for $(Nym_O, Nym_I, Nym_U, cred, L)$, the simulator extracts $vk_O, vk_I, vk_U$ from the pseudonyms and adds $(vk_O, L+1, vk_I, vk_U)$ to a list ValidCredentialChains. The adversary wins if it outputs a valid triple $(Nym_O, credproof, Nym)$, from which can be extracted $(vk_0, \ldots, vk_L)$ s.t. $(vk_0, i, vk_{i-1}, vk_i) \notin$ ValidCredentialChains for some $i$ and $vk_{i-1}$ is an honest user key.

### 5.2 Our Instantiation

In the instantiation from [BCC+09] the system parameters are a Groth-Sahai (GS) commitment key and parameters for an authentication scheme. Each user holds a secret key $x$ for the authentication scheme. A pseudonym is made up of two GS commitments to $H^x$ and $U^x$ (from which $x$ cannot be extracted), respectively, for parameters $H$ and $U$. To issue and delegate, the issuer and the user jointly compute a proof of knowledge of an *authenticator* on the user's secret key, which is valid under the issuer's secret key. The authors define a complex interactive two-party

[5] Since, as opposed to [BCC+09], we consider non-interactive delegation, Issue and Obtain are *non-interactive* algorithms; the output of Issue is *credproof* which is an additional input for Obtain.

protocol for this. A credential is then a chain of pseudonyms and committed authenticators with GS proofs of validity.

We replace the authentication scheme by an automorphic signature scheme. A non-anonymous credential for $vk_L$ rooted at $vk_0$ is a chain of public keys and signatures $(\Sigma_1, vk_1, \Sigma_2 \ldots, vk_{L-1}, \Sigma_L)$, where $\Sigma_i$ is a signature on $vk_i$ under $vk_{i-1}$. To achieve anonymity, the public keys and signatures in the credential are committed to and proofs of validity are added. Using commuting signatures, given a commitment to a public key, the issuer can directly make a commitment to a signature on it and a validity proof. This is what enables non-interactive delegation.

**Commuting Signatures with Partially Public Messages.** To instantiate credentials, merely signing user public keys does not suffice. The issuer of a credential might want to add public information to the credential, such as attributes. For delegatable credentials it is also required to include the originator's pseudonym and the delegation level in each certificate to prevent combining different credentials and changing the order within a credential.

In Sect. 9.1, we give an automorphic signature scheme $\mathbf{Sig}''$, where in addition to the message, the signer can specify some public value. The message space of $\mathbf{Sig}''$ is $\mathbb{Z}_p \times \mathcal{M}$. Our scheme only extends the parameters of *pp* by one group element, but is otherwise as efficient as $\mathbf{Sig}$, in particular, $\mathbf{Sig}$- and $\mathbf{Sig}''$ signatures have the same size. We also define $\mathsf{VK}$ that on input a signing key outputs the corresponding verification key, which allows us to comply with the formal definition of credentials in [BCC+09]. In Sect. 9.2, we define $\mathsf{SigCom}''$, which is $\mathsf{SigCom}$ adapted to $\mathbf{Sig}''$ and thus has the public part of the message as additional input. Moreover, we show that all other algorithms defined in Def. 2 and instantiated in Sect. 8 work equally for $\mathbf{Sig}$ and $\mathbf{Sig}''$.

For our instantiation, we assume a collision-resistant hash function $\mathcal{H}\colon \mathcal{C}_\mathcal{M} \times \mathbb{N} \to \mathbb{Z}_p$.

**More Intuition.** We informally describe how our algorithms work. $\mathsf{Setup}_\mathrm{C}$ generates a key for $\mathbf{Com}$ and parameters for $\mathbf{Sig}''$, $\mathsf{KeyGen}_\mathrm{C}$ outputs a secret key for $\mathbf{Sig}''$, and given a secret key, $\mathsf{NymGen}$ outputs a commitment to the corresponding public key and the used randomness as auxiliary information. A level $L$ credential from $Nym_0$ to $Nym_L$ has the form

$$cred = (\mathbf{c}_1, \pi_1, Nym_1, \mathbf{c}_2, \pi_2, \ldots, Nym_{L-1}, \mathbf{c}_L, \pi_L) \ , \tag{1}$$

where $\mathbf{c}_i$ is a commitment to a signature $\Sigma_i$ on the public value $\mathcal{H}(Nym_0, i)$ and the key committed in $Nym_i$, valid under the key committed in $Nym_{i-1}$; and $\pi_i$ is a proof of validity of $\Sigma_i$. We call it a *credential* if it is valid on a *trivial* $Nym_L$, i.e., when $Nym_L = \mathsf{Com}(ck, vk_L, 0)$, and speak of a *credential proof* otherwise.

$\mathsf{CredProve}$ takes a credential and turns it into a credential proof by randomizing all its components, using *aux* s.t. $Nym_L = \mathsf{Com}(ck, \mathsf{VK}(sk), aux)$ for the last component. $\mathsf{CredVerify}$ verifies a *credproof* by checking the proofs contained in it. Given a level $L$ credential, $\mathsf{Issue}$ extends it by one level to a credential for $Nym_{L+1}$: if it is not an original issuing, it first makes a *credproof* for the issuer's pseudonym $Nym_I$; using $\mathsf{SigCom}''$ it produces $(\mathbf{c}_{L+1}, \pi_{L+1})$ for $Nym_{L+1}$, and turns the proof into a proof for the committed verification key $Nym_I$ by running $\mathsf{AdPrC}_\mathcal{K}$ on randomness $aux_I$. $\mathsf{Obtain}$ turns this *credproof* into a *cred* by adapting the randomness to make it valid for a trivial $Nym_L$.

**Algorithm Specification.** We now formally define the algorithms of our scheme $\mathbf{Cred}$.

$\mathsf{Setup}_\mathrm{C}(1^\lambda)$. Run $ck \leftarrow \mathsf{Setup}$; $pp_\mathrm{S} \leftarrow \mathsf{Setup}''_\mathrm{S}$; return $pp := (ck, pp_\mathrm{S})$

$\mathsf{KeyGen}_\mathrm{C}(pp)$. Parse $pp \rightsquigarrow (ck, pp_\mathrm{S})$; run $(vk, sk) \leftarrow \mathsf{KeyGen}''_\mathrm{S}(pp_\mathrm{S})$; return $sk$

$\mathsf{NymGen}(pp, sk)$. Choose $aux \leftarrow \mathcal{R}_\mathcal{M}$; return $\big(Nym := \mathsf{Com}_\mathcal{M}(pp, \mathsf{VK}(sk), aux),\ aux\big)$

$\mathsf{Issue}(pp, Nym_O, sk_I, Nym_I, aux_I, cred, Nym_U, L)$.

- If $L = 0$, and $cred \neq \emptyset$ or $Nym_O \neq Nym_I$ then abort;
  if $L > 0$ then set $credproof \leftarrow \mathsf{CredProve}(pp, Nym_O, cred, sk_I, Nym_I, aux_I, L)$ and abort if it fails
- Parse *cred* as in (1); if $Nym_L := Nym_I \neq \mathsf{Com}_\mathcal{M}(pp, \mathsf{VK}(sk_I), aux)$ or $Nym_U \notin \mathcal{C}_\mathcal{M}(pp)$ then abort
- $(\mathbf{c}_{L+1}, \pi_{L+1}) \leftarrow \mathsf{SigCom}''(pp, sk, \mathcal{H}(Nym_O, L+1), Nym_U)$
  $\pi'_{L+1} \leftarrow \mathsf{AdPrC}_\mathcal{K}\big(pp, (\mathsf{VK}(sk_I), aux_I), Nym_U, \mathbf{c}_{L+1}, \pi_{L+1}\big)$
- Return $credproof \,\|\, (Nym_I, \mathbf{c}_{L+1}, \pi'_{L+1})$

$\mathsf{Obtain}(pp, Nym_O, sk_U, Nym_U, aux_U, Nym_I, L;\ credproof')$.

- Parse $pp \rightsquigarrow (ck, pp_S)$; parse $credproof' \rightsquigarrow credproof_L' \,\|\, (Nym_I', \mathbf{c}_{L+1}', \pi_{L+1}')$. If $Nym_I \neq Nym_I'$ or $Nym_U \neq \mathsf{Com}_{\mathcal{M}}(pp, \mathsf{VK}(sk_U), aux_U)$ or $\mathsf{CredVerify}(pp, Nym_O, credproof', Nym_U, L+1) = 0$ then abort
- $\pi_{L+1} \leftarrow \mathsf{RdProof}\big(ck, \mathrm{E}_{\mathsf{Ver}''_{pp_S}(\cdot,\mathcal{H}(Nym_O,L+1),\cdot,\cdot)}, (Nym_I, 0), (Nym_U, -aux_U), (\mathbf{c}_{L+1}, 0), \pi_{L+1}'\big)$
- Return $credproof_L' \,\|\, (Nym_I', \mathbf{c}_{L+1}', \pi_{L+1})$

$\mathsf{CredProve}(pp, Nym_O, cred, sk, Nym, aux, L)$.

- Parse $pp \rightsquigarrow (ck, pp_S)$ and $cred$ as in (1)
- If $Nym \neq \mathsf{Com}_{\mathcal{M}}(pp, \mathsf{VK}(sk), aux)$ or $\mathsf{CredVerify}\big(pp, Nym_O, cred, \mathsf{Com}_{\mathcal{M}}(pp, \mathsf{VK}(sk), 0), L\big) = 0$ then abort
- For $i = 1 \dots L$, pick $\nu_i \leftarrow \mathcal{R}_{\mathcal{M}}, \gamma_i \leftarrow \mathcal{R}$. Set $Nym_0 := Nym_O,\ \nu_0 := 0,\ Nym_L := Nym_U,\ \nu_L := aux$
- For $i = 1 \dots L$ do

$$Nym_i' := \mathsf{RdCom}_{\mathcal{M}}(pp, Nym_i, \nu_i);\ \ \mathbf{c}_i' := \mathsf{RdCom}(ck, \mathbf{c}_i, \gamma_i)$$
$$\pi_i' \leftarrow \mathsf{RdProof}\big(ck, \mathrm{E}_{\mathsf{Ver}''_{pp_S}(\cdot,\mathcal{H}(Nym_O,i),\cdot,\cdot)}, (Nym_{i-1}, \nu_{i-1}), (Nym_i, \nu_i), (\mathbf{c}_i, \gamma_i), \pi_i\big)$$

- Return $(\mathbf{c}_1', \pi_1', Nym_1', \mathbf{c}_2', \pi_2', \dots, Nym_{L-1}', \mathbf{c}_L', \pi_L')$

$\mathsf{CredVerify}(pp, Nym_O, credproof, Nym, L)$

- Parse $pp \rightsquigarrow (ck, pp_S)$, $credproof \rightsquigarrow (\mathbf{c}_1, \pi_1, Nym_1, \dots, \mathbf{c}_L, \pi_L)$, let $Nym_0 := Nym_O, Nym_L := Nym$
- If $\forall\, 1 \leq i \leq L : \mathsf{Verify}\big(ck, \mathrm{E}_{\mathsf{Ver}''_{pp_S}(\cdot,\mathcal{H}(Nym_O,i),\cdot,\cdot)}, Nym_{i-1}, Nym_i, \mathbf{c}_i, \pi_i\big) = 1$ and $Nym_i \in \mathcal{C}_{\mathcal{M}}(pp)$, return 1

**Theorem 1.** *Let* $(\mathbf{Com}, \mathbf{Proof})$ *be a randomizable, extractable, composable zero-knowledge non-interactive proof-of-knowledge system, let* $\mathbf{Sig}''$ *be a strongly unforgeable automorphic signature scheme, and let* $\mathcal{H}$ *be collision resistant. Then* $\mathbf{Cred}$ *as defined above is a secure anonymous delegatable credential scheme.*

*Proof sketch.* We refer to the full version of [BCC+09] for the formal definition of the model, which is quite involved. Since the overall construction of our scheme is similar to the BCCKLS construction, in particular the use of (randomizable and simulatable) Groth-Sahai proofs to commit to a delegation chain and prove validity, our scheme is proved to satisfy the security definitions analogously. Our proof is a lot simpler though, since our certificates are on *public keys* and one can extract a complete certification chain from our credentials, avoiding thus partial-extractability notions. Moreover, our construction does not make use of interactive secure two-party protocols. We give a sketch of the security proof, highlighting the differences.

CORRECTNESS. Correctness of our scheme follows from a straightforward argument using the correctness of the underlying building blocks.

ANONYMITY. A witness-indistinguishability based definition of anonymity is an immediate consequence of perfectly hiding commitments and proofs, when $ck$ is produced by $\mathsf{WISetup}$: pseudonyms $Nym$ information-theoretically hide the committed value and the proofs in $cred$ do not contain information either. We thus define $\mathsf{SimSetup}_{\mathrm{C}}$ using $\mathsf{WISetup}$. The algorithms $\mathsf{CredProve}$, $\mathsf{Issue}$ and $\mathsf{Obtain}$ can be simulated without knowledge of any private information; this follows from the zero-knowledge property of Groth-Sahai proofs; in particular, in the witness-indistinguishable (WI) setting, given the simulation trapdoor $sim$ of GS proofs, we can make perfectly hiding commitments and proofs for any equation with certain properties—which are satisfied by ours. (See Sect. 10.)

Our simulator $\mathsf{SimCredProve}$ is the exact analogue of $\mathsf{SimProve}$, defined in the anonymity proof of the BCCKLS scheme: it constructs a simulated certificate chain from $Nym_O$ to $Nym_I$ of length $L$, by simulating the intermediate pseudonyms, i.e., the $\mathbf{Com}_{\mathcal{M}}$ commitments (cf. Sect. 10.2), the commitments in $\mathbf{c}_i$, and the proofs $\pi_i$. $\mathsf{SimIssue}$ is defined as $\mathsf{SimCredProve}$ for $L+1$ except that it sets $Nym_L$ as $Nym_I$. Since $\mathsf{Obtain}$ does not interact with the issuer, $\mathsf{SimObtain}$ is the empty algorithm. Our proof is considerably simpler than that of [BCC+09], due to the fact that $\mathsf{Issue}$ readily outputs a credential rather than engaging in a two-party protocol with $\mathsf{Obtain}$. All we need to do is simulate GS commitments and proofs. We refer to Sect. 10 for a discussion on how to actually simulate such proofs.

UNFORGEABILITY. Soundness and extractability of GS proofs, unforgeability of $\mathbf{Sig}''$ and simulatability of $\mathsf{SigCom}''$ imply that our scheme is unforgeable in the sense of [BCC+09]: (I) For $\mathsf{ExSetup}_\mathrm{C}$, we substitute $\mathsf{Setup}$ by $\mathsf{ExSetup}$ in $\mathsf{Setup}_\mathrm{C}$. This generates an identically distributed key *ck* (which leads to perfectly binding commitments) and an extraction key *ek* that allows to extract the committed chain of public keys ("identities") and certificates from a credential.

The notion defined in (II) is reduced to unforgeability of $\mathbf{Sig}$: given a verification key *vk* and a signing oracle, we simulate the game as follows: we guess which honest user the adversary will "frame"; we compute the parameters with $\mathsf{ExSetup}_\mathrm{C}$, and use extraction, the signing oracle and $\mathsf{SmSigCom}$ to simulate $\mathsf{Issue}$ for that user. Let $(\mathit{Nym}_0, \mathit{credproof}, \mathit{Nym}_L)$ be a successful forgery, thus when we extract $(\mathit{vk}_0, \Sigma_1, \mathit{vk}_1, \ldots, \Sigma_L, \mathit{vk}_L)$ from it then for some $i$: $(\mathit{vk}_0, i, \mathit{vk}_{i-1}, \mathit{vk}_i) \notin \mathsf{ValidCredentialChains}$ and $\mathit{vk}_{i-1}$ is honest. If we guessed correctly (i.e., $\mathit{vk}_{i-1} = \mathit{vk}$) then we can return the $\mathbf{Sig}''$ forgery $(\mathit{vk}_i, \Sigma_i)$, since (by collision resistance of $\mathcal{H}$) we have never queried our signing oracle on $(\mathcal{H}(\mathit{Nym}, i), \mathit{vk}_i)$ for any *Nym* of $\mathit{vk}_0$. □

**Optimizing the Black-Box Construction for Concrete Commuting Signatures.** When using our implementation of commuting signatures (Sect. 6 and 8), we can make the following optimizations. In the instantiation we have $\mathcal{M} \subseteq \mathbb{G}_1 \times \mathbb{G}_2$ for an asymmetric bilinear group. A $\mathbf{Com}_\mathcal{M}$ commitment to a message $(M, N) \in \mathcal{M}$ is defined as $\mathbf{c}_M := \mathsf{Com}(\mathit{ck}, M, \mu)$, $\mathbf{c}_N := \mathsf{Com}(\mathit{ck}, N, \nu)$, $\pi_N \leftarrow \mathsf{Prove}(\mathit{ck}, \mathrm{E}_{\mathcal{DH}}, (M, \mu), (N, \nu))$ and additional components which enable the signer to make a committed signature on $(M, N)$ and a proof of validity by running $\mathsf{SigCom}$. These additional components are however not required for the *Nym*'s contained in the credential, where giving $(\mathbf{c}_M, \mathbf{c}_N, \pi_M)$ is sufficient. Moreover, when issuing a new credential, the "public key" used to verify it can be given in the clear, i.e., as $(X, Y) \in \mathcal{M}$, rather than as a commitment.

### 5.3 A Comparison to the BCCKLS Instantiation

The key building block of a delegatable-credential scheme is a certification scheme signing (or authenticating) user keys and some public information. The certificates (or "authenticator") on user secret keys in the BCCKLS instantiation [BCC+09] are in $\mathbb{G}_1^8 \times \mathbb{G}_2^3$ and are verified by evaluating 16 pairings. Our certificates on a user public key (and a public value) are in $\mathbb{G}_1^3 \times \mathbb{G}_2^2$ and verified by evaluating 7 pairings. Proving validity of a committed certificate requires Groth-Sahai proofs (which are in $\mathbb{G}_1^{2\times 2} \times \mathbb{G}_2^{2\times 2}$ for the SXDH instantiation) for 8 equations for the BCCKLS instantiation and 3 equations for ours. The pseudonyms in [BCC+09] consist of two Groth-Sahai commitments and one proof, and are thus in $\mathbb{G}_1^6 \times \mathbb{G}_2^6$, as are the pseudonyms contained in our credentials. The pseudonyms enabling non-interactive delegation are the size of two optimized pseudonyms and 3 $\mathbb{G}_1$ elements.[6] A tuple $(\mathbf{c}_i, \pi_i, \mathit{Nym}_i)$ contained in a warrant is thus in $\mathbb{G}_1^{50} \times \mathbb{G}_2^{40}$ for the BCCKLS instantiation, whereas it is in $\mathbb{G}_1^{20} \times \mathbb{G}_2^{18}$ for ours.[7] We conclude that the size of our credentials is less than half the size of BCCKLS credentials.

Most importantly, issuing and delegation in our scheme is substantially more efficient than in the BCCKLS scheme. In the latter the issuer and the user run a secure two-party protocol to jointly compute a proof of knowledge of an authenticator on the user's secret key. This protocol uses homomorphic encryption and interactive ZK proofs asserting that certain blinding values are in the correct ranges. Since these tools are not made explicit it is not clear how many rounds the protocol requires nor what amount of data needs to be sent in each of them. In contrast, in our instantiation the issuer simply rerandomizes his credential and runs $\mathsf{SigCom}$ and $\mathsf{AdPrC}_\mathcal{K}$. She then sends a ready credential to the user.

Concerning the assumptions on which security is based, they are both *non*-interactive, "$q$-type" assumptions and part of the generalized "Uber-Assumption" family [Boy08]. What is more, both are comparable variants of the *strong Diffie-Hellman* assumption [BB04], as was argued in [Fuc09] (cf. the discussion in Appendix C.1, ibid.).

[6] Note that if the size of pseudonyms is to be minimized, user could publish $\mathit{Nym} = (\mathbf{c}_M, \mathbf{c}_N, \pi_M)$ and send the remaining elements $(\mathbf{c}_P, \mathbf{c}_Q, \pi_P, U, \pi_{U_S})$ as a first step in $\mathsf{Obtain}$ (see Sect. 10.2).

[7] This analysis is for the case when Groth-Sahai proofs are applied in a straightforward way and not considering simulatability. To make *inhomogeneous* equations simulatable, a Groth-Sahai proof is augmented by one commitment in $\mathbb{G}_1^2$, one commitment in $\mathbb{G}_2^2$ and one proof from $\mathbb{G}_1^4 \times \mathbb{G}_2^2$ (see Sect. 10.2). Of our 3 equations only 1 is inhomogeneous, the size of a triple $(\mathbf{c}_i, \pi_i, \mathit{Nym}_i)$ is thus augmented by 6 $\mathbb{G}_1$ elements and 4 $\mathbb{G}_2$ elements. We note that the 8 equations for BCCKLS authenticators contain 3 inhomogeneous equations.

# 6 Instantiation of the Building Blocks

In this section we give instantiations of the building blocks $\mathbf{Com}$, $\mathbf{Proof}$ and $\mathbf{Sig}$ (Sect. 6.2, 6.3 and 6.4, respectively) on which we base our commuting signatures. We start with introducing bilinear groups and the assumptions under which our instantiations are secure.

## 6.1 Bilinear Groups and Assumptions

A *bilinear group* (cf. e.g. [GPS08]) is a tuple $grp = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, G_1, G_2)$ where $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ are cyclic groups of prime order $p$, $G_1$ and $G_2$ generate $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively, and $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is an efficient non-degenerate bilinear map, i.e., $\forall\, X \in \mathbb{G}_1\ \forall\, Y \in \mathbb{G}_2\ \forall\, a, b \in \mathbb{Z} : e(X^a, Y^b) = e(X, Y)^{ab}$, and $e(G_1, G_2)$ generates $\mathbb{G}_T$. We assume that there exists a probabilistic polynomial-time algorithm $\mathsf{GrpGen}$ that on input $1^\lambda$ outputs a bilinear group *grp* for which $p$ is a $\lambda$-bit prime.

**Assumption 1** (SXDH)**.** *The* Symmetric External Diffie-Hellman Assumption *states that given* $(G_1^r, G_1^s, G_1^t)$ *for random* $r, s \in \mathbb{Z}_p$*, it is hard to decide whether* $t = rs$ *or* $t$ *is random; moreover, given* $(G_2^{r'}, G_2^{s'}, G_2^{t'})$ *for random* $r', s' \in \mathbb{Z}_p$*, it is hard to decide whether* $t' = r's'$ *or* $t'$ *is random.*

The $q$-*Asymmetric Double Hidden Strong Diffie-Hellman assumption* was introduced in [Fuc09] and is a variant of $q$-HSHD [BW07] in asymmetric bilinear groups. It was shown in [FPV09] that under the $q$-SDH assumption [BB04], given $q-1$ tuples $((K \cdot G^{v_i})^{1/(x+c_i)}, c_i, v_i)$ for random $c_i, v_i \leftarrow \mathbb{Z}_p$, it is hard to produce a new tuple of this form. The assumption below states that if $c_i$ and $v_i$ are given in a *hidden* form $(F^{c_i}, H^{c_i})$ and $(G^{v_i}, H^{v_i})$, respectively, it is intractable to produce a new tuple $((K \cdot G^v)^{1/(x+c)}, F^c, H^c, G^v, H^v)$.

**Assumption 2** ($q$-ADHSDH)**.** *Given* $(G, F, K, X = G^x;\ H, Y = H^x) \leftarrow \mathbb{G}_1^4 \times \mathbb{G}_2^2$,

$$\big(A_i = (K \cdot G^{v_i})^{\frac{1}{x+c_i}},\ B_i = F^{c_i},\ V_i = G^{v_i},\ D_i = H^{c_i},\ W_i = H^{v_i}\big)_{i=1}^{q-1}, \quad \textit{for } c_i, v_i \leftarrow \mathbb{Z}_p,$$

*it is hard to output a new tuple* $(A, B, V, D, W) \in \mathbb{G}_1^3 \times \mathbb{G}_2^2$ *of this form, i.e., a tuple that satisfies*

$$e(A, Y \cdot D) = e(K \cdot V, H) \qquad e(B, H) = e(F, D) \qquad e(V, H) = e(G, W) \tag{2}$$

The next assumption was also introduced in [Fuc09] and is the weakest variant of the various *flexible CDH* assumptions, adapted to asymmetric bilinear groups. It states that given $G, G^a$ and $H$, it is hard to output $(G^r, G^{ra}, H^r, H^{ra})$ for an arbitrary $r \neq 0$.

**Assumption 3** (AWFCDH)**.** *Given random generators* $G \in \mathbb{G}_1$ *and* $H \in \mathbb{G}_2$*, and* $A = G^a$ *for* $a \leftarrow \mathbb{Z}_p$*, it is hard to output* $(G^r, G^{ra}, H^r, H^{ra}) \in (\mathbb{G}_1^*)^2 \times (\mathbb{G}_2^*)^2$*, i.e., a tuple* $(R, M, S, N)$ *that satisfies*

$$e(A, S) = e(M, H) \qquad e(M, H) = e(G, N) \qquad e(R, H) = e(G, S) \tag{3}$$

Throughout the paper, we will assume two fixed generators $G$ and $H$ of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. We call a pair $(A, B) \in \mathbb{G}_1 \times \mathbb{G}_2$ a *Diffie-Hellman pair* (w.r.t. $(G, H)$), if there exists $a \in \mathbb{Z}_p$ such that $A = G^a$ and $B = H^a$. Using the bilinear map $e$, such pairs are efficiently decidable by checking

$$\mathrm{E}_{\mathcal{DH}}(A; B)\ :\ e(G^{-1}, \underline{B})\, e(\underline{A}, H) = 1\ .$$

We let $\mathcal{DH} := \{(G^a, H^a) \,|\, a \in \mathbb{Z}_p\}$ denote the set of DH pairs and implicitly assume them to be w.r.t. $G$ and $H$.

## 6.2 SXDH Commitments.

We instantiate $\mathbf{Com}$, defined in Sect. 3.1, by the commitment scheme based on SXDH given in [GS08].

$\mathsf{Setup}$, on input $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, G_1, G_2)$ chooses $\alpha_1, \alpha_2, t_1, t_2 \leftarrow \mathbb{Z}_p$ and returns $ck = (grp, \mathbf{u}_1, \mathbf{u}_2, \mathbf{v}_1, \mathbf{v}_2)$ with

$$\mathbf{u}_1 := (G_1, G_1^{\alpha_1}) \qquad \mathbf{u}_2 := (G_1^{t_1}, G_1^{\alpha_1 t_1}) \qquad \mathbf{v}_1 := (G_2, G_2^{\alpha_2}) \qquad \mathbf{v}_2 := (G_2^{t_2}, G_2^{\alpha_2 t_2})$$

Value and random space are defined as $\mathcal{V} := \mathbb{G}_1 \cup \mathbb{G}_2$ and $\mathcal{R} := \mathbb{Z}_p^2$.

$\mathsf{Com}(ck, X, r)$ is defined as follows: for $X \in \mathbb{G}_1$, $\mathsf{Com}(ck, X, r) := (\mathbf{u}_{1,1}^{r_1} \cdot \mathbf{u}_{2,1}^{r_2},\ X \cdot \mathbf{u}_{1,2}^{r_1} \cdot \mathbf{u}_{2,2}^{r_2})$; for $X \in \mathbb{G}_2$, $\mathsf{Com}(ck, X, r) := (\mathbf{v}_{1,1}^{r_1} \cdot \mathbf{v}_{2,1}^{r_2},\ X \cdot \mathbf{v}_{1,2}^{r_1} \cdot \mathbf{v}_{2,2}^{r_2})$.

$\mathsf{RdCom}(ck, \mathbf{c}, r')$ returns $\mathbf{c} \odot \mathsf{Com}(ck, 1, r') = (\mathbf{c}_1 \cdot \mathbf{u}_{1,1}^{r'_1} \cdot \mathbf{u}_{2,1}^{r'_2},\ \mathbf{c}_2 \cdot \mathbf{u}_{1,2}^{r'_1} \cdot \mathbf{u}_{2,2}^{r'_2})$, when $\mathbf{c} \in \mathbb{G}_1^2$ and similarly for the case when $\mathbf{c} \in \mathbb{G}_2^2$. (" $\odot$ " denotes component-wise multiplication.)

$\mathsf{ExSetup}$ constructs $ck$ as in $\mathsf{Setup}$ and in addition outputs the extraction key $ek := (\alpha_1, \alpha_2)$.

$\mathsf{Extr}(ek, \mathbf{c})$ is defined as follows. If $\mathbf{c} \in \mathbb{G}_1^2$ then set output $c_2 \cdot c_1^{-\alpha_1}$; if $\mathbf{c} \in \mathbb{G}_2^2$ then output $c_2 \cdot c_1^{-\alpha_2}$

$\mathsf{WISetup}$ produces $ck$ as $\mathsf{Setup}$, but sets $\mathbf{u}_{2,2}$ and $\mathbf{v}_{2,2}$ as $G_1^{\alpha_1 t_1 - 1}$ and $G_2^{\alpha_2 t_2 - 1}$, respectively (which is indistinguishable by SXDH). This results in perfectly hiding commitments.

**Remark 3.** $\mathbf{Com}$ commitments are *homomorphic*: $\mathsf{Com}(ck, X, r) \odot \mathsf{Com}(ck, X', r') = \mathsf{Com}(ck, X \cdot X', r + r')$; therefore if $\mathbf{c} = \mathsf{Com}(ck, X, r)$ then $\mathsf{RdCom}(ck, \mathbf{c}, r') = \mathsf{Com}(ck, X, r + r')$.

**Security.** *The scheme is perfectly binding, computationally hiding and randomizable as defined in Sect. 3.1.*

### 6.3 SXDH Groth-Sahai Proofs for Pairing-Product Equations

In order to instantiate $\mathbf{Proof}$, defined in Sect. 3.2, we use the proof system introduced in [GS08]. The class of equations $\mathcal{E}$ for our proof system are *pairing-product equations* (PPE). A PPE over variables $X_1, \ldots, X_m \in \mathbb{G}_1$ and $Y_1, \ldots, Y_n \in \mathbb{G}_2$ is an equation of the form

$$\mathrm{E}(X_1, \ldots, X_m; Y_1, \ldots, Y_n) \ :\ \prod_{i=1}^{n} e(A_i, \underline{Y_i}) \prod_{i=1}^{m} e(\underline{X_i}, B_i) \prod_{i=1}^{m} \prod_{j=1}^{n} e(\underline{X_i}, \underline{Y_j})^{\gamma_{i,j}} \ = \ \mathbf{t}_T \ , \tag{4}$$

defined by $A_j \in \mathbb{G}_1$, $B_i \in \mathbb{G}_2$, $\gamma_{i,j} \in \mathbb{Z}_p$, for $1 \leq i \leq m$ and $1 \leq j \leq n$, and $\mathbf{t}_T \in \mathbb{G}_T$.

**Proofs.** We define $\mathsf{Prove}\big(ck, \mathrm{E}, (X_i, r_i)_{i=1}^m, (Y_j, s_j)_{j=1}^n\big)$ for $X_i \in \mathbb{G}_1, Y_j \in \mathbb{G}_2$ and $r_i, s_j \in \mathbb{Z}_p^2$, and equation $\mathrm{E}$ given by the values $\big((A_j)_{j=1}^n, (B_i)_{i=1}^m, (\gamma_{i,j})_{i,j} \in \mathbb{Z}_p^{m \times n}, \mathbf{t}_T \in \mathbb{G}_T\big)$. For notational convenience, let us first define the following two shortcuts for $Z = (z_{ij})_{ij} \in \mathbb{Z}_p^{2\times 2}, \vec{\mathbf{u}} \in \mathbb{G}_1^{2\times 2}, \vec{\mathbf{v}} \in \mathbb{G}_2^{2\times 2}$.

$$Z \otimes \vec{\mathbf{u}} := \begin{bmatrix} u_{11}^{z_{11}} u_{21}^{z_{12}} & u_{12}^{z_{11}} u_{22}^{z_{12}} \\ u_{11}^{z_{21}} u_{21}^{z_{22}} & u_{12}^{z_{21}} u_{22}^{z_{22}} \end{bmatrix} \qquad\qquad Z \otimes \vec{\mathbf{v}} := \begin{bmatrix} v_{11}^{-z_{11}} v_{21}^{-z_{21}} & v_{12}^{-z_{11}} v_{22}^{-z_{21}} \\ v_{11}^{-z_{12}} v_{21}^{-z_{22}} & v_{12}^{-z_{12}} v_{22}^{-z_{22}} \end{bmatrix} \tag{5}$$

$\mathsf{Prove}$ chooses $Z = ((z_{11}, z_{12}), (z_{21}, z_{22}))^\top \leftarrow \mathbb{Z}_p^{2\times 2}$ and defines

$$\begin{aligned} t_{11} &:= \textstyle\sum_{j=1}^n \sum_{i=1}^m r_{i1}\gamma_{ij}s_{j1} \qquad & t_{12} &:= \textstyle\sum_{j=1}^n \sum_{i=1}^m r_{i1}\gamma_{ij}s_{j2} \\ t_{21} &:= \textstyle\sum_{j=1}^n \sum_{i=1}^m r_{i2}\gamma_{ij}s_{j1} \qquad & t_{22} &:= \textstyle\sum_{j=1}^n \sum_{i=1}^m r_{i2}\gamma_{ij}s_{j2} \end{aligned} \tag{6}$$

The output $(\phi, \theta) \in \mathbb{G}_2^{2\times 2} \times \mathbb{G}_1^{2\times 2}$ of $\mathsf{Prove}$ is then defined as:

$$\begin{aligned} \phi &:= \begin{bmatrix} v_{11}^{t_{11}} v_{21}^{t_{12}} & \big(\prod_{i=1}^m B_i^{r_{i1}}\big)\big(\prod_{j=1}^n Y_j^{\sum_{i=1}^m r_{i1}\gamma_{ij}}\big) v_{12}^{t_{11}} v_{22}^{t_{12}} \\ v_{11}^{t_{21}} v_{21}^{t_{22}} & \big(\prod_{i=1}^m B_i^{r_{i2}}\big)\big(\prod_{j=1}^n Y_j^{\sum_{i=1}^m r_{i2}\gamma_{ij}}\big) v_{12}^{t_{21}} v_{22}^{t_{22}} \end{bmatrix} \odot (Z \otimes \vec{\mathbf{v}}) \\ \theta &:= \begin{bmatrix} 1 & \big(\prod_{j=1}^n A_j^{s_{j1}}\big)\big(\prod_{i=1}^m X_i^{\sum_{j=1}^n s_{j1}\gamma_{ij}}\big) \\ 1 & \big(\prod_{j=1}^n A_j^{s_{j2}}\big)\big(\prod_{i=1}^m X_i^{\sum_{j=1}^n s_{j2}\gamma_{ij}}\big) \end{bmatrix} \odot (Z \otimes \vec{\mathbf{u}}) \end{aligned} \tag{7}$$

We write $\mathsf{Prove}\big(ck, \mathrm{E}, (X_i, r_i)_{i=1}^m, (Y_j, s_j)_{j=1}^n;\ Z\big)$ if we want to make the internal randomness $Z \in \mathbb{Z}_p^{2\times 2}$ explicit.

**Randomization.** Randomization of a commitment $\mathbf{c} = \mathsf{Com}(ck, X, r)$ via $\mathsf{RdCom}(ck, \mathbf{c}, r')$ replaces randomness $r$ by randomness $r + r'$; and similarly for $\mathbf{d} = \mathsf{Com}(ck, Y, s)$. Adaptation of a proof by $\mathsf{RdProof}$ must thus do the same to a proof $\pi = (\phi, \theta)$. We formally define $\mathsf{RdProof}\big(ck, \mathrm{E}, (\mathbf{c}_i, r_i)_{i=1}^m, (\mathbf{d}_j, s_j)_{j=1}^n, \pi\big)$: choose $Z = ((z_{11}, z_{12}), (z_{21}, z_{22}))^\top \leftarrow \mathbb{Z}_p^{2\times 2}$, define $(t_{11}, t_{12}, t_{21}, t_{22})$ as in (6) and output $(\phi', \theta') \in \mathbb{G}_2^{2\times 2} \times \mathbb{G}_1^{2\times 2}$ defined as:

$$\phi' := \phi \odot \begin{bmatrix} \big(\prod_{j=1}^n d_{j1}^{\sum_{i=1}^m r_{i1}\gamma_{ij}}\big) v_{11}^{t_{11}} v_{21}^{t_{12}} & \big(\prod_{i=1}^m B_i^{r_{i1}}\big)\big(\prod_{j=1}^n d_{j2}^{\sum_{i=1}^m r_{i1}\gamma_{ij}}\big) v_{12}^{t_{11}} v_{22}^{t_{12}} \\ \big(\prod_{j=1}^n d_{j1}^{\sum_{i=1}^m r_{i2}\gamma_{ij}}\big) v_{11}^{t_{21}} v_{21}^{t_{22}} & \big(\prod_{i=1}^m B_i^{r_{i2}}\big)\big(\prod_{j=1}^n d_{j2}^{\sum_{i=1}^m r_{i2}\gamma_{ij}}\big) v_{12}^{t_{21}} v_{22}^{t_{22}} \end{bmatrix} \odot (Z \otimes \vec{\mathbf{v}})$$

$$\theta' := \theta \odot \begin{bmatrix} \big(\prod_{i=1}^m c_{i1}^{\sum_{j=1}^n s_{j1}\gamma_{ij}}\big) & \big(\prod_{j=1}^n A_j^{s_{j1}}\big)\big(\prod_{i=1}^m c_{i2}^{\sum_{j=1}^n s_{j1}\gamma_{ij}}\big) \\ \big(\prod_{i=1}^m c_{i1}^{\sum_{j=1}^n s_{j2}\gamma_{ij}}\big) & \big(\prod_{j=1}^n A_j^{s_{j2}}\big)\big(\prod_{i=1}^m c_{i2}^{\sum_{j=1}^n s_{j2}\gamma_{ij}}\big) \end{bmatrix} \odot (Z \otimes \vec{\mathbf{u}})$$

which has the same distribution as the output of $\mathsf{Prove}\big(ck, \mathrm{E}, (X_i, r'_i + r_i)_{i=1}^m, (Y_j, s'_j + s_j)_{j=1}^n\big)$, where $r'_i$ and $s'_j$ are such that $\mathbf{c}_i = \mathsf{Com}(ck, X_i, r'_i)$ and $\mathbf{d}_j = \mathsf{Com}(ck, Y_j, s'_j)$, as we will show now.

**Remark 4.** In additive notation, the commitments and proofs can be written as follows (cf. the full version of [GS08]), when the cumulated randomness for all variables is $R = (r_{ik}) \in \mathbb{Z}_p^{m\times 2}$ and $S = (s_{jk}) \in \mathbb{Z}_p^{n\times 2}$, and we set $\Gamma = (\gamma_{ij}) \in \mathbb{Z}_p^{m\times n}$ and define $\iota_i(\vec{X}) := [\,\vec{0}\,|\,\vec{X}\,]$.

$$\vec{\mathbf{c}} = \iota(\vec{X}) + R\vec{\mathbf{u}} \qquad\qquad \phi = R^\top \iota(\vec{B}) + R^\top \Gamma \iota(\vec{Y}) + (R^\top \Gamma S - Z^\top)\vec{\mathbf{v}}$$
$$\vec{\mathbf{d}} = \iota(\vec{Y}) + S\vec{\mathbf{v}} \qquad\qquad \theta = S^\top \iota(\vec{A}) + S^\top \Gamma^\top \iota(\vec{X}) + Z\vec{\mathbf{u}}$$

To randomize the commitments and proofs, choose $\hat{R} \leftarrow \mathbb{Z}_p^{m\times 2},\ \hat{S} \leftarrow \mathbb{Z}_p^{n\times 2},\ \hat{Z} \leftarrow \mathbb{Z}_p^{2\times 2}$ and set

$$\vec{\mathbf{c}}' := \vec{\mathbf{c}} + \hat{R}\vec{\mathbf{u}} = \iota(\vec{X}) + (R + \hat{R})\vec{\mathbf{u}} \qquad\qquad \vec{\mathbf{d}}' := \vec{\mathbf{d}} + \hat{S}\vec{\mathbf{v}} = \iota(\vec{Y}) + (S + \hat{S})\vec{\mathbf{v}}$$

$$\begin{aligned}
\phi' &:= \phi + \hat{R}^\top \iota(\vec{B}) + \hat{R}^\top \Gamma \vec{\mathbf{d}} + (\hat{R}^\top \Gamma \hat{S} - \hat{Z}^\top)\vec{\mathbf{v}} \\
&= \big[R^\top \iota(\vec{B}) + R^\top \Gamma \iota(\vec{Y}) + (R^\top \Gamma S - Z^\top)\vec{\mathbf{v}}\big] + \hat{R}^\top \iota(\vec{B}) + \hat{R}^\top \Gamma \big[\iota(\vec{Y}) + S\vec{\mathbf{v}}\big] + (\hat{R}^\top \Gamma \hat{S} - \hat{Z}^\top)\vec{\mathbf{v}} \\
&= (R + \hat{R})^\top \iota(\vec{B}) + (R + \hat{R})^\top \Gamma \iota(\vec{Y}) + \big[(R + \hat{R})^\top \Gamma (S + \hat{S}) - \underbrace{(Z^\top + \hat{Z}^\top + R^\top \Gamma \hat{S})}_{=(Z + \hat{Z} + \hat{S}^\top \Gamma^\top R)^\top =: (Z')^\top}\,\big]\vec{\mathbf{v}} \\
\theta' &:= \theta + \hat{S}^\top \iota(\vec{A}) + \hat{S}^\top \Gamma^\top \vec{\mathbf{c}} + \hat{Z}\vec{\mathbf{u}} \\
&= \big[S^\top \iota(\vec{A}) + S^\top \Gamma^\top \iota(\vec{X}) + Z\vec{\mathbf{u}}\big] + \hat{S}^\top \iota(\vec{A}) + \hat{S}^\top \Gamma^\top \big[\iota(\vec{X}) + R\vec{\mathbf{u}}\big] + \hat{Z}\vec{\mathbf{u}} \\
&= (S + \hat{S})^\top \iota(\vec{A}) + (S + \hat{S})^\top \Gamma^\top \iota(\vec{X}) + \underbrace{(Z + \hat{Z} + \hat{S}^\top \Gamma^\top R)}_{=Z'}\vec{\mathbf{u}}
\end{aligned}$$

The output of $\mathsf{RdProof}\big(ck, \mathrm{E}, (\mathbf{c}_i, \hat{r}_i)_{i=1}^m, (\mathbf{c}_j, \hat{s}_j)_{j=1}^n, \pi\big)$ using randomness $((\hat{z}_{11}, \hat{z}_{12}), (\hat{z}_{21}, \hat{z}_{22}))^\top$ is therefore the same as that of $\mathsf{Prove}\big(ck, \mathrm{E}, (X_i, r_i + \hat{r}_i)_{i=1}^m, (Y_j, s_j + \hat{s}_j)\big)$ when the randomness used is

$$\begin{bmatrix} z_{11} + \hat{z}_{11} + \sum\sum \hat{s}_{j1}\gamma_{ij} r_{i1} & z_{12} + \hat{z}_{12} + \sum\sum \hat{s}_{j1}\gamma_{ij} r_{i2} \\ z_{21} + \hat{z}_{21} + \sum\sum \hat{s}_{j2}\gamma_{ij} r_{i1} & z_{22} + \hat{z}_{22} + \sum\sum \hat{s}_{j2}\gamma_{ij} r_{i2} \end{bmatrix}\ , \tag{8}$$

which is uniformly distributed over $\mathbb{Z}_p^{2\times 2}$ if $\hat{Z}$ is.

**Verification.** Let $ck = (\mathbf{u}_1, \mathbf{u}_2, \mathbf{v}_1, \mathbf{v}_2) \in \mathbb{G}_1^{2\times 2} \times \mathbb{G}_2^{2\times 2}$ be a commitment key, let $\vec{\mathbf{c}} \in \mathbb{G}_1^{m\times 2}, \vec{\mathbf{d}} \in \mathbb{G}_2^{n\times 2}$ be vectors of commitments, and let $(\phi, \theta) \in \mathbb{G}_2^{2\times 2} \times \mathbb{G}_1^{2\times 2}$ be a proof for an equation E given by $\vec{A} \in \mathbb{G}_1^n$, $\vec{B} \in \mathbb{G}_2^m$, $\Gamma = (\gamma_{i,j})_{i,j} \in \mathbb{Z}_p^{m\times n}$, and $\mathbf{t}_T \in \mathbb{G}_T$. $\mathsf{Verify}(ck, \mathrm{E}, \vec{\mathbf{c}}, \vec{\mathbf{d}}, (\phi, \theta))$ outputs 1 if and only if the following 4 equations hold.

$$\begin{aligned}
\textstyle\prod_{i=1}^m e\big(c_{i,1},\, \prod_{j=1}^n d_{j,1}^{\gamma_{i,j}}\big) &= e(u_{1,1}, \phi_{1,1})\, e(u_{2,1}, \phi_{2,1})\, e(\theta_{1,1}, v_{1,1})\, e(\theta_{2,1}, v_{2,1}) \\
\textstyle\prod_{i=1}^m e\big(c_{i,1},\, B_i \prod_{j=1}^n d_{j,2}^{\gamma_{i,j}}\big) &= e(u_{1,1}, \phi_{1,2})\, e(u_{2,1}, \phi_{2,2})\, e(\theta_{1,1}, v_{1,2})\, e(\theta_{2,1}, v_{2,2}) \\
\textstyle\prod_{j=1}^n e\big(A_j \prod_{i=1}^m c_{i,2}^{\gamma_{i,j}},\, d_{j,1}\big) &= e(u_{1,2}, \phi_{1,1})\, e(u_{2,2}, \phi_{2,1})\, e(\theta_{1,2}, v_{1,1})\, e(\theta_{2,2}, v_{2,1}) \\
\textstyle\prod_{j=1}^n e(A_j, d_{j,2}) \prod_{i=1}^m e\big(c_{i,2},\, B_i \prod_{j=1}^n d_{j,2}^{\gamma_{i,j}}\big) &= \mathbf{t}_T\, e(u_{1,2}, \phi_{1,2})\, e(u_{2,2}, \phi_{2,2})\, e(\theta_{1,2}, v_{1,2})\, e(\theta_{2,2}, v_{2,2})
\end{aligned}$$

**Remark 5.** Blazy et al. [BFI+10] show that by using techniques of *batch verification*, the number of pairing computations can be reduced from $4m + n + 16$ to $2m + n + 8$.

**Security.** *It follows from the results of [GS08] and Remark 4 that* $(\mathsf{Prove}, \mathsf{Verify}, \mathsf{RdProof})$ *is a randomizable witness-indistinguishable proof system for* $\mathbf{Com}$ *from Sect. 6.2, as defined in Sect. 3.2.*

### 6.4 Automorphic Signatures

We instantiate the signature scheme $\mathbf{Sig} = (\mathsf{Setup_S}, \mathsf{KeyGen_S}, \mathsf{Sign}, \mathsf{Ver})$ with the scheme from [Fuc09]. It is compatible since signature components are in $\mathcal{V} = \mathbb{G}_1 \cup \mathbb{G}_2$, the space for committed values, and the verification equations are pairing-product equations, thus in $\mathcal{E}$. It is moreover *automorphic* since the verification keys lie in the message space.

**Scheme 1 (Sig).** $\mathsf{Setup_S}$ *has input grp* $= (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, G, H)$ *and outputs grp and additional generators* $F, K, T \leftarrow \mathbb{G}_1$. *The message space is* $\mathcal{DH} := \{(G^m, H^m) \,|\, m \in \mathbb{Z}_p\}$.

$\mathsf{KeyGen_S}$ *chooses* $x \leftarrow \mathbb{Z}_p$ *and outputs* $(\mathit{vk} = (G^x, H^x),\ \mathit{sk} = x)$.

$\mathsf{Sign}$ *has inputs a secret key* $x$ *and a message* $(M, N) \in \mathcal{DH}$. *It chooses random* $c, r \leftarrow \mathbb{Z}_p$ *and outputs*

$$\big(A := (K \cdot T^r \cdot M)^{\frac{1}{x+c}},\ B := F^c,\ D := H^c,\ R := G^r,\ S := H^r\big)$$

$\mathsf{Ver}$ *on inputs a public key* $(X, Y) \in \mathcal{DH}$, *a message* $(M, N) \in \mathcal{DH}$ *and a signature* $(A, B, D, R, S)$ *outputs 1 (and 0 otherwise) iff the following hold:*

$$e(A, Y \cdot D) = e(K \cdot M, H)\, e(T, S) \qquad e(B, H) = e(F, D) \qquad e(R, H) = e(G, S) \tag{9}$$

**Security.** *Under q-ADHSDH and AWFCDH,* $\mathbf{Sig}$ *is strongly existentially unforgeable against adversaries making up to* $q - 1$ *adaptive chosen-message queries [Fuc09].*

**Automorphic Signatures on Two Messages.** Fuchsbauer [Fuc09] shows how to transform the above construction into an automorphic signature scheme that signs two messages at once—if we restrict the message space to $\mathcal{DH}^* := \{(G^m, H^m) \,|\, m \in \mathbb{Z}_p \setminus \{0\}\}$. $\mathsf{Sign}^*(\mathit{sk}, (V, W), (M, N))$ for $(V, W), (M, N) \in \mathcal{DH}^*$ is defined as follows: pick a key pair $(\mathit{vk}^*, \mathit{sk}^*) \leftarrow \mathsf{KeyGen_S}$ and output[8]

$$\big(\mathit{vk}^*,\ \mathsf{Sign}(\mathit{sk},\ \mathit{vk}^*),\ \mathsf{Sign}(\mathit{sk}^*,\ (M, N)),\ \mathsf{Sign}(\mathit{sk}^*,\ (V, W) \odot (M, N)),\ \mathsf{Sign}(\mathit{sk}^*,\ (V, W)^3 \odot (M, N))\big)\ .$$

$\mathsf{Ver}^*\big(\mathit{vk}, (V, W), (M, N), \Sigma\big)$ parses $\Sigma$ as $(\mathit{vk}^*, \Sigma_0, \Sigma_1, \Sigma_2, \Sigma_3)$ and outputs

$$\mathsf{Ver}(\mathit{vk}, \mathit{vk}^*, \Sigma_0)\ \cdot\ \mathsf{Ver}(\mathit{vk}^*, (M, N), \Sigma_1)\ \cdot\ \mathsf{Ver}(\mathit{vk}^*, (V, W) \odot (M, N), \Sigma_2)\ \cdot\ \mathsf{Ver}(\mathit{vk}^*, (V, W)^3 \odot (M, N), \Sigma_3)\ .$$

$\mathbf{Sig}^* = (\mathsf{Setup_S}, \mathsf{KeyGen_S}, \mathsf{Sign}^*, \mathsf{Ver}^*)$ is strongly unforgeable under ADHSDH and AWFCDH [Fuc09].

In Sect. 9.1, we give a variant of the scheme $\mathbf{Sig}$ with messages in $\mathbb{Z}_p \times \mathcal{DH}$ (required by our application to credentials in Sect. 5) which does *not* increase the size of a signature.

## 7 Additional Properties of Groth-Sahai Proofs

We identify five properties of Groth-Sahai proofs that will allow us to instantiate commuting signatures. The first is that proofs are constructed independently of the right-hand side of the equation; if the equation is *linear*, i.e., if $\gamma_{ij} = 0$ for all $i, j$, then they are even independent of the committed values. Given two independent (i.e., with no common variables) equations and commitments and proofs for them then the product of the proofs is a proof of the "product of the equations" and the concatenated vectors of commitments. The fourth property states that if we change a committed value by exponentiation then we can adapt the proof. And lastly, given commitments and a proof for an equation, if we commit to a constant of the equation then we can turn the proof into one for the set of commitments extended by the new commitment and the equation where the constant is now a variable.

[8] Exponentiation of a DH pair is defined componentwise: $(M, N)^k := (M^k, N^k)$.

### 7.1 Independence of Proofs

In general, proofs are independent of the right-hand side of the equation; moreover, proofs for linear equations are independent of the committed values.

**Lemma 1.** *Consider equation* $\mathrm{E}$ *from* (4). *Then the output of* $\mathsf{Prove}(\cdot, \mathrm{E}, \cdot, \cdot)$ *is independent of* $\mathbf{t}_T$.

*Proof.* The result follows by inspection of the proof definition in (7), or, more generally, the one in Remark 4, which also encompasses other instantiations of Groth-Sahai proofs. $\square$

For concreteness, we will give the proofs of the next lemmas for the SXDH instantiation, but we note that they also hold for the other instantiations.

**Lemma 2.** *Linear proofs depend only on the randomness of the commitments, but not on the committed values.*

*Proof.* For an equation $\mathrm{E}$ for which $\gamma_{ij} = 0$ for all $i$ and $j$ the proof simplifies to

$$\phi := \begin{bmatrix} 1 & \prod_{i=1}^m B_i^{r_{i1}} \\ 1 & \prod_{i=1}^m B_i^{r_{i2}} \end{bmatrix} \odot (Z \otimes \vec{\mathbf{v}}) \qquad\qquad \theta := \begin{bmatrix} 1 & \prod_{j=1}^n A_j^{s_{j1}}) \\ 1 & \prod_{j=1}^n A_j^{s_{j2}}) \end{bmatrix} \odot (Z \otimes \vec{\mathbf{u}})$$

which does not contain values $X_i$ and $Y_j$. $\square$

### 7.2 Proofs for Composed Equations

Groth-Sahai proofs are homomorphic w.r.t. the equations in the following sense. Given equations

$$\mathrm{E} \;:\; \prod_{i=1}^{n} e(A_i, \underline{Y_i}) \prod_{i=1}^{m} e(\underline{X_i}, B_i) \prod_{i=1}^{m}\prod_{j=1}^{n} e(\underline{X_i}, \underline{Y_j})^{\gamma_{i,j}} \;=\; \mathbf{t}_T$$

$$\mathrm{E}' \;:\; \prod_{i=1}^{n'} e(A'_i, \underline{Y'_i}) \prod_{i=1}^{m'} e(\underline{X'_i}, B'_i) \prod_{i=1}^{m'}\prod_{j=1}^{n'} e(\underline{X'_i}, \underline{Y'_j})^{\gamma'_{i,j}} \;=\; \mathbf{t}'_T$$

and a proof $\pi$ for commitments $(\vec{\mathbf{c}}, \vec{\mathbf{d}})$ for equation $\mathrm{E}$ and a proof $\pi'$ for commitments $(\vec{\mathbf{c}}', \vec{\mathbf{d}}')$ for equation $\mathrm{E}'$, then $\pi'' := \pi \odot \pi'$ is a proof for commitments $((\vec{\mathbf{c}}, \vec{\mathbf{c}}'), (\vec{\mathbf{d}}, \vec{\mathbf{d}}'))$ and equation $\mathrm{E}''$ (for arbitrary $\mathbf{t}''_T \in \mathbb{G}_T$):

$$E'' \;:\; \prod_{i=1}^{n} e(A_i, \underline{Y_i}) \prod_{i=1}^{n'} e(A'_i, \underline{Y'_i}) \prod_{i=1}^{m} e(\underline{X_i}, B_i) \prod_{i=1}^{m'} e(\underline{X'_i}, B'_i) \prod_{i=1}^{m}\prod_{j=1}^{n} e(\underline{X_i}, \underline{Y_j})^{\gamma_{i,j}} \prod_{i=1}^{m'}\prod_{j=1}^{n'} e(\underline{X'_i}, \underline{Y'_j})^{\gamma'_{i,j}} \;=\; \mathbf{t}''_T$$

**Lemma 3.** *If* $\pi = \mathsf{Prove}(ck, \mathrm{E}, (X_i, r_i)_{i=1}^m, (Y_j, s_j)_{j=1}^n;\ Z)$ *and* $\pi' = \mathsf{Prove}(ck, \mathrm{E}', (X'_i, r'_i)_{i=1}^{m'}, (Y'_j, s'_j)_{j=1}^{n'};\ Z')$ *then* $\pi \odot \pi' = \mathsf{Prove}(ck, \mathrm{E}'', (X_i, r_i)_{i=1}^m, (X'_i, r'_i)_{i=1}^{m'}, (Y_j, s_j)_{j=1}^n, (Y'_j, s'_j)_{j=1}^{n'};\ Z + Z')$

*Proof.* Equation $\mathrm{E}''$ over $(X_1, \ldots, X_m, X'_1, \ldots, X'_{m'}; Y_1, \ldots, Y_n, Y'_1, \ldots, Y'_{n'})$ is determined by the constants $\vec{A}'' := (\vec{A}, \vec{A}')$, $\vec{B}'' := (\vec{B}, \vec{B}')$ and $\Gamma'' := \begin{bmatrix} \Gamma & 0 \\ 0 & \Gamma' \end{bmatrix}$. The proof $\pi'' = \pi \odot \pi'$ looks as follows (with $t''_{ij} := t_{ij} + t'_{ij}$)

$$\phi'' \;:=\; \begin{bmatrix} v_{11}^{t''_{11}} v_{21}^{t''_{12}} & \big(\prod_{i=1}^m B_i^{r_{i1}}\big)\big(\prod_{i=1}^{m'} (B'_i)^{r'_{i1}}\big)\big(\prod_{j=1}^n Y_j^{\sum_{i=1}^m r_{i1}\gamma_{ij}}\big)\big(\prod_{j=1}^{n'} (Y'_j)^{\sum_{i=1}^{m'} r'_{i1}\gamma'_{ij}}\big) v_{12}^{t''_{11}} v_{22}^{t''_{12}} \\ v_{11}^{t''_{21}} v_{21}^{t''_{22}} & \big(\prod_{i=1}^m B_i^{r_{i2}}\big)\big(\prod_{i=1}^{m'} (B'_i)^{r'_{i2}}\big)\big(\prod_{j=1}^n Y_j^{\sum_{i=1}^m r_{i2}\gamma_{ij}}\big)\big(\prod_{j=1}^{n'} (Y'_j)^{\sum_{i=1}^{m'} r'_{i2}\gamma'_{ij}}\big) v_{12}^{t''_{21}} v_{22}^{t''_{22}} \end{bmatrix} \odot ((Z + Z') \otimes \vec{\mathbf{v}})$$

$$\theta'' \;:=\; \begin{bmatrix} 1 & \big(\prod_{j=1}^n A_j^{s_{j1}}\big)\big(\prod_{j=1}^{n'} (A'_j)^{s'_{j1}}\big)\big(\prod_{i=1}^m X_i^{\sum_{j=1}^n s_{j1}\gamma_{ij}}\big)\big(\prod_{i=1}^{m'} (X'_i)^{\sum_{j=1}^{n'} s'_{j1}\gamma'_{ij}}\big) \\ 1 & \big(\prod_{j=1}^n A_j^{s_{j2}}\big)\big(\prod_{j=1}^{n'} (A'_j)^{s'_{j2}}\big)\big(\prod_{i=1}^m X_i^{\sum_{j=1}^n s_{j2}\gamma_{ij}}\big)\big(\prod_{i=1}^{m'} (X'_i)^{\sum_{j=1}^{n'} s'_{j2}\gamma'_{ij}}\big) \end{bmatrix} \odot ((Z + Z') \otimes \vec{\mathbf{u}})$$

which is a proof for $(\vec{\mathbf{c}}, \vec{\mathbf{c}}', \vec{\mathbf{d}}, \vec{\mathbf{d}}')$ for $\mathrm{E}''$ with internal randomness $Z + Z'$. $\square$

### 7.3 Changing the Committed Value and Adapting Proofs

We give a special case which we require to randomize commitments to *non-trivial* messages in Appendix C.1. We start with some notation. Let $\vec{\mathbf{w}} \in \mathbb{G}^{m\times n}$, $Z \in \mathbb{Z}_p^{m\times n}$ and $k \in \mathbb{Z}_p$. Then by $\vec{\mathbf{w}}^k$ we denote componentwise exponentiation, and by $k\cdot Z$ we denote standard scalar multiplication, i.e.,

$$\text{if } \vec{\mathbf{w}} = (w_{ij})_{\substack{1\le i\le m\\1\le j\le n}} \text{ then } \vec{\mathbf{w}}^k = (w_{ij}^k)_{\substack{1\le i\le m\\1\le j\le n}} \qquad \text{if } Z = (z_{ij})_{\substack{1\le i\le m\\1\le j\le n}} \text{ then } k\cdot Z = (kz_{ij})_{\substack{1\le i\le m\\1\le j\le n}}$$

Consider equation $\mathrm{E}^*\ :\ e(\underline{X},\underline{Y}) = \mathbf{t}_T$; then given a proof $\pi$ for $\mathrm{E}^*$, $\mathsf{Com}(ck,X,r)$ and $\mathsf{Com}(ck,Y,s)$, $\pi^k$ is a proof for $e(\underline{X},\underline{Y}) = \mathbf{t}_T^k$ and $\mathsf{Com}(ck,X^k,k\cdot r)$ and $\mathsf{Com}(ck,Y,s)$.

**Lemma 4.** *If* $\pi = \mathsf{Prove}\big(ck,\mathrm{E}^*,(X,r),(Y,s);\ Z\big)$ *then* $\pi^k = \mathsf{Prove}\big(ck,\mathrm{E}^*,(X^k,k\cdot r),(Y,s);\ k\cdot Z\big)$.

*Proof.* By (5), we have $(Z\otimes\vec{\mathbf{u}})^k = (k\cdot Z)\otimes\vec{\mathbf{u}}$ and $(Z\circledast\vec{\mathbf{v}})^k = (k\cdot Z)\circledast\vec{\mathbf{v}}$ for $Z\in\mathbb{Z}_p^{2\times 2}$ and $k\in\mathbb{Z}_p$. The proof $\mathsf{Prove}(ck,\mathrm{E}^*,(X,r),(Y,s);\ Z)$ is defined as

$$\pi_1 = \begin{bmatrix} v_{11}^{r_1s_1}v_{21}^{r_1s_2} & Y^{r_1}v_{12}^{r_1s_1}v_{22}^{r_1s_2}\\ v_{11}^{r_2s_1}v_{21}^{r_2s_2} & Y^{r_2}v_{12}^{r_2s_1}v_{22}^{r_2s_2}\end{bmatrix} \odot (Z\circledast\vec{\mathbf{v}}) \qquad \pi_2 = \begin{bmatrix}1 & X^{s_1}\\ 1 & X^{s_2}\end{bmatrix} \odot (Z\otimes\vec{\mathbf{u}})$$

so we have

$$\pi_1^k = \begin{bmatrix} v_{11}^{kr_1s_1}v_{21}^{kr_1s_2} & Y^{kr_1}v_{12}^{kr_1s_1}v_{22}^{kr_1s_2}\\ v_{11}^{kr_2s_1}v_{21}^{kr_2s_2} & Y^{kr_2}v_{12}^{kr_2s_1}v_{22}^{kr_2s_2}\end{bmatrix} \odot \big((k\cdot Z)\circledast\vec{\mathbf{v}}\big) \qquad \pi_2^k = \begin{bmatrix}1 & (X^k)^{s_1}\\ 1 & (X^k)^{s_2}\end{bmatrix} \odot \big((k\cdot Z)\otimes\vec{\mathbf{u}}\big)$$

which is the definition of $\mathsf{Prove}\big(ck,\mathrm{E}^*,(X^k,k\cdot r),(Y,s);\ k\cdot Z\big)$. □

### 7.4 Committing to Constants and Adapting Proofs

Given a proof for an equation, one can commit to one of the constants and adapt the proof. Consider an equation $\mathrm{E}(X_1,\ldots,X_m;\,Y_1,\ldots,Y_n)$ as in (4) and a proof $(\phi,\theta)$ for commitments $(\mathbf{c}_1,\ldots,\mathbf{c}_m;\,\mathbf{d}_1,\ldots,\mathbf{d}_n)$. Some calculation shows that $(\phi,\theta)$ is also a proof for equation

$$\mathrm{E}'(\vec{X},A_k;\,\vec{Y})\ :\ \prod_{\substack{i=1\\i\ne k}}^{n} e(A_i,\underline{Y_i}) \prod_{i=1}^{m} e(\underline{X_i},B_i) \prod_{i=1}^{m}\prod_{j=1}^{n} e(\underline{X_i},\underline{Y_j})^{\gamma_{i,j}}\, e(\underline{A_k},\underline{Y_k}) \ =\ \mathbf{t}_T$$

and commitments $(\mathbf{c}_1,\ldots,\mathbf{c}_m,\mathsf{Com}(ck,A_k,0);\ \mathbf{d}_1,\ldots,\mathbf{d}_n)$. This yields the following result.

**Lemma 5.** *Let* $\pi \leftarrow \mathsf{Prove}\big(ck,\mathrm{E},(X_i,r_i)_{i=1}^m,(Y_j,s_j)_{j=1}^n\big)$ *and for all* $i,j$ *let* $\mathbf{c}_i = \mathsf{Com}(ck,X_i,r_i)$ *and* $\mathbf{d}_j = \mathsf{Com}(ck,Y_j,s_j)$. *Then* $\mathsf{RdProof}\big(ck,\mathrm{E}',(\mathbf{c}_i,0)_{i=1}^m,(\mathsf{Com}(ck,A_k,0),r),(\mathbf{d}_j,0)_{j=1}^n,\pi\big)$ *yields a proof that is distributed as the output of* $\mathsf{Prove}\big(ck,\mathrm{E}',(X_i,r_i)_{i=1}^m,(A_k,r),(Y_j,s_j)_{j=1}^n\big)$. *An analogous result holds for committing to a constant* $B_k\in\mathbb{G}_2$.

## 8 Instantiation of Commuting Signatures

In [Fuc09], a blind signature scheme is constructed from the scheme **Sig** (Sect. 6.4) as follows. The user, who wishes to obtain a signature on a message $(M,N)\in\mathcal{DH}$, chooses a random $t\leftarrow\mathbb{Z}_p$ and *blinds* the first message component by the factor $T^t$. The user then sends the following: $U := T^t\cdot M$, commitments $\mathbf{c}_M$ and $\mathbf{c}_N$ to $M$ and $N$, respectively, and commitments $\mathbf{c}_P$ and $\mathbf{c}_Q$ to $G^t$ and $H^t$, respectively; moreover, proofs $\pi_M,\pi_P$ and $\pi_U$ of well-formedness of $(M,N)$, $(P,Q)$ and $U$, respectively. The signer replies with a "pre-signature" on $U$ (which is constructed as a signature on $U$, but on a message that lacks the second component):

$$A := (K\cdot T^r\cdot U)^{\frac{1}{x+c}} \qquad B := F^c \qquad D := H^c \qquad R' := G^r \qquad S' := H^r$$

Knowing $t$, the user can fabricate an actual signature on $(M,N)$ from this "pre-signature" by setting $R := R'\cdot G^t$ and $S := S'\cdot H^t$. Then $(A,B,D,R,S)$ is a signature with randomness $(c, r+t)$ because $A = (K\cdot T^r\cdot U)^{1/(x+c)} = (K\cdot T^{r+t}\cdot M)^{1/(x+c)}$, $R = G^{r+t}$, and $S = H^{r+t}$. To prevent linking a signature to the signing session, the blind signature is defined as a Groth-Sahai proof of knowledge of the signature. Now to turn this into a commuting signature, there are two key observations.

1. The values $(\mathbf{c}_M, \mathbf{c}_N, \pi_M, \mathbf{c}_P, \mathbf{c}_Q, \pi_P, U, \pi_U)$ which the user sends to the signer can actually be considered as a *commitment* to the message $(M,N)$, which is extractable and randomizable, and which perfectly hides the message when the values are produced using a key $ck^* \leftarrow \mathsf{WISetup}$.
2. Since $\mathbf{Com}$ is homomorphic, the values $\mathbf{c}_P$ and $\mathbf{c}_Q$ can be used to produce commitments on the actual signature components $R$ and $S$. Moreover, we show how $\pi_P$ and $\pi_U$ can be used to produce a proof of validity of the committed values using the results from Lemmas 1, 2, 3 and 5.

For the blind signature scheme in [Fuc09], the values $\mathbf{c}_p, \mathbf{c}_Q, \pi_P$ and $\pi_U$ are mainly used in the proof of unforgeability, when the simulator needs to extract the message, query it to its signing oracle and then use the values $P$ and $Q$ to turn the signature into a pre-signature. We show that all these values can be directly used by the signer to produce commitments to the signature components and even a proof of validity.

Our exposition will use Groth-Sahai proofs and the results from Sect. 7 in a black-box manner. We refer to Appendix A for a detailed and self-contained presentation of the instantiations.

### 8.1 Commitments to Messages

We define a *commitment* on a message $(M,N) \in \mathcal{DH}$ as the values $\mathbf{C} = (\mathbf{c}_M, \mathbf{c}_N, \pi_M, \mathbf{c}_P, \mathbf{c}_Q, \pi_P, U, \pi_U)$ the user sends to the signer in the issuing protocol for blind signatures from [Fuc09]. We then show how to randomize a commitment and how to extract the committed value. Since the committed values are the messages of $\mathbf{Sig}$, the algorithms also get the parameters $pp_{\mathrm{S}}$ as input.

$\mathsf{Com}_{\mathcal{M}}$ has inputs $pp = (ck, grp, F, K, T)$, $(M,N) \in \mathcal{DH}$ and $(t,\mu,\nu,\rho,\sigma) \in \mathbb{Z}_p^9 =: \mathcal{R}_{\mathcal{M}}$. We define the following equations:

$$\mathrm{E}_{\mathcal{DH}}(M,N) \;:\; e(G^{-1},\underline{N})\, e(\underline{M},H) = 1 \tag{10}$$

$$\mathrm{E}_U(M,Q) \;:\; e(T^{-1},\underline{Q})\, e(\underline{M},H^{-1}) = e(U,H)^{-1} \tag{11}$$

$\mathsf{Com}_{\mathcal{M}}(pp,(M,N),(t,\mu,\nu,\rho,\sigma))$ defines $P = G^t$ and $Q = H^t$, computes

$$\begin{array}{lll}
\mathbf{c}_M := \mathsf{Com}(ck,M,\mu) & \mathbf{c}_N := \mathsf{Com}(ck,N,\nu) & \pi_M \leftarrow \mathsf{Prove}\big(ck, \mathrm{E}_{\mathcal{DH}},\ (M,\mu),(N,\nu)\big) \\
\mathbf{c}_P := \mathsf{Com}(ck,P,\rho) & \mathbf{c}_Q := \mathsf{Com}(ck,Q,\sigma) & \pi_P \leftarrow \mathsf{Prove}\big(ck, \mathrm{E}_{\mathcal{DH}},\ (P,\rho),(Q,\sigma)\big) \\
U := T^t\cdot M & & \pi_U \leftarrow \mathsf{Prove}\big(ck, \mathrm{E}_U,\ (M,\mu),(Q,\sigma)\big)
\end{array}$$

and returns $\mathbf{C} = (\mathbf{c}_M, \mathbf{c}_N, \pi_M, \mathbf{c}_P, \mathbf{c}_Q, \pi_P, U, \pi_U) \in \mathcal{C}_{\mathcal{M}}(pp)$.

$\mathsf{RdCom}_{\mathcal{M}}$, on input $(ck, pp_{\mathrm{S}})$, $\mathbf{C}$ and randomness $(t',\mu',\nu',\rho',\sigma')$, first defines $\hat{\mathbf{c}}_P := \mathbf{c}_P \circledcirc \mathsf{Com}(ck, G^{t'}, 0)$, $\hat{\mathbf{c}}_Q := \mathbf{c}_Q \circledcirc \mathsf{Com}(ck, H^{t'}, 0)$ and $U' := U\cdot T^{t'}$. Then it sets

$$\begin{array}{ll}
\mathbf{c}'_M := \mathsf{RdCom}(ck,\mathbf{c}_M,\mu') & \pi'_M \leftarrow \mathsf{RdProof}(ck, \mathrm{E}_{\mathcal{DH}}, (\mathbf{c}_M,\mu'), (\mathbf{c}_N,\nu'), \pi_M) \\
\mathbf{c}'_N := \mathsf{RdCom}(ck,\mathbf{c}_N,\nu') & \pi'_P \leftarrow \mathsf{RdProof}(ck, \mathrm{E}_{\mathcal{DH}}, (\hat{\mathbf{c}}_P,\rho'), (\hat{\mathbf{c}}_Q,\sigma'), \pi_P) \\
\mathbf{c}'_P := \mathsf{RdCom}(ck,\hat{\mathbf{c}}_P,\rho') & \pi'_U \leftarrow \mathsf{RdProof}(ck, \mathrm{E}_U, (\mathbf{c}_M,\mu'), (\hat{\mathbf{c}}_Q,\sigma'), \pi_U) \\
\mathbf{c}'_Q := \mathsf{RdCom}(ck,\hat{\mathbf{c}}_Q,\sigma') &
\end{array}$$

and returns $\mathbf{C}' = (\mathbf{c}'_M, \mathbf{c}'_N, \pi'_M, \mathbf{c}'_P, \mathbf{c}'_Q, \pi'_P, U', \pi'_U) \in \mathcal{C}_{\mathcal{M}}(pp)$. Randomness $(t,\mu,\nu,\rho,\sigma)$ was thus replaced by $(t+t', \mu+\mu', \nu+\nu', \rho+\rho', \sigma+\sigma')$.

$\mathsf{Extr}_{\mathcal{M}}$ has inputs $ek$ and $\mathbf{C}$. It returns $\big(\mathsf{Extr}(ek,\mathbf{c}_M), \mathsf{Extr}(ek,\mathbf{c}_N)\big)$.

$\mathcal{C}_\mathcal{M}(pp)$, the space of valid $\mathbf{Com}_\mathcal{M}$ commitments under parameters $pp = (ck, pp_\mathrm{S})$ is defined as

$$\mathcal{C}_\mathcal{M}(pp) := \big\{(\mathbf{c}_M, \mathbf{c}_N, \pi_M, \mathbf{c}_P, \mathbf{c}_Q, \pi_P, U, \pi_U) \in \mathbb{G}_1^{17} \times \mathbb{G}_2^{16} \mid \mathsf{Verify}(ck, \mathrm{E}_{\mathcal{DH}}, \mathbf{c}_M, \mathbf{c}_N, \pi_M) \\ \wedge \mathsf{Verify}(ck, \mathrm{E}_{\mathcal{DH}}, \mathbf{c}_P, \mathbf{c}_Q, \pi_P) \wedge \mathsf{Verify}(ck, \mathrm{E}_U, \mathbf{c}_M, \mathbf{c}_Q, \pi_U)\big\} \ .$$

See Appendix A.1 for a proof that $\mathbf{Com}_\mathcal{M}$ is a randomizable extractable commitment scheme that is perfectly binding and computationally hiding.

### 8.2 Making Committents to a Signature on a Committed Message and a Proof of Validity

We show how the signer can use the values in $\mathbf{C}$ to produce a proof of knowledge

$$(\mathbf{c}_A, \mathbf{c}_B, \mathbf{c}_D, \mathbf{c}_R, \mathbf{c}_S, \pi_A, \pi_B, \pi_R) \in \mathbb{G}_1^{18} \times \mathbb{G}_2^{16}$$

of a signature $(A, B, D, R, S)$, where $\pi_A, \pi_B$ and $\pi_R$ are proofs that the committed values satisfy the equations in (9), respectively, i.e.,

$$\begin{aligned} \mathrm{E}_A(A, M;\, S, D) &: e(T^{-1}, \underline{S})\, e(\underline{A}, Y) e(\underline{M}, H^{-1})\, e(\underline{A}, \underline{D}) = e(K, H) \\ \mathrm{E}_B(B;\, D) &: e(F^{-1}, \underline{D})\, e(\underline{B}, H) = 1 \\ \mathrm{E}_R(R;\, S) &: e(G^{-1}, \underline{S})\, e(\underline{R}, H) = 1 \end{aligned} \tag{12}$$

In the blind signature from [Fuc09], on receiving $\mathbf{C}$, the signer checks the proofs contained in it, and then produces a pre-signature by choosing $c, r \leftarrow \mathbb{Z}_p$ and computing

$$A := (K \cdot T^r \cdot U)^{\frac{1}{x+c}} \qquad B := F^c \qquad D := H^c \qquad R' := G^r \qquad S' := H^r$$

Knowing $t$ s.t. $U = T^t \cdot M$, these values are turned into a signature by setting $R := R' \cdot G^t$ and $S := S' \cdot H^t$. Since the commitments are homomorphic, the signer can—without knowledge of the values $P = G^t$ and $Q = H^t$—make *commitments* to $R$ and $S$:

$$\mathbf{c}_R := \mathbf{c}_P \odot \mathsf{Com}(ck, R', 0) = \mathsf{Com}(ck, R, \rho) \qquad \mathbf{c}_S := \mathbf{c}_Q \odot \mathsf{Com}(ck, S', 0) = \mathsf{Com}(ck, S, \sigma)$$

The signer also chooses $\alpha, \beta, \delta \leftarrow \mathbb{Z}_p^2$, and makes the remaining commitments:

$$\mathbf{c}_A := \mathsf{Com}(ck, A, \alpha) \qquad \mathbf{c}_B := \mathsf{Com}(ck, B, \beta) \qquad \mathbf{c}_D := \mathsf{Com}(ck, D, \delta) \tag{13}$$

The vector $\vec{\mathbf{c}}_\Sigma := (\mathbf{c}_A, \mathbf{c}_B, \mathbf{c}_D, \mathbf{c}_R, \mathbf{c}_S)$ is thus a commitment on the *actual* signature $\Sigma = (A, B, D, R, S)$. It remains to construct proofs $\pi_A, \pi_B$ and $\pi_R$ that the committed values satisfy the 3 equations in (12)—without knowledge of $\mu, \rho$ and $\sigma$, the randomness of the commitments $\mathbf{c}_M, \mathbf{c}_R$ and $\mathbf{c}_S$, respectively! This can be done using the following observations:

1. Equation $\mathrm{E}_R(R;\, S)$ is actually $\mathrm{E}_{\mathcal{DH}}(R;\, S)$ from (10). Since $\mathbf{c}_R$ and $\mathbf{c}_P$ have the same randomness $\rho$, and $\mathbf{c}_S$ and $\mathbf{c}_Q$ have the same randomness $\sigma$, and since by Lemma 2, the proof for the *linear* equation $\mathrm{E}_{\mathcal{DH}}$ is independent of the committed values, we can set $\pi_R := \pi_P$.

2. Lemmas 1 and 2 state that linear proofs only depend on the randomness of the commitments. Since $\mathbf{c}_S = \mathsf{Com}(ck, S, \sigma)$ and $\mathbf{c}_Q = \mathsf{Com}(ck, Q, \sigma)$ have the same randomness, $\pi_U$ is a proof for $\mathrm{E}_U(M; S)$ for $\mathbf{c}_M$ and $\mathbf{c}_S$. Moreover, define

$$\mathrm{E}_{A^\dagger}(A;\, D) \ : \ e(\underline{A}, Y)\, e(\underline{A}, \underline{D}) = 1 \tag{14}$$

   and let $\pi_{A^\dagger} \leftarrow \mathsf{Prove}(ck, \mathrm{E}_{A^\dagger}, (A, \alpha), (D, \delta))$. Since the product of the left-hand sides of $\mathrm{E}_U(M; S)$ and $\mathrm{E}_{A^\dagger}(A; D)$ is the left-hand side of $\mathrm{E}_A(A, M;\, S, D)$, by Lemma 3 we have $\pi_A := \pi_U \odot \pi_{A^\dagger}$.

**SigCom**$(ck, sk, \mathbf{C})$ Parse $\mathbf{C}$ as $(\mathbf{c}_M, \mathbf{c}_N, \pi_M, \mathbf{c}_P, \mathbf{c}_Q, \pi_P, U, \pi_U)$ and *sk* as $x$. If $\pi_M, \pi_P$ and $\pi_U$ are valid then choose $c, r \leftarrow \mathbb{Z}_p$ and $\alpha, \beta, \delta, \rho', \sigma' \leftarrow \mathbb{Z}_p^2$ and compute the following values:

$$A := (K \cdot T^r \cdot U)^{\frac{1}{x+c}} \qquad \mathbf{c}_B := \mathsf{Com}(ck, F^c, \beta) \qquad \mathbf{c}_R := \mathbf{c}_P \odot \mathsf{Com}(ck, G^r, \rho')$$

$$\mathbf{c}_A := \mathsf{Com}(ck, A, \alpha) \qquad \mathbf{c}_D := \mathsf{Com}(ck, H^c, \delta) \qquad \mathbf{c}_S := \mathbf{c}_Q \odot \mathsf{Com}(ck, H^r, \sigma')$$

$\pi'_A \leftarrow \pi_U \odot \mathsf{Prove}(ck, \mathrm{E}_{A^\dagger}, (A, \alpha), (H^c, \delta))$ (with $\mathrm{E}_{A^\dagger}$ being Equation (14))

$\pi_A \leftarrow \mathsf{RdProof}(ck, \mathrm{E}_A, (\mathbf{c}_A, 0), (\mathbf{c}_D, 0), (\mathbf{c}_M, 0), (\mathbf{c}_S, \sigma'), \pi'_A)$

$\pi_R \leftarrow \mathsf{RdProof}(ck, \mathrm{E}_R, (\mathbf{c}_R, \rho'), (\mathbf{c}_S, \sigma'), \pi_P)$ $\qquad \pi_B \leftarrow \mathsf{Prove}(ck, \mathrm{E}_{\mathcal{DH}}, (F^c, \beta), (H^c, \delta))$

Return $(\mathbf{c}_A, \mathbf{c}_B, \mathbf{c}_D, \mathbf{c}_R, \mathbf{c}_S, \pi_A, \pi_B, \pi_R)$.

Figure 2: Making commitments to a signature and proving knowledge.

The remaining proof $\pi_B$ can be constructed regularly, since randomness $(\beta, \delta)$ is known to the signer. Finally, to get a *random* proof of knowledge, the signer randomizes all commitments and proofs using RdCom and RdProof as defined in Sect. 6.3. Algorithm SigCom is summarized in Fig. 2. In Appendix A.2, we formally prove that the output of SigCom distributed as required by Def. 2.

**Instantiation of** SmSigCom**.** This algorithm is similar to SigCom but instead of the signing key *sk* it is directly given a signature $(A, B, D, R, S)$. It proceeds like SigCom but starting from a signature instead of producing a pre-signature: choose $\alpha, \beta, \delta \leftarrow \mathcal{R}$ and set $\mathbf{c}_A, \mathbf{c}_B$ and $\mathbf{c}_D$ as in (13); use *ek* to extract $P$ and $Q$ from $\mathbf{C}$ and set

$$\mathbf{c}_R := \mathbf{c}_P \odot \mathsf{Com}(ck, R \cdot P^{-1}, 0) = \mathsf{Com}(ck, R, \rho) \qquad \mathbf{c}_S := \mathbf{c}_Q \odot \mathsf{Com}(ck, S \cdot Q^{-1}, 0) = \mathsf{Com}(ck, S, \sigma)$$

Now $\pi_A, \pi_B$ and $\pi_R$ can be computed as in SigCom (see Fig. 2).

### 8.3 Instantiations of Proof Adaptation for Committing and Decommitting

We define equations $\mathrm{E}_{\tilde{A}}$ and $\mathrm{E}_{\bar{A}}$ and recall $\mathrm{E}_A$:

$$\begin{aligned}
\mathrm{E}_A(A, M;\ S, D) &:\ e(T^{-1}, \underline{S})\, e(\underline{A}, Y) e(\underline{M}, H^{-1})\, e(\underline{A}, \underline{D}) = e(K, H) \\
\mathrm{E}_{\tilde{A}}(A;\ S, D) &:\ e(T^{-1}, \underline{S})\, e(\underline{A}, Y)\, e(\underline{A}, \underline{D}) = e(K \cdot M, H) \\
\mathrm{E}_{\bar{A}}(M) &:\ e(\underline{M}, H^{-1}) = e(A, Y \cdot D)^{-1} e(K, H)\, e(T, S)
\end{aligned}$$

Recall equations $\mathrm{E}_B$ and $\mathrm{E}_R$ from (12). Then we have

$$\begin{aligned}
\mathrm{E}_{\mathsf{Verify}((X,Y),\cdot,\cdot)}((M, N), (A, B, D, R, S)) &\equiv \mathrm{E}_A(A, M;\ S, D) \wedge \mathrm{E}_B(B;\ D) \wedge \mathrm{E}_R(R;\ S) \\
\mathrm{E}_{\mathsf{Verify}((X,Y),(M,N),\cdot)}(A, B, D, R, S) &\equiv \mathrm{E}_{\tilde{A}}(A;\ S, D) \wedge \mathrm{E}_B(B;\ D) \wedge \mathrm{E}_R(R;\ S) \\
\mathrm{E}_{\mathsf{Verify}((X,Y),\cdot,(A,B,D,R,S))}(M, N) &\equiv \mathrm{E}_{\bar{A}}(M)
\end{aligned}$$

Since the product of the left-hand sides of $\mathrm{E}_{\tilde{A}}$ and $\mathrm{E}_{\bar{A}}$ is the left-hand side of $\mathrm{E}_A$, by Lemma 3 we have

$$\pi_A = \pi_{\tilde{A}} \odot \pi_{\bar{A}} \ ,$$

which allows us to implement the algorithms AdPrC, $\mathsf{AdPrC}_{\mathcal{M}}$, AdPrDC and $\mathsf{AdPrDC}_{\mathcal{M}}$ as follows:

$\mathsf{AdPrC}(pp, vk, \mathbf{C}, ((A, B, D, R, S), (\alpha, \beta, \delta, \rho, \sigma)), \bar{\pi})$. The proof $\bar{\pi}$ is a proof for equation $\mathrm{E}_{\bar{A}}$. The algorithm sets

$$\pi_{\tilde{A}} \leftarrow \mathsf{Prove}(ck, \mathrm{E}_{\tilde{A}}, (A, \alpha), (S, \sigma), (D, \delta)) \qquad \begin{aligned} \pi_B &\leftarrow \mathsf{Prove}(ck, \mathrm{E}_B, (B, \beta), (D, \delta)) \\ \pi_R &\leftarrow \mathsf{Prove}(ck, \mathrm{E}_R, (R, \rho), (S, \sigma)) \end{aligned}$$

for $\mathrm{E}_B$ and $\mathrm{E}_R$ as defined in (12). It then returns $\pi := (\pi_{\tilde{A}} \odot \pi_{\bar{A}}, \pi_B, \pi_R)$.

$\mathsf{AdPrC}_{\mathcal{M}}(pp, vk, ((M,N),(t,\mu,\nu,\rho,\sigma)), \mathbf{c}_\Sigma, \widetilde{\pi})$. The proof $\widetilde{\pi}$ is of the form $(\pi_{\widetilde{A}}, \pi_B, \pi_R)$. The algorithm sets $\pi_{\bar{A}} \leftarrow \mathsf{Prove}(ck, \mathrm{E}_{\bar{A}}, (M,\mu))$ and returns a randomization of $\pi := (\pi_{\widetilde{A}} \circ \pi_{\bar{A}}, \pi_B, \pi_R)$.

$\mathsf{AdPrDC}(pp, vk, \mathbf{C}, ((A,B,D,R,S),(\alpha,\beta,\delta,\rho,\sigma)), \pi)$. The proof $\pi$ is of the form $(\pi_A, \pi_B, \pi_R)$. The algorithm sets $\pi_{\widetilde{A}} \leftarrow \mathsf{Prove}(ck, \mathrm{E}_{\widetilde{A}}, (A,\alpha),(S,\sigma),(D,\delta))$ and returns $\bar{\pi} := \pi_A \oslash \pi_{\widetilde{A}}$ (where "$\oslash$" denotes component-wise division, that is: replace all the components of the second argument by their inverses and then multiply them with those of the first argument).

$\mathsf{AdPrDC}_{\mathcal{M}}(pp, vk, ((M,N),(t,\mu,\nu,\rho,\sigma)), \mathbf{c}_\Sigma, \pi)$. The proof $\pi$ is of the form $(\pi_A, \pi_B, \pi_R)$. The algorithm produces $\pi_{\bar{A}} \leftarrow \mathsf{Prove}(ck, \mathrm{E}_{\bar{A}}, (M,\mu))$ and returns a randomization of $\widetilde{\pi} := (\pi_A \oslash \pi_{\bar{A}}, \pi_B, \pi_R)$.

**Instantiation of** $\mathsf{AdPrC}_{\mathcal{K}}$ **and** $\mathsf{AdPrDC}_{\mathcal{K}}$**.** In applications (such as the credentials in Sect. 5) where the signer wants to remain anonymous, she makes a commitment

$$\mathbf{c}_{vk} := \big(\mathbf{c}_X = \mathsf{Com}(ck, X, \xi),\ \mathbf{c}_Y = \mathsf{Com}(ck, Y, \psi),\ \pi_X = \mathsf{Prove}(ck, \mathrm{E}_{\mathcal{DH}}, (X,\xi),(Y,\psi))\big)$$

to her public key $vk = (X,Y) \in \mathcal{DH}$ and wishes to prove that the values in $\mathbf{c}_\Sigma$ are a valid signature on the value $(M,N)$ in $\mathbf{C}$ under the public key that is committed in $\mathbf{c}_{vk}$. The first equation of verification is thus

$$\mathrm{E}_{\widehat{A}}(A, M;\ S, Y, D)\ :\ e(T^{-1}, \underline{S})\, e(\underline{M}, H^{-1})\, e(\underline{A}, \underline{Y}) e(\underline{A}, \underline{D}) = e(K, H)\ \ ,$$

whereas $\mathrm{E}_B$ and $\mathrm{E}_R$ remain unchanged. Given a commitment $\mathbf{C}$ to a message, $\mathbf{c}_\Sigma = (\mathbf{c}_A, \mathbf{c}_B, \mathbf{c}_D, \mathbf{c}_R, \mathbf{c}_S)$, a commitment to a signature, and a proof $\pi = (\pi_A, \pi_B, \pi_R)$ of validity, $\pi_A$ can be adapted to $\pi_{\widehat{A}}$ using Lemma 5 from Sect. 7: set $\pi_{\widehat{A}} \leftarrow \mathsf{RdProof}\big(ck, \mathrm{E}_{\widehat{A}}, (\mathbf{c}_A, 0), (\mathbf{c}_M, 0), (\mathbf{c}_S, 0), (\mathsf{Com}(ck, Y, 0), \psi), (\mathbf{c}_D, 0), \pi_A\big)$. (See Appendix A.3 for the details.) To adapt to a decommitment of $\mathbf{c}_{vk}$, we have to reset the randomness of $\mathbf{c}_Y$ to 0. $\mathsf{AdPrDC}_{\mathcal{K}}$ does thus the converse: it sets $\pi_A \leftarrow \mathsf{RdProof}\big(ck, \mathrm{E}_{\widehat{A}}, (\mathbf{c}_A, 0), (\mathbf{c}_M, 0), (\mathbf{c}_S, 0), (\mathsf{Com}(ck, Y, 0), -\psi), (\mathbf{c}_D, 0), \pi_{\widehat{A}}\big)$.

# 9 Commuting Signatures with Partially Public Messages

## 9.1 Automorphic Signatures on an Integer and a Message

The scheme **Sig** from Sect. 6.4 can be adapted to sign a value from $\mathbb{Z}_p$ and an element from $\mathcal{DH}$ at the same time, as it is required for our application to delegatable credentials. Note that while this requires one extra element in the parameters it does *not* increase the size of a signature.

**Intuition.** ADHSDH states that given "weak signatures" $((K{\cdot}V)^{\frac{1}{x+c}}, F^c, H^c)$ on random messages $(V,W) \in \mathcal{DH}$, it is hard to forge such a signature on a new message. Now to turn this into a CMA secure scheme, Fuchsbauer [Fuc09] implicitly defines a *trapdoor commitment* $\mathsf{TCom}((M,N), r) := M{\cdot}T^r$ with opening $(G^r, H^r)$. The actual signature is then a weak signature on $\mathsf{TCom}((M,N), r)$ together with the opening $(G^r, H^r)$. AWFCDH implies that it is hard to open a $\mathsf{TCom}$ commitment in two different ways, thus $\mathsf{TCom}$ is computationally binding.

In order to sign a message *pair* consisting of an integer value $v$ and a DH-pair $(M,N)$, we replace $\mathsf{TCom}$ by $\mathsf{TCom}''$ having an additional parameter $L$: $\mathsf{TCom}''(v, (M,N), r) := L^v \cdot M \cdot T^r$, which is also computationally binding by AWFCDH: consider an adversary producing $(v, (M,N), (R,S))$ and $(v', (M',N'), (R',S')$ with $\mathsf{TCom}''(v, (M,N), r) = \mathsf{TCom}''(v', (M',N'), r')$; then the case $r \neq r'$ is reducible to AWFCDH—as for $\mathsf{TCom}$—and $r = r'$ is reducible to CDH, which is implied by AWFCDH.

**Scheme 2** ($\mathbf{Sig}''$)**.** $\mathsf{Setup}''_{\mathsf{S}}$ *has input* $grp = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, G, H)$ *and outputs* $grp$ *and additional generators* $F, K, L, T \leftarrow \mathbb{G}_1$. *The message space is* $\mathcal{DH} := \{(G^m, H^m)\,|\, m \in \mathbb{Z}_p\}$.

$\mathsf{KeyGen}''_{\mathsf{S}}$ *chooses* $x \leftarrow \mathbb{Z}_p$ *and outputs* ($vk = \mathsf{VK}(x)$, $sk = x$), *with* $\mathsf{VK}(x) := (G^x, H^x)$

$\mathsf{Sign}''$ *has inputs a secret key* $x$ *and a message* $(v, (M,N)) \in \mathbb{Z}_p \times \mathcal{DH}$. *It chooses random* $c, r \leftarrow \mathbb{Z}_p$ *and outputs*

$$\big(A := (K{\cdot}L^v{\cdot}M{\cdot}T^r)^{\frac{1}{x+c}},\ B := F^c,\ D := H^c,\ R := G^r,\ S := H^r\big)\ \ .$$

$\mathsf{Ver}''$ *on inputs a public key* $(X,Y) \in \mathcal{DH}$*, a message* $(v,(M,N)) \in \mathbb{Z}_p \times \mathcal{DH}$ *and a signature* $(A,B,D,R,S)$ *outputs 1 (and 0 otherwise) iff the following hold:*

$$e(A, Y\cdot D) = e(K\cdot L^v\cdot M, H)\, e(T,S) \qquad e(B,H) = e(F,D) \qquad e(R,H) = e(G,S) \tag{15}$$

**Theorem 2.** *Assuming q-ADHSDH and AWFCDH,* $\mathbf{Sig}_\mathrm{A}$ *is strongly existentially unforgeable against adversaries making at most* $q-1$ *adaptive chosen-message queries.*

A formal proof of Theorem 2 can be found in Appendix B.

### 9.2 Verifiably Encrypting a Signature on a Public Integer and a Committed Message

A commitment to a signature on an integer $v$ and a message committed in $\mathbf{C}$ is of the form $(\mathbf{c}_A, \mathbf{c}_B, \mathbf{c}_D, \mathbf{c}_R, \mathbf{c}_S)$ and a proof of validity is $(\pi_{A''}, \pi_B, \pi_R)$ for equations $\mathrm{E}_B$ and $\mathrm{E}_R$ as in (12) and $\mathrm{E}_{A''}$ defined as

$$\mathrm{E}_{A''}(A,M;\ S,D)\ :\ e(T^{-1},\underline{S})\, e(\underline{A},Y) e(\underline{M},H^{-1})\, e(\underline{A},\underline{D}) = e(K\cdot L^v, H) \tag{16}$$

By Lemma 1, proofs are independent of the right-hand side of the equation, thus $\pi_{A''}$ is defined like $\pi_A$. This also holds for proofs about all other equations such as $\mathrm{E}_{\widetilde{A}}, \mathrm{E}_{\bar{A}}$ and $\mathrm{E}_{\widehat{A}}$ and their variants for $\mathbf{Sig}''$, since going from $\mathbf{Sig}$ to $\mathbf{Sig}''$ only affects the right-hand sides of the equations.

Proofs for $\mathrm{E}_{\mathsf{Ver}(\cdots)}$ and $\mathrm{E}_{\mathsf{Ver}''(\cdots)}$ are thus the same for all combinations of keys, messages and signatures given as commitments or in the clear. This means that the proof-adaptation algorithms $\mathsf{AdPrC}$, $\mathsf{AdPrC}_\mathcal{M}$, $\mathsf{AdPrC}_\mathcal{K}$, $\mathsf{AdPrDC}$, $\mathsf{AdPrDC}_\mathcal{M}$ and $\mathsf{AdPrDC}_\mathcal{K}$ defined in Sect. 8.3 can all be used for proofs about committed $\mathsf{Sig}''$ signatures as well. The only functionality that has to be slightly adapted is $\mathsf{SigCom}$. We define $\mathsf{SigCom}''(\mathit{ck}, \mathit{sk}, v, \mathbf{C})$ as $\mathsf{SigCom}$ in Fig. 2, except that $A$ is defined as $A := (K\cdot L^v\cdot T^r\cdot U)$. We do not need to modify $\mathsf{SmSigCom}$, since $\Sigma$ is given as input to it.

Note that if in the construction of a blind signature in Sect. 4.2 we replace $\mathbf{Sig}$ and $\mathsf{SigCom}$ by $\mathbf{Sig}''$ and $\mathsf{SigCom}''$, respectively, we obtain *partially blind signatures* [AF96], where the signer controls part of the message.

## 10 A Note on Simulatability of Proofs

Groth and Sahai [GS08] show that pairing-product equations with a right-hand side $\mathbf{t}_T$ of the form

$$\mathbf{t}_T \ =\ e(P_1,Q_1)\cdots e(P_n,Q_n) \tag{17}$$

can be simulated: in the witness-indistinguishability setting (i.e., when $\mathit{ck}^* \leftarrow \mathsf{WISetup}$; cf. Sect. 6.2), given as simulation trapdoor *sim* the values $(\alpha_1, t_1, \alpha_2, t_2)$ used to construct $\mathit{ck}^*$ one can construct commitments and proofs of validity for an equation of the above form *without knowing a witness*, i.e., elements that satisfy the equation.

Equations with right-hand side 1 ("*homogeneous equations*") can be simulated directly, since they have a trivial witness. Equations with a non-trivial right-hand side as in (17) must be transformed to a new set of equations to be simulatable: in the original equation the values $P_i$ are replaced by *variables* $V_i$ (which makes the equation homogeneous) and for each $i$ we add the *multi-scalar multiplication equation*[9] $V_i^d \cdot P_i^{-d} = 1$, where the commitment for $d$ will be a *trivial* commitment to 1 (Since the randomness for the commitment of $d$ is 0, we can check that the committed value is 1, which gives us $V_i = P_i$ from the additional equations, and thus soundness of the construction.) In the simulation, we can now set all variables from $\mathbb{G}_1$ and $\mathbb{G}_2$ to 1 (which is a satisfying witness for our transformed PPE), and can thus give commitments and proofs. The additional equations can be simulated, since given the trapdoor *sim*, the commitment to $d$ can be trapdoor-opened to 0 (see [GS08] for the details).

In Sect. 10.2 we show that modifying our verification equations for commuting signatures does not interfere with its functionality; thus we get simulatability, as required for anonymity of our credentials.

[9] An equation of the form $\mathrm{E}(X_1,\ldots,X_m;\ y_1,\ldots,y_n)\ :\ \prod_{i=1}^n A_i^{y_i} \prod_{i=1}^m X_i^{b_i} \prod_{i=1}^m \prod_{j=1}^n X_i^{\gamma_{ij} y_j} = T$, over $X_i \in \mathbb{G}_1$ and $y_j \in \mathbb{Z}_p$, is called multi-scalar-multiplication equation in $\mathbb{G}_1$. [GS08] show how to construct WI proofs for this type of equation.

## 10.1 Simulating Proofs of Knowledge with Given Commitments.

Groth and Sahai show that given *sim* in the WI setting, for any simulatable PPE, a simulator can produce commitments and a proof of validity. However, they do not consider the case where some of the commitments are given to the simulator, i.e., it cannot produce them itself and in particular, it does not know their randomness.

In [BCC$^+$09], to prove anonymity, simulations of this kind are required (see, e.g. the definition of SimProve in Appendix B of the full version). However, the authors do not explain how to achieve such simulation. We will show that the proofs used in our construction can all be simulated even when some of the commitments are fixed in advance.

**Lemma 6.** *Let* $\mathrm{E}$ *be as in* (4) *with* $\mathbf{t}_T = 1$ *and* $A_j = 1$ *for indices* $j \in J \subseteq \{1, \ldots, n\}$. *Given commitments* $\mathbf{d}_j$ *for* $j \in J$*, we can simulate* $\mathbf{c}_1, \ldots, \mathbf{c}_m$ *and* $\mathbf{d}_j$ *for* $j \notin J$ *and a proof* $\pi$ *for* $\mathrm{E}$ *and* $(\mathbf{c}_1, \ldots, \mathbf{c}_m, \mathbf{d}_1, \ldots, \mathbf{d}_m)$ *if we are given the simulation trapdoor sim for* $ck^*$. *A symmetric result holds for* $\mathbf{c}_i$ *and* $\mathbf{d}_j$ *interchanged, and* $A_j$ *replaced with* $B_i$.

*Proof.* If the simulator can choose all the commitments $\mathbf{c}_i$ and $\mathbf{d}_j$, it sets the committed values to 1. Since these values satisfy an homogeneous equation, the simulator can make an honest proof using the randomness for the commitments. But if the commitments $(\mathbf{d}_j)_{j\in J}$ are fixed and given to the simulator, it does not know the randomness $s_j$ s.t. $\mathbf{d}_j = \mathsf{Com}(ck^*, 1, s_j)$ for all $j \in J$! We show that the proof can nontheless be construct. Let us look at the definition of $\mathsf{Prove}\big(ck, \mathrm{E}, (X_i, r_i)_{i=1}^m, (Y_j, s_j)_{j=1}^n;\, Z\big)$ in (7). For the case when $X_i = 1 = Y_j$ we have

$$t_{11} := \textstyle\sum_{j=1}^n \sum_{i=1}^m r_{i1}\gamma_{ij}s_{j1} \qquad\qquad t_{12} := \sum_{j=1}^n \sum_{i=1}^m r_{i1}\gamma_{ij}s_{j2}$$

$$t_{21} := \textstyle\sum_{j=1}^n \sum_{i=1}^m r_{i2}\gamma_{ij}s_{j1} \qquad\qquad t_{22} := \sum_{j=1}^n \sum_{i=1}^m r_{i2}\gamma_{ij}s_{j2}$$

$$\phi := \begin{bmatrix} v_{11}^{t_{11}} v_{21}^{t_{12}} & \big(\prod_{i=1}^m B_i^{r_{i1}}\big) v_{12}^{t_{11}} v_{22}^{t_{12}} \\ v_{11}^{t_{21}} v_{21}^{t_{22}} & \big(\prod_{i=1}^m B_i^{r_{i2}}\big) v_{12}^{t_{21}} v_{22}^{t_{22}} \end{bmatrix} \odot (Z \otimes \vec{\mathbf{v}}) \qquad \theta := \begin{bmatrix} 1 & \big(\prod_{j=1}^n A_j^{s_{j1}}\big) \\ 1 & \big(\prod_{j=1}^n A_j^{s_{j2}}\big) \end{bmatrix} \odot (Z \otimes \vec{\mathbf{u}})$$

which has to be constructed without knowledge of $(s_j)_{j\in J}$, i.e., the values satisfying $\mathbf{d}_j = \mathsf{Com}(ck^*, 1, s_j)$. Let the simulation trapdoor $sim = (\alpha_1, \alpha_2, \beta_1, \beta_2)$ be s.t. $\mathbf{v}_1 = (G_2, G_2^{\alpha_2})$ and $\mathbf{v}_2 = (G_2^{\beta_2}, G_2^{\alpha_2\beta_2 - 1})$ (see Sect. 6.2). Let $(k_j, l_j)$ be the (unknown) logarithms of $\mathbf{d}_j$, i.e., $\mathbf{d}_{j,1} = G_2^{k_j}$ and $\mathbf{d}_{j,2} = G_2^{l_j}$. Then we have

$$(G_2^{k_j}, G_2^{l_j}) = \mathbf{d}_j = \mathsf{Com}(ck^*, 1, s_j) = (v_{11}^{s_{j1}} v_{21}^{s_{j2}}, v_{12}^{s_{j1}} v_{22}^{s_{j2}}) = (G_2^{s_{j1} + \beta_1 s_{j2}}, G_2^{\alpha_2 s_{j1} + \alpha_2\beta_2 s_{j2} - s_{j2}})$$

Solving for $s_{j1}$ and $s_{j2}$ we get $s_{j1} = k_j - \alpha_2\beta_2 k_j + \beta_2 l_j$ and $s_{j2} = \alpha_2 k_j - l_j$. The simulator can thus compute

$$G_2^{s_{j1}} = \mathbf{d}_{j1}^{(1-\alpha_2\beta_2)} \cdot \mathbf{d}_{j2}^{\beta_2} \qquad\qquad G_2^{s_{j2}} = \mathbf{d}_{j1}^{\alpha_2} \cdot \mathbf{d}_{j2}^{-1} \tag{18}$$

and use these values to compute $\phi$, since it knows the logarithms of all $v_{ij}$ as well as all $r_{ij}$ and $\gamma_{ij}$, and $\theta$, e.g.

$$\begin{aligned} v_{22}^{t_{22}} &= v_{22}^{\sum_{j=1}^n s_{j2} \sum_{i=1}^m r_{i2}\gamma_{ij}} = \big(G_2^{\sum_{j=1}^n s_{j2} \sum_{i=1}^m r_{i2}\gamma_{ij}}\big)^{\alpha_2\beta_2 - 1} = \textstyle\prod_{j=1}^n (G_2^{s_{j2}})^{(\alpha_2\beta_2-1)\sum_{i=1}^m r_{i2}\gamma_{ij}} \\ &= \big(\textstyle\prod_{j\in J} (\mathbf{d}_{j1}^{\alpha_2} \cdot \mathbf{d}_{j2}^{-1})^{(\alpha_2\beta_2-1)\sum_{i=1}^m r_{i2}\gamma_{ij}}\big)\big(\prod_{j\notin J} G_2^{s_{j2}(\alpha_2\beta_2-1)\sum_{i=1}^m r_{i2}\gamma_{ij}}\big) \enspace . \end{aligned}$$

Since $A_j = 1$ for $j \in J$, it is straightforward to compute $\theta$. □

The above lemma lets us simulate a committed message, a committed signature and a proof of validity for a *given* committed public key (since in $\mathrm{E}_{\widehat{A}''_S}$ given in (19) below, the (implicit) constant that is paired with $Y$ is 1, thus the premise of the lemma is satisfied). To prove anonymity of our construction of a delegatable-credential scheme in Sect. 5 we moreover need to simulate a verifiably encrypted signature on a given committed message; this requires simulation of a proof for equation $\mathrm{E}_{\widehat{A}''_S}$, where the constant $(H^{-1})$ that is paired with the value whose commitment is given $(M)$ is not trivial; however, its logarithm $-1$ is known to the simulator.

We give a strengthening of Lemma 6, where the $A_j$ are of the form $G_1^{a_j}$ with $a_j$ known to the simulator.

**Lemma 7.** *Let* $\mathrm{E}$ *be as in* (4) *with* $\mathbf{t}_T = 1$ *and* $A_j = G_1^{a_j}$ *(with* $a_j$ *known) for indices* $j \in J$*. Given commitments* $\mathbf{d}_j$ *for* $j \in J$*, we can simulate* $\mathbf{c}_1, \dots, \mathbf{c}_n$ *and* $\mathbf{d}_j$ *for* $j \notin J$ *and a proof* $\pi$ *for* $\mathrm{E}$ *and* $(\mathbf{c}_1, \dots, \mathbf{c}_m, \mathbf{d}_1, \dots, \mathbf{d}_m)$ *if we are given the simulation trapdoor sim for ck***. A symmetric result holds for* $\mathbf{c}_i$ *and* $\mathbf{d}_j$ *interchanged, and* $A_j = G_1^{a_j}$ *replaced with* $B_i = G_2^{b_i}$*.*

*Proof.* For simplicity, we give a proof in the additive notation of Remark 4. Since $\vec{X} = (0, \dots, 0)^\top = \vec{Y}$, we have

$$\vec{\mathbf{c}} = R\vec{\mathbf{u}} \qquad\qquad \phi = R^\top \iota(\vec{B}) + (R^\top \Gamma S - Z^\top)\vec{\mathbf{v}}$$
$$\vec{\mathbf{d}} = S\vec{\mathbf{v}} \qquad\qquad \theta = S^\top \iota(\vec{A}) + Z\vec{\mathbf{u}}$$

Let us denote by $\vec{A}'$ the vector $\vec{A}$ where all $A_j$ with $j \notin J$ are replaced by 0, and by $\vec{A}''$ the vector $\vec{A}$ where all $A_j$ with $j \in J$ are replaced by 0, and let $S'$ and $S''$ be defined analogously. We have $\vec{A} = \vec{A}' + \vec{A}''$ and $S = S' + S''$, and moreover $S^\top \iota(\vec{A}) = (S')^\top \iota(\vec{A}') + (S'')^\top \iota(\vec{A}'')$. Note that the simulator only knows the logarithms of $\vec{A}'$ and the values in $S''$.

Since in the WI setting the matrix $\vec{\mathbf{u}}$ is invertible, there exists $\Omega \in \mathbb{Z}_p^{2\times 2}$ s.t. $\iota(\vec{A}') = \Omega\vec{\mathbf{u}}$. The logarithms of $\iota(\vec{A}')$ and $\vec{\mathbf{u}}$ being known to the simulator, it can efficiently compute $\Omega$. We now show how the simulator computes the proof $(\phi, \theta)$: it chooses $\widehat{Z} \leftarrow \mathbb{Z}_p^{2\times 2}$ and (knowing the values in $S''$) computes $\theta := (S'')^\top \iota(\vec{A}'') + \widehat{Z}\vec{\mathbf{u}} = S^\top \iota(\vec{A}) - (S')^\top \iota(\vec{A}') + \widehat{Z}\vec{\mathbf{u}}$, which is a proof $\theta = S^\top \iota(\vec{A}) + Z\vec{\mathbf{u}}$ with $Z := \widehat{Z} - (S')^\top \Omega$. The first part of the proof is then

$$\phi = R^\top \iota(\vec{B}) + (R^\top \Gamma S - \widehat{Z}^\top + \Omega^\top S')\,\vec{\mathbf{v}} \ ,$$

which can be constructed using the techniques of the proof of Lemma 6: it suffices to construct the elements in (18) and use the known logarithms of $\vec{\mathbf{v}}$ as well as the known values $R$ and $\Omega$. □

## 10.2 Making the Equations for $\mathsf{Ver}''$ Simulatable

In our application in Sect. 5.2 we have to simulate proofs for the equations of $\mathrm{E}_{\mathsf{Ver}''(\cdot,v,\cdot,\cdot)}(vk, (M,N), \Sigma)$, *when the commitments for* $vk = (X,Y)$ *or* $(M,N)$ *(or both!) are given to the simulator.*

While $\mathrm{E}_B$ and $\mathrm{E}_R$ from (12) have a trivial right-hand side, we replace $\mathrm{E}_{A''}$ from (16) by the equations

$$\mathrm{E}_{A''_S}(A;\, W, S, N, D) \ :\ e(K\cdot L^v, \underline{W})\, e(T^{-1}, \underline{S})\, e(G^{-1}, \underline{N})\, e(\underline{A}, Y)\, e(\underline{A}, \underline{D}) = 1$$
$$\mathrm{E}_{d_S}(d;\, W) \ :\ W^d \cdot H^{-d} = 1$$

where, besides transforming $\mathrm{E}_{A''}$ into a homogeneous equation and a multi-scalar multiplication equation[10] as described by [GS08], we replaced $e(M, H^{-1})$ by $e(G^{-1}, N)$ which by $\mathrm{E}_M(M; N)$ is equal. Accordingly, we replace $\mathrm{E}_U$ by

$$\mathrm{E}_{U_S}(Q, N) \ :\ e(T^{-1}, \underline{Q})\, e(G^{-1}, \underline{N}) = e(U, H)^{-1}$$

which, together with $\mathrm{E}_M$ and $\mathrm{E}_P$, still asserts that $U = T^t \cdot M$. Note that in addition, this equation is *linear* in the sense of Groth-Sahai and a proof $\pi_{U_S}$ thus reduces to an element from $\mathbb{G}_1^2$, whereas $\pi_U \in \mathbb{G}_1^4 \times \mathbb{G}_2^4$.

Next, we show how to adapt $\mathsf{SigCom}''$ (which is the algorithm in Fig. 2 with $A$ defined as $(K\cdot L^v \cdot T^r \cdot U)^{\frac{1}{x+c}}$ at the beginning). All that needs to be done to define $\mathsf{SigCom}''_S$ is replacing $\mathrm{E}_{\mathcal{A}^\dagger}$ by

$$\mathrm{E}_{A^\dagger_S}(A;\, W, D) \ :\ e(K\cdot L^v, \underline{W})\, e(\underline{A}, Y)\, e(\underline{A}, \underline{D}) = 1 \ .$$

Setting $\pi'_{A_S} \leftarrow \pi_{U_S} \circledcirc \mathsf{Prove}(ck, \mathrm{E}_{A^\dagger_S}, (A, \alpha), (W, \omega), (H^c, \delta))$ in Fig. 2 yields thus a proof for $\mathrm{E}_{A''_S}$ by Lemma 3, since the product of the left-hand sides of $\mathrm{E}_{A^\dagger_S}$ and $\mathrm{E}_{U_S}$ is the left-hand side of $\mathrm{E}_{A''_S}$. The proof for the additional (multi-scalar multiplication) equation can be produced by the signer herself.

We demonstrated how our instantiation of commuting signatures based on $\mathbf{Sig}''$ can be adapted to make the equations for $\mathsf{Ver}''$ simulatable. Below, we show that they can even be simulated when commitments to the verification key and/or the message are given to the simulator.

[10] We chose to turn $H$ into a variable, since proofs for equations in $\mathbb{G}_2$ are in $\mathbb{G}_1^4 \times \mathbb{G}_2^2$ and thus smaller than proofs for equations in $\mathbb{G}_1$.

**Simulating** $\mathsf{Com}_{\mathcal{M}}$**.** When the simulator needs to simulate a $\mathsf{Com}_{\mathcal{M}}$ commitment it does the following: set $\mathbf{c}_M$ and $\mathbf{c}_N$ to commitments to 1. This enables simulation of other proofs for equations about $M$ (such as those in $\mathsf{Ver}''$). Since $\mathsf{Com}_{\mathcal{M}}$ also contains the value $U = T^t \cdot M$, the simulator has to choose $t$ randomly, which defines $P$ and $Q$. Now the simulator can produce $\mathbf{c}_P, \mathbf{c}_Q, \pi_M, \pi_P, \pi_U$ honestly. Note that the fact that $\mathbf{c}_P$ and $\mathbf{c}_Q$ were not produced as commitments to 1 is not a problem, as they are never used outside of a $\mathsf{Com}_{\mathcal{M}}$ commitment.

**Simulating** $\mathsf{Ver}''(\cdot,\cdot,\cdot)$ **for Fixed Commitments.** In the proof of anonymity of our credential scheme, we have to construct algorithms SimCredProve and SimIssue that output Groth-Sahai proofs without being given witnesses. The proofs are for validity of certificates contained in credentials, thus about the equations in $\mathsf{Ver}''$ from Scheme 2. The only equation that contains parts of a verification key or the message of is the following

$$\mathrm{E}_{\widehat{A}''_S}(A;\, W,S,N,Y,D) \;:\; e(K\cdot L^v, \underline{W})\, e(T^{-1}, \underline{S})\, e(G^{-1}, \underline{N})\, e(\underline{A}, \underline{Y})\, e(\underline{A}, \underline{D}) = 1 \;. \tag{19}$$

**Corollary 1.** *Given commitments* $\mathbf{c}_{vk}$ *and* $\mathbf{C}$*, the simulator can produce* $(\mathbf{c}_\Sigma, \widehat{\pi})$ *that is distributed as*

$$\Big[\Sigma \leftarrow \mathsf{Sign}''(sk, v, (M,N));\ \rho \leftarrow \mathcal{R}\ :\ \big(\mathsf{Com}(ck^*, \Sigma, \rho),\ \mathsf{Prove}\big(ck^*, \mathrm{E}_{\mathsf{Ver}''(\cdot,v,\cdot,\cdot)}, ((X,\xi),(Y,\psi)), ((M,\mu),(N,\nu)), (\Sigma,\rho)\big)\big)\Big]\ ,$$

*where* $vk = (X,Y)$ *and* $(\xi,\psi)$ *are such that* $\mathbf{c}_{vk} = \mathsf{Com}(ck^*, vk, (\xi,\psi))$*, sk is such that* $vk = \mathsf{VK}(sk)$*, and* $M, N, \mu$ *and* $\nu$ *are such that* $\mathbf{C} = (\mathsf{Com}(ck^*, M, \mu), \mathsf{Com}(ck^*, N, \nu), \ldots)$*.*

*Proof.* Simulating $\mathsf{Ver}''$ means simulating $\mathrm{E}_{\widehat{A}''_S}, \mathrm{E}_{d_S}, \mathrm{E}_B$ and $\mathrm{E}_R$. The simulator makes commitments $\mathbf{c}_\Sigma$ to $(1,\ldots,1)$. Proofs $\pi_B$ and $\pi_R$ are computed honestly and the first equation satisfies the premises of Lemma 7: the constants (the "$A_j$" in (4)) that are paired with $N$ and $Y$ are $G^{-1}$ and 1, respectively, and thus have known logarithms. $\pi_{\widehat{A}''_S}$ can thus be simulated. $\pi_{d_S}$ is simulated by opening $\mathbf{c}_d := \mathsf{Com}(ck^*, 1, 0)$ to 0, as described in [GS08]. □

## 11 Conclusions

In this paper we defined and instantiated a new primitive we call *commuting signatures*. They allow users to encrypt different components of a triple consisting of a verification key, a message and a signature, and prove validity of the encrypted values. Most importantly, they enable signers that are given an encrypted message to produce an encryption of a signature on it together with a proof of validity.

We showed that this primitive enables the first instantiation of delegatable anonymous credentials with non-interactive issuing and delegation. Moreover, using our instantiation, the efficiency of the credential scheme improves significantly compared to the (only) previous instantiation. We believe that commuting signatures are an important tool in the construction of privacy-preserving primitives and that they will find further applications.

## Acknowledgments

This work was supported by EADS, the French ANR 07-TCOM-013-04 PACE Project, and the European Commission through the ICT Program under Contract ICT-2007-216646 ECRYPT II.

## References

[AF96] Masayuki Abe and Eiichiro Fujisaki. How to date blind signatures. In Kwangjo Kim and Tsutomu Matsumoto, editors, *ASIACRYPT'96*, volume 1163 of *LNCS*, pages 244–251. Springer, November 1996.

[BB04] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 56–73. Springer, May 2004.

[BCC+09] Mira Belenkiy, Jan Camenisch, Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Hovav Shacham. Randomizable proofs and delegatable anonymous credentials. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 108–125. Springer, August 2009. Full version available at `http://eprint.iacr.org/2008/428`.

[BCKL08] Mira Belenkiy, Melissa Chase, Markulf Kohlweiss, and Anna Lysyanskaya. P-signatures and noninteractive anonymous credentials. In Ran Canetti, editor, *TCC 2008*, volume 4948 of *LNCS*, pages 356–374. Springer, March 2008.

[BFI+10] Olivier Blazy, Georg Fuchsbauer, Malika Izabachène, Amandine Jambert, Hervé Sibert, and Damien Vergnaud. Batch Groth-Sahai. Cryptology ePrint Archive, Report 2010/040, 2010. `http://eprint.iacr.org/`.

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications. In *STOC*, pages 103–112. ACM Press, 1988.

[BGLS03] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 416–432. Springer, May 2003.

[BHY09] Mihir Bellare, Dennis Hofheinz, and Scott Yilek. Possibility and impossibility results for encryption and commitment secure under selective opening. In Antoine Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 1–35. Springer, April 2009.

[Boy08] Xavier Boyen. The uber-assumption family (invited talk). In Steven D. Galbraith and Kenneth G. Paterson, editors, *PAIRING 2008*, volume 5209 of *LNCS*, pages 39–56. Springer, September 2008.

[Bra99] Stefan Brands. Rethinking public key infrastructure and digital certificates—building privacy. PhD thesis, Eindhoven Inst. of Tech., The Netherlands, 1999.

[BW07] Xavier Boyen and Brent Waters. Full-domain subgroup hiding and constant-size group signatures. In Tatsuaki Okamoto and Xiaoyun Wang, editors, *PKC 2007*, volume 4450 of *LNCS*, pages 1–15. Springer, April 2007.

[Cha82] David Chaum. Blind signatures for untraceable payments. In *CRYPTO*, pages 199–203, 1982.

[Cha85] David Chaum. Security without identification: Transaction systems to make big brother obsolete. *Commun. ACM*, 28(10):1030–1044, 1985.

[CHK+06] Jan Camenisch, Susan Hohenberger, Markulf Kohlweiss, Anna Lysyanskaya, and Mira Meyerovich. How to win the clonewars: Efficient periodic n-times anonymous authentication. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 06*, pages 201–210. ACM Press, October / November 2006.

[CL01] Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 93–118. Springer, May 2001.

[CL02] Jan Camenisch and Anna Lysyanskaya. A signature scheme with efficient protocols. In Stelvio Cimato, Clemente Galdi, and Giuseppe Persiano, editors, *SCN 02*, volume 2576 of *LNCS*, pages 268–289. Springer, September 2002.

[CL04] Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 56–72. Springer, August 2004.

[CL06] Melissa Chase and Anna Lysyanskaya. On signatures of knowledge. In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 78–96. Springer, August 2006.

[Dam90] Ivan Damgård. Payment systems and credential mechanisms with provable security against abuse by individuals. In Shafi Goldwasser, editor, *CRYPTO'88*, volume 403 of *LNCS*, pages 328–335. Springer, August 1990.

[Fis06] Marc Fischlin. Round-optimal composable blind signatures in the common reference string model. In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 60–77. Springer, August 2006.

[FP08] Georg Fuchsbauer and David Pointcheval. Anonymous proxy signatures. In Rafail Ostrovsky, Roberto De Prisco, and Ivan Visconti, editors, *SCN 08*, volume 5229 of *LNCS*, pages 201–217. Springer, September 2008.

[FPV09] Georg Fuchsbauer, David Pointcheval, and Damien Vergnaud. Transferable constant-size fair e-cash. In Juan A. Garay, Atsuko Miyaji, and Akira Otsuka, editors, *CANS 09*, volume 5888 of *LNCS*, pages 226–247. Springer, December 2009. Full version available at `http://eprint.iacr.org/2009/146`.

[Fuc09] Georg Fuchsbauer. Automorphic signatures in bilinear groups. Cryptology ePrint Archive, Report 2009/320, 2009. `http://eprint.iacr.org/`.

[GPS08] Steven D. Galbraith, Kenneth G. Paterson, and Nigel P. Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, 156(16):3113–3121, 2008.

[GS08] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 415–432. Springer, April 2008. Full version available at `http://eprint.iacr.org/2007/155`.

[LRSW00] Anna Lysyanskaya, Ronald L. Rivest, Amit Sahai, and Stefan Wolf. Pseudonym systems. In Howard M. Heys and Carlisle M. Adams, editors, *SAC 1999*, volume 1758 of *LNCS*, pages 184–199. Springer, August 2000.

[RS09] Markus Rückert and Dominique Schröder. Security of verifiably encrypted signatures and a construction without random oracles. In Hovav Shacham and Brent Waters, editors, *PAIRING 2009*, volume 5671 of *LNCS*, pages 17–34. Springer, August 2009.

# A Details and Proofs of our Commuting-Signature Instantiation

## A.1 Commitments on Messages

We define a *commitment* on a message $(M, N) \in \mathcal{DH}$ as the values $\mathbf{C} = (\mathbf{c}_M, \mathbf{c}_N, \pi_M, \mathbf{c}_P, \mathbf{c}_Q, \pi_P, U, \pi_U)$ that the user sends to the signer in the issuing protocol for blind signatures from [Fuc09]. We then show how to randomize a commitment and how to extract the committed value. Since the committed values are the messages of **Sig**, the algorithms also get the parameters $pp_{\mathrm{S}}$ as input.

$\mathsf{Com}_{\mathcal{M}}$ has inputs $pp = (ck, grp, F, K, T)$, $(M, N) \in \mathcal{DH}$ and $(t, \mu, \nu, \rho, \sigma) \in \mathbb{Z}_p^9 =: \mathcal{R}_{\mathcal{M}}$. Recall that $ck$ is composed of $\vec{\mathbf{u}} = (\mathbf{u}_1, \mathbf{u}_2) \in \mathbb{G}_1^{2\times 2}$ and $\vec{\mathbf{v}} = (\mathbf{v}_1, \mathbf{v}_2) \in \mathbb{G}_2^{2\times 2}$. We define the following equations:

$$\mathrm{E}_{\mathcal{DH}}(M, N) \; : \; e(G^{-1}, \underline{N})\, e(\underline{M}, H) = 1 \tag{20}$$

$$\mathrm{E}_U(M, Q) \; : \; e(T^{-1}, \underline{Q})\, e(\underline{M}, H^{-1}) = e(U, H)^{-1} \tag{21}$$

$\mathsf{Com}_{\mathcal{M}}(pp, (M, N), (t, \mu, \nu, \rho, \sigma))$ defines $P = G^t$ and $Q = H^t$, computes

$$\begin{aligned}
\mathbf{c}_M &:= \mathsf{Com}(ck, M, \mu) = (u_{11}^{\mu_1} u_{21}^{\mu_2}, M u_{12}^{\mu_1} u_{22}^{\mu_2}) & \mathbf{c}_N &:= \mathsf{Com}(ck, N, \nu) = (v_{11}^{\nu_1} v_{21}^{\nu_2}, N v_{12}^{\nu_1} v_{22}^{\nu_2}) \\
\mathbf{c}_P &:= \mathsf{Com}(ck, P, \rho) = (u_{11}^{\rho_1} u_{21}^{\rho_2}, P u_{12}^{\rho_1} u_{22}^{\rho_2}) & \mathbf{c}_Q &:= \mathsf{Com}(ck, Q, \sigma) = (v_{11}^{\sigma_1} v_{21}^{\sigma_2}, Q v_{12}^{\sigma_1} v_{22}^{\sigma_2}) \\
\pi_M &\leftarrow \mathsf{Prove}\big(ck, \mathrm{E}_{\mathcal{DH}},\ (M, \mu), (N, \nu)\big) & \pi_P &\leftarrow \mathsf{Prove}\big(ck, \mathrm{E}_{\mathcal{DH}},\ (P, \rho), (Q, \sigma)\big) \\
U &:= T^t \cdot M & \pi_U &\leftarrow \mathsf{Prove}\big(ck, \mathrm{E}_U,\ (M, \mu), (Q, \sigma)\big)
\end{aligned}$$

and returns $\mathbf{C} = (\mathbf{c}_M, \mathbf{c}_N, \pi_M, \mathbf{c}_P, \mathbf{c}_Q, \pi_P, U, \pi_U) \in \mathcal{C}_{\mathcal{M}}$.

$\mathsf{RdCom}_{\mathcal{M}}$ on input $(ck, pp_{\mathrm{S}})$, $\mathbf{C}$ and $(t', \mu', \nu', \rho', \sigma')$, the algorithm first defines $\hat{\mathbf{c}}_P := \mathbf{c}_P \odot (1, G^{t'})$, $\hat{\mathbf{c}}_Q := \mathbf{c}_Q \odot (1, H^{t'})$ and $U' := U \cdot T^{t'}$. It sets

$$\begin{aligned}
\mathbf{c}'_M &:= \mathsf{RdCom}(ck, \mathbf{c}_M, \mu') & \pi'_M &\leftarrow \mathsf{RdProof}(ck, \mathrm{E}_{\mathcal{DH}}, (\mathbf{c}_M, \mu'), (\mathbf{c}_N, \nu'), \pi_M) \\
\mathbf{c}'_N &:= \mathsf{RdCom}(ck, \mathbf{c}_N, \nu') & \pi'_P &\leftarrow \mathsf{RdProof}(ck, \mathrm{E}_{\mathcal{DH}}, (\hat{\mathbf{c}}_P, \rho'), (\hat{\mathbf{c}}_Q, \sigma'), \pi_P) \\
\mathbf{c}'_P &:= \mathsf{RdCom}(ck, \hat{\mathbf{c}}_P, \rho') & \pi'_U &\leftarrow \mathsf{RdProof}(ck, \mathrm{E}_U, (\mathbf{c}_M, \mu'), (\hat{\mathbf{c}}_Q, \sigma'), \pi_U) \\
\mathbf{c}'_Q &:= \mathsf{RdCom}(ck, \hat{\mathbf{c}}_Q, \sigma') & &
\end{aligned}$$

and returns $\mathbf{C}' = (\mathbf{c}'_M, \mathbf{c}'_N, \pi'_M, \mathbf{c}'_P, \mathbf{c}'_Q, \pi'_P, U', \pi'_U) \in \mathcal{C}_{\mathcal{M}}$.

$\mathsf{Extr}_{\mathcal{M}}$ has inputs $ek$ and $\mathbf{C}$. It returns $\big(\mathsf{Extr}(ek, \mathbf{c}_M), \mathsf{Extr}(ek, \mathbf{c}_N)\big)$.

$\mathbf{C} \in \mathcal{C}_{\mathcal{M}}$ is efficiently verifiable by parsing it as $(\mathbf{c}_M, \mathbf{c}_N, \pi_M, \mathbf{c}_P, \mathbf{c}_Q, \pi_P, U, \pi_U)$ and checking the proofs $\pi_M, \pi_P$ and $\pi_U$.

**Theorem 3.** $\mathbf{Com}_{\mathcal{M}}$ *is a randomizable extractable commitment scheme that is perfectly binding and computationally hiding.*

*Proof.* The commitment $\mathbf{C} = (\mathbf{c}_M, \mathbf{c}_N, \pi_M, \mathbf{c}_P, \mathbf{c}_Q, \pi_P, U, \pi_U)$ is binding by the corresponding property of SXDH commitments. A correctly constructed commitment contains valid proofs; in particular, we have $e(U, H) = e(T^t, H)\, e(M, H) = e(T, Q)\, e(M, H)$, thus (21) is satisfied.

The scheme is computationally hiding as defined Sect. 3.1: let $ck^* \leftarrow \mathsf{WISetup}$. Then for every $(M, N) \in \mathcal{DH}$ there exists $t$ s.t. $U = T^t \cdot M$. Moreover there exist $\mu, \nu, \rho$ and $\sigma$ s.t. $\mathbf{c}_M = \mathsf{Com}(ck, M, \mu)$, $\mathbf{c}_N := \mathsf{Com}(ck, N, \nu)$, $\mathbf{c}_P := \mathsf{Com}(ck, G^t, \rho)$, and $\mathbf{c}_Q := \mathsf{Com}(ck, H^t, \sigma)$. So for every $\mathbf{C}$ and every $(M, N) \in \mathcal{DH}$ there exists $r := (t, \mu, \nu, \rho, \sigma) \in \mathcal{R}_{\mathcal{M}}$ s.t. $\mathbf{C} = \mathsf{Com}_{\mathcal{M}}(ck^*, (M, N), r)$.

Moreover, $\mathsf{RdCom}_{\mathcal{M}}$ randomizes a commitment. When $(U, \mathbf{c}_P, \mathbf{c}_Q)$ is replaced by $(U', \hat{\mathbf{c}}_P, \hat{\mathbf{c}}_Q)$ in the first step, $t$ is replaced by $t + t'$ (since the commitments are homomorphic, $\hat{\mathbf{c}}_P$ is a commitment to $P \cdot G^{t'}$ and $\mathbf{c}'_Q$ commits to $Q \cdot H^{t'}$; note that $\pi_P$ and $\pi_U$ do not depend on $t$ but only on the randomness of the commitments—which is not changed in the first step.) In the second step, $(\mu, \nu, \rho, \sigma)$ is replaced by $(\mu + \mu', \nu + \nu', \rho + \rho', \sigma + \sigma')$. $\square$

### A.2 Making Committents to a Signature and a Proof of Validity

We show how the signer can use the values in $\mathbf{C}$ to produce a proof of knowledge

$$(\mathbf{c}_A, \mathbf{c}_B, \mathbf{c}_D, \mathbf{c}_R, \mathbf{c}_S, \pi_A, \pi_B, \pi_R) \in \mathbb{G}_1^{18} \times \mathbb{G}_2^{16}$$

of a signature $(A, B, D, R, S)$ where $\pi_A, \pi_B$ and $\pi_R$ are proofs that the committed values satisfy the equations in (9), respectively, i.e.

$$\begin{aligned} \mathrm{E}_A(A, M;\ S, D)\ &:\ e(T^{-1}, \underline{S})\, e(\underline{A}, Y) e(\underline{M}, H^{-1})\, e(\underline{A}, \underline{D}) = e(K, H) \\ \mathrm{E}_B(B;\ D)\ &:\ e(F^{-1}, \underline{D})\, e(\underline{B}, H) = 1 \\ \mathrm{E}_R(R;\ S)\ &:\ e(G^{-1}, \underline{S})\, e(\underline{R}, H) = 1 \end{aligned} \tag{22}$$

Instantiating the formulas from Sect. 6.3, for equations $\mathrm{E}_{\mathcal{DH}}$ and $\mathrm{E}_U$ (as defined in (20) and (21)), the proofs $\pi_M, \pi_P$ and $\pi_U$ are computed by choosing random $Z_M, Z_P, Z_U \leftarrow \mathbb{Z}_p^{2\times 2}$ and (using the notation defined in (5)) setting

$$\begin{aligned} \pi_{M,1} &= \begin{bmatrix} 1 & H^{\mu_1} \\ 1 & H^{\mu_2} \end{bmatrix} \odot (Z_M \otimes \vec{\mathbf{v}}) \quad \pi_{P,1} = \begin{bmatrix} 1 & H^{\rho_1} \\ 1 & H^{\rho_2} \end{bmatrix} \odot (Z_P \otimes \vec{\mathbf{v}}) \quad \pi_{U,1} = \begin{bmatrix} 1 & H^{-\mu_1} \\ 1 & H^{-\mu_2} \end{bmatrix} \odot (Z_U \otimes \vec{\mathbf{v}}) \\ \pi_{M,2} &= \begin{bmatrix} 1 & G^{-\nu_1} \\ 1 & G^{-\nu_2} \end{bmatrix} \odot (Z_M \otimes \vec{\mathbf{u}}) \quad \pi_{P,2} = \begin{bmatrix} 1 & G^{-\sigma_1} \\ 1 & G^{-\sigma_2} \end{bmatrix} \odot (Z_P \otimes \vec{\mathbf{u}}) \quad \pi_{U,2} = \begin{bmatrix} 1 & T^{-\sigma_1} \\ 1 & T^{-\sigma_2} \end{bmatrix} \odot (Z_U \otimes \vec{\mathbf{u}}) \end{aligned} \tag{23}$$

In the blind signature from [Fuc09], on receiving $\mathbf{C}$, the signer checks the proofs contained in it, and then produces a pre-signature by choosing $c, r \leftarrow \mathbb{Z}_p$ and computing

$$A := (K \cdot T^r \cdot U)^{\frac{1}{x+c}} \qquad B := F^c \qquad D := H^c \qquad R' := G^r \qquad S' := H^r$$

Knowing $t$ s.t. $U = T^t \cdot M$, these values can be turned into a signature on $(M, N)$ by setting $R := R' \cdot G^t$ and $S := S' \cdot H^t$. (Because $A = (K \cdot T^r \cdot U)^{1/(x+c)} = (K \cdot T^{r+t} \cdot M)^{1/(x+c)}$, $R = G^{r+t}$, and $S = H^{r+t}$.) Since the commitments are homomorphic, the signer can—without knowledge of the values $P = G^t$ and $Q = H^t$—make *commitments* on $R$ and $S$:

$$\mathbf{c}_R := \mathbf{c}_P \odot \mathsf{Com}(ck, R', 0) = \mathsf{Com}(ck, R, \rho) \qquad \mathbf{c}_S := \mathbf{c}_Q \odot \mathsf{Com}(ck, S', 0) = \mathsf{Com}(ck, S, \sigma)$$

The signer also chooses $\alpha, \beta, \delta \leftarrow \mathbb{Z}_p^2$, and makes the remaining commitments:

$$\mathbf{c}_A := \mathsf{Com}(ck, A, \alpha) \qquad \mathbf{c}_C := \mathsf{Com}(ck, B, \beta) \qquad \mathbf{c}_D := \mathsf{Com}(ck, D, \delta)$$

The vector $\vec{\mathbf{c}}_\Sigma := (\mathbf{c}_A, \mathbf{c}_B, \mathbf{c}_D, \mathbf{c}_R, \mathbf{c}_S)$ are thus commitments on the *actual* signature $\Sigma = (A, B, D, R, S)$. It remains to construct proofs $\pi_A, \pi_B$ and $\pi_R$ that the committed values satisfy the 3 equations in (22). Instantiating the proofs given in (7) for the concrete equations, we get the following:

$$\begin{aligned} \pi_{A,1} &= \begin{bmatrix} v_{11}^{\alpha_1\delta_1} v_{21}^{\alpha_1\delta_2} & (YD)^{\alpha_1} H^{-\mu_1} v_{12}^{\alpha_1\delta_1} v_{22}^{\alpha_1\delta_2} \\ v_{11}^{\alpha_2\delta_1} v_{21}^{\alpha_2\delta_2} & (YD)^{\alpha_2} H^{-\mu_2} v_{12}^{\alpha_2\delta_1} v_{22}^{\alpha_2\delta_2} \end{bmatrix} \odot (Z_A \otimes \vec{\mathbf{v}}) \qquad \pi_{R,1} = \begin{bmatrix} 1 & H^{\rho_1} \\ 1 & H^{\rho_2} \end{bmatrix} \odot (Z_R \otimes \vec{\mathbf{v}}) \\ \pi_{A,2} &= \begin{bmatrix} 1 & T^{-\sigma_1} A^{\delta_1} \\ 1 & T^{-\sigma_2} A^{\delta_2} \end{bmatrix} \odot (Z_A \otimes \vec{\mathbf{u}}) \qquad \pi_{R,2} = \begin{bmatrix} 1 & G^{-\sigma_1} \\ 1 & G^{-\sigma_2} \end{bmatrix} \odot (Z_R \otimes \vec{\mathbf{u}}) \end{aligned} \tag{24}$$

**ComSig**$(ck, sk, \mathbf{C})$ Parse $\mathbf{C}$ as $(\mathbf{c}_M, \mathbf{c}_N, \pi_M, \mathbf{c}_P, \mathbf{c}_Q, \pi_P, U, \pi_U)$ and $sk$ as $x$. If $\pi_M, \pi_P$ and $\pi_U$ are valid then choose $c, r \leftarrow \mathbb{Z}_p$ and $\alpha, \beta, \delta, \rho', \sigma' \leftarrow \mathbb{Z}_p^2$ and compute the following values.

$$\mathbf{c}_A := \mathsf{Com}(ck, (K \cdot T^r \cdot U)^{\frac{1}{x+c}}, \alpha) \qquad \mathbf{c}_B := \mathsf{Com}(ck, F^c, \beta) \qquad \mathbf{c}_D := \mathsf{Com}(ck, H^c, \delta)$$
$$\mathbf{c}_R := \mathbf{c}_P \odot \mathsf{Com}(ck, G^r, \rho') \qquad \mathbf{c}_S := \mathbf{c}_Q \odot \mathsf{Com}(ck, H^r, \sigma')$$

$$\pi'_{A,1} := \pi_{U,1} \odot \begin{bmatrix} v_{11}^{\alpha_1\delta_1} v_{21}^{\alpha_1\delta_2} & (YD)^{\alpha_1} v_{12}^{\alpha_1\delta_1} v_{22}^{\alpha_1\delta_2} \\ v_{11}^{\alpha_2\delta_1} v_{21}^{\alpha_2\delta_2} & (YD)^{\alpha_2} v_{12}^{\alpha_2\delta_1} v_{22}^{\alpha_2\delta_2} \end{bmatrix} \qquad \pi'_{A,2} := \pi_{U,2} \odot \begin{bmatrix} 1 & A^{\delta_1} \\ 1 & A^{\delta_2} \end{bmatrix}$$

$$\pi_A \leftarrow \mathsf{RdProof}(ck, \mathrm{E}_A, (\mathbf{c}_A, 0), (\mathbf{c}_D, 0), (\mathbf{c}_M, 0), (\mathbf{c}_S, \sigma'), \pi'_A) \qquad \pi_B \leftarrow \mathsf{Prove}(ck, \mathrm{E}_{\mathcal{DH}}, (F^c, \beta), (H^c, \delta))$$
$$\pi_R \leftarrow \mathsf{RdProof}(ck, \mathrm{E}_R, (\mathbf{c}_R, \rho'), (\mathbf{c}_S, \sigma'), \pi_P)$$

Return $(\mathbf{c}_A, \mathbf{c}_B, \mathbf{c}_D, \mathbf{c}_R, \mathbf{c}_S, \pi_A, \pi_B, \pi_R)$.

Figure 3: Making commitments to a signature and proving knowledge.

and $\pi_B = \mathsf{Prove}(ck, \mathrm{E}_{\mathcal{DH}}, (B, \beta), (D, \delta))$, which is built analogously to $\pi_R$. The signer must produce all these proofs *without knowledge of* $\mu, \rho$ *and* $\sigma$. He can do so by recycling the proofs in $\mathbf{C}$ given in (23). In particular, he sets

$$\pi_{A,1} := \pi_{U,1} \odot \begin{bmatrix} v_{11}^{\alpha_1\delta_1} v_{21}^{\alpha_1\delta_2} & (YD)^{\alpha_1} v_{12}^{\alpha_1\delta_1} v_{22}^{\alpha_1\delta_2} \\ v_{11}^{\alpha_2\delta_1} v_{21}^{\alpha_2\delta_2} & (YD)^{\alpha_2} v_{12}^{\alpha_2\delta_1} v_{22}^{\alpha_2\delta_2} \end{bmatrix} \qquad \pi_{A,2} = \pi_{U,2} \odot \begin{bmatrix} 1 & A^{\delta_1} \\ 1 & A^{\delta_2} \end{bmatrix} \tag{25}$$

which is $\pi_A$ using randomness $Z_A := Z_U$. Moreover, he produces $\pi_B$, for which he knows the values $\beta$ and $\delta$, and sets $\pi_R := \pi_P$ (both $\pi_R$ and $\pi_P$ are proofs for commitments with the same randomness $\rho$ and $\sigma$, and the proofs are such that they do not depend on the committed value). Finally, to get a *random* proof of knowledge, the signer randomizes all commitments and proofs using RdCom and RdProof as defined in Sect. 6.3. The final proof of knowledge of a signature is $(\mathbf{c}'_A, \mathbf{c}'_B, \mathbf{c}'_D, \mathbf{c}'_R, \mathbf{c}'_S, \pi'_A, \pi'_B, \pi'_R)$. See Fig. 3 for a formal description of SigCom.

**Theorem 4.** SigCom*, as defined in Fig. 3 (and Fig. 2), is commuting.*

*Proof.* The algorithm SigCom optimizes the steps discussed informally above by constructing and randomizing $\mathbf{c}_R$ and $\mathbf{c}_S$ in one step. Moreover, $\mathbf{c}_A, \mathbf{c}_B$ and $\mathbf{c}_D$ need not be randomized since SigCom chooses their randomness; in addition, being produced "freshly", $\pi_B$ need not be randomized either.

We formally prove that the output of the algorithm is correctly distributed. Recall that by $\mathsf{Prove}(\ldots; Z)$ we denote the output of Prove when $Z \in \mathbb{Z}_p^{2\times 2}$ is the randomness used.

Let $\mathbf{C} = (\mathbf{c}_M, \mathbf{c}_N, \pi_M, \mathbf{c}_P, \mathbf{c}_Q, \pi_P, U, \pi_U)$ and let $(M, N) \in \mathcal{DH}$ and $rand := (t, \mu, \nu, \rho, \sigma)$ be such that $\mathbf{C} = \mathsf{Com}_{\mathcal{M}}(ck, (M, N), rand)$; in particular, since $\mathbf{Com}_{\mathcal{M}}$ is perfectly binding, we have $U = T^t \cdot M$ and

$$\mathbf{c}_M = \mathsf{Com}(ck, M, \mu) \qquad \mathbf{c}_N = \mathsf{Com}(ck, N, \nu) \qquad \mathbf{c}_P = \mathsf{Com}(ck, G^t, \rho) \qquad \mathbf{c}_Q = \mathsf{Com}(ck, H^t, \sigma)$$

Let moreover $Z_M, Z_P$ and $Z_U$ be such that

$$\pi_M := \mathsf{Prove}\big(ck, \mathrm{E}_{\mathcal{DH}}, (M, \mu), (N, \nu);\ Z_M\big) \qquad \pi_U := \mathsf{Prove}\big(ck, \mathrm{E}_U, (M, \mu), (H^t, \sigma);\ Z_U\big)$$
$$\pi_P := \mathsf{Prove}\big(ck, \mathrm{E}_{\mathcal{DH}}, (G^t, \rho), (H^t, \sigma);\ Z_P\big)$$

CORRECTNESS. Let $c, r, \alpha, \beta, \delta, \rho', \sigma'$ be the values chosen by SigCom. We have

$$\mathbf{c}_A = \mathsf{Com}(ck, (K \cdot T^{r+t} \cdot M)^{\frac{1}{x+c}}, \alpha) \qquad \mathbf{c}_B = \mathsf{Com}(ck, F^c, \beta) \qquad \mathbf{c}_D = \mathsf{Com}(ck, H^c, \delta)$$
$$\mathbf{c}_R = \mathsf{Com}(ck, G^{t+r}, \rho + \rho') \qquad \mathbf{c}_S = \mathsf{Com}(ck, H^{t+s}, \sigma + \sigma')$$

where the first equation follows from the definition of $U$ and the last two from the homomorphic property of $\mathsf{Com}$. Note that values $(A, B, D, R, S)$ committed in $(\mathbf{c}_A, \mathbf{c}_B, \mathbf{c}_D, \mathbf{c}_R, \mathbf{c}_S)$ compose a valid signature, in particular

$$(A, B, D, R, S) = \mathsf{Sign}(x, (M, N);\ (c, r + t)) \ .$$

Define $\pi'_A$ as in Fig. 3 and let $\pi'_R := \pi_P$. In the discussion above we showed the following:

$$\pi'_A = \mathsf{Prove}\big(ck, \mathrm{E}_A, (A, \alpha), (M, \mu), (S, \sigma), (D, \delta);\ Z_U\big) \qquad \pi'_R := \mathsf{Prove}\big(ck, \mathrm{E}_{\mathcal{DH}}, (P, \rho), (Q, \sigma);\ Z_P\big)$$

Let $Z_A, Z_B$ and $Z_R$ be the randomness used by $\mathsf{RdProof}$ (or $\mathsf{Prove}$) in the construction of $\pi_A, \pi_B$ and $\pi_R$, respectively. By the properties of $\mathsf{RdProof}$ (cf. Remark 4) we have the following:

$$\begin{aligned}
\pi_A &:= \mathsf{Prove}(ck, \mathrm{E}_A, (A, \alpha), (M, \mu), (S, \sigma + \sigma'), (D, \delta);\ Z_U + Z_A + Z') \\
\pi_R &:= \mathsf{Prove}(ck, \mathrm{E}_R, (R, \rho + \rho'), (S, \sigma + \sigma');\ Z_P + Z_R + Z'') \\
\pi_B &:= \mathsf{Prove}(ck, \mathrm{E}_{\mathcal{DH}}, (B, \beta), (D, \delta);\ Z_B)
\end{aligned}$$

where the $Z'$ is defined by $\alpha, \mu$ and $\sigma'$ and $Z''$ is defined by $\rho$ and $\sigma$ (cf. equation (8) in Remark 4).

The resulting output $(\mathbf{c}_A, \mathbf{c}_B, \mathbf{c}_D, \mathbf{c}_R, \mathbf{c}_S, \pi_A, \pi_B, \pi_R)$ of $\mathsf{SigCom}$ is thus the same as the values constructed the following way:

$$\begin{aligned}
(A, B, D, R, S) &:= \Sigma = \mathsf{Sign}(x, (M, N), (\hat{c}, \hat{r})) \\
(\mathbf{c}_A, \mathbf{c}_B, \mathbf{c}_D, \mathbf{c}_R, \mathbf{c}_S) &:= \mathsf{Com}(ck, (A, B, D, R, S), (\hat{\alpha}, \hat{\beta}, \hat{\delta}, \hat{\rho}, \hat{\sigma})) \\
\pi_A &:= \mathsf{Prove}\big(ck, \mathrm{E}_A, (A, \hat{\alpha}), (M, \mu), (S, \hat{\sigma}), (D, \hat{\delta});\ \hat{Z}_A\big) \\
\pi_B &:= \mathsf{Prove}\big(ck, \mathrm{E}_B, (B, \hat{\beta}), (D, \hat{\delta});\ \hat{Z}_B\big) \\
\pi_R &:= \mathsf{Prove}\big(ck, \mathrm{E}_R, (R, \hat{\rho}), (S, \hat{\sigma});\ \hat{Z}_R\big)
\end{aligned}$$

when $(\hat{\alpha}, \hat{\beta}, \hat{\delta}, \hat{\rho}, \hat{\sigma}, \hat{Z}_A, \hat{Z}_B, \hat{Z}_R)$ are defined as

$$\begin{array}{lllll}
\hat{c} = c & \hat{r} = r + t & \hat{\alpha} = \alpha & \hat{\beta} = \beta & \hat{\delta} = \delta \\
\hat{\rho} = \rho + \rho' & \hat{\sigma} = \sigma + \sigma' & \hat{Z}_A = Z_U + Z_A + Z' & \hat{Z}_B = Z_B & \hat{Z}_R = Z_P + Z_R + Z''
\end{array}$$

All these values are uniformly random since $c, r, \alpha, \beta, \delta, \rho', \sigma', Z_A, Z_B$, and $Z_R$ are chosen uniformly and independently at random by $\mathsf{SigCom}$. $\square$

**Instantiation of** $\mathsf{SmSigCom}$**.** This algorithm does what $\mathsf{SigCom}$ does but instead of being given the signing key *sk*, it is directly given a signature. It proceeds similarly to $\mathsf{SigCom}$ but starting from a signature instead of a pre-signature. It first uses *ek* to extract $P$ and $Q$ from $\mathbf{C}$. It then chooses $\alpha, \beta, \delta \leftarrow \mathcal{R}$ and sets $\mathbf{c}_A := \mathsf{Com}(ck, A, \alpha)$, $\mathbf{c}_B := \mathsf{Com}(ck, B, \beta)$, and $\mathbf{c}_D := \mathsf{Com}(ck, D, \delta)$. It moreover sets

$$\mathbf{c}_R := \mathbf{c}_P \circledcirc \mathsf{Com}(ck, R \cdot P^{-1}, 0) = \mathsf{Com}(ck, R, \rho) \qquad \mathbf{c}_S := \mathbf{c}_Q \circledcirc \mathsf{Com}(ck, S \cdot Q^{-1}, 0) = \mathsf{Com}(ck, S, \sigma)$$

It can now define $\pi_A$ as in (25) and $\pi_R$ as $\pi_P$ from $\mathbf{C}$, and produce $\pi_B$ using $\beta$ and $\delta$. Randomize everything and output $(\mathbf{c}_A, \mathbf{c}_B, \mathbf{c}_D, \mathbf{c}_R, \mathbf{c}_S, \pi_A, \pi_B, \pi_R)$.

### A.3 Instantiations of Proof Adaptation for Committing and Decommitting

**Adaptation of Proofs for Committing and Decommitting to Signatures and Messages.** In Fig. 4 we rewrote the proofs contained in a commitment $\mathbf{C}$ and the proofs for the first verification relation of the signatures, depending on which elements are committed. $\pi_A$ is the proof when both signature and message are committed, $\pi_{\widetilde{A}}$ when only the signature is committed and $\pi_{\bar{A}}$ for when only the message is committed. We also give $\pi_{A^\dagger}$, a proof for when only the elements $A$ and $D$ of a signature are committed.

**C** Proofs contained in a $\mathbf{Com}_{\mathcal{M}}$ commitment, for equations $\mathrm{E}_P = \mathrm{E}_M = \mathrm{E}_{\mathcal{DH}}(M,N) : e(G^{-1}, \underline{N})\, e(\underline{M}, H) = 1$ and $\mathrm{E}_U(M,Q) \ : \ e(T^{-1}, \underline{Q}) e(\underline{M}, H^{-1}) = e(U,H)^{-1}$

$$\pi_{M,1} = \begin{bmatrix} 1 & H^{\mu_1} \\ 1 & H^{\mu_2} \end{bmatrix} \odot (Z_M \otimes \vec{\mathbf{v}}) \quad \pi_{P,1} = \begin{bmatrix} 1 & H^{\rho_1} \\ 1 & H^{\rho_2} \end{bmatrix} \odot (Z_P \otimes \vec{\mathbf{v}}) \quad \pi_{U,1} = \begin{bmatrix} 1 & H^{-\mu_1} \\ 1 & H^{-\mu_2} \end{bmatrix} \odot (Z_U \otimes \vec{\mathbf{v}})$$

$$\pi_{M,2} = \begin{bmatrix} 1 & G^{-\nu_1} \\ 1 & G^{-\nu_2} \end{bmatrix} \odot (Z_M \otimes \vec{\mathbf{u}}) \quad \pi_{P,2} = \begin{bmatrix} 1 & G^{-\sigma_1} \\ 1 & G^{-\sigma_2} \end{bmatrix} \odot (Z_P \otimes \vec{\mathbf{u}}) \quad \pi_{U,2} = \begin{bmatrix} 1 & T^{-\sigma_1} \\ 1 & T^{-\sigma_2} \end{bmatrix} \odot (Z_U \otimes \vec{\mathbf{u}})$$

$\pi_A$ Proof for equation $\mathrm{E}_A(A, M;\ S, D) \ : \ e(T^{-1}, \underline{S})\, e(\underline{A}, Y) e(\underline{M}, H^{-1})\, e(\underline{A}, \underline{D}) = e(K, H)$

$$\pi_{A,1} = \begin{bmatrix} v_{11}^{\alpha_1\delta_1} v_{21}^{\alpha_1\delta_2} & (YD)^{\alpha_1} H^{-\mu_1} v_{12}^{\alpha_1\delta_1} v_{22}^{\alpha_1\delta_2} \\ v_{11}^{\alpha_2\delta_1} v_{21}^{\alpha_2\delta_2} & (YD)^{\alpha_2} H^{-\mu_2} v_{12}^{\alpha_2\delta_1} v_{22}^{\alpha_2\delta_2} \end{bmatrix} \odot (Z_A \otimes \vec{\mathbf{v}}) \qquad \pi_{A,2} = \begin{bmatrix} 1 & T^{-\sigma_1} A^{\delta_1} \\ 1 & T^{-\sigma_2} A^{\delta_2} \end{bmatrix} \odot (Z_A \otimes \vec{\mathbf{u}})$$

$\pi_{\widetilde{A}}$ Proof for equation $\mathrm{E}_{\widetilde{A}}(A;\ S, D) \ : \ e(T^{-1}, \underline{S})\, e(\underline{A}, Y)\, e(\underline{A}, \underline{D}) = e(K\cdot M, H)$

$$\pi_{\widetilde{A},1} = \begin{bmatrix} v_{11}^{\alpha_1\delta_1} v_{21}^{\alpha_1\delta_2} & (YD)^{\alpha_1} v_{12}^{\alpha_1\delta_1} v_{22}^{\alpha_1\delta_2} \\ v_{11}^{\alpha_2\delta_1} v_{21}^{\alpha_2\delta_2} & (YD)^{\alpha_2} v_{12}^{\alpha_2\delta_1} v_{22}^{\alpha_2\delta_2} \end{bmatrix} \odot (Z_{\widetilde{A}} \otimes \vec{\mathbf{v}}) \qquad \pi_{\widetilde{A},2} = \begin{bmatrix} 1 & T^{-\sigma_1} A^{\delta_1} \\ 1 & T^{-\sigma_2} A^{\delta_2} \end{bmatrix} \odot (Z_{\widetilde{A}} \otimes \vec{\mathbf{u}})$$

$\pi_{A^\dagger}$ Proof for equation $\mathrm{E}_{A^\dagger}(A;\ D) \ : \ e(\underline{A}, Y)\, e(\underline{A}, \underline{D}) = e(K\cdot M, H)\, e(T, S)$

$$\pi_{A^\dagger,1} = \begin{bmatrix} v_{11}^{\alpha_1\delta_1} v_{21}^{\alpha_1\delta_2} & (YD)^{\alpha_1} v_{12}^{\alpha_1\delta_1} v_{22}^{\alpha_1\delta_2} \\ v_{11}^{\alpha_2\delta_1} v_{21}^{\alpha_2\delta_2} & (YD)^{\alpha_2} v_{12}^{\alpha_2\delta_1} v_{22}^{\alpha_2\delta_2} \end{bmatrix} \odot (Z_{A^\dagger} \otimes \vec{\mathbf{v}}) \qquad \pi_{A^\dagger,2} = \begin{bmatrix} 1 & A^{\delta_1} \\ 1 & A^{\delta_2} \end{bmatrix} \odot (Z_{A^\dagger} \otimes \vec{\mathbf{u}})$$

$\pi_{\bar{A}}$ Proof for equation $\mathrm{E}_{\bar{A}}(M) \ : \ e(\underline{M}, H^{-1}) = e(A, Y\cdot D)^{-1} e(K, H)\, e(T, S)$

$$\pi_{\bar{A},1} = \begin{bmatrix} 1 & H^{-\mu_1} \\ 1 & H^{-\mu_2} \end{bmatrix} \odot (Z_{\bar{A}} \otimes \vec{\mathbf{v}}) \qquad \pi_{\bar{A},2} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \odot (Z_{\bar{A}} \otimes \vec{\mathbf{u}})$$

Figure 4: Overview of different variants of the proof for the first verification equation.

The reason for $\pi_{A^\dagger}$ is to illustrate how the proof $\pi_A$ in SigCom is actually constructed. What the signer does is to produce $\pi_{A^\dagger}$ (for which he knows the randomness $(\alpha, \delta)$) and then set $\pi_A := \pi_U \odot \pi_{A^\dagger}$ in (25). In particular, we observe that the following relations hold between the proofs:

$$\begin{aligned} \pi_A &= \pi_U \odot \pi_{A^\dagger} \\ \pi_A &= \pi_{\widetilde{A}} \odot \pi_{\bar{A}} \end{aligned}$$

The last equation allows us to implement the algorithms AdPrC, $\mathsf{AdPrC}_{\mathcal{M}}$, AdPrDC and $\mathsf{AdPrDC}_{\mathcal{M}}$.

$\mathsf{AdPrC}(pp, vk, \mathbf{C}, ((A,B,D,R,S), (\alpha,\beta,\delta,\rho,\sigma)), \bar{\pi})$. The proof $\bar{\pi}$ is a proof for equation $\mathrm{E}_{\bar{A}}$. The algorithm sets

$$\pi_{\widetilde{A}} \leftarrow \mathsf{Prove}(ck, \mathrm{E}_{\widetilde{A}}, (A,\alpha), (S,\sigma), (D,\delta)) \qquad \begin{aligned} \pi_B &\leftarrow \mathsf{Prove}(ck, \mathrm{E}_B, (B,\beta), (D,\delta)) \\ \pi_R &\leftarrow \mathsf{Prove}(ck, \mathrm{E}_R, (R,\rho), (S,\sigma)) \end{aligned}$$

for $\mathrm{E}_B$ and $\mathrm{E}_R$ as defined in (22). It then returns $\pi := (\pi_{\widetilde{A}} \odot \pi_{\bar{A}}, \pi_B, \pi_R)$.

$\mathsf{AdPrC}_{\mathcal{M}}(pp, vk, ((M,N), (t,\mu,\nu,\rho,\sigma)), \mathbf{c}_\Sigma, \widetilde{\pi})$. The proof $\widetilde{\pi}$ is of the form $(\pi_{\widetilde{A}}, \pi_B, \pi_R)$. The algorithm sets $\pi_{\bar{A}} \leftarrow \mathsf{Prove}(ck, \mathrm{E}_{\bar{A}}, (M,\mu))$ and returns a randomization of $\pi := (\pi_{\widetilde{A}} \odot \pi_{\bar{A}}, \pi_B, \pi_R)$.

$\mathsf{AdPrDC}(pp, vk, \mathbf{C}, ((A,B,D,R,S), (\alpha,\beta,\delta,\rho,\sigma)), \pi)$. The proof $\pi$ is of the form $(\pi_A, \pi_B, \pi_R)$. The algorithm sets $\pi_{\widetilde{A}} \leftarrow \mathsf{Prove}(ck, \mathrm{E}_{\widetilde{A}}, (A,\alpha), (S,\sigma), (D,\delta))$ and returns $\bar{\pi} := \pi_A \oslash \pi_{\widetilde{A}}$ (where "$\oslash$" denotes component-wise division, that is: replace all the components of the second argument by their inverses and then multiply them with those of the first argument).

$\mathsf{AdPrDC}_{\mathcal{M}}(pp, vk, ((M,N),(t,\mu,\nu,\rho,\sigma)), \mathbf{c}_\Sigma, \pi)$. The proof $\pi$ is of the form $(\pi_A, \pi_B, \pi_R)$. The algorithm produces $\pi_{\bar{A}} \leftarrow \mathsf{Prove}(ck, \mathrm{E}_{\bar{A}}, (M,\mu))$ and returns a randomization of $\widetilde{\pi} := (\pi_A \oslash \pi_{\bar{A}}, \pi_B, \pi_R)$.

**Instantiation of Proof Adaptation when Committing to the Verification Key** In some applications (such as the one discussed in Sect. 5), in order to remain anonymous, the signer makes a commitment

$$\big(\mathbf{c}_X = \mathsf{Com}(ck, X, \xi),\ \mathbf{c}_Y = \mathsf{Com}(ck, Y, \psi),\ \pi_X = \mathsf{Prove}(ck, \mathrm{E}_{\mathcal{DH}}, (X,\xi), (Y,\psi))\big)$$

to his public key and wishes to prove that the values in $\vec{\mathbf{c}}_\Sigma$ are a valid signature on the values $(M,N)$ in $\mathbf{C}$ under the public key that is committed in $(\mathbf{c}_X, \mathbf{c}_Y, \pi_X)$. The first equation of verification is thus

$$\mathrm{E}_{\widehat{A}}(A, M;\ S, Y, D)\ :\ e(T^{-1}, \underline{S})\, e(\underline{M}, H^{-1})\, e(\underline{A}, \underline{Y}) e(\underline{A}, \underline{D}) = e(K, H)\ \ ,$$

for which, by (7), the proof is

$$\begin{aligned}
\pi_{\widehat{A},1} &:= \begin{bmatrix} v_{11}^{\alpha_1(\psi_1+\delta_1)} v_{21}^{\alpha_1(\psi_2+\delta_2)} & (YD)^{\alpha_1} H^{-\mu_1} v_{12}^{\alpha_1(\psi_1+\delta_1)} v_{22}^{\alpha_1(\psi_2+\delta_2)} \\ v_{11}^{\alpha_2(\psi_1+\delta_1)} v_{21}^{\alpha_2(\psi_2+\delta_2)} & (YD)^{\alpha_2} H^{-\mu_2} v_{12}^{\alpha_2(\psi_1+\delta_1)} v_{22}^{\alpha_2(\psi_2+\delta_2)} \end{bmatrix} \odot (Z_{\widehat{A}} \otimes \vec{\mathbf{v}}) \\
\pi_{\widehat{A},2} &:= \begin{bmatrix} 1 & T^{-\sigma_1} A^{\psi_1+\delta_1} \\ 1 & T^{-\sigma_2} A^{\psi_2+\delta_2} \end{bmatrix} \odot (Z_{\widehat{A}} \otimes \vec{\mathbf{u}})
\end{aligned} \tag{26}$$

whereas $\mathrm{E}_B$ and $\mathrm{E}_R$ do not change. Given a commitment $\mathbf{C}$ to a message, a commitment $\mathbf{c}_\Sigma = (\mathbf{c}_A, \mathbf{c}_B, \mathbf{c}_D, \mathbf{c}_R, \mathbf{c}_S)$ to a signature and a proof $\pi = (\pi_A, \pi_B, \pi_R)$ of validity, $\pi_A$ can be adapted to $\pi_{\widehat{A}}$ using the technique discussed in Sect. 7.4. Using the randomness $\psi$ for $\mathbf{c}_Y$, choose $Z' \in \mathbb{Z}_p^{2\times 2}$ and set

$$\begin{aligned}
\pi_{\widehat{A},1} &:= \pi_{A,1} \odot (Z' \otimes \vec{\mathbf{v}}) \\
\pi_{\widehat{A},2} &:= \pi_{A,2} \odot \begin{bmatrix} \mathbf{c}_{A,1}^{\psi_1} & \mathbf{c}_{A,2}^{\psi_1} \\ \mathbf{c}_{A,1}^{\psi_2} & \mathbf{c}_{A,2}^{\psi_2} \end{bmatrix} \odot (Z' \otimes \vec{\mathbf{u}})
\end{aligned}$$

We show why this yields a proof for $\mathrm{E}_{\widehat{A}}$. Let $Z \in \mathbb{Z}_p^{2\times 2}$ denote the randomness used in $\pi_A$. Set

$$\hat{Z} := \begin{bmatrix} z_{11} + z'_{11} + \alpha_1\psi_1 & z_{12} + z'_{12} + \alpha_2\psi_1 \\ z_{21} + z'_{21} + \alpha_1\psi_2 & z_{22} + z'_{22} + \alpha_2\psi_2 \end{bmatrix}\ .$$

Then, using the definition of $\pi_A$ from (24), we have

$$\begin{aligned}
\pi_{\widehat{A},1} &:= \begin{bmatrix} v_{11}^{\alpha_1\delta_1} v_{21}^{\alpha_1\delta_2} & (YD)^{\alpha_1} H^{-\mu_1} v_{12}^{\alpha_1\delta_1} v_{22}^{\alpha_1\delta_2} \\ v_{11}^{\alpha_2\delta_1} v_{21}^{\alpha_2\delta_2} & (YD)^{\alpha_2} H^{-\mu_2} v_{12}^{\alpha_2\delta_1} v_{22}^{\alpha_2\delta_2} \end{bmatrix} \odot \begin{bmatrix} v_{11}^{\alpha_1\psi_1} v_{21}^{\alpha_1\psi_2} & v_{12}^{\alpha_1\psi_1} v_{22}^{\alpha_1\psi_2} \\ v_{11}^{\alpha_2\psi_1} v_{21}^{\alpha_2\psi_2} & v_{12}^{\alpha_2\psi_1} v_{22}^{\alpha_2\psi_2} \end{bmatrix} \odot (\hat{Z} \otimes \vec{\mathbf{v}}) \\
\pi_{\widehat{A},2} &:= \begin{bmatrix} 1 & T^{-\sigma_1} A^{\delta_1} \\ 1 & T^{-\sigma_2} A^{\delta_2} \end{bmatrix} \odot \begin{bmatrix} (u_{11}^{\alpha_1} u_{21}^{\alpha_2})^{\psi_1} & (A u_{12}^{\alpha_1} u_{22}^{\alpha_2})^{\psi_1} \\ (u_{11}^{\alpha_1} u_{21}^{\alpha_2})^{\psi_2} & (A u_{12}^{\alpha_1} u_{22}^{\alpha_2})^{\psi_2} \end{bmatrix} \odot ((Z + Z') \otimes \vec{\mathbf{u}}) = \begin{bmatrix} 1 & T^{-\sigma_1} A^{\delta_1+\psi_1} \\ 1 & T^{-\sigma_2} A^{\delta_2+\psi_2} \end{bmatrix} \odot (\hat{Z} \otimes \vec{\mathbf{u}})
\end{aligned}$$

which is a proof for equation $\mathrm{E}_{\widehat{A}}$ as detailed in (26) when $Z_{\widehat{A}} := \hat{Z}$.

# B Proof of Theorem 2

Consider an adversary that after receiving parameters $(G, F, K, L, T, H)$ and public key $(X, Y)$ is allowed to ask for $q-1$ signatures $(A_i, B_i, D_i, R_i, S_i)$ on messages $(u_i, (M_i, N_i)) \in \mathbb{Z}_p \times \mathcal{DH}$ of its choice and then outputs $(u, (M, N)) \in \mathbb{Z}_p \times \mathcal{DH}$ and a valid signature $(A, B, D, R, S)$ on it, such that either $(u, (M, N))$ was never queried, or $(u,(M,N)) = (u_i,(M_i,N_i))$ and $(A,B,D,R,S) \neq (A_i,B_i,D_i,R_i,S_i)$. We distinguish three kinds of forgers: An adversary is called of Type I if its output satisfies the following

$$\forall\, 1 \le i \le q-1 : \big[e(T, S\cdot S_i^{-1}) \neq e(L^{u_i}\cdot M_i\cdot L^{-u}\cdot M^{-1}, H) \,\vee\, B \neq B_i\big] \tag{27}$$

An adversary is called of Type IIa if its output satisfies

$$\exists\, 1 \le i \le q-1 : \big[e(T, S\cdot S_i^{-1}) = e(L^{u_i}\cdot M_i\cdot L^{-u}\cdot M^{-1}, H) \,\wedge\, B = B_i \,\wedge\, S \neq S_i\big] \tag{28}$$

otherwise it is called of Type IIb. We will use the first type to break $q$-ADHSDH, Type IIa to break AWFCDH and Type IIb to break CDH, which is implied by AWFCDH.

**Type I** Let $\big(G,F,K,X,H,Y,(A_i,B_i,V_i,D_i,W_i)_{i=1}^{q-1}\big)$ be a $q$-ADHSDH challenge. It satisfies thus

$$e(A_i, Y\cdot D_i) = e(K\cdot V_i, H) \qquad e(B_i,H) = e(F,D_i) \qquad e(V_i,H) = e(G,W_i) \tag{29}$$

Let $\mathcal{A}$ be a forger of Type I. Choose $t,l \leftarrow \mathbb{Z}_p$ and give parameters $(G,F,K,L := G^l, T := G^t, H)$ and the public key $(X,Y)$ to $\mathcal{A}$. The $i$-th signing query for $(u,(M_i,N_i)) \in \mathbb{Z}_p \times \mathcal{DH}$ is answered as

$$\big(A_i, B_i, D_i, R_i := (V_i\cdot G^{-l\cdot u_i}\cdot M_i^{-1})^{\frac{1}{t}}, S_i = (W_i\cdot H^{-l\cdot u_i}\cdot N_i^{-1})^{\frac{1}{t}}\big).$$

It is easily verified that it satisfies (15); and it is correctly distributed since $v_i = \log_G V_i$ is uniformly random in the ADHSDH instance. If the adverseray produces a valid pair $\big((A,B,D,R,S),(u,(M,N))\big)$ then by the last 2 equations of (15), there exist $c,r$ s.t. $B = F^c, D = H^c, R = G^r, S = H^r$, and

$$e(A, Y\cdot D) = e(K\cdot L^u\cdot M, H)\, e(T,S)\ . \tag{30}$$

The tuple $(A,B,D,V := G^{l\cdot u}\cdot M\cdot R^t, W := H^{l\cdot u}\cdot N\cdot S^t)$ satisfies (2), since $(B,D)$ and $(V,W)$ are Diffie-Hellman pairs and $e(K\cdot V,H) = e(K\cdot L^u\cdot M\cdot (G^r)^t, H) = e(K\cdot L^u\cdot M, H)\, e(T,S) \overset{(30)}{=} e(A, Y\cdot D)$. Moreover, it is a solution for the ADHSDH instance, since it is a *new* tuple: assume that for some $i$ we have $B = B_i$ and $W = W_i$, that is $H^{l\cdot u}\cdot N\cdot S^t = H^{l\cdot u_i}\cdot N_i\cdot S_i^t$. Since $(M,N),(M_i,N_i) \in \mathcal{DH}$, we have $e(T,S)\, e(L^u\cdot M, H) = e(T,S)\, e(G, H^{l\cdot u}\cdot N) = e(G, H^{l\cdot u}\cdot N\cdot S^t) = e(G, H^{l\cdot u_i}\cdot N_i\cdot S_i^t) = e(T,S_i)\, e(G, H^{l\cdot u_i}\cdot N_i) = e(T,S_i)$ $e(L^{u_i}\cdot M_i, H)$. We have thus $e(T, S\cdot S_i^{-1}) = e(L^{u_i}\cdot M_i\cdot L^{-u}\cdot M^{-1}, H)$ and $B = B_i$ which contradicts (27) and thus the fact that $\mathcal{A}$ is of Type I.

**Type IIa** Let $(G,H,T = G^t)$ be an AWFCDH instance; let $\mathcal{A}$ be a forger of Type IIa. Pick $F,K \leftarrow \mathbb{G}_1$ and $l,x \leftarrow \mathbb{Z}_p$, set $X := G^x$, $Y := H^x$ and give the adversary parameters $(G,F,K,L := G^l, T, H)$ and public key $(X,Y)$. Answer a signing query on $(u_i,(M_i,N_i)) \in \mathbb{Z}_p \times \mathcal{DH}$ by returning a signature $(A_i,B_i,D_i,R_i,S_i)$ produced by $\mathsf{Sign}_{\mathrm{A}}(x,\cdot)$. Suppose $\mathcal{A}$ returns $\big((A,B,D,R,S),(u,(M,N))\big)$ satisfying (15) s.t. $e(T, S\cdot S_i^{-1}) = e(L^{u_i}\cdot M_i\cdot L^{-u}\cdot M^{-1}, H)$, $B = B_i$ and $S \neq S_i$ for some $i$. Then $(M^* := L^{u_i}\cdot M_i\cdot L^{-u}\cdot M^{-1}, N^* := H^{l\cdot u_i}\cdot N_i\cdot H^{-l\cdot u}\cdot N^{-1}, R^* := R\cdot R_i^{-1}, S^* := S\cdot S_i^{-1})$ is a AWFCDH solution: $(S^*,M^*)$, $(M^*,N^*)$ and $(R^*,S^*)$ satisfy the respective equations in (3), and since $S \neq S_i$ it is non-trivial.

**Type IIb** Let $(G,H,L := G^l)$ be a CDH instance, i.e., we have to produce $H^l$. Let $\mathcal{A}$ be a forger of Type IIb. Pick $F,K,T \leftarrow \mathbb{G}_1$ and $x \leftarrow \mathbb{Z}_p$, set $X := G^x$, $Y := H^x$ and give the adversary parameters $(G,F,K,L,T,H)$ and public key $(X,Y)$. Answer a signing query on $(u_i,(M_i,N_i)) \in \mathbb{Z}_p \times \mathcal{DH}$ by returning a signature $(A_i,B_i,D_i,R_i,S_i)$ produced by $\mathsf{Sign}_{\mathrm{A}}(x,\cdot)$. Suppose $\mathcal{A}$ returns $\big((A,B,D,R,S),(u,(M,N))\big)$ satisfying (15) of Type IIa, i.e., $e(T, S\cdot S_i^{-1}) = e(L^{u_i}\cdot M_i\cdot L^{-u}\cdot M^{-1}, H)$, $B = B_i$ and $S = S_i$ for some $i$; which implies $L^{u_i}\cdot M_i = L^u\cdot M$.

We first show that $u \neq u_i$: Suppose $u = u_i$; then by the above we have $M = M_i$, and moreover $B = B_i$ and $S = S_i$. Since these values completely determine $A,D,R$ and $N$, we have $(A,B,D,R,S,u,M,N) = (A_i,B_i,D_i,R_i,S_i,u_i,M_i,N_i)$, which means that $\mathcal{A}$ did not break strong unforgeability.

From $L^{u_i}\cdot M_i = L^u M$ we have $L^{u-u_i} = M_i\cdot M^{-1}$ and since $u \neq u_i$ we have $L = (M_i\cdot M^{-1})^{\frac{1}{u-u_i}}$, which for $m := \log_G M = \log_H N$, $m_i := \log_G M_i = \log_H N_i$ can be written as $G^{\frac{m_i-m}{u-u_i}}$. Thus $(N_i\cdot N^{-1})^{\frac{1}{u-u_i}} = H^{\frac{m_i-m}{u-u_i}}$ is a CDH solution. $\square$

# C Additional Tools and Their Instantiations

In Sect. 6.4, we introduced the scheme from [Fuc09] to sign two public keys at once, which can easily be extended for arbitrary many messages. The scheme $\mathsf{Sig}^*$ however has message space $\mathcal{M}^* := \mathcal{DH}^* = \mathcal{DH} \setminus \{(1,1)\}$. In this section, we define randomizable, extractable commitments to elements from $\mathcal{M}^*$, and show how to make a commitments to a signature and a proof of validity on a clear and a committed message from $\mathcal{M}^*$. We define the following:

**Committing to $\mathcal{M}^*$ Elements.** We define $\mathbf{Com}^*_{\mathcal{M}}$ that has the same properties as $\mathbf{Com}_{\mathcal{M}}$, but with value space $\mathcal{M}^*$ rather than $\mathcal{M}$. By $\mathcal{C}^*_{\mathcal{M}}$ we denote the commitment space and by $\mathcal{R}^*_{\mathcal{M}}$ the space of randomness. Our instantiation is given in Appendix C.1.

**Partially Blind Automorphic Signatures.** Since $\mathbf{Sig}^*$ (cf. Sect. 6.4) signs two messages, we can define a variant of $\mathsf{SigCom}$ that gets one message in the clear and one committed message. Based on $\mathbf{Sig}^*$ and $\mathbf{Com}^*_{\mathcal{M}}$, we define $\mathsf{PSigCom}$ that is given a message $M \in \mathcal{M}^*$ and a $\mathbf{Com}^*_{\mathcal{M}}$ commitment and outputs a proof of knowledge of a $\mathbf{Sig}^*$ signature on $M$ and the committed value:

$\mathsf{PSigCom}(ck, sk, M, \mathbf{C})$. If $M \in \mathcal{M}^*$ and $\mathbf{C} \in \mathcal{C}^*_{\mathcal{M}}$ then the algorithm outputs a commitment to a signature and a proof of validity $(\mathbf{c}_\Sigma, \pi)$ which is distributed as

$$\Big[\Sigma \leftarrow \mathsf{Sign}^*(sk, (M, V));\ \rho \leftarrow \mathcal{R}\ :\ \big(\mathsf{Com}(ck, \Sigma, \rho),\ \mathsf{Prove}(ck, \mathrm{E}_{\mathsf{Ver}^*(vk,(M,\cdot),\cdot)}, (V, \nu), (\Sigma, \rho))\big)\Big]\ ,$$

where $V$ and $\nu$ are such that $\mathbf{C} = \mathsf{Com}^*_{\mathcal{M}}(ck, V, \nu)$.

Note that if in the construction of a blind signature in Sect. 4.2 we replace $\mathbf{Sig}$, $\mathbf{Com}_{\mathcal{M}}$ and $\mathsf{SigCom}$ by $\mathbf{Sig}^*$, $\mathbf{Com}^*_{\mathcal{M}}$ and $\mathsf{PSigCom}$, we obtain *partially blind signatures* [AF96] (where the signer controls part of the message), which are automorphic themselves.

## C.1 Commitments to Non-Trivial Messages

We instantiate $\mathbf{Com}^*_{\mathcal{M}}$, with message space $\mathcal{M}^* := \mathcal{DH}^* = \{(G^m, H^m) \mid m \in \mathbb{Z}_p \setminus \{0\}\}$. To guarantee that the committed value is not $(1,1)$, $\mathsf{Com}^*_{\mathcal{M}}$ contains additional elements. Intuitively, given $(M,N) \in \mathcal{M}^*$ if we choose $l \leftarrow \mathbb{Z}^*_p$ and publish $W = N^l$ then $W \neq 1$ iff $N \neq 1$. We add a commitment $\mathbf{c}_L$ to $G^l$ and a proof $\pi_W$ that $e(G^l, N) = e(G, W)$, which proves well-formedness of $W$. In the WI setting $W$, $\mathbf{c}_L$ and $\pi_W$ perfectly hide $N$.

Randomization of a $\mathbf{Com}^*_{\mathcal{M}}$ commitment is a bit trickier. Since $l$ must be from $\mathbb{Z}^*_p$, we randomize it *multiplicatively*, that is, we choose $l' \leftarrow \mathbb{Z}^*_p$ and replace $l$ by $l \cdot l'$. This also enables randomization of $W$ as $W' := W^{l'}$ which without knowledge of $N$ cannot be done additively. Finally, Lemma 4 (Sect. 7.4) shows how to adapt $\mathbf{c}_L$ and $\pi_W$ to the new value $l \cdot l'$.

We define $\mathbf{Com}^*_{\mathcal{M}}$ by extending $\mathbf{Com}_{\mathcal{M}}$ from Sect. 8.1:

$\mathsf{Com}^*_{\mathcal{M}}(pp, (M,N), (\kappa = (t, \mu, \nu, \rho, \sigma), \eta, l))$. If $(M,N) \in \mathcal{DH}^*$, $(\kappa, \eta, l) \in \mathcal{R}_{\mathcal{M}} \times \mathcal{R} \times \mathbb{Z}^*_p =: \mathcal{R}^*_{\mathcal{M}}$ then define

$$W := N^l \qquad \mathbf{c}_L := \mathsf{Com}(ck, G^l, \eta) \qquad \pi_W \leftarrow \mathsf{Prove}(ck, \mathrm{E}_W,\ (G^l, \eta), (N, \nu))$$

with $\mathrm{E}_W(L;\ N) :\ e(\underline{L}, \underline{N}) = e(G, W)$, and output $\big(\mathsf{Com}_{\mathcal{M}}(ck, (M,N), \kappa), W, \mathbf{c}_L, \pi_W\big)$.

The space of valid commitments is $\mathcal{C}^*_{\mathcal{M}} := \big\{(\mathbf{C}, W, \mathbf{c}_L, \pi_W) \mid \mathbf{C} \in \mathcal{C}_{\mathcal{M}} \wedge W \neq 1 \wedge \mathsf{Verify}(ck, \mathrm{E}_W, \mathbf{c}_L, \mathbf{c}_N, \pi_W)\big\}$. $\mathbf{Com}^*_{\mathcal{M}}$ is shown to be binding and computationally hiding analogously to $\mathbf{Com}_{\mathcal{M}}$; in particular if $ck^*$ is a WI key then for every $(M,N) \in \mathcal{DH}^*$, given $(\mathbf{C}, W, \mathbf{c}_L, \pi_W)$ there exists $(\kappa, \eta, l)$ such that $\mathbf{C} = \mathsf{Com}_{\mathcal{M}}(ck^*, (M,N), \kappa)$, $W = N^l$, $\mathbf{c}_L = \mathsf{Com}(ck^*, G^l, \eta)$ and $\pi_W \leftarrow \mathsf{Prove}(ck, \mathrm{E}_W,\ (G^l, \eta), (N, \nu))$ with $\kappa = (t, \mu, \nu, \rho, \sigma)$.

Using the results from Sect. 7, we define $\mathsf{RdCom}^*_{\mathcal{M}}$ by extending $\mathsf{RdCom}_{\mathcal{M}}$:

$\mathsf{RdCom}^*_{\mathcal{M}}(pp, (\mathbf{C}, W, \mathbf{c}_L, \pi_W), (\kappa', \eta', l'))$ returns a commitment $(\mathbf{C}', W', \mathbf{c}'_L, \pi'_W)$ that is equivalent to the output of $\mathsf{Com}^*_{\mathcal{M}}(pp, (M,N), (\kappa + \kappa', l' \cdot \eta + \eta', l \cdot l'))$ defined as $\mathbf{C}' := \mathsf{RdCom}_{\mathcal{M}}(pp, \mathbf{C}, \kappa')$ and

$$W' := W^{l'} \qquad \mathbf{c}'_L := \mathsf{RdCom}(ck, (\mathbf{c}_L^{l'}), \eta') \qquad \pi'_W \leftarrow \mathsf{RdProof}(ck, \mathrm{E}_W, (\mathbf{c}_L^{l'}, \eta'), (\mathbf{c}_N, \nu'), \pi_W^{l'}))$$

$\mathsf{RdCom}^*_{\mathcal{M}}$ works as follows: the part $\mathbf{C}$ of a commitment is randomized by $\mathsf{RdCom}_{\mathcal{M}}$ which replaces $\kappa$ by $\kappa + \kappa'$. Now $W$ is replaced by $W^{l'}$, which implicitly replaces $l$ by $l \cdot l'$. Setting $\hat{\mathbf{c}}_L := \mathbf{c}_L^{l'}$, we get $\hat{\mathbf{c}}_L = \mathsf{Com}(ck, L^{l'}, l' \cdot \eta)$ and by Lemma 4, we have that $\hat{\pi}_W := \pi_W^{l'}$ is a proof for $\mathrm{E}_W$ and $(\hat{\mathbf{c}}_L, \mathbf{c}_N)$. In $W', \hat{\mathbf{c}}_L$ and $\hat{\pi}_W$, randomness $l$ has thus consistently been replaced by $l \cdot l'$. The final step is to set $\mathbf{c}'_L = \mathsf{RdCom}(ck, \hat{\mathbf{c}}_L, \eta')$ and $\pi'_W \leftarrow \mathsf{RdProof}(ck, \mathrm{E}_W, (\hat{\mathbf{c}}_L, \eta'), (\mathbf{c}_N, \nu'), \hat{\pi}_W)$. Note that $\mathbf{c}'_L$ is thus a commitment to $L^{l'}$ under randomness $l' \cdot \eta + \eta'$.

If $(\kappa, \eta, l)$ and $(\kappa', \eta', l')$ are both uniformly chosen from $\mathcal{R}^*_{\mathcal{M}}$ then the randomness after randomization is also uniform in $\mathcal{R}^*_{\mathcal{M}}$.

### C.2 Making Commitments to a Signature on a Public and a Committed Message and a Proof of Validity

We give an instantiation of $\mathsf{PSigCom}$ defined at the beginning of the section. We start by giving a variant of $\mathsf{SigCom}$ that has inputs $(ck, sk, (V, W), \mathbf{C})$ with $(V, W) \in \mathcal{M}^*$ and $\mathbf{C} \in \mathcal{C}^*_{\mathcal{M}}$ and outputs a proof of knowledge of a signature on $(V, W) \odot (M, N)$, where $(M, N)$ is the message committed in $\mathbf{C}$. The verification of a signature on such a product $\mathsf{Ver}'((X, Y), (V, W), (M, N), (A, B, D, R, S))$ are the equations $\mathrm{E}_{A'}$, defined as

$$\mathrm{E}_{A'}(A, M;\, S, D) \;:\; e(T^{-1}, \underline{S})\, e(\underline{A}, Y) e(\underline{M}, H^{-1})\, e(\underline{A}, \underline{D}) = e(K \cdot V, H) \;\;,$$

and $\mathrm{E}_B$, $\mathrm{E}_R$ as defined in (12). Since the left-hand sides of $\mathrm{E}_{A'}$ and $\mathrm{E}_A$ from (12) are equivalent, by Lemma 1 both equations have the same proofs: $\pi_A = \pi_{A'}$. The *only* thing that changes w.r.t. $\mathsf{SigCom}$ is thus the value $A$ of the pre-signature.

$\mathsf{SigCom}'(ck, x, (V, W), \mathbf{C})$**.** This variant is defined as $\mathsf{SigCom}$ in Fig. 2, except that $A := (K \cdot T^r \cdot U \cdot V)^{\frac{1}{x+c}}$.

Using $\mathsf{SigCom}'$, the definition of $\mathsf{PSigCom}$ is straightforward. Note that $\mathsf{AdPrC}_{\mathcal{K}}$ can also be applied to outputs of $\mathsf{SigCom}'$ since proofs only depend on the left-hand sides of their equation. Let $\mathrm{E}_{\widehat{A}'}$ be $\mathrm{E}_{A'}$ with $Y$ being a variable. Since $\mathrm{E}_{A'}$ and $\mathrm{E}_A$ as well as $\mathrm{E}_{\widehat{A}}$ and $\mathrm{E}_{\widehat{A}'}$ have the same left-hand sides, $\mathsf{AdPrC}_{\mathcal{K}}$ also transforms a proof for $\mathrm{E}_{A'}$ into one for $\mathrm{E}_{\widehat{A}'}$.

$\mathsf{PSigCom}(ck, sk, (V, W), \mathbf{C})$**.**

- $(vk^*, sk^*) \leftarrow \mathsf{KeyGen}_{\mathsf{S}};\ \tau \leftarrow \mathcal{R}_{\mathcal{M}};\ \mathbf{C}_{vk^*} := \mathsf{Com}_{\mathcal{M}}(ck, vk^*, \tau);\ (\mathbf{c}_{\Sigma_0}, \pi_0) \leftarrow \mathsf{SigCom}(ck, sk, \mathbf{C}_{vk^*})$
- $(\mathbf{c}_{\Sigma_1}, \pi'_1) \leftarrow \mathsf{SigCom}(ck, sk^*, \mathbf{C});\ \pi_1 \leftarrow \mathsf{AdPrC}_{\mathcal{K}}(ck, (vk^*, \tau), \mathbf{C}, \mathbf{c}_{\Sigma_1}, \pi'_1)$
  $(\mathbf{c}_{\Sigma_2}, \pi'_2) \leftarrow \mathsf{SigCom}'(ck, sk^*, (V, W), \mathbf{C});\ \pi_2 \leftarrow \mathsf{AdPrC}_{\mathcal{K}}(ck, (vk^*, \tau), \mathbf{C}, \mathbf{c}_{\Sigma_2}, \pi'_2)$
  $(\mathbf{c}_{\Sigma_3}, \pi'_3) \leftarrow \mathsf{SigCom}'(ck, sk^*, (V, W)^3, \mathbf{C});\ \pi_3 \leftarrow \mathsf{AdPrC}_{\mathcal{K}}(ck, (vk^*, \tau), \mathbf{C}, \mathbf{c}_{\Sigma_3}, \pi'_3)$
- Return $\big(\mathbf{c}_\Sigma = (\mathbf{C}_{vk^*}, \mathbf{c}_{\Sigma_0}, \mathbf{c}_{\Sigma_1}, \mathbf{c}_{\Sigma_2}, \mathbf{c}_{\Sigma_3}), \pi = (\pi_0, \pi_1, \pi_2, \pi_3)\big)$

A proof of knowledge of a signature $(\mathbf{c}_\Sigma, \pi)$ under $vk$ on the message pair $(V, W) \in \mathcal{M}^*$ and $(M, N)$, which is given as a commitment $\mathbf{C} \in \mathcal{C}^*_{\mathcal{M}}$ is then verified by checking the following:

$$\begin{aligned} \mathbf{C}_{vk^*} \overset{?}{\in} \mathcal{C}_{\mathcal{M}},\ & \mathsf{Verify}(ck, \mathrm{E}_{\mathsf{Ver}(vk,\cdot,\cdot)}, \mathbf{C}_{vk^*}, \mathbf{c}_{\Sigma_0}, \pi_0),\ \mathsf{Verify}(ck, \mathrm{E}_{\mathsf{Ver}(\cdot,\cdot,\cdot)}, \mathbf{C}_{vk^*}, \mathbf{C}, \mathbf{c}_{\Sigma_1}, \pi_1), \\ & \mathsf{Verify}(ck, \mathrm{E}_{\mathsf{Ver}^*(\cdot,(V,W),\cdot,\cdot)}, \mathbf{C}_{vk^*}, \mathbf{C}, \mathbf{c}_{\Sigma_2}, \pi_2),\ \mathsf{Verify}(ck, \mathrm{E}_{\mathsf{Ver}^*(\cdot,(V,W)^3,\cdot,\cdot)}, \mathbf{C}_{vk^*}, \mathbf{C}, \mathbf{c}_{\Sigma_3}, \pi_3) \;\; . \end{aligned}$$

Proof adaptation $\mathsf{AdPrC}^*_{\mathcal{K}}$ for a verifiable encrypted signature of the above form $\big(\mathbf{c}_\Sigma = (\mathbf{C}_{vk^*}, \mathbf{c}_{\Sigma_0}, \mathbf{c}_{\Sigma_1}, \mathbf{c}_{\Sigma_2}, \mathbf{c}_{\Sigma_3}),$ $\pi = (\pi_0, \pi_1, \pi_2, \pi_3)\big)$ is done by running $\widehat{\pi}_0 \leftarrow \mathsf{AdPrC}_{\mathcal{K}}(ck, (vk, \xi), \mathbf{C}_{vk^*}, \mathbf{c}_{\Sigma_0}, \pi_0)$ and outputting $(\widehat{\pi}_0, \pi_1, \pi_2, \pi_3)$.