Commuting Signatures and Verifiable Encryption

Georg Fuchsbauer*

Dept. Computer Science, University of Bristol, UK
georg@cs.bris.ac.uk

Abstract. Verifiable encryption allows one to encrypt a signature while preserving its public verifiability. We introduce a new primitive called *commuting signatures and verifiable encryption* that extends this in multiple ways, such as enabling encryption of both signature and message while proving validity. More importantly, given a ciphertext, a signer can create a verifiably encrypted signature on the encrypted (unknown) message, which leads to the same result as first signing the message and then verifiably encrypting the message/signature pair; thus, signing and encrypting commute. Our instantiation is based on the recently introduced *automorphic signatures* and Groth-Sahai proofs, which we show to be homomorphic. We also prove a series of other properties and provide a novel approach to simulation.

As an application, we give an instantiation of *delegatable anonymous credentials*, a primitive introduced by Belenkiy et al. Our construction is arguably simpler than theirs and it is the first to provide *non-interactive* (and thus concurrently secure) issuing and delegation protocols, which are significantly more efficient. Moreover, the size of our credentials and the cost of verification are less than half of those of the previous instantiation. All our constructions are proven secure in the standard model under known non-interactive assumptions.

Keywords: Verifiably encrypted signatures, blind signatures, anonymous credentials, Groth-Sahai proofs.

1 Introduction

Verifiably encrypted signatures let us sign a message, encrypt the signature, and make a proof asserting that the ciphertext contains a valid signature. Suppose the message is only available as an encryption. We cannot make a signature on the plaintext, as this would contradict the security of the encryption scheme[1]. But could we instead, given a ciphertext, produce a *verifiable encryption* of a signature on the plaintext?

We show that 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 a verifiably encrypted certificate for the key, which acts as a credential.

* Work done while at École normale supérieure, Paris, France. The author has been supported by the French ANR 07-TCOM-013-04 PACE Project, the European Commission through the ICT Program under Contract ICT-2007-216676 ECRYPT II and EPSRC Grant EP/H043454/1.

[1] Given two messages and the encryption of one of them, a signature on the plaintext could be used to decide which one was encrypted.

K.G. Paterson (Ed.): Eurocrypt 2011, LNCS 6632, pp. 224–245, 2011.

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 usually does not issue credentials directly to each user, but relies on intermediate layers in the hierarchy. For example, a system administrator issues credentials to webmasters for using his server; the latter may then create forums and delegate rights to moderators, who can give posting privileges to users.

Web-based social network services are enjoying a huge popularity and represent another area of application for credentials. Registered users can be given credentials to access services, which they delegate to introduce and recommend friends and friend of friends. The recent rise of concern about protection of privacy in such networks motivates *anonymous* credentials: a user can obtain a credential and prove possession of it without revealing neither her identity nor that of the user who delegated it to her.

In practice, (non-anonymous) delegation of rights is usually realized by certifying (i.e., signing) the public key of the delegated user. Consecutive delegation leads to a certification chain, consisting of public keys and certificates linking them, starting with the *original issuer* of the credential. A user in the chain can delegate the credential by signing the delegatee's public key and appending the certificate to the credential.

Anonymous credentials [Cha85, Dam90, Bra99, LRSW00, CL01, CL04, BCKL08] aim to provide a functionality similar to certificates without revealing information about the user's identity when obtaining or showing a credential. However, the goal of reconciling delegatability and anonymity remained elusive—until recently. Chase and Lysyanskaya [CL06] show theoretical feasibility of delegatable anonymous credentials, but their size is exponential in the number of delegations. A breakthrough was then made in [BCC$^+$09], where Belenkiy et al. (BCCKLS) introduce a new approach using a non-interactive zero-knowledge (NIZK) proof system [BFM88] with *randomizable* proofs: anyone can transform such a proof into a new proof of the same statement that cannot be linked to the original one. A credential is a *proof of knowledge* of a certification chain that can be randomized before being delegated or shown, which guarantees anonymity and unlinkability.

In their model, each user holds a secret key which can be used to produce multiple unlinkable pseudonyms *Nym*. A user A can be known to user O as $Nym_A^{(O)}$ and to B as $Nym_A^{(B)}$. Given a credential issued by O for $Nym_A^{(O)}$, A can transform it into a credential from O for $Nym_A^{(B)}$ and show it to B. Moreover, A can delegate the credential to user C, known to A as $Nym_C^{(A)}$. C can then show a credential from O for $Nym_C^{(D)}$ to user D (without revealing neither $Nym_A^{(C)}$ nor $Nym_C^{(A)}$), or redelegate it. Delegation preserves anonymity, in that delegator and delegatee learn nothing more about each other than their respective pseudonyms. This is formalized by requiring that there exist a *simulator* that can produce pseudonyms and credentials for them without knowing any secrets.

In the instantiation of [BCC$^+$09], delegation is fairly complex and interactive—in contrast 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 bridge this gap by giving an instantiation that enables *non-interactive* delegation: given a pseudonym *Nym*, a delegator can produce a ready credential for the holder of *Nym* without any interaction. Note that a non-interactive delegation protocol immediately yields security against

concurrent attacks, where an adversary might simultaneously run protocols for delegating and being delegated credentials with honest users. This was not considered in the BCCKLS model.

Commuting Signatures. Our main building block for non-interactively delegatable anonymous credentials is a new primitive we call *commuting signature and verifiable encryption* (or *commuting signature* for short), which we sketch in the following and formally define in Section 3. It combines a digital signature scheme, an encryption scheme and a proof system with the following properties: given a verification key *vk*, a message M and a signature Σ on M under *vk*, we can encrypt any subset of $\{vk, M, \Sigma\}$ and add a proof (which leaks no more information) that the plaintexts are a key, a message and a valid signature—which makes the encryptions verifiable.

For consistency with the Groth-Sahai methodology [GS08], we also say *commitment* instead of *encryption*, as their commitments to group elements, which we will use, are *extractable*, i.e., we can recover the committed value (and thus "decrypt") using an *extraction key*. An extractable commitment to a signature together with a proof of validity is a *proof of knowledge* (PoK) of a signature, and at the same time a *verifiably encrypted signature* (VES) [BGLS03, RS09][2].

We denote committing to signatures by Com, committing to messages by $\mathsf{Com}_{\mathcal{M}}$, and proving validity by Prove. A proof for a committed signature is denoted by $\widetilde{\pi}$, and for a committed message by $\bar{\pi}$. If both are committed, we write π, and if the verification key is committed too, we write $\widehat{\pi}$ ("$\sim$" for signature and "$\wedge$" for $\underline{vk}$). Besides allowing us to prove validity of committed values, a commuting signature scheme provides the following functionalities, neither of which requires knowledge of the extraction 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 Σ on M under *sk*, and a proof π that the content of $\mathbf{c}_\Sigma$ is a valid signature on the content of $\mathbf{C}_M$.

$\mathsf{AdC}_{\mathcal{S}}$. Given a commitment $\mathbf{C}_M$ to M, a signature Σ on M and a proof $\bar{\pi}$ of validity of Σ on the content of $\mathbf{C}_M$, we can make a commitment $\mathbf{c}_\Sigma$ to Σ using randomness ρ_Σ. Then $\mathsf{AdC}_{\mathcal{S}}$ ("adapt when committing to signature") allows us to adapt $\bar{\pi}$ to a proof for $\mathbf{C}_M$ and $\mathbf{c}_\Sigma$: given $(\mathbf{C}_M, \Sigma, \rho_\Sigma, \bar{\pi})$ it returns a proof π that the content of $\mathbf{c}_\Sigma$ is a valid signature on the content of $\mathbf{C}_M$. $\mathsf{AdD}_{\mathcal{S}}$ ("adapt proof when *de*committing") does the converse: given a committed message $\mathbf{C}_M$, a committed signature $\mathbf{c}_\Sigma$ together with the used randomness ρ_Σ, and a proof π for $\mathbf{C}_M$ and $\mathbf{c}_\Sigma$, $\mathsf{AdD}_{\mathcal{S}}$ outputs a proof $\bar{\pi}$ of validity of the signature Σ on the committed message.

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

$\mathsf{AdC}_{\mathcal{K}}$. We can also adapt proofs when (de)committing to the *verification key*. Given commitments $\mathbf{C}_M$ and $\mathbf{c}_\Sigma$, a proof of validity π w.r.t. a verification key *vk*, and

[2] While for VES, encryption suffices to be one-way (*opacity* means it is hard to extract a signature), we require verifiable encryptions of different signatures to be *indistinguishable*.

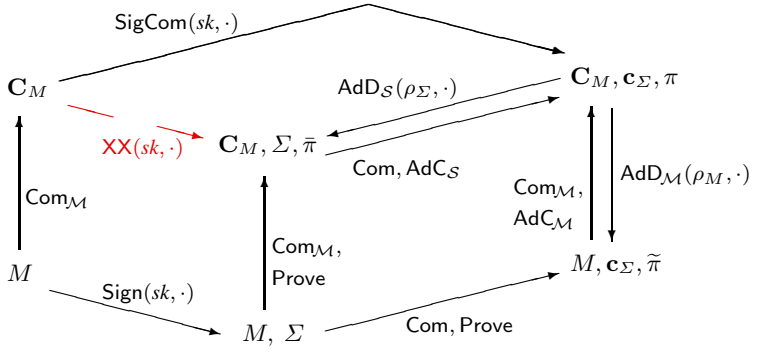

Fig. 1. Diagram representing a system of commuting signatures and verifiable encryption

randomness ρ_{vk}, $\mathsf{AdC}_{\mathcal{K}}$ outputs a proof $\widehat{\pi}$ that the content of $\mathbf{c}_{\Sigma}$ is a signature on the content of $\mathbf{C}_M$ valid under vk given as a commitment $\mathbf{c}_{vk}$ with randomness ρ_{vk}. $\mathsf{AdD}_{\mathcal{K}}$ is given $(vk, \rho_{vk}, \mathbf{C}_M, \mathbf{c}_{\Sigma}, \widehat{\pi})$ and adapts the proof $\widehat{\pi}$ for $(\mathbf{c}_{vk}, \mathbf{C}_M, \mathbf{c}_{\Sigma})$, where $\mathbf{c}_{vk}$ commits to vk with randomness ρ_{vk}, to a proof for $(vk, \mathbf{C}_M, \mathbf{c}_{\Sigma})$.

We require that committing, signing and the above functionalities all *commute* with each other, that is, it does not matter in which order they are executed; 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 SigCom. Thus, the diagram in Figure 1 commutes. Note that due to the argument given in Footnote 1, there cannot exist a functionality XX that is given a commitment $\mathbf{C}_M$ to a message M and a secret key sk, and outputs a signature Σ on M.

Besides verifiably encrypted signatures, commuting signatures imply *blind signatures*, and moreover *CL signatures* [CL02] and *P-signatures* [BCKL08], both building blocks for protocols providing privacy. They let a user obtain a signature on a committed value from a signer by running an *issuing* protocol. The user can then make a proof of knowledge of that signature, which is verifiable given the commitment. SigCom provides a non-interactive issuing, which directly gives the user a (randomizable) proof of knowledge of such a signature (see the full version [Fuc10]).

Instantiating Commuting Signatures. Blind signatures [Cha83, PS96] enable a user to obtain a signature on a message in a way that the signer cannot link the resulting message/signature pair to its issuing. In [Fuc09, AFG+10] the author gives an efficient implementation with *round-optimal* issuing [Fis06], where after sending information to the signer, the user can immediately derive the blind signature from the signer's response. In this scheme, the user *randomizes* the message, makes (extractable) commitments to the message and the randomness, and adds a witness-indistinguishable (WI) proof that the commitments contain the correct values.

The user sends these values to the signer, who learns nothing about the message from them. The signer fabricates a "pre-signature", which the user, knowing the values used to randomize the message, can transform into a signature on the message. The actual

blind signature is a WI proof of knowledge (PoK) of this signature, which prevents the signer from linking it to the signing session. This PoK is instantiated with Groth-Sahai (GS) commitments and proofs [GS08] for *pairing-product equations* (PPE), and the message space consists of pairs of group elements.

We require a lot more: the signer, without knowing the randomness used to hide the message, should not only make a commitment to a signature (which he cannot know—Footnote 1) on an unknown message, but in addition give a proof that this signature is valid. While the described blind signature scheme has the nice property that during issuing the user obtains an actual signature on the message, we show that its true potential has not yet been exploited. We first observe that the values the user sends to the signer for a blind signature can be seen as a commitment to the message. We then show that they actually suffice for the signer to *directly*—without the help of the user—construct a proof of knowledge of a signature on the message.

This is made possible by the specific structure of the signature, the fact that the commitments are homomorphic and a series of properties of the proof system. We prove that, besides being randomizable, Groth-Sahai proofs are *homomorphic*[3] w.r.t. the statement they prove (the product of two proofs is a proof for the product of the equations they prove), they are independent of parts of the statement—in some cases even of the committed value—, and there are ways to "blindly" transform a proof for one statement into a proof for another statement.

Instantiating Delegatable Anonymous Credentials. Belenkiy et al. [BCC+09] show that Groth-Sahai (GS) 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. To issue or delegate, the issuer and the user jointly compute a PoK of an authenticator on the content of the user's pseudonym. In the case of delegation, the issuer prepends her own credential, after randomizing it. Their authentication scheme must satisfy strong security notions (*F-unforgeability* and *certification security*), since secret keys cannot be extracted from the commitments, and an adversary against it must be allowed to ask for authenticators *on* as well as *under* the attacked key.

We avoid these notions and interactivity of delegation by following a more modular approach replacing the authenticators on secret keys by commuting signatures on verification keys. The underlying signatures are *automorphic* [Fuc09], which means that they are Groth-Sahai compatible and their verification keys lie in the message space—which is a requirement for delegation. A credential is then a chain of *verification 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 issuing and delegation: given a user's pseudonym Nym_U (i.e., a commitment to his verification key), the issuer can produce a commitment $\mathbf{c}_\Sigma$ to a signature on the value committed in Nym_U and a proof π of validity using SigCom. In the case of issuing, the credential is $(\mathbf{c}_\Sigma, \pi)$ and is verified by checking π on the issuer's public key, Nym_U and $\mathbf{c}_\Sigma$. In the case of delegation, the issuer also randomizes her own credential $cred_I$, yielding a credential $cred_I{}'$ on her pseudonym Nym_I that is unlinkable to $cred_I$. Running $\mathsf{AdC}_\mathcal{K}$, the issuer adapts the proof π (which

[3] For linear equations the homomorphic property of GS proofs was also noted in [DHLW10].

is valid under her verification key) to a proof $\widehat{\pi}$ of validity of the signature contained in $\mathbf{c}_\Sigma$ on the content of the pseudonym Nym_U under the content of the issuer's pseudonym Nym_I. The credential for the user is then $cred_I{}' \parallel Nym_I \parallel (\mathbf{c}_\Sigma, \widehat{\pi})$.

Comparing Our Results to Previous Ones. Replacing the authenticators from the BCCKLS scheme with our automorphic signatures already more than doubles the efficiency. In the full version [Fuc10] we revise the approach to achieving simulatability of credentials. Groth and Sahai show how to simulate *proofs of satisfiability* of equations, consisting of commitments and proofs for the committed values, which are produced by the simulator. However, in order to simulate credentials for a given pseudonym, the simulator has to construct proofs for *given* commitments. Belenkiy et al. therefore double some of the commitments and provide proofs of consistency. We show that our credentials can be directly simulated even if some of the commitments are fixed beforehand.

Finally, our issuing (and delegation) protocol is significantly more efficient. While in [BCC$^+$09], the issuer and the user run a complex two-party protocol using homomorphic encryption and interactive ZK proofs, in our instantiation the issuer simply sends a PoK of a signature. Both schemes are proven secure under the SXDH assumption and different "hidden" variants of the *strong Diffie-Hellman* assumption [BB04] (which are thus "q-type" assumptions): BB-CDH and BB-HSDH, introduced in [BCC$^+$09], for their scheme and ADH-SDH [AFG$^+$10] for ours (see Section 4.1).

Automorphic signatures were combined with GS proofs in [AFG$^+$10] to construct *anonymous proxy signatures* (APS) [FP08]. They also allow one to prove rights in an anonymous way, but there is no anonymity between the delegator and the delegated user. If in our credential scheme we give the extraction key to a tracing authority, and define a *proxy signing algorithm* similar to delegation but outputting a committed signature on a *clear* message, we get an instantiation of APS with *mutually anonymous* delegation.

Subsequent to our work, Blazy et al. [BFPV11] defined a primitive similar to commuting signatures, called *extractable signatures on randomizable ciphertexts*. While their instantiation solely relies on the *decision linear* assumption (DLIN) [BBS04], it is only efficient for small message spaces due to bit-by-bit techniques.

2 Preliminaries

We briefly recall the definitions and security requirements for the relevant primitives from the literature (and refer to the full version [Fuc10] for more details).

Commitments. We will use a (non-interactive) *randomizable extractable commitment scheme* $\mathbf{Com}$ which is composed of the algorithms Setup, Com, RdCom, $\mathsf{ExSetup}$, Extr, and $\mathsf{WISetup}$. By $\mathcal{V}$ we denote the space of "committable" values, by $\mathcal{R}$ the randomness space and by $\mathcal{C}$ the space of commitments. On input the security parameter 1^λ, Setup and $\mathsf{WISetup}$ output a *commitment key* ck, and $\mathsf{ExSetup}$ outputs (ck, ek), where ck is distributed as the output of Setup; ek is called the *extraction key*. On input ck, a message $M \in \mathcal{V}$ and randomness $\rho \in \mathcal{R}$, Com outputs a commitment $\mathbf{c} \in \mathcal{C}$.

The scheme is *perfectly binding*, i.e., for any $ck \leftarrow \mathsf{Setup}$ and any $\mathbf{c} \in \mathcal{C}$ there exists exactly one $M \in \mathcal{V}$ s.t. $\mathbf{c} = \mathsf{Com}(ck, M, \rho)$ for some ρ. If $(ck, ek) \leftarrow \mathsf{ExSetup}$ then $\mathsf{Extr}(ek, \mathbf{c})$ extracts that value M from $\mathbf{c}$. The keys output by $\mathsf{WISetup}$ are computationally indistinguishable from those output by Setup and generate *perfectly hiding*

commitments: for any $ck^* \leftarrow \mathsf{WISetup}$, $\mathbf{c} \in \mathcal{C}$ and $M \in \mathcal{V}$, there exists a $\rho \in \mathcal{R}$ s.t. $\mathbf{c} = \mathsf{Com}(ck^*, M, \rho)$. Finally, we have $\mathsf{RdCom}(ck, \mathsf{Com}(ck, M, \rho), \rho') = \mathsf{Com}(ck, M, \rho + \rho')$; thus RdCom *randomizes* commitments[4].

Proofs for Committed Values. We define a proof system that allows one to prove that committed values satisfy an equation. The proofs are constructed from the committed values and the used randomness, and they are witness indistinguishable, which means they do not reveal which satisfying values were used. Given a proof for a set of commitments, the proof can be adapted to a randomization of the commitments without knowledge of the committed values.

A *randomizable witness-indistinguishable proof system* $\mathbf{Proof}$ for a commitment scheme $\mathbf{Com}$ for a class $\mathcal{E}$ of equations consists of the algorithms Prove, Verify and RdProof. On input *ck*, an equation $\mathrm{E} \in \mathcal{E}$, values $M_1, \ldots, M_n \in \mathcal{V}$ satisfying E and $\rho_1, \ldots, \rho_n \in \mathcal{R}$, Prove outputs a proof π for the values $\mathsf{Com}(ck, M_1, \rho_1), \ldots, \mathsf{Com}(ck, M_n, \rho_n)$. On input *ck*, E, $\mathbf{c}_1, \ldots, \mathbf{c}_n$ and π, Verify outputs 0 or 1, indicating rejection or acceptance of π. Every proof generated for commitments to values satisfying an equation is accepted by Verify. Given *ck*, $\mathbf{c}_1, \ldots, \mathbf{c}_n$, a proof π for $(\mathbf{c}_1, \ldots, \mathbf{c}_n)$ and E, and $\rho'_1, \ldots, \rho'_n \in \mathcal{R}$, algorithm RdProof outputs a proof for the randomizations $\mathbf{c}'_i := \mathsf{RdCom}(ck, \mathbf{c}_i, \rho'_i)$; in particular, $\mathsf{RdProof}(ck, \mathrm{E}, (\mathbf{c}_1, \rho'_1), \ldots, (\mathbf{c}_n, \rho'_n), \pi)$ is distributed as $\mathsf{Prove}(ck, \mathrm{E}, (M_1, \rho_1 + \rho'_1), \ldots, (M_n, \rho_n + \rho'_n))$.

Soundness states that if there is a valid proof for a set of commitments for $\mathrm{E} \in \mathcal{E}$ then Extr extracts a set of values satisfying E. *Witness indistinguishability* is defined as follows: if the commitment key is output by WISetup then a set of commitments and a valid proof for them for an equation E reveals nothing, in an information-theoretical sense, about the committed values, except that they satisfy E.

Signatures. A signature scheme $\mathbf{Sig}$ consists of the following algorithms: $\mathsf{Setup}_\mathsf{S}$ takes as input the security parameter 1^λ and outputs parameters *pp*, which define a message space $\mathcal{M}$. On input *pp*, $\mathsf{KeyGen}_\mathsf{S}$ outputs a pair (vk, sk) of verification and signing key. For $M \in \mathcal{M}$, $\mathsf{Sign}(sk, M)$ outputs a signature Σ, which is verified by $\mathsf{Ver}(vk, M, \Sigma)$. If $pp \leftarrow \mathsf{Setup}_\mathsf{S}$ and $(vk, sk) \leftarrow \mathsf{KeyGen}_\mathsf{S}(pp)$ then $\mathsf{Ver}(vk, M, \mathsf{Sign}(sk, M)) = 1$ for all $M \in \mathcal{M}$. *Strong unforgeability* means that given *vk* and an oracle that queried on a message M_i returns a signature Σ_i on M_i, it is infeasible to output a pair (M, Σ), s.t. $\mathsf{Ver}(vk, M, \Sigma) = 1$ and $(M, \Sigma) \neq (M_i, \Sigma_i)$ for all i.

We require that $\mathbf{Sig}$ be *compatible* with $\mathbf{Com}$ and $\mathbf{Proof}$: the messages, verification keys and signatures are composed of values in $\mathcal{V}$ (the value space of $\mathbf{Com}$) and the signature verification predicate is a conjunction of equations from $\mathcal{E}$ (the class of equations for $\mathbf{Proof}$). We note that from a compatible triple $(\mathbf{Com}, \mathbf{Proof}, \mathbf{Sig})$ one can easily construct a *verifiably encrypted signature* scheme; see [Fuc10].

For our application to delegatable credentials we require furthermore that $\mathbf{Sig}$ be *automorphic*, that is, besides being compatible, its verification keys must lie in its message space $\mathcal{M}$. We let $\mathrm{E}_{\mathsf{Ver}}$ denote the verification equations for $\mathbf{Sig}$. When, for example, Σ is considered a variable we write $\mathrm{E}_{\mathsf{Ver}(vk, M, \cdot)}(\Sigma)$.

[4] Commitment schemes with two types of keys were called *perfectly hiding with extraction* in [GOS06] and *strongly computationally hiding* in [BCKL08]. Note that a scheme $\mathbf{Com}$ with the described properties is at the same time a *lossy encryption scheme* [BHY09].

3 Commuting Signatures and Verifiable Encryption

Commuting signatures extend a commitment scheme $\mathbf{Com}$, an associated proof system $\mathbf{Proof}$ and a compatible signature scheme $\mathbf{Sig}$ by the following functionalities: $\mathbf{Com}_{\mathcal{M}}$ is a commitment scheme with the same keys as $\mathbf{Com}$ and whose message space is that of $\mathbf{Sig}$. SigCom takes a $\mathbf{Com}_{\mathcal{M}}$ commitment and a signing key, and produces a commitment to a signature on the committed message and a proof of validity. SmSigCom simulates SigCom and is given a signature instead of the signing key. Moreover, the algorithms AdC and AdD *adapt proofs when (de)committing* to a signature (subscript $\mathcal{S}$), a message (subscript $\mathcal{M}$) or a verification key (subscript $\mathcal{K}$).

Definition 1. *A system of* ***commuting signatures and verifiable encryption*** *consists of an extractable commitment scheme* $\mathbf{Com} = (\mathsf{Setup}, \mathsf{Com}, \mathsf{RdCom}, \mathsf{ExSetup}, \mathsf{Extr}, \mathsf{WISetup})$ *with value space* $\mathcal{V}$ *and randomness space* $\mathcal{R}$*, a randomizable WI proof system* $\mathbf{Proof} = (\mathsf{Prove}, \mathsf{Verify}, \mathsf{RdProof})$ *for* $\mathbf{Com}$*, a compatible signature scheme* $\mathbf{Sig} = (\mathsf{Setup}_{\mathrm{S}}, \mathsf{KeyGen}_{\mathrm{S}}, \mathsf{Sign}, \mathsf{Ver})$ *and the following algorithms. We let* $ck \leftarrow \mathsf{Setup}$*,* $pp_{\mathrm{S}} \leftarrow \mathsf{Setup}_{\mathrm{S}}$*,* $(vk, sk) \leftarrow \mathsf{KeyGen}_{\mathrm{S}}(pp_{\mathrm{S}})$*,* $M \in \mathcal{M}$*,* $\mu \in \mathcal{R}_{\mathcal{M}}$ *and* $pp := (ck, pp_{\mathrm{S}})$*.*

$\mathbf{Com}_{\mathcal{M}}$*. On input pp, 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 pp,* $\mathbf{C}$ *and* $\mu' \in \mathcal{R}_{\mathcal{M}}$ *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* $\mathbf{Com}_{\mathcal{M}} := (\mathsf{Setup}, \mathsf{Com}_{\mathcal{M}}, \mathsf{RdCom}_{\mathcal{M}}, \mathsf{ExSetup}, \mathsf{Extr}_{\mathcal{M}}, \mathsf{WISetup})$ *to be a commitment scheme as defined in Section 2 and to be* compatible *with* $\mathbf{Proof}$*, i.e.,* Prove *and* RdProof *accept inputs from* $\mathcal{R}_{\mathcal{M}}$ *and* Verify *accepts* $\mathbf{Com}_{\mathcal{M}}$ *commitments.*

$\mathsf{AdC}_{\mathcal{S}}(pp, vk, \mathbf{C}, (\Sigma, \rho), \bar{\pi})$*. If* $\mathsf{Verify}(ck, \mathrm{E}_{\mathsf{Ver}(vk,\cdot,\Sigma)}, \mathbf{C}, \bar{\pi}) = 1$ *then the algorithm outputs* π *which is distributed as* $[\mathsf{Prove}(ck, \mathrm{E}_{\mathsf{Ver}(vk,\cdot,\cdot)}, (M, \mu), (\Sigma, \rho))]$*, where* M *and* μ *are such that* $\mathbf{C} = \mathsf{Com}_{\mathcal{M}}(pp, M, \mu)$*.*

$\mathsf{AdD}_{\mathcal{S}}(pp, vk, \mathbf{C}, (\Sigma, \rho), \pi)$*. If* $\mathsf{Verify}(ck, \mathrm{E}_{\mathsf{Ver}(vk,\cdot,\cdot)}, \mathbf{C}, \mathsf{Com}(ck, \Sigma, \rho), \pi) = 1$ *then the algorithm outputs* $\bar{\pi}$ *which is distributed as* $[\mathsf{Prove}(ck, \mathrm{E}_{\mathsf{Ver}(vk,\cdot,\Sigma)}, (M, \mu))]$*, where* M *and* μ *are such that* $\mathbf{C} = \mathsf{Com}_{\mathcal{M}}(pp, M, \mu)$*.*

$\mathsf{AdC}_{\mathcal{M}}(pp, vk, (M, \mu), \mathbf{c}_{\Sigma}, \widetilde{\pi})$*. If* $\mathsf{Verify}(ck, \mathrm{E}_{\mathsf{Ver}(vk,M,\cdot)}, \mathbf{c}_{\Sigma}, \widetilde{\pi}) = 1$ *then the algorithm outputs* π *which is distributed as* $[\mathsf{Prove}(ck, \mathrm{E}_{\mathsf{Ver}(vk,\cdot,\cdot)}, (M, \mu), (\Sigma, \rho))]$*, where* Σ *and* ρ *are such that* $\mathbf{c}_{\Sigma} = \mathsf{Com}(ck, \Sigma, \rho)$*.*

$\mathsf{AdD}_{\mathcal{M}}(pp, vk, (M, \mu), \mathbf{c}_{\Sigma}, \pi)$*. If* $\mathsf{Verify}(ck, \mathrm{E}_{\mathsf{Ver}(vk,\cdot,\cdot)}, \mathsf{Com}_{\mathcal{M}}(pp, M, \mu), \mathbf{c}_{\Sigma}, \pi) = 1$*, the algorithm outputs* $\widetilde{\pi}$ *which is distributed as* $[\mathsf{Prove}(ck, \mathrm{E}_{\mathsf{Ver}(vk,M,\cdot)}, (\Sigma, \rho))]$*, where* Σ *and* ρ *are such that* $\mathbf{c}_{\Sigma} = \mathsf{Com}(ck, \Sigma, \rho)$*.*

$\mathsf{AdC}_{\mathcal{K}}(pp, (vk, \xi), \mathbf{C}, \mathbf{c}_{\Sigma}, \pi)$*. 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* $[\mathsf{Prove}(ck, \mathrm{E}_{\mathsf{Ver}(\cdot,\cdot,\cdot)}, (vk, \xi), (M, \mu), (\Sigma, \rho)))]$*, where* M, μ, Σ *and* ρ *are such that* $\mathbf{C} = \mathsf{Com}_{\mathcal{M}}(pp, M, \mu)$ *and* $\mathbf{c}_{\Sigma} = \mathsf{Com}(ck, \Sigma, \rho)$*.*

$\mathsf{AdD}_{\mathcal{K}}(pp, (vk, \xi), \mathbf{C}, \mathbf{c}_{\Sigma}, \widehat{\pi})$*. 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* π*, distributed as* $[\mathsf{Prove}(ck, \mathrm{E}_{\mathsf{Ver}(vk,\cdot,\cdot)}, (M, \mu), (\Sigma, \rho)))]$*, where* M, μ, Σ *and* ρ *are such that* $\mathbf{C} = \mathsf{Com}_{\mathcal{M}}(pp, M, \mu)$ *and* $\mathbf{c}_{\Sigma} = \mathsf{Com}(ck, \Sigma, \rho)$*.*

$\mathsf{SigCom}(pp, sk, \mathbf{C})$. *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* μ *are such that* $\mathbf{C} = \mathsf{Com}_{\mathcal{M}}(pp, M, \mu)$.

$\mathsf{SmSigCom}(pp, ek, vk, \mathbf{C}, \Sigma)$. *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* $[\rho \leftarrow \mathcal{R} : (\mathsf{Com}(ck, \Sigma, \rho), \mathsf{Prove}(ck, \mathrm{E}_{\mathsf{Ver}(vk,\cdot,\cdot)}, (M, \mu), (\Sigma, \rho)))]$, *where* M *and* μ *are such that* $\mathbf{C} = \mathsf{Com}_{\mathcal{M}}(pp, M, \mu)$.

By $\mathbf{Algs} := (\mathsf{AdC}_{\mathcal{S}}, \mathsf{AdD}_{\mathcal{S}}, \mathsf{AdC}_{\mathcal{M}}, \mathsf{AdD}_{\mathcal{M}}, \mathsf{AdC}_{\mathcal{K}}, \mathsf{AdD}_{\mathcal{K}}, \mathsf{SigCom}, \mathsf{SmSigCom})$ we denote the algorithms of the system that extend $\mathbf{Com}, \mathbf{Proof}, \mathbf{Sig}$ and $\mathbf{Com}_{\mathcal{M}}$. When verifying a signature Σ on a message M by $\mathsf{Ver}(vk, M, \Sigma)$, we implicitly assume that Ver 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 $\mathbf{C} \in \mathcal{C}_{\mathcal{M}}$.

Definition 1 implies that running SigCom on a commitment to M yields the same (the output is distributed identically) as running $\Sigma \leftarrow \mathsf{Sign}(sk, M)$, $\mathsf{Com}_{\mathcal{M}}$ on M, Com on Σ and Prove for $\mathrm{E}_{\mathsf{Ver}(vk,\cdot,\cdot)}$; or running Sign, $\mathsf{Com}_{\mathcal{M}}$ on M and Prove for $\mathrm{E}_{\mathsf{Ver}(vk,\cdot,\Sigma)}$, and then Com on Σ and $\mathsf{AdC}_{\mathcal{S}}$, etc. This means that the diagram in Figure 1 *commutes*.

SmSigCom allows us to prove the following unforgeability property: Consider an adversary that is given (pp, ek, vk) for $(vk, sk) \leftarrow \mathsf{KeyGen}_{\mathsf{S}}$ and access to an oracle that on input $\mathbf{C}_i$ returns $(\mathbf{c}_i, \pi_i) \leftarrow \mathsf{SigCom}(pp, sk, \mathbf{C}_i)$. Then it cannot output a valid triple $(\mathbf{C}, \mathbf{c}_\Sigma, \pi)$ such that the committed message/signature pair is different from every pair committed in $(\mathbf{C}_i, \mathbf{c}_i)$ from the oracle calls. Commuting signatures moreover yield a round-optimal blind signature scheme: The user sends a commitment to the message to the signer, who runs SigCom to produce $(\mathbf{c}_\Sigma, \pi)$. The user computes a proof $\widetilde{\pi}$ for $\mathbf{c}_\Sigma$ and M via $\mathsf{AdD}_{\mathcal{M}}$, and outputs $(\mathbf{c}_\Sigma, \widetilde{\pi})$ as the blind signature (see [Fuc10]).

4 Instantiation of the Building Blocks

4.1 Bilinear Groups and Assumptions

A *bilinear group* is a tuple $\mathcal{G} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, G_1, G_2)$ 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: $\forall X \in \mathbb{G}_1\ \forall Y \in \mathbb{G}_2\ \forall a, b \in \mathbb{Z} : e(X^a, Y^b) = e(X, Y)^{ab}$, and $e(G_1, G_2)$ generates $\mathbb{G}_T$.

We denote group elements by capital letters and 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 $B = H^a$. We let $\mathcal{DH} := \{(G^a, H^a) \mid a \in \mathbb{Z}_p\}$ denote the set of DH pairs. Using the bilinear map e, $\mathcal{DH}$ is efficiently decidable by checking $e(G^{-1}, \underline{B})\, e(\underline{A}, H) = 1$. We will make the following assumptions.

Assumption 1 (SXDH). *The* Symmetric External Diffie-Hellman assumption *for* $\mathcal{G}$ *states that given* $(G_1, G_1^r, G_1^s, G_1^t)$ *for random* $r, s \in \mathbb{Z}_p$, *it is hard to decide whether* $t = rs$ *or* t *is random; likewise, given* $(G_2, G_2^{r'}, G_2^{s'}, G_2^{t'})$ *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* (ADH-SDH) was introduced in [AFG+10] and proven to hold in the generic-group model for any type of pairing. 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. Similarly to q-HSDH [BW07], ADH-SDH states that if c_i and v_i are given in a *hidden* form $(F^{c_i}, H^{c_i}, G^{v_i}, H^{v_i})$, it is intractable to produce another such tuple $((K\cdot G^{v})^{1/(x+c)}, F^{c}, H^{c}, G^{v}, H^{v})$.

Assumption 2 (q-ADH-SDH). *For randomly chosen $G, F, K \leftarrow \mathbb{G}_1, H \leftarrow \mathbb{G}_2$ and $x, c_i, v_i \leftarrow \mathbb{Z}_p$, given $(G, F, K, X=G^x;\ H, Y=H^x)$ and, for $1 \le i \le q-1$,*

$$\left(A_i = (K\cdot G^{v_i})^{\frac{1}{x+c_i}},\ B_i = F^{c_i},\ D_i = H^{c_i},\ V_i = G^{v_i},\ W_i = H^{v_i}\right)\ ,$$

it is hard to output $((K\cdot G^{v})^{1/(x+c)}, F^{c}, H^{c}, G^{v}, H^{v})$ with $(c, v) \neq (c_i, v_i)$ for all i.

The next assumption is a *weak* variant of the various *flexible CDH* assumptions, generalized to *asymmetric* bilinear groups. It was also introduced in [AFG+10] and shown to be implied by DDH in $\mathbb{G}_1$, and thus by SXDH.

Assumption 3 (AWF-CDH). *Given random generators $G \leftarrow \mathbb{G}_1$ and $H \leftarrow \mathbb{G}_2$, and $A = G^a$ for $a \leftarrow \mathbb{Z}_p$, it is hard to output $(G^r, G^{ra}, H^r, H^{ra})$ with $r \neq 0$.*

4.2 Groth-Sahai Proofs and Automorphic Signatures

Commitments. We instantiate $\mathbf{Com}$, defined in Section 2, by the commitment scheme based on SXDH from [GS08]. Setup, on input $\mathcal{G} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, G_1, G_2)$, outputs a commitment key $ck \in \mathbb{G}_1^{2\times 2} \times \mathbb{G}_2^{2\times 2}$. Value and randomness space are defined as $\mathcal{V} := \mathbb{G}_1 \cup \mathbb{G}_2$ and $\mathcal{R} := \mathbb{Z}_p^2$. $\mathsf{Com}(ck, X, r)$ takes randomness $r \in \mathcal{R}$ and an element $X \in \mathcal{V}$; commitments to $\mathbb{G}_1$-elements are in $\mathbb{G}_1^2$ and commitments to $\mathbb{G}_2$-elements are in $\mathbb{G}_2^2$. We have $\mathsf{Com}(ck, X, r) \circ \mathsf{Com}(ck, X', r') = \mathsf{Com}(ck, X\cdot X', r+r')$, where "$\circ$" denotes component-wise multiplication; the commitments are thus *homomorphic*.

$\mathsf{RdCom}(ck, \mathbf{c}, r')$ returns $\mathbf{c}' := \mathbf{c} \circ \mathsf{Com}(ck, 1, r')$, which for $\mathbf{c} = \mathsf{Com}(ck, X, r)$ is $\mathbf{c}' = \mathsf{Com}(ck, X, r+r')$. $\mathsf{ExSetup}$ constructs ck as in Setup and in addition outputs the used randomness as ek, and $\mathsf{Extr}(ek, \mathbf{c})$ outputs the committed value. $\mathsf{WISetup}$ produces a commitment key ck^* that is indistinguishable from outputs of Setup under the SXDH assumption. Commitments under ck^* are independent of the committed value.

Proofs for Committed Values. In order to instantiate $\mathbf{Proof}$ for $\mathbf{Com}$, we use the proof system from [GS08], which was shown to be randomizable in [BCC+09]. 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[5]

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

[5] For a more concise exposition we will underline the variables of an equation.

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 $\mathbf{t}_T \in \mathbb{G}_T$. We refer to [GS08] or [Fuc10] for a description of the implementations of the following: Prove, which chooses internal randomness $Z \leftarrow \mathbb{Z}_p^{2\times 2}$, and outputs $\pi \in \mathbb{G}_2^{2\times 2} \times \mathbb{G}_1^{2\times 2}$ (we write $\mathsf{Prove}(ck, \mathrm{E}, (X_i, r_i)_{i=1}^m, (Y_j, s_j)_{j=1}^n; Z)$ if we want to make Z explicit); $\mathsf{RdProof}(ck, \mathrm{E}, (\mathbf{c}_i, r_i)_{i=1}^m, (\mathbf{d}_j, s_j)_{j=1}^n, \pi)$, which adapts a proof π to the new commitments output by $\mathsf{RdCom}(ck, \mathbf{c}_i, r_i)$ and $\mathsf{RdCom}(ck, \mathbf{d}_j, s_j)$; and $\mathsf{Verify}(ck, \mathrm{E}, \vec{\mathbf{c}}, \vec{\mathbf{d}}, \pi)$.

Signatures. We instantiate $\mathbf{Sig}$ with the automorphic signature scheme presented in [AFG+10]. It is compatible, as signature components are in the space for committed values $\mathcal{V} = \mathbb{G}_1 \cup \mathbb{G}_2$, and the verification equations are pairing-product equations, thus in $\mathcal{E}$. Moreover, the verification keys lie in its message space, the set of Diffie-Hellman pairs. Under q-ADH-SDH and AWF-CDH (which is implied by SXDH), $\mathbf{Sig}$ is strongly unforgeable against adversaries making up to $q-1$ adaptive chosen-message queries, as shown in [Fuc09].

Scheme 1 (Sig). $\mathsf{Setup}_\mathrm{S}$ has input a bilinear group $\mathcal{G}$ and outputs random generators $F, K, T \leftarrow \mathbb{G}_1$. The message space is $\mathcal{DH} := \{(G^m, H^m) \,|\, m \in \mathbb{Z}_p\}$.

$\mathsf{KeyGen}_\mathrm{S}$ chooses $x \leftarrow \mathbb{Z}_p$ and outputs $vk = (G^x, H^x)$ and $sk = x$.

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

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

Ver on input a verification key $(X, Y) \in \mathcal{DH}$, a message $(M, N) \in \mathcal{DH}$ and a signature (A, B, D, R, S) outputs 1 if and only if the following equalities hold:

$$e(A, Y \cdot D) = e(K \cdot M, H)\, e(T, S) \qquad \begin{matrix} e(B, H) = e(F, D) \\ e(R, H) = e(G, S) \end{matrix} \tag{2}$$

Under SXDH and ADH-SDH, $\mathbf{Com}, \mathbf{Proof}$ and $\mathbf{Sig}$ are instantiations of the primitives defined in Section 2. Note that if we based GS proofs on DLIN instead of SXDH, the security of our constructions would follow from DLIN, ADH-SDH and AWF-CDH, which can all be made for bilinear groups of every type (1, 2 and 3) from [GPS08].

5 Additional Properties of Groth-Sahai Proofs

We identify four properties of Groth-Sahai (GS) proofs which will allow us to instantiate commuting signatures. We refer to [Fuc10] for the proofs and further results. First, proofs are independent of the right-hand side of the equation, and if the equation does not contain pairings of two variables, i.e., $\gamma_{ij} = 0$ for all i, j in (1), then they are even independent of the committed values.

Lemma 1. *For any equation* $\mathrm{E} \in \mathcal{E}$ *the output of* $\mathsf{Prove}(\cdot, \mathrm{E}, \cdot, \cdot)$ *is independent of* $\mathbf{t}_T$.

Lemma 2. *Proofs for equations for which* $\gamma_{ij} = 0$ *for all* i, j *depend only on the randomness of the commitments.*

Groth-Sahai (GS) proofs are homomorphic w.r.t. the equations, in that the product of two proofs is a proof for the "product of the respective equations". More precisely, given two 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 π for commitments $(\vec{\mathbf{c}}, \vec{\mathbf{d}})$ for E and a proof π' for commitments $(\vec{\mathbf{c}}', \vec{\mathbf{d}}')$ for E′, then $\pi'' := \pi \circ \pi'$ is a proof for commitments $((\vec{\mathbf{c}}, \vec{\mathbf{c}}'), (\vec{\mathbf{d}}, \vec{\mathbf{d}}'))$ and equation E″ defined as $\prod e(A_i, \underline{Y_i}) \prod e(A'_i, \underline{Y'_i}) \prod e(\underline{X_i}, B_i) \prod e(\underline{X'_i}, B'_i) \prod\prod e(\underline{X_i}, \underline{Y_j})^{\gamma_{i,j}} \cdot \prod\prod e(\underline{X'_i}, \underline{Y'_j})^{\gamma'_{i,j}} \;=\; \mathbf{t}''_T$ (for arbitrary $\mathbf{t}''_T \in \mathbb{G}_T$).

Lemma 3. *For* E, E′ *and* E″ *as above, 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 the following equation holds:* $\pi \circ \pi' = \mathsf{Prove}(ck, \mathrm{E}'', (X_i, r_i)_{i=1}^m, (X'_i, r'_i)_{i=1}^{m'}, (Y_j, s_j)_{j=1}^n, (Y'_j, s'_j)_{j=1}^{n'};\ Z + Z')$.

Given a proof for an equation, one can commit to its constants and adapt the proof. Consider $\mathrm{E}(X_1, \ldots, X_m;\ Y_1, \ldots, Y_n)$ as in (1) and a proof π for E and commitments $(\mathbf{c}_1, \ldots, \mathbf{c}_m;\ \mathbf{d}_1, \ldots, \mathbf{d}_n)$. Then π is also a proof for $\mathrm{E}'(\vec{X}, A_k;\ \vec{Y})$ defined as

$$\prod_{\substack{i=1\\ i\neq k}}^{n} e(A_i, \underline{Y_i}) \prod_{i=1}^{m} e(\underline{X_i}, B_i) \prod_{i=1}^{m}\prod_{j=1}^{n} e(\underline{X_i}, \underline{Y_j})^{\gamma_{i,j}}\, e(\underline{A_k}, \underline{Y_k}) \;=\; \mathbf{t}_T$$

and commitments $(\mathbf{c}_1, \ldots, \mathbf{c}_m, \mathsf{Com}(ck, A_k, 0);\ \mathbf{d}_1, \ldots, \mathbf{d}_n)$. We have thus:

Lemma 4. *Let* $\pi \leftarrow \mathsf{Prove}\big(ck, \mathrm{E}, (X_i, r_i)_{i=1}^m, (Y_j, s_j)_{j=1}^n\big)$, $\mathbf{c}_i = \mathsf{Com}(ck, X_i, r_i)$ *and* $\mathbf{d}_j = \mathsf{Com}(ck, Y_j, s_j)$ *for all* i, 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* $\mathsf{Prove}\big(ck, \mathrm{E}', (X_i, r_i)_{i=1}^m, (A_k, r), (Y_j, s_j)_{j=1}^n\big)$. *An analogous result holds for committing to a constant* $B_k \in \mathbb{G}_2$.

6 Instantiation of Commuting Signatures

We explain how to implement commuting signatures; due to space constraints we refer to [Fuc10] for more details and the proofs. In [Fuc09, AFG+10], a blind signature scheme is constructed from the scheme **Sig** (Scheme 1) as follows. Given parameters (G, H, F, K, T) and a message $(M, N) \in \mathcal{DH} := \{(G^m, H^m) \,|\, m \in \mathbb{Z}_p\}$, the user chooses a random $t \leftarrow \mathbb{Z}_p$ and *blinds* the first message component by the factor T^t. He then sends the following to the signer: $U := T^t \cdot M$, commitments $\mathbf{c}_M, \mathbf{c}_N, \mathbf{c}_P$ and $\mathbf{c}_Q$ to $M, N, P := G^t$ and $Q := H^t$, respectively; and proofs π_M, π_P and π_U proving $(M, N), (P, Q) \in \mathcal{DH}$ and well-formedness of U. With

$$\mathrm{E}_{\mathcal{DH}}(M, N) \;:\; e(G^{-1}, \underline{N})\, e(\underline{M}, H) = 1 \tag{3}$$
$$\mathrm{E}_U(M, Q) \;:\; e(T^{-1}, \underline{Q})\, e(\underline{M}, H^{-1}) = e(U, H)^{-1} \tag{4}$$

π_M proves $\mathrm{E}_{\mathcal{DH}}(M, N)$, π_P proves $\mathrm{E}_{\mathcal{DH}}(P, Q)$, and π_U proves $\mathrm{E}_U(M, Q)$, which asserts $U = T^t \cdot M$.

The signer replies with a "pre-signature" on U, defined in (6) below, which the user converts into a signature and outputs a GS proof of knowledge of it. Now to turn this into a commuting signature, there are two key observations.

1. 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 can 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 under a key $ck^* \leftarrow \mathsf{WISetup}$.
2. As $\mathbf{Com}$ is homomorphic, the values $\mathbf{c}_P$ and $\mathbf{c}_Q$ contained in the commitment $\mathbf{C}$ to (M, N) can be used by the signer to compute commitments to an actual signature. Moreover, below we show how π_P and π_U can be used to make a proof of validity using Lemmas 1, 2, 3 and 4.

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

Commitments to Messages. To instantiate $\mathbf{Com}_{\mathcal{M}}$, we define a commitment to a message $(M, N) \in \mathcal{DH}$ as $\mathbf{C}$ discussed above. For parameters $pp = (\mathcal{G}, ck, F, K, T)$ the space of valid commitments is thus defined as

$$\mathcal{C}_{\mathcal{M}}(pp) := \big\{(\mathbf{c}_M, \mathbf{c}_N, \pi_M, \mathbf{c}_P, \mathbf{c}_Q, \pi_P, U, \pi_U) \;\big|\; \mathsf{Verify}(ck, \mathrm{E}_{\mathcal{DH}}, \mathbf{c}_M, \mathbf{c}_N, \pi_M) \\ \wedge\; \mathsf{Verify}(ck, \mathrm{E}_{\mathcal{DH}}, \mathbf{c}_P, \mathbf{c}_Q, \pi_P) \;\wedge\; \mathsf{Verify}(ck, \mathrm{E}_U, \mathbf{c}_M, \mathbf{c}_Q, \pi_U)\big\}\ ,$$

and the randomness space is $\mathcal{R}_{\mathcal{M}} := \mathbb{Z}_p \times \mathcal{R}^4$. The algorithms $\mathsf{Com}_{\mathcal{M}}, \mathsf{RdCom}_{\mathcal{M}}$ and $\mathsf{Extr}_{\mathcal{M}}$ defining $\mathbf{Com}_{\mathcal{M}}$ are given in Figure 2.

Committing to a Signature on a Committed Message and Proving Validity. We now show how the signer can use the values in $\mathbf{C}$ to produce a proof of knowledge $(\mathbf{c}_A, \mathbf{c}_B, \mathbf{c}_D, \mathbf{c}_R, \mathbf{c}_S, \pi_A, \pi_B, \pi_R)$ of a signature (A, B, D, R, S) (i.e., a verifiably encrypted signature) on the message committed in $\mathbf{C}$. The proofs π_A, π_B and π_R attest that the values committed in $(\mathbf{c}_A, \mathbf{c}_B, \mathbf{c}_D, \mathbf{c}_R, \mathbf{c}_S)$ and $\mathbf{c}_M$ contained in $\mathbf{C}$ satisfy the verification equations in (2):

$$\begin{aligned} \mathrm{E}_A(A, M;\, S, D) \;&:\; e(T^{-1}, \underline{S})\, e(\underline{A}, Y) e(\underline{M}, H^{-1})\, e(\underline{A}, \underline{D}) = e(K, H) \\ \mathrm{E}_B(B;\, D) \;&:\; e(F^{-1}, \underline{D})\, e(\underline{B}, H) = 1 \\ \mathrm{E}_R(R;\, S) \;&:\; e(G^{-1}, \underline{S})\, e(\underline{R}, H) = 1 \end{aligned} \tag{5}$$

In the blind signature scheme, after receiving $\mathbf{C}$, the signer checks the proofs contained in it, and then produces a pre-signature, which is constructed as a signature on U, but on a message that lacks the second component: choose $c, r \leftarrow \mathbb{Z}_p$ and compute

$$A := (K \cdot T^r \cdot U)^{1/(x+c)} \qquad B := F^c \qquad D := H^c \qquad R' := G^r \qquad S' := H^r \tag{6}$$

Since $U := T^t \cdot M$, we have $A = (K \cdot T^r \cdot U)^{1/(x+c)} = (K \cdot T^{r+t} \cdot M)^{1/(x+c)}$, which is the first component of a signature on (M, N) with randomness $r + t$. Knowing t,

$\mathbf{Com}_{\mathcal{M}}$ on input pp, $(M,N) \in \mathcal{DH}$, $(t,\mu,\nu,\rho,\sigma) \in \mathcal{R}_{\mathcal{M}}$ sets $P = G^t$, $Q = H^t$ and returns

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

$\mathbf{RdCom}_{\mathcal{M}}$ has input pp, $\mathbf{C}$ and $(t',\mu',\nu',\rho',\sigma') \in \mathcal{R}_{\mathcal{M}}$, and returns $\mathbf{C}'$. It replaces t by $t+t'$ setting $U' := U \cdot T^{t'}$, $\hat{\mathbf{c}}_P := \mathbf{c}_P \circ \mathsf{Com}(ck, G^{t'}, 0)$, and $\hat{\mathbf{c}}_Q := \mathbf{c}_Q \circ \mathsf{Com}(ck, H^{t'}, 0)$. It then replaces the remaining randomness (μ,ν,ρ,σ) by $(\mu+\mu', \nu+\nu', \rho+\rho', \sigma+\sigma')$, setting

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

$\mathbf{Extr}_{\mathcal{M}}(ek, \mathbf{C} = (\mathbf{c}_M, \mathbf{c}_N, \pi_M, \mathbf{c}_P, \mathbf{c}_Q, \pi_P, U, \pi_U))$ outputs $(\mathsf{Extr}(ek, \mathbf{c}_M), \mathsf{Extr}(ek, \mathbf{c}_N))$.

$\mathbf{SigCom}(pp, sk, \mathbf{C})$. 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 π_M, π_P and π_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:

$$\begin{array}{lll} A := (K \cdot T^r \cdot U)^{\frac{1}{x+c}} & \mathbf{c}_B := \mathsf{Com}(ck, F^c, \beta) & \mathbf{c}_R := \mathbf{c}_P \circ \mathsf{Com}(ck, G^r, \rho') \\ \mathbf{c}_A := \mathsf{Com}(ck, A, \alpha) & \mathbf{c}_D := \mathsf{Com}(ck, H^c, \delta) & \mathbf{c}_S := \mathbf{c}_Q \circ \mathsf{Com}(ck, H^r, \sigma') \\ \pi'_A := \pi_U \circ \mathsf{Prove}(ck, \mathrm{E}_{A^\dagger}, (A,\alpha), (H^c,\delta);\ 0) & & \text{(with } \mathrm{E}_{A^\dagger} \text{ being Equation (9))} \\ \pi_A \leftarrow \mathsf{RdProof}(ck, \mathrm{E}_A, (\mathbf{c}_A,0), (\mathbf{c}_D,0), (\mathbf{c}_M,0), (\mathbf{c}_S,\sigma'), \pi'_A) & & \\ \pi_R \leftarrow \mathsf{RdProof}(ck, \mathrm{E}_R, (\mathbf{c}_R,\rho'), (\mathbf{c}_S,\sigma'), \pi_P) & \pi_B \leftarrow \mathsf{Prove}(ck, \mathrm{E}_{\mathcal{DH}}, (F^c,\beta), (H^c,\delta)) & \end{array}$$

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

Fig. 2. Committing to messages and making commitments to a signature on a committed value

the user can fabricate an actual signature on (M,N) from the pre-signature by setting $R := R' \cdot G^t = G^{r+t}$ and $S := S' \cdot H^t = H^{r+t}$. (A,B,D,R,S) is then a signature on (M,N) with randomness $(c, r+t)$.

Let μ,ρ and σ denote the randomness of the respective commitments $\mathbf{c}_M, \mathbf{c}_P$ and $\mathbf{c}_Q$, contained in $\mathbf{C}$. Since the commitments are homomorphic, the signer can—without knowing $P = G^t$ and $Q = H^t$—compute commitments to R and S from $\mathbf{c}_P$ and $\mathbf{c}_Q$:

$$\begin{array}{l} \mathbf{c}_R := \mathsf{Com}(ck, R', 0) \circ \mathbf{c}_P = \mathsf{Com}(ck, R, \rho) \\ \mathbf{c}_S := \mathsf{Com}(ck, S', 0) \circ \mathbf{c}_Q = \mathsf{Com}(ck, S, \sigma) \end{array} \tag{7}$$

The signer then chooses $\alpha,\beta,\delta \leftarrow \mathcal{R}$, and makes the remaining commitments:

$$\mathbf{c}_A := \mathsf{Com}(ck, A, \alpha) \qquad \mathbf{c}_B := \mathsf{Com}(ck, B, \beta) \qquad \mathbf{c}_D := \mathsf{Com}(ck, D, \delta) \tag{8}$$

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 to the signature $\Sigma = (A,B,D,R,S)$ on (M,N). It remains to construct proofs π_A, π_B and π_R for the 3

equations in (5)—without knowledge of the randomness μ, ρ and σ of the commitments $\mathbf{c}_M, \mathbf{c}_R$ and $\mathbf{c}_S$! This can be done by observing the following:

1. Equation $\mathrm{E}_R(R;\ S)$ is actually $\mathrm{E}_{\mathcal{DH}}(R;\ S)$ from (3). Since by (7) $\mathbf{c}_R$ and $\mathbf{c}_P$ have the same randomness ρ, and $\mathbf{c}_S$ and $\mathbf{c}_Q$ have the same randomness σ, and since by Lemma 2 proofs for $\mathrm{E}_{\mathcal{DH}}$ are independent of the committed values, π_P (the proof for $\mathbf{c}_P$ and $\mathbf{c}_Q$ for $\mathrm{E}_{\mathcal{DH}}$) is also a proof for $\mathbf{c}_R$ and $\mathbf{c}_S$; we thus set $\pi_R := \pi_P$.
2. Lemmas 1 and 2 yield that proofs for E_U (Equation (4)) 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, π_U is not only a proof for $\mathrm{E}_U(M; Q)$ but also for

$$\mathrm{E}_{U^\dagger}(M;\ S)\ :\ e(T^{-1}, \underline{S})\, e(\underline{M}, H^{-1}) = \mathbf{t}_T$$

(for an arbitrary $\mathbf{t}_T \in \mathbb{G}_T$)[6] for $\mathbf{c}_M$ and $\mathbf{c}_S$. Moreover, the signer, knowing A, D, α and δ, can produce a proof $\pi_{A^\dagger} \leftarrow \mathsf{Prove}(ck, \mathrm{E}_{A^\dagger}, (A, \alpha), (D, \delta))$ for

$$\mathrm{E}_{A^\dagger}(A;\ D)\ :\ e(\underline{A}, Y)\, e(\underline{A}, \underline{D}) = \mathbf{t}_T \tag{9}$$

(for any $\mathbf{t}_T$). Since the product of the left-hand sides of $\mathrm{E}_{U^\dagger}(M; S)$ and $\mathrm{E}_{A^\dagger}(A; D)$ is the left-hand side of $\mathrm{E}_A(A, M;\ S, D)$ from (5), Lemma 3 yields that $\pi_A := \pi_U \circledcirc \pi_{A^\dagger}$ is a proof for E_A.

We have thus shown how the signer can construct π_A and π_R. The remaining proof π_B can be made regularly, since the required randomness (β, δ) was chosen by the signer. Finally, to get a *random* proof of knowledge, the signer randomizes all commitments and proofs using RdCom and $\mathsf{RdProof}$ as defined in Section 4.2. Algorithm SigCom, with some optimizations, is summarized in Figure 2.

Instantiation of SmSigCom. This algorithm is similar to SigCom but instead of the signing key *sk* it is given a signature (A, B, D, R, S) and the extraction key. It proceeds like SigCom but starting from a signature instead of producing a pre-signature: choose $\alpha, \beta, \delta, \rho', \sigma' \leftarrow \mathcal{R}$ and set $\mathbf{c}_A, \mathbf{c}_B$ and $\mathbf{c}_D$ as in (8); use *ek* to extract P and Q from $\mathbf{C}$ and set $\mathbf{c}_R := \mathbf{c}_P \circledcirc \mathsf{Com}(ck, R \cdot P^{-1}, \rho') = \mathsf{Com}(ck, R, \rho + \rho')$ and $\mathbf{c}_S := \mathbf{c}_Q \circledcirc \mathsf{Com}(ck, S \cdot Q^{-1}, \sigma') = \mathsf{Com}(ck, S, \sigma + \sigma')$. Now π_A, π_B and π_R can be computed as in SigCom in Figure 2.

Instantiations of Proof Adaptation for Committing and Decommitting. We define equations $\mathrm{E}_{\tilde{A}}$ and $\mathrm{E}_{\bar{A}}$ and recall E_A, which all represent the first verification equation in (2) but with different elements being variables.

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

[6] Technically, π_U is only a *proof* for $\mathrm{E}_{U^\dagger}$ when $\mathbf{t}_T$ is s.t. M and S satisfy it. However, since the proofs are independent of the right-hand side, the prover need not know the appropriate $\mathbf{t}_T$.

With E_B and E_R defined in (5) we can write the following

$$\mathrm{E}_{\mathsf{Ver}((X,Y),\cdot,\cdot)}((M,N),(A,B,D,R,S)) \equiv \mathrm{E}_A(A,M;\ S,D) \wedge \mathrm{E}_B(B;\ D) \wedge \mathrm{E}_R(R;\ S) \quad (10)$$

$$\mathrm{E}_{\mathsf{Ver}((X,Y),(M,N),\cdot)}(A,B,D,R,S) \equiv \mathrm{E}_{\widetilde{A}}(A;\ S,D) \wedge \mathrm{E}_B(B;\ D) \wedge \mathrm{E}_R(R;\ S) \quad (11)$$

$$\mathrm{E}_{\mathsf{Ver}((X,Y),\cdot,(A,B,D,R,S))}(M,N) \equiv \mathrm{E}_{\bar{A}}(M) \quad (12)$$

$\mathsf{AdC}_{\mathcal{S}}$ transforms proofs $\bar{\pi}$ for $\mathrm{E}_{\mathsf{Ver}(vk,\cdot,\Sigma)}$ (12) into proofs π for $\mathrm{E}_{\mathsf{Ver}(vk,\cdot,\cdot)}$ (10), $\mathsf{AdC}_{\mathcal{M}}$ transforms proofs $\widetilde{\pi}$ for $\mathrm{E}_{\mathsf{Ver}(vk,(M,N),\cdot)}$ (11) into proofs π for $\mathrm{E}_{\mathsf{Ver}(vk,\cdot,\cdot)}$, whereas $\mathsf{AdD}_{\mathcal{S}}$ and $\mathsf{AdD}_{\mathcal{M}}$ do the converse.

Since the product of the left-hand sides of $\mathrm{E}_{\widetilde{A}}$ and $\mathrm{E}_{\bar{A}}$ is the left-hand side of E_A, by Lemma 3 we have $\pi_A = \pi_{\widetilde{A}} \circ \pi_{\bar{A}}$, which lets us transform proofs for equations $\mathrm{E}_{\widetilde{A}}$ and $\mathrm{E}_{\bar{A}}$ into proofs for E_A and vice versa, and thus implement the four algorithms. Note that when a proof is multiplied by a freshly generated proof, it is uniformly distributed: by Lemma 3, if Z is the internal randomness of the proof and Z' that of the fresh proof then the randomness of the product of proofs is $Z + Z'$. However, when reusing a proof it must be randomized first.

$\mathsf{AdC}_{\mathcal{S}}(pp, vk, \mathbf{C}, ((A,B,D,R,S),(\alpha,\beta,\delta,\rho,\sigma)), \bar{\pi})$. Proof $\bar{\pi}$ being for (12), it sets

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

for E_B and E_R defined in (5). It then returns $\pi := (\pi_{\widetilde{A}} \circ \pi_{\bar{A}}, \pi_B, \pi_R)$.

$\mathsf{AdD}_{\mathcal{S}}(pp, vk, \mathbf{C}, ((A,B,D,R,S),(\alpha,\beta,\delta,\rho,\sigma)), \pi)$. The proof π is of the form (π_A, π_B, π_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 componentwise division, i.e., multiplying every component of π_A with the inverse of that component of $\pi_{\widetilde{A}}$).

$\mathsf{AdC}_{\mathcal{M}}(pp, vk, ((M,N),(t,\mu,\nu,\rho,\sigma)), (\mathbf{c}_A,\mathbf{c}_B,\mathbf{c}_D,\mathbf{c}_R,\mathbf{c}_S), \widetilde{\pi})$. The proof $\widetilde{\pi}$ is of the form $(\pi_{\widetilde{A}}, \pi_B, \pi_R)$. The algorithm returns $\pi := (\pi_{\widetilde{A}} \circ \pi_{\bar{A}}, \pi'_B, \pi'_R)$ with

$$\pi_{\bar{A}} \leftarrow \mathsf{Prove}(ck, \mathrm{E}_{\bar{A}}, (M,\mu)) \qquad \begin{array}{l} \pi'_B \leftarrow \mathsf{RdProof}(ck, \mathrm{E}_B, (\mathbf{c}_B,0),(\mathbf{c}_D,0),\pi_B) \\ \pi'_R \leftarrow \mathsf{RdProof}(ck, \mathrm{E}_R, (\mathbf{c}_R,0),(\mathbf{c}_S,0),\pi_R) \end{array}$$

$\mathsf{AdD}_{\mathcal{M}}(pp, vk, ((M,N),(t,\mu,\nu,\rho,\sigma)), (\mathbf{c}_A,\mathbf{c}_B,\mathbf{c}_D,\mathbf{c}_R,\mathbf{c}_S), \pi)$. The proof π is of the form (π_A, π_B, π_R). The algorithm computes $\pi_{\bar{A}}, \pi'_B$ and π'_R as for $\mathsf{AdC}_{\mathcal{M}}$ above, and returns $\widetilde{\pi} := (\pi_A \oslash \pi_{\bar{A}}, \pi'_B, \pi'_R)$.

Instantiation of $\mathsf{AdC}_{\mathcal{K}}$ and $\mathsf{AdD}_{\mathcal{K}}$. A commitment $\mathbf{c}_{vk}$ to $vk = (X,Y) \in \mathcal{DH}$ is defined as $(\mathsf{Com}(ck,X,\xi), \mathsf{Com}(ck,Y,\psi), \mathsf{Prove}(ck, \mathrm{E}_{\mathcal{DH}}, (X,\xi),(Y,\psi)))$. The equations for $\mathrm{E}_{\mathsf{Ver}(\cdot,\cdot,\cdot)}((X,Y),(M,N),(A,B,D,R,S))$, i.e., when the key, the message and the signature are committed, are represented by

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

and E_B and E_R from (5). 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, (X, Y) and a proof $\pi = (\pi_A, \pi_B, \pi_R)$ of validity, by Lemma 4 the component π_A can be adapted to $\pi_{\widehat{A}}$ for $\mathbf{c}_Y = \mathsf{Com}(ck, Y, \psi)$ setting

$$\pi_{\widehat{A}} \leftarrow \mathsf{RdProof}\big(ck, \mathrm{E}_{\widehat{A}}, (\mathbf{c}_A, 0), (\mathbf{c}_M, 0), (\mathbf{c}_S, 0), (\mathsf{Com}(ck, Y, 0), \psi), (\mathbf{c}_D, 0), \pi_A\big) \ .$$

To adapt a proof to a decommitment of $\mathbf{c}_{vk}$, we have to reset the randomness of $\mathbf{c}_Y$ to 0. $\mathsf{AdD}_{\mathcal{K}}$ does thus the converse: it sets

$$\pi_A \leftarrow \mathsf{RdProof}(ck, \mathrm{E}_{\widehat{A}}, (\mathbf{c}_A, 0), (\mathbf{c}_M, 0), (\mathbf{c}_S, 0), (\mathsf{Com}(ck, Y, 0), -\psi), (\mathbf{c}_D, 0), \pi_{\widehat{A}}) \ .$$

We conclude this section by summarizing the results by the following theorem.

Theorem 1. *Under the ADH-SDH and the SXDH assumption,* $\big(\mathbf{Com}, \mathbf{Proof}, \mathbf{Sig}, \mathbf{Com}_{\mathcal{M}}, \mathsf{AdC}_{\mathcal{S}}, \mathsf{AdD}_{\mathcal{S}}, \mathsf{AdC}_{\mathcal{M}}, \mathsf{AdD}_{\mathcal{M}}, \mathsf{AdC}_{\mathcal{K}}, \mathsf{AdD}_{\mathcal{K}}, \mathsf{SigCom}, \mathsf{SmSigCom}\big)$ *is a system of commuting signatures and verifiable encryption as defined in Definition 1.*

7 Non-interactively Delegatable Anonymous Credentials

7.1 The BCCKLS Model

Functionality. We give an overview of the model for delegatable credentials defined in [BCC+09]. The system parameters are set up by a trusted party. Every user generates a secret key *sk*, of which they can publish *pseudonyms Nym*. Any user can become *originator* of a credential by publishing a pseudonym Nym_O as the *public key*. To issue or delegate a credential, the issuer and the user (both known to each other under their respective pseudonyms) run a protocol at the end of which the user holds a credential. The holder can produce a *credential proof* for any of his pseudonyms, which proves that the owner of that pseudonym holds a credential rooted at a public key Nym_O.

In our *non-interactive* instantiation we have the following: To delegate (or to issue) a credential to a user known to the delegator under Nym_U, the delegator produces (without interacting with the user) a ready credential proof for Nym_U. The user can turn this credential proof into a credential, which (as in the BCCKLS model) she can then use to make a credential proof for another pseudonym.

A (non-interactively) delegatable anonymous credential system consists of the following algorithms. On input 1^λ, $\mathsf{Setup}_{\mathrm{C}}$ generates the parameters *pp*, which are input to all other algorithms. $\mathsf{KeyGen}_{\mathrm{C}}$ generates user secret keys *sk*, of which NymGen outputs pseudonyms *Nym* and auxiliary information *aux* related to *Nym*.

Issuing and delegation is done via Issue, which on input the issuer's secret key sk_I, pseudonym Nym_I and corresponding aux_I, a level-L credential *cred* for the issuer rooted at Nym_O (if $L = 0$ then $cred = \varepsilon$) and a user pseudonym Nym_U, outputs a credential proof *credproof* for Nym_U. From this the user can obtain a credential *cred* by running Obtain on his secret key sk_U, pseudonym Nym_U and aux_U, the issuer's and delegator's pseudonyms Nym_O and Nym_I, respectively, and *credproof*. Running CredProve on $(pp, Nym_O, cred, sk, Nym, aux, L)$ permits a user to make a credential proof for the pseudonym *Nym* related to *sk* and *aux*. CredVerify verifies a proof

credproof rooted at Nym_O for *Nym*. Note that CredProve outputs a *credproof* for the user that runs it, while Issue outputs a *credproof* for the user one issues or delegates to.

Security. Security is defined by *correctness*, *anonymity* and *unforgeability*. Run honestly, Issue and Obtain must produce credentials on which, for all user pseudonyms, CredProve outputs a proof that is accepted by CredVerify.

Anonymity means that an adversary interacting with honest users cannot distinguish the real game from an ideal game: There are *simulated parameters* which are indistinguishable from the real ones but lead to pseudonyms, credentials and proofs that are *independent* of users' secret keys. Given a trapdoor for these parameters, Issue, Obtain and CredProve can be simulated without the secret inputs *cred*, *sk* and *aux*.

To break unforgeability, an adversary must produce a proof that some *Nym* has a credential although such a credential has never been issued to any pseudonym of the owner of *Nym*. To formalize the notion of "owner", Belenkiy et al. define an algorithm that extracts from a pseudonym a user identity *vk*, which is uniquely defined by the secret key. Moreover, from a credential proof it extracts the identities that represent the underlying delegation chain. We say that a forgery occurs if the adversary produces a credential for authority vk_0 from which are extracted $(vk_1, \ldots, vk_{L-1}, vk_L)$ such that vk_{L-1} is an honest user that never delegated a level-L credential rooted at vk_0 to vk_L.

7.2 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 *sk* for the authentication scheme, and a pseudonym is a GS commitment to $f(sk)$ for a one-way function f. To issue and delegate, the issuer and the user run an interactive two-party protocol to compute a proof of knowledge of an *authenticator* on the user's secret key, which is valid under the issuer's secret key. A credential is then a chain of pseudonyms and committed authenticators with GS proofs of validity.

We replace the authenticators (consisting of 11 group elements and verified by 8 pairing-product equations) by automorphic signatures (5 group elements satisfying 3 PPEs). A non-anonymous level-L credential for vk_L rooted at vk_0 is a chain of verification keys and signatures $(\Sigma_1, vk_1, \Sigma_2 \ldots, vk_{L-1}, \Sigma_L)$, where Σ_i is a signature on vk_i under vk_{i-1}. To achieve anonymity, the keys and signatures in the credential are committed to and proofs of validity are added. Using commuting signatures, given a commitment to a key, the issuer can directly make a commitment to a signature on it and a validity proof. This is what enables non-interactive delegation.

However, merely signing user keys does not suffice, as 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 the full version [Fuc10] we therefore give a simple extension of $\mathbf{Sig}$: Scheme $\mathbf{Sig}''$ has message space $\mathbb{Z}_p \times \mathcal{M}$, allowing the signer to specify a public value in addition to M. Its parameters contain one additional group element, but the signatures have the same size as those of $\mathbf{Sig}$. We also define SigCom″, an adaptation of SigCom which

has the public value in $\mathbb{Z}_p$ as additional input, and show that all the other algorithms defined in Definition 1 and instantiated in Section 6 work equally for $\mathbf{Sig}$ and $\mathbf{Sig}''$[7].

Our Scheme. Let $\mathcal{H}\colon \mathcal{C}_{\mathcal{M}} \times \mathbb{N} \to \mathbb{Z}_p$ be a collision-resistant hash function. Then our scheme $\mathbf{Cred}$ can be sketched as follows: $\mathsf{Setup}_{\mathrm{C}}$ generates a key for $\mathbf{Com}$ and parameters for $\mathbf{Sig}''$; $\mathsf{KeyGen}_{\mathrm{C}}$ outputs a signing key for $\mathbf{Sig}''$; and given such a key, NymGen outputs a commitment to the corresponding verification key and the used randomness as auxiliary information. A level-L credential proof from Nym_0 for Nym_L has the form

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

where $\mathbf{c}_i$ is a commitment to a $\mathbf{Sig}''$ signature Σ_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 π_i is a proof of validity of Σ_i. We call *credproof* a credential if it is valid on a "trivial" Nym_L, i.e., when $Nym_L = \mathsf{Com}(ck, vk_L, 0)$.

$\mathsf{CredProve}$ takes a credential and turns it into a credential proof for Nym_L by randomizing all its components, using as randomness for the last component the value *aux* s.t. $Nym_L = \mathsf{Com}(ck, vk_L, aux)$. $\mathsf{CredVerify}$ verifies a *credproof* by checking the proofs contained in it. Given a level-L credential, Issue extends it by one level making a credential proof for the delegatee's pseudonym Nym_{L+1}: In case of a delegation ($L > 0$), it first makes a *credproof* for the issuer's pseudonym Nym_I; otherwise *credproof* $:= \varepsilon$. Running SigCom'' on sk_I, $\mathcal{H}(Nym_0, L+1)$ and Nym_{L+1}, it produces $(\mathbf{c}_{L+1}, \pi'_{L+1})$, a verifiably encrypted signature under vk_I. It then runs $\mathsf{AdC}_{\mathcal{K}}$ on (vk_I, aux_I) to adapt π'_{L+1} to a proof π_{L+1}, which is valid for the committed verification key Nym_I. Finally, Issue outputs $credproof \,\|\, Nym_I \,\|\, (\mathbf{c}_{L+1}, \pi_{L+1})$. Obtain turns a level-L credential proof into a credential by adapting the randomness to make it valid for a trivial Nym_L.

Simulatability. While unforgeability of our scheme is implied by the soundness of $\mathbf{Proof}$ and unforgeability of $\mathbf{Sig}''$, anonymity is shown as follows. We generate the simulated parameters by replacing Setup with $\mathsf{WISetup}$; this makes the commitments perfectly hiding and thus pseudonyms and credential proofs independent of secret keys.

We then have to simulate issuing and $\mathsf{CredProve}$ for pseudonyms of the adversary's choice. This essentially means to simulate credential proofs for given *Nym*'s; thus, simulating commitments to a signature and a proof of validity w.r.t. commitments to a key and to a message which are both given to the simulator.

While Groth and Sahai [GS08] show simulation of *proofs of satisfiability* of equations, where the simulator produces all the commitments, we require a novel type of simulation: given commitments to certain variables of an equation, we have to simulate the remaining commitments and the proof of validity. In the full version [Fuc10] we show that this can be done for a class $\mathcal{E}' \subset \mathcal{E}$ of equations, in which the equations for validity of committed signatures fall. We call a commuting signature scheme *simulatable* if given a commitment to a key and a commitment to a message the simulator can create a verifiably encrypted signature for them.

In [Fuc10] we define a simulatable variant of our scheme from Section 6. We also give a formal description of our credential scheme $\mathbf{Cred}$ and a proof of the following.

[7] Note that replacing SigCom by SigCom'' in the construction of a blind signature at the end of Section 3 yields a *partially blind signature* [AF96].

Theorem 2. *Let* $(\mathbf{Com}, \mathbf{Proof}, \mathbf{Sig}'', \mathbf{Com}_{\mathcal{M}}, \mathbf{Algs})$ *be a system of commuting signatures which is automorphic and simulatable, and let* $\mathcal{H}$ *be a collision-resistant hash function. Then* $\mathbf{Cred}$ *is a secure delegatable anonymous credential scheme.*

8 Conclusion

We introduced and instantiated *commuting signatures and verifiable encryption*. Given an encryption $\mathbf{C}$ of M and a signing key, they let us produce an encryption $\mathbf{c}_\Sigma$ of a signature on M and a proof that $\mathbf{c}_\Sigma$ contains a valid signature on the content of $\mathbf{C}$. We used them to give the first instantiation of delegatable anonymous credentials with non-interactive issuing and delegation and believe that they are a useful tool in the construction of privacy-preserving primitives that will find further applications.

Acknowledgements

The author would like to thank David Pointcheval, Elizabeth Quaglia and Damien Vergnaud for many helpful discussions and the anonymous referees of CCS 2010 and EUROCRYPT for their valuable comments.

References

[AF96] Abe, M., Fujisaki, E.: How to date blind signatures. In: Kim, K., Matsumoto, T. (eds.) ASIACRYPT 1996. LNCS, vol. 1163, pp. 244–251. Springer, Heidelberg (1996)

[AFG+10] Abe, M., Fuchsbauer, G., Groth, J., Haralambiev, K., Ohkubo, M.: Structure-Preserving Signatures and Commitments to Group Elements. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 209–236. Springer, Heidelberg (2010)

[BB04] Boneh, D., Boyen, X.: Short signatures without random oracles. In: Cachin, C., Camenisch, J. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 56–73. Springer, Heidelberg (2004)

[BBS04] Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 41–55. Springer, Heidelberg (2004)

[BCC+09] Belenkiy, M., Camenisch, J., Chase, M., Kohlweiss, M., Lysyanskaya, A., Shacham, H.: Randomizable proofs and delegatable anonymous credentials. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 108–125. Springer, Heidelberg (2009)

[BCKL08] Belenkiy, M., Chase, M., Kohlweiss, M., Lysyanskaya, A.: P-signatures and noninteractive anonymous credentials. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 356–374. Springer, Heidelberg (2008)

[BFM88] Blum, M., Feldman, P., Micali, S.: Non-interactive zero-knowledge and its applications. In: STOC, pp. 103–112. ACM Press, New York (1988)

[BFPV11] Blazy, O., Fuchsbauer, G., Pointcheval, D., Vergnaud, D.: Signatures on randomizable ciphertexts. In: Gennaro, R. (ed.) PKC 2011. LNCS, vol. 6571, pp. 403–422. Springer, Heidelberg (2011)

[BGLS03] Boneh, D., Gentry, C., Lynn, B., Shacham, H.: Aggregate and verifiably encrypted signatures from bilinear maps. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 416–432. Springer, Heidelberg (2003)

[BHY09] Bellare, M., Hofheinz, D., Yilek, S.: Possibility and impossibility results for encryption and commitment secure under selective opening. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 1–35. Springer, Heidelberg (2009)

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

[BW07] Boyen, X., Waters, B.: Full-domain subgroup hiding and constant-size group signatures. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 1–15. Springer, Heidelberg (2007)

[Cha83] Chaum, D.: Blind signatures for untraceable payments. In: Chaum, D., Rivest, R.L., Sherman, A.T. (eds.) CRYPTO 1982, pp. 199–203. Plenum Press, New York (1983)

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

[CL01] Camenisch, J., Lysyanskaya, A.: An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 93–118. Springer, Heidelberg (2001)

[CL02] Camenisch, J., Lysyanskaya, A.: A signature scheme with efficient protocols. In: Cimato, S., Galdi, C., Persiano, G. (eds.) SCN 2002. LNCS, vol. 2576, pp. 268–289. Springer, Heidelberg (2003)

[CL04] Camenisch, J., Lysyanskaya, A.: Signature schemes and anonymous credentials from bilinear maps. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 56–72. Springer, Heidelberg (2004)

[CL06] Chase, M., Lysyanskaya, A.: On signatures of knowledge. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 78–96. Springer, Heidelberg (2006)

[Dam90] Damgård, I.: Payment systems and credential mechanisms with provable security against abuse by individuals. In: Goldwasser, S. (ed.) CRYPTO 1988. LNCS, vol. 403, pp. 328–335. Springer, Heidelberg (1990)

[DHLW10] Dodis, Y., Haralambiev, K., López-Alt, A., Wichs, D.: Cryptography against continuous memory attacks. In: FOCS, pp. 511–520. IEEE Computer Society, Los Alamitos (2010)

[Fis06] Fischlin, M.: Round-optimal composable blind signatures in the common reference string model. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 60–77. Springer, Heidelberg (2006)

[FP08] Fuchsbauer, G., Pointcheval, D.: Anonymous proxy signatures. In: Ostrovsky, R., De Prisco, R., Visconti, I. (eds.) SCN 2008. LNCS, vol. 5229, pp. 201–217. Springer, Heidelberg (2008)

[FPV09] Fuchsbauer, G., Pointcheval, D., Vergnaud, D.: Transferable constant-size fair E-cash. In: Garay, J.A., Miyaji, A., Otsuka, A. (eds.) CANS 2009. LNCS, vol. 5888, pp. 226–247. Springer, Heidelberg (2009)

[Fuc09] Fuchsbauer, G.: Automorphic signatures in bilinear groups and an application to round-optimal blind signatures. Cryptology ePrint Archive, Report 2009/320 (2009), `http://eprint.iacr.org/2009/320`, an extended abstract appeared as part of [AFG+10]

[Fuc10] Fuchsbauer, G.: Commuting signatures and verifiable encryption and an application to non-interactively delegatable credentials. Cryptology ePrint Archive, Report 2010/233 (2010), `http://eprint.iacr.org/2010/233`

[GOS06] Groth, J., Ostrovsky, R., Sahai, A.: Perfect non-interactive zero knowledge for NP. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 339–358. Springer, Heidelberg (2006)

[GPS08] Galbraith, S.D., Paterson, K.G., Smart, N.P.: Pairings for cryptographers. Discrete Applied Mathematics 156(16), 3113–3121 (2008)

[GS08] Groth, J., Sahai, A.: Efficient non-interactive proof systems for bilinear groups. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 415–432. Springer, Heidelberg (2008)

[LRSW00] Lysyanskaya, A., Rivest, R.L., Sahai, A., Wolf, S.: Pseudonym systems. In: Heys, H.M., Adams, C.M. (eds.) SAC 1999. LNCS, vol. 1758, pp. 184–199. Springer, Heidelberg (2000)

[PS96] Pointcheval, D., Stern, J.: Provably secure blind signature schemes. In: Kim, K., Matsumoto, T. (eds.) ASIACRYPT 1996. LNCS, vol. 1163, pp. 252–265. Springer, Heidelberg (1996)

[RS09] Rückert, M., Schröder, D.: Security of verifiably encrypted signatures and a construction without random oracles. In: Shacham, H., Waters, B. (eds.) Pairing 2009. LNCS, vol. 5671, pp. 17–34. Springer, Heidelberg (2009)