Fine-Tuning Groth-Sahai Proofs

Alex Escala[1] and Jens Groth[2,⋆]

[1] Scytl Secure Electronic Voting, Spain
[2] University College London, United Kingdom

Abstract. Groth-Sahai proofs are efficient non-interactive zero-knowledge proofs that have found widespread use in pairing-based cryptography. We propose efficiency improvements of Groth-Sahai proofs in the SXDH setting, which is the one that yields the most efficient non-interactive zero-knowledge proofs.

- We replace some of the commitments with ElGamal encryptions, which reduces the prover's computation and for some types of equations reduces the proof size.
- Groth-Sahai proofs are zero-knowledge when no public elements are paired to each other. We observe that they are also zero-knowledge when base elements for the groups are paired to public constants.
- The prover's computation can be reduced by letting her pick her own common reference string. By giving a proof she has picked a valid common reference string this does not compromise soundness.
- We define a type-based commit-and-prove scheme, which allows commitments to be reused in many different proofs.

Keywords: Non-interactive zero-knowledge proofs, commit-and-prove schemes, Groth-Sahai proofs, type-based commitments.

1 Introduction

Non-interactive zero-knowledge (NIZK) proofs [BFM88] can be used to demonstrate a statement is true without revealing any other information. NIZK proofs are fundamental building blocks in cryptography and are used in numerous cryptographic schemes. It is therefore important to increase their efficiency since even small improvements will lead to significant performance gains when aggregated over many applications.

NIZK proofs were invented more than two decades ago but early constructions [BFM88, FLS99, Dam92, KP98] were very inefficient. This changed when Groth, Ostrovsky and Sahai [GOS12] introduced pairing-based techniques for constructing NIZK proofs. In a series of works [BW06, Gro06, BW07, GS12] pairing-friendly NIZK proofs were developed. This line of research culminated

[⋆] The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement n. 307937 and the Engineering and Physical Sciences Research Council grants EP/G013829/1 and EP/J009520/1.

H. Krawczyk (Ed.): PKC 2014, LNCS 8383, pp. 630–649, 2014.

in Groth and Sahai [GS12] that gave efficient and practical NIZK proofs that are now widely used in pairing-based cryptography.

Groth-Sahai proofs [GS12] can be instantiated in many ways with either symmetric or asymmetric pairings and over groups that may have either composite order or prime order. The asymmetric setting with prime order groups yields the smallest group elements [GPS08]. We will therefore focus on improving Groth-Sahai proofs for prime order asymmetric bilinear groups, since the better efficiency makes it the most important setting for use in practice.

Let us give some more details of what can be done with Groth-Sahai proofs. The setting they consider is a bilinear group $(p, \hat{\mathbb{G}}, \check{\mathbb{H}}, \mathbb{T}, e, \hat{g}, \check{h})$, where $\hat{\mathbb{G}}, \check{\mathbb{H}}, \mathbb{T}$ are prime order p groups, $\hat{g}$ and $\check{h}$ are generators of $\hat{\mathbb{G}}$ and $\check{\mathbb{H}}$ respectively and $e : \hat{\mathbb{G}} \times \check{\mathbb{H}} \to \mathbb{T}$ is a non-degenerate bilinear map. The prover wants to show that there are values $\hat{x}_i \in \hat{\mathbb{G}}, \check{y}_j \in \check{\mathbb{H}}, x_i, y_j \in \mathbb{Z}_p$ simultaneously satisfying a set of equations. Groth and Sahai formulate four types of equations, which using additive notation for group operations and multiplicative notation for the bilinear map e can be written as follows.

Pairing-product equation: Public constants $\hat{a}_j \in \hat{\mathbb{G}}, \check{b}_i \in \check{\mathbb{H}}, \gamma_{ij} \in \mathbb{Z}_p, t_{\mathbb{T}} \in \mathbb{T}$.

$$\sum_i \hat{x}_i \cdot \check{b}_i + \sum_j \hat{a}_j \cdot \check{y}_j + \sum_i \sum_j \gamma_{ij} \hat{x}_i \cdot \check{y}_j = t_{\mathbb{T}}.$$

Multi-scalar multiplication equation in $\hat{\mathbb{G}}$: Public constants $\hat{a}_j \in \hat{\mathbb{G}}, b_i \in \mathbb{Z}_p, \gamma_{ij} \in \mathbb{Z}_p, \hat{t} \in \hat{\mathbb{G}}$.

$$\sum_i \hat{x}_i b_i + \sum_j \hat{a}_j y_j + \sum_i \sum_j \gamma_{ij} \hat{x}_i y_j = \hat{t}.$$

Multi-scalar multiplication equation in $\check{\mathbb{H}}$: Public constants $a_j \in \mathbb{Z}_p, \check{b}_i \in \check{\mathbb{H}}, \gamma_{ij} \in \mathbb{Z}_p, \check{t} \in \check{\mathbb{H}}$.

$$\sum_i x_i \check{b}_i + \sum_j a_j \check{y}_j + \sum_i \sum_j \gamma_{ij} x_i \check{y}_j = \check{t}.$$

Quadratic equation in $\mathbb{Z}_p$: Public constants $a_j \in \mathbb{Z}_p, b_i \in \mathbb{Z}_p, \gamma_{ij} \in \mathbb{Z}_p, t \in \mathbb{Z}_p$.

$$\sum_i x_i b_i + \sum_j a_j y_j + \sum_i \sum_j \gamma_{ij} x_i y_j = t.$$

These four types of equations express in a direct way statements arising in pairing-based cryptography. For this reason Groth-Sahai proofs are used in numerous pairing-based protocols including group signatures [Gro07], anonymous credentials [BCKL08, BCC$^+$09], e-cash [FPV09], etc.

Groth-Sahai proofs are witness-indistinguishable proofs that enable a prover to convince a verifier that a statement is true without revealing which witness the prover knows. For a slightly more restricted set of statements where all pairing-product equations have $t_{\mathbb{T}} = 0_{\mathbb{T}}$, Groth-Sahai proofs are actually zero-knowledge proofs that leak no information besides the truth of the statement.

There have been several papers that extend or improve the Groth-Sahai proof system in different directions. [Mei09] suggested how to create perfectly extractable commitments, something which is not given by the commitments used by Groth and Sahai. [CHP07, BFI$^+$10] reduced the computational cost of the verification of the proofs using batch techniques, at the cost of trading perfect soundness for statistical soundness. [Seo12] gave another map for verifying proofs in the symmetric setting which reduces the computational cost of the verification of the proofs. On the other hand, they prove that the map proposed by Groth and Sahai in the asymmetric setting is optimal. [GSW10] proposed another assumption on which Groth-Sahai proofs can be based. [BCKL08, BCC$^+$09] exploited rerandomization properties of Groth-Sahai proofs, which they used in anonymous credentials. [Fuc11] proposed a witness-indistinguishable commit-and-prove scheme based on Groth-Sahai proofs in the symmetric setting. [CKLM12] introduced a new notion of malleable proof systems, which can be built from Groth-Sahai proofs. While there has been significant research effort devoted to pairing-based NIZK proofs, Groth-Sahai proofs still remain the most efficient NIZK proofs that are based on standard intractability assumptions and there has not been any progress in reducing their size or the prover's computation except for special purpose statements [EHKRV13, JR13].

1.1 Our Contributions

We focus on improving efficiency and propose several ways to fine-tune Groth-Sahai zero-knowledge proofs in the asymmetric bilinear group setting.

- Groth-Sahai proofs use public constants and committed variables. We introduce two new types of values: public base elements and encrypted variables. This reduces the size of proofs for statements involving these values.
- We recast Groth-Sahai proofs as a commit-and-prove scheme. This makes it possible to reuse commitments in the proofs of different statements even when these statements depend on previous commitments and proofs.
- We show that the prover's computation can be reduced by letting her pick her own provably correct common reference string.

Encrypted Variables. The common reference string in Groth-Sahai proofs contains a public commitment key that the prover uses to commit to variables. The prover then proceeds to prove that the committed variables satisfy the equations in the statement. In our scheme we allow the prover to encrypt variables using ElGamal encryption as an alternative to the commitment scheme. ElGamal encryption reduces the prover's computation when compared to the commitment operation. Moreover, equations that use ElGamal ciphertexts instead of commitments have simpler proofs. However, using ElGamal encryption means we cannot get perfect zero-knowledge, so we rely on the Decision Diffie-Hellman (DDH) assumption to get computational zero-knowledge and we place some restrictions on the types of equations where ElGamal encryptions can be used.

Base Elements. We observe that the commitment keys can be set up to allow simulation in pairing-product equations where $t_{\mathbb{T}} = \hat{a} \cdot \check{h} + \hat{g} \cdot \check{b}$ for public constants $\hat{a} \in \hat{\mathbb{G}}$ and $\check{b} \in \check{\mathbb{H}}$. This extension of Groth-Sahai proofs comes at no extra cost, so we save the costly rewriting of the equations proposed in [GS12] which was required to get zero-knowledge in those kinds of equations.

In addition, a similar observation allows us to have shorter Groth-Sahai zero-knowledge proofs for multi-scalar multiplications equations in $\hat{\mathbb{G}}$ or in $\check{\mathbb{H}}$ in which all the field elements are the constants $\hat{t} = \hat{g}$ or $\check{t} = \check{h}$.

Using Commitment Keys with Known Discrete Logarithms. In Groth-Sahai proofs, a common reference string created by a trusted entity is shared between the prover and the verifier. We show how to reduce the prover's computation by allowing her to choose her *own common reference string*, which we think of as her public key. This change reduces the cost of computing her commitments from 4 scalar multiplications to 2 scalar multiplications and it also reduces the cost of computing proofs.

To enforce soundness, the prover will give a Groth-Sahai proof to the prover, using a common reference string the verifier does trust, for the public key being correct. The cost of such proof is 12 group elements in total, which is a one-off cost as the public key can be used for many commitments and proofs.

Viewing the common reference string as the prover's public key gives us some flexibility in the setup. Instead of proving the public key correct in the common reference string model, the prover could use the multi-string model [GO07] where we only assume a majority out of n common reference strings are honest. Alternatively, the prover could give a zero-knowledge proof of knowledge to a trusted third party that the public key is correct and get a certificate on the public key.

Type-Based Commit-and-Prove Schemes. A natural generalization of zero-knowledge proofs are commit-and-prove schemes [Kil90, CLOS02], where the prover can commit to values and prove statements about the committed values. Commit-and-prove schemes provide extra flexibility and reduce communication; it is for instance possible to choose values to be committed to in an adaptive fashion that depends on previous commitments or proofs. The traditional definition of zero-knowledge proofs would require the prover to make an entirely new set of commitments for each statement to be proven.

Groth-Sahai proofs can be used to build a non-interactive commit-and-prove scheme in a natural way; Belenkiy et al. [BCKL08] for instance explicitly let the commitments be part of the statements and define witness-indistinguishable proofs for such statements. Fuchsbauer [Fuc11] defines a witness-indistinguishable Groth-Sahai based commit-and-prove scheme and uses it in the construction of delegatable anonymous credentials. Our definition of a non-interactive commit-and-prove scheme will resemble Fuchsbauer's [Fuc11]. However, we are in a different situation because we have more types of elements that we want to commit to. A group element in $\hat{\mathbb{G}}$ may for instance be committed using the perfectly binding/perfectly hiding commitment scheme or using ElGamal encryption.

To give a generally applicable definition of non-interactive commit-and-prove schemes, we propose the notion of *type-based* commitments. A type-based commitment scheme enables the prover to commit to a message m with a publicly known type t and we require that the type and message pair (t, m) belong to a message space $\mathcal{M}_{ck}$. One example of a type could for instance be $t = \mathrm{enc}_{\hat{\mathbb{G}}}$ meaning the value m should be encrypted (as opposed to using the more expensive commitment operation) and it should be done in group $\hat{\mathbb{G}}$. This increases the flexibility of the commitment scheme, we can for instance create a type $(\mathrm{pub}_{\hat{\mathbb{G}}}, x)$ that publicly declares the committed value x. Since the type is public this commitment is no longer hiding, however, as we shall see it simplifies our commit-and-prove scheme because we can now commit to both public constants and secret variables without having to treat them differently.

Applications. To illustrate the advantages of our fine-tuned Groth-Sahai proofs we give an example based on the weak Boneh-Boyen signature scheme [BB04], which is widely used in pairing-based protocols. The verification key is an element $\hat{v} \in \hat{\mathbb{G}}$ and a signature on a message $m \in \mathbb{Z}_p$ is a group element $\check{\sigma} \in \check{\mathbb{H}}$ such that

$$(\hat{v} + m\hat{g}) \cdot \check{\sigma} = \hat{g} \cdot \check{h}.$$

Suppose the prover has commitments to $\hat{v}$ and $\check{\sigma}$ and wants to demonstrate that they satisfy the verification equation for a (public) message m. With traditional Groth-Sahai proofs the commitments c and d to $\hat{v}$ and $\check{\sigma}$ would be treated as part of the statement and one would carefully demonstrate the existence of openings of c and d to $\hat{v}$ and $\check{\sigma}$ satisfying the pairing-product equation. With a commit-and-prove system, we can instead jump directly to demonstrating that the values inside $\hat{v}$ and $\check{\sigma}$ satisfy the verification equation without having to treat the openings of the commitments as part of the witness. This saves several group elements each time one of the commitments is used.

Next, observe that the pairing-product equation has $t_{\mathbb{T}} = \hat{g} \cdot \check{h}$. A direct application of Groth-Sahai proofs would therefore not yield a zero-knowledge proof but only give witness-indistinguishability. To get zero-knowledge we could use the workaround suggested by Groth-Sahai, which would consist of committing to a new variable $\check{y}$, prove that $\check{y} = \check{h}$ and simultaneously $(\hat{v} + m\hat{g}) \cdot \check{\sigma} - \hat{g} \cdot \check{y} = 0_{\mathbb{T}}$. This workaround would increase the cost of the proof from 8 group elements to 16 group elements, so we save 8 group elements by enabling a direct proof.

Now assume the prover has created her own common reference string pk and has already sent it together with the well-formedness proof to the verifier. The prover could now use pk to compute the zero-knowledge proof for the equation $(\hat{v} + m\hat{g}) \cdot \check{\sigma} = \hat{g}\check{h}$. By using pk, she would need to do 10 scalar multiplications in $\hat{\mathbb{G}}$ and 6 scalar multiplications in $\check{\mathbb{H}}$ to compute the proof. In contrast, if she was computing the proof using the commitment key ck, she would need to do 12 scalar multiplications in $\hat{\mathbb{G}}$ and 10 scalar multiplications in $\check{\mathbb{H}}$. As the operations in $\check{\mathbb{H}}$ are usually significantly more expensive than the operations in $\hat{\mathbb{G}}$, the prover is essentially saving 4 expensive operations of the 10 that she would need to do if she used ck. Therefore, our techniques reduce the computational cost of creating the zero-knowledge proof by roughly 40%. In addition, the computational cost of computing the commitments to $\hat{v}$ and $\breve{\sigma}$ would also be reduced by 50%.

Finally, we can obtain a saving by encrypting one of the variables instead of committing to it. If we encrypt $\hat{v}$ for instance, the ciphertext is 2 group elements just as a commitment would be, but the cost of the proof for the pairing-product equation is reduced from 8 group elements to 6 group elements. In total we have reduced the cost by 63% from 16 group elements to 6 group elements.

In the full paper [EG13] we give two concrete examples of existing schemes using Groth-Sahai proofs where our techniques can improve efficiency.

2 Commit-and-Prove Scheme Definitions

Let R_L be a polynomial time verifiable relation containing triples (ck, x, w). We will call ck the commitment key or the common reference string, x the statement and w the witness. We define the key-dependent language L_{ck} as the set of statements x for which there exists a witness w such that $(ck, x, w) \in R_L$.

We will now define a commit-and-prove scheme for a relation R_L. In the commit-and-prove scheme, we may commit to different values $w_1, \ldots, w_N$ and prove for different statements x that a subset of the committed values $w = (w_{i_1}, \ldots, w_{i_n})$ constitute a witness for $x \in L_{ck}$, i.e., $(ck, x, w) \in R_L$.

We will divide each committed value into two parts $w_i = (t_i, m_i)$. The first part t_i can be thought of as a public part that does not need to be kept secret, while the second part m_i can be thought of as a secret value that our commit-and-prove scheme should not reveal. The first part t_i will be useful later on to specify the type of value m_i is, for instance a group element or a field element, and to specify which type of commitment we should make to m_i. This is a natural and useful generalization of standard commitment schemes.

A commit-and-prove scheme CP = (Gen, Com, Prove, Verify) consists of four polynomial time algorithms. The algorithms Gen, Prove are probabilistic and the algorithms Com, Verify are deterministic.

$\mathrm{Gen}(1^k)$: Generates a commitment key ck. The commitment key specifies a message space $\mathcal{M}_{ck}$, a randomness space $\mathcal{R}_{ck}$ and a commitment space $\mathcal{C}_{ck}$. Membership of either space can be decided efficiently.

$\mathrm{Com}_{ck}(t, m; r)$: Given a commitment key ck, a message $(t, m) \in \mathcal{M}_{ck}$ and randomness r such that $(t, r) \in \mathcal{R}_{ck}$ returns a commitment c such that $(t, c) \in \mathcal{C}_{ck}$.

$\mathrm{Prove}_{ck}(x, (t_1, m_1, r_1), \ldots, (t_n, m_n, r_n))$: Given a commitment key ck, statement x and commitment openings such that $(t_i, m_i) \in \mathcal{M}_{ck}, (t_i, r_i) \in \mathcal{R}_{ck}$ and $(ck, x, t_1, m_1, \ldots, t_n, m_n) \in R_L$ returns a proof π.

$\mathrm{Verify}_{ck}(x, (t_1, c_1), \ldots, (t_n, c_n), \pi)$: Given a commitment key ck, a statement x, a proof π and commitments $(t_i, c_i) \in \mathcal{C}_{ck}$ returns 1 (accept) or 0 (reject).

Definition 1 (Perfect correctness). *The commit-and-prove system* CP *is (perfectly) correct if for all adversaries* $\mathcal{A}$

$$\Pr\left[\begin{array}{l} ck \leftarrow \mathrm{Gen}(1^k)\ ;\ (x, w_1, r_1, \ldots, w_n, r_n) \leftarrow \mathcal{A}(ck)\ ;\ c_i \leftarrow \mathrm{Com}_{ck}(w_i; r_i)\ ; \\ \pi \leftarrow \mathrm{Prove}_{ck}(x, w_1, r_1, \ldots, w_n, r_n) : \mathrm{Verify}_{ck}(x, (t_1, c_1), \ldots, (t_n, c_n), \pi) = 1 \end{array}\right] = 1,$$

where $\mathcal{A}$ outputs w_i, r_i such that $w_i = (t_i, m_i) \in \mathcal{M}_{ck}, (t_i, r_i) \in \mathcal{R}_{ck}$ and $(ck, x, w_1, \ldots, w_n) \in R_L$.

We say a commit-and-prove scheme is sound if it is impossible to prove a false statement. Strengthening the usual notion of soundness, we will associate unique values to the commitments, and these values will constitute a witness for the statement. This means that not only does a valid proof guarantee the truth of the statement, but also each commitment will always contribute a consistent witness towards establishing the truth of the statement.

Definition 2 (Perfect soundness). *The commit-and-prove system* CP *is (perfectly) sound if there exists a deterministic (unbounded) opening algorithm* Open *such that for all adversaries $\mathcal{A}$*

$$\Pr\left[\begin{array}{l} ck \leftarrow \mathrm{Gen}(1^k)\ ;\ (x, t_1, c_1, \ldots, t_n, c_n, \pi) \leftarrow \mathcal{A}(ck)\ ;\ m_i \leftarrow \mathrm{Open}_{ck}(t_i, c_i) : \\ \mathrm{Verify}_{ck}(x, t_1, c_1, \ldots, t_n, c_n, \pi) = 0\ \vee\ (ck, x, (t_1, m_1), \ldots, (t_n, m_n)) \in R_L \end{array}\right] = 1.$$

Extending the notion of soundness we may define a proof of knowledge as one where it is possible to efficiently extract a witness for the truth of the statement proven when given an extraction key xk. Actually, the commit-and-prove schemes we construct will not allow the extraction of all types of witnesses due to the hardness of the discrete logarithm problem. However, following Belenkiy et al. [BCKL08] we can specify a function F such that we can extract $F(ck, w)$ from a commitment. Efficient extraction of a witness corresponds to the special case where $F(ck, w) = m$, with m being the secret part of the witness $w = (t, m)$.

Definition 3 (Perfect F-extractability). *Let in the following* ExtGen *and* Ext *be two algorithms as described below.*

- ExtGen *is a probabilistic polynomial time algorithm that on 1^k returns (ck, xk). We call ck the commitment key and xk the extraction key. We require that the probability distributions of ck made by* ExtGen *and* Gen *are identical.*
- Ext *is a deterministic polynomial time algorithm that given an extraction key xk and $(t, c) \in \mathcal{C}_{ck}$ returns a value.*

The commit-and-prove scheme CP *with perfect soundness for opening algorithm* Open *is F-extractable if for all adversaries $\mathcal{A}$*

$$\Pr\left[\begin{array}{l} (ck, xk) \leftarrow \mathrm{ExtGen}(1^k)\ ;\ (t, c) \leftarrow \mathcal{A}(ck, xk) : \\ (t, c) \notin \mathcal{C}_{ck} \vee \mathrm{Ext}_{xk}(t, c) = F(ck, (t, \mathrm{Open}(t, c))) \end{array}\right] = 1.$$

A commit-and-prove scheme is zero-knowledge if it does not leak information about the secret parts of the committed messages besides what is known from the public parts. This is defined as the ability to simulate commitments and proofs without knowing the secret parts of the messages (the types are known) if instead some secret simulation trapdoor is known.

Following [Gro06, GOS12] we define a strong notion of zero-knowledge called composable zero-knowledge. Composable zero-knowledge says the commitment key can be simulated, and if the commitment key is simulated it is not possible to distinguish real proofs from simulated proofs even if the simulation trapdoor is known.

Definition 4 (Composable zero-knowledge). *The commit-and-prove system* CP *is (computationally) composable zero-knowledge if there exist probabilistic polynomial time algorithms* SimGen, SimCom, SimProve *such that for all non-uniform polynomial time stateful interactive adversaries* $\mathcal{A}$ [1]

$$\Pr\left[ck \leftarrow \mathrm{Gen}(1^k) : \mathcal{A}(ck) = 1\right] \approx \Pr\left[(ck, tk) \leftarrow \mathrm{SimGen}(1^k) : \mathcal{A}(ck) = 1\right]$$

and

$$\Pr\left[\begin{array}{l}(ck, tk) \leftarrow \mathrm{SimGen}(1^k); (x, i_1, \ldots, i_n) \leftarrow \mathcal{A}^{\mathrm{Com}_{ck}(\cdot)}(ck, tk);\\ \pi \leftarrow \mathrm{Prove}_{ck}(x, w_{i_1}, r_{i_1}, \ldots, w_{i_n}, r_{i_n}) : \mathcal{A}(\pi) = 1\end{array}\right]$$
$$\approx \Pr\left[\begin{array}{l}(ck, tk) \leftarrow \mathrm{SimGen}(1^k); (x, i_1, \ldots, i_n) \leftarrow \mathcal{A}^{\mathrm{SimCom}_{tk}(\cdot)}(ck, tk);\\ \pi \leftarrow \mathrm{SimProve}_{tk}(x, t_{i_1}, s_{i_1}, \ldots, t_{i_n}, s_{i_n}) : \mathcal{A}(\pi) = 1\end{array}\right],$$

where

- tk *is a trapdoor key used to construct simulated proofs*
- $\mathrm{Com}_{ck}(\cdot)$ *on* $w_i = (t_i, m_i) \in \mathcal{M}_{ck}$ *picks uniformly random* r_i *such that* $(t_i, r_i) \in \mathcal{R}_{ck}$ *and returns* $c_i = \mathrm{Com}_{ck}(w_i; r_i)$
- $\mathrm{SimCom}_{tk}(\cdot)$ *on* $w_i = (t_i, m_i) \in \mathcal{M}_{ck}$ *runs* $(c_i, s_i) \leftarrow \mathrm{SimCom}_{tk}(t_i)$ *and returns* c_i*, where* s_i *is some auxiliary information used to construct simulated proofs*
- $\mathcal{A}$ *picks* $(x, i_1, \ldots, i_n)$ *such that* $(ck, x, w_{i_1}, \ldots, w_{i_n}) \in R_L$

3 Preliminaries

3.1 Bilinear Group

Let $\mathcal{G}$ be a probabilistic polynomial time algorithm that on input 1^k returns $(p, \hat{\mathbb{G}}, \check{\mathbb{H}}, \mathbb{T}, e, \hat{g}, \check{h})$, where $\hat{\mathbb{G}}, \check{\mathbb{H}}$ and $\mathbb{T}$ are groups of prime order p, $\hat{g}$ and $\check{h}$ generate $\hat{\mathbb{G}}$ and $\check{\mathbb{H}}$ respectively, and $e : \hat{\mathbb{G}} \times \check{\mathbb{H}} \rightarrow \mathbb{T}$ is an efficiently computable, non-degenerate bilinear map.

Notation. We will write elements $\hat{x} \in \hat{\mathbb{G}}$ with a hat and elements $\check{y} \in \check{\mathbb{H}}$ with an inverted hat to make it easy to distinguish elements from the two groups. We denote the neutral elements in the groups $\hat{\mathbb{G}}, \check{\mathbb{H}}$ and $\mathbb{T}$ with $\hat{0}, \check{0}$ and $0_{\mathbb{T}}$.

[1] Given two functions $f, g : \mathbb{N} \rightarrow [0, 1]$ we write $f(k) \approx g(k)$ when $|f(k) - g(k)| = O(k^{-c})$ for every positive integer c. We say that f is *negligible* when $f(k) \approx 0$ and that it is *overwhelming* when $f(k) \approx 1$.

It will be convenient to use additive notation for all three groups $\hat{\mathbb{G}}, \check{\mathbb{H}}$ and $\mathbb{T}$. This notation deviates from standard practice ($\hat{\mathbb{G}}, \check{\mathbb{H}}$ are sometimes written multiplicatively and $\mathbb{T}$ is usually written multiplicatively) but will greatly simplify our paper and make it possible to use linear algebra concepts such as vectors and matrices in a natural way. We stress that even though we are using additive notation it is hard to compute discrete logarithms in the groups.

It will also be convenient to write the pairing e with multiplicative notation. So we define

$$\hat{x} \cdot \check{y} = e(\hat{x}, \check{y}).$$

Writing the pairing multiplicatively allows us to use linear algebra notation in a natural way, we have for instance

$$\hat{x} \cdot \begin{pmatrix} \check{0} & \check{y} \\ \check{z} & \check{0} \end{pmatrix} \boldsymbol{e}^\top = \begin{pmatrix} \hat{x} \cdot \check{y} \\ 0_{\mathbb{T}} \end{pmatrix},$$

for $\hat{x} \in \hat{\mathbb{G}}, \check{y}, \check{z} \in \check{\mathbb{H}}$ and $\boldsymbol{e} = (0, 1)$. Note that as $\hat{x}a \cdot \check{y} = \hat{x} \cdot a\check{y}$ we will use the simpler notation $\hat{x}a\check{y} = \hat{x}a \cdot \check{y} = \hat{x} \cdot a\check{y}$.

3.2 SXDH Assumption

Let $(p, \hat{\mathbb{G}}, \check{\mathbb{H}}, \mathbb{T}, e, \hat{g}, \check{h})$ be a bilinear group. The Decision Diffie-Hellman (DDH) problem in $\hat{\mathbb{G}}$ is to distinguish the two distributions $(\hat{g}, \xi\hat{g}, \rho\hat{g}, \xi\rho\hat{g})$ and $(\hat{g}, \xi\hat{g}, \rho\hat{g}, \kappa\hat{g})$, where $\xi, \rho, \kappa \leftarrow \mathbb{Z}_p$. The DDH problem in $\check{\mathbb{H}}$ is defined in a similar way.

Definition 5. *The Symmetric eXternal Diffie-Hellman (SXDH) assumption holds relative to $\mathcal{G}$ if the DDH problems are computationally hard in both $\hat{\mathbb{G}}$ and $\check{\mathbb{H}}$ for $(p, \hat{\mathbb{G}}, \check{\mathbb{H}}, \mathbb{T}, e, \hat{g}, \check{h}) \leftarrow \mathcal{G}(1^k)$.*

3.3 ElGamal Encryption

The ElGamal encryption scheme [EG84] is a public key encryption scheme given by the following algorithms:

- **Setup:** on input a security parameter 1^k, output a cyclic group $\hat{\mathbb{G}}$ of prime order p, an element $\hat{g} \in \hat{\mathbb{G}}$ and an element $\xi \leftarrow \mathbb{Z}_p^*$. Then, define the public key as $pk = (\hat{\mathbb{G}}, \hat{\boldsymbol{v}})$, where $\hat{\boldsymbol{v}} = (\xi\hat{g}, \hat{g})^\top \in \hat{\mathbb{G}}^{2\times 1}$ and the secret decryption key as $xk = (pk, \boldsymbol{\xi})$, where $\boldsymbol{\xi} = (-\xi^{-1} \bmod p, 1)$.
- **Encrypt:** the encryption algorithm takes as input the public key pk and a message $\hat{x} \in \hat{\mathbb{G}}$, picks a random $r \leftarrow \mathbb{Z}_p$ and outputs the ciphertext $\hat{\boldsymbol{c}} = \boldsymbol{e}^\top \hat{x} + \hat{\boldsymbol{v}} r \in \hat{\mathbb{G}}^{2\times 1}$, where $\boldsymbol{e} = (0, 1)$.
- **Decrypt:** the decryption algorithm takes as input the secret key xk and a ciphertext $\hat{\boldsymbol{c}} \in \hat{\mathbb{G}}^{2\times 1}$ and outputs $\hat{x} = \boldsymbol{\xi}\hat{\boldsymbol{c}}$. Note $\boldsymbol{\xi}\boldsymbol{e}^\top = 1$ and $\boldsymbol{\xi}\hat{\boldsymbol{v}} = 0$ so simple linear algebra shows decryption is correct.

The ElGamal encryption scheme is IND-CPA secure if the DDH problem is computationally hard in $\hat{\mathbb{G}}$ [TY98]. ElGamal encryption can be defined similarly in $\check{\mathbb{H}}$ and if the SXDH assumption holds we then have IND-CPA secure encryption schemes in both $\hat{\mathbb{G}}$ and $\check{\mathbb{H}}$.

3.4 Pairing-Product Equations and Other Types of Equations

Using the linear algebra friendly additive notation for group operations and multiplicative notation for the pairing, we can express the four types of equations given in the introduction (Sec. 1) in a compact way.

Consider elements $\hat{x}_1, \ldots, \hat{x}_m \in \hat{\mathbb{G}}$ and $\check{y}_1, \ldots, \check{y}_n \in \check{\mathbb{H}}$, which may be publicly known constants (called $\hat{a}_j$ and $\check{b}_i$ in the introduction) or secret variables. Let furthermore the matrix $\Gamma = \{\gamma_{ij}\}_{i=1,j=1}^{m,n} \in \mathbb{Z}_p^{m \times n}$ and $t_{\mathbb{T}} \in \mathbb{T}$ be public values. We can now write the pairing product equation simply as

$$\hat{\boldsymbol{x}} \Gamma \check{\boldsymbol{y}} = t_{\mathbb{T}},$$

where $\hat{\boldsymbol{x}} = (\hat{x}_1, \ldots, \hat{x}_m)$ and $\check{\boldsymbol{y}} = (\check{y}_1, \ldots, \check{y}_n)^\top$.

We can in a similar fashion write multi-scalar multiplication equations in $\hat{\mathbb{G}}$, multi-scalar multiplication equations in $\check{\mathbb{H}}$, and quadratic equations in $\mathbb{Z}_p$ as

$$\hat{\boldsymbol{x}} \Gamma \boldsymbol{y} = \hat{t} \qquad \boldsymbol{x} \Gamma \check{\boldsymbol{y}} = \check{t} \qquad \boldsymbol{x} \Gamma \boldsymbol{y} = t$$

for suitable choices of $m, n \in \mathbb{N}, \Gamma \in \mathbb{Z}_p^{m \times n}, \hat{\boldsymbol{x}} \in \hat{\mathbb{G}}^{1 \times m}, \check{\boldsymbol{y}} \in \check{\mathbb{H}}^{n \times 1}, \boldsymbol{x} \in \mathbb{Z}_p^{1 \times m}, \boldsymbol{y} \in \mathbb{Z}_p^{n \times 1}, \hat{t} \in \hat{\mathbb{G}}, \check{t} \in \check{\mathbb{H}}$ and $t \in \mathbb{Z}_p$. The vectors $\hat{\boldsymbol{x}}, \check{\boldsymbol{y}}, \boldsymbol{x}, \boldsymbol{y}$ may contain a mix of known public values and secret variables.

Groth and Sahai [GS12] made the useful observation that by subtracting $\hat{t} \cdot 1, 1 \cdot \check{t}$ and $1 \cdot t$ on both sides of the respective equations we may without loss of generality assume $\hat{t} = \hat{0}, \check{t} = \check{0}$ and $t = 0$ in all multi-scalar multiplication equations and quadratic equations.

To get zero-knowledge proofs, we will in addition like Groth and Sahai restrict ourselves to $t_{\mathbb{T}} = 0_{\mathbb{T}}$ in all pairing product equations. Groth and Sahai [GS12] do not allow pairings of public constants in the pairing product equations in their zero-knowledge proofs, which we express by requiring the matrix Γ to contain entries $\gamma_{i,j} = 0$ whenever $\hat{x}_i$ and $\check{y}_j$ both are public values. This is because their zero-knowledge simulator breaks down when public values are paired. Groth and Sahai offers a work-around to deal with public values being paired with each other but it involves introducing additional multi-scalar multiplication equations and therefore increases the complexity of the zero-knowledge proof by many group elements. We will show that zero-knowledge simulation is possible when base elements $\hat{g}$ or $\check{h}$ are paired with each other or other public values. Since we do not need the additional multi-scalar multiplication equation used in Groth and Sahai's work-around this yields a significant efficiency gain whenever $\hat{g}$ or $\check{h}$ are paired with each other or other public values.

4 Commitment Keys and Commitments

Like in Groth-Sahai proofs, commitment keys come in two flavours: extraction keys that give perfect soundness and simulation keys that give zero-knowledge. The two types of key generation algorithms are given in Fig. 1 and by the

ExtGen(1^k)		SimGen(1^k)	
$(p, \hat{\mathbb{G}}, \check{\mathbb{H}}, \mathbb{T}, e, \hat{g}, \check{h}) \leftarrow \mathcal{G}(1^k)$		$(p, \hat{\mathbb{G}}, \check{\mathbb{H}}, \mathbb{T}, e, \hat{g}, \check{h}) \leftarrow \mathcal{G}(1^k)$	
$\rho \leftarrow \mathbb{Z}_p, \xi \leftarrow \mathbb{Z}_p^*$	$\sigma \leftarrow \mathbb{Z}_p, \psi \leftarrow \mathbb{Z}_p^*$	$\rho \leftarrow \mathbb{Z}_p, \xi \leftarrow \mathbb{Z}_p^*$	$\sigma \leftarrow \mathbb{Z}_p, \psi \leftarrow \mathbb{Z}_p^*$
$\hat{\boldsymbol{v}} \leftarrow (\xi\hat{g}, \hat{g})^\top$	$\check{\boldsymbol{v}} \leftarrow (\psi\check{h}, \check{h})$	$\hat{\boldsymbol{v}} \leftarrow (\xi\hat{g}, \hat{g})^\top$	$\check{\boldsymbol{v}} \leftarrow (\psi\check{h}, \check{h})$
$\hat{\boldsymbol{w}} \leftarrow \rho\hat{\boldsymbol{v}}$	$\check{\boldsymbol{w}} \leftarrow \sigma\check{\boldsymbol{v}}$	$\hat{\boldsymbol{w}} \leftarrow \rho\hat{\boldsymbol{v}} - (\hat{0}, \hat{g})^\top$	$\check{\boldsymbol{w}} \leftarrow \sigma\check{\boldsymbol{v}} - (\check{0}, \check{h})$
$\hat{\boldsymbol{u}} \leftarrow \hat{\boldsymbol{w}} + (\hat{0}, \hat{g})^\top$	$\check{\boldsymbol{u}} \leftarrow \check{\boldsymbol{w}} + (\check{0}, \check{h})$	$\hat{\boldsymbol{u}} \leftarrow \hat{\boldsymbol{w}} + (\hat{0}, \hat{g})^\top$	$\check{\boldsymbol{u}} \leftarrow \check{\boldsymbol{w}} + (\check{0}, \check{h})$
$\boldsymbol{\xi} \leftarrow (-\xi^{-1} \bmod p, 1)$	$\boldsymbol{\psi} \leftarrow (-\psi^{-1} \bmod p, 1)^\top$		
$ck \leftarrow (p, \hat{\mathbb{G}}, \check{\mathbb{H}}, \mathbb{T}, e, \hat{\boldsymbol{u}}, \hat{\boldsymbol{v}}, \hat{\boldsymbol{w}}, \check{\boldsymbol{u}}, \check{\boldsymbol{v}}, \check{\boldsymbol{w}})$		$ck \leftarrow (p, \hat{\mathbb{G}}, \check{\mathbb{H}}, \mathbb{T}, e, \hat{\boldsymbol{u}}, \hat{\boldsymbol{v}}, \hat{\boldsymbol{w}}, \check{\boldsymbol{u}}, \check{\boldsymbol{v}}, \check{\boldsymbol{w}})$	
$xk \leftarrow (ck, \boldsymbol{\xi}, \boldsymbol{\psi})$		$tk \leftarrow (ck, \rho, \sigma)$	
Return (ck, xk)		Return (ck, tk)	

Fig. 1. Generator algorithms

SXDH assumption extraction keys and simulation keys are computationally indistinguishable.[2]

The column vectors $\hat{\boldsymbol{v}}, \hat{\boldsymbol{w}}, \hat{\boldsymbol{u}} \in \hat{\mathbb{G}}^{2\times 1}$ will be used to make commitments $\hat{\boldsymbol{c}}$ to group elements $\hat{x} \in \hat{\mathbb{G}}$ and scalars $x \in \mathbb{Z}_p$. Commitments to group elements and scalars are computed as

$$\hat{\boldsymbol{c}} \leftarrow \boldsymbol{e}^\top \hat{x} + \hat{\boldsymbol{v}} r + \hat{\boldsymbol{w}} s \qquad \text{and} \qquad \hat{\boldsymbol{c}} \leftarrow \hat{\boldsymbol{u}} x + \hat{\boldsymbol{v}} r,$$

where $r, s \in \mathbb{Z}_p$. Commitments, usually denoted $\check{\boldsymbol{d}}$, to group elements $\check{y} \in \check{\mathbb{H}}$ and scalars $y \in \mathbb{Z}_p$ are made analogously using the row vectors $\check{\boldsymbol{v}}, \check{\boldsymbol{w}}, \check{\boldsymbol{u}} \in \check{\mathbb{H}}^{1\times 2}$.

The commitment scheme is similar to [GS12], however, we will have several different types of commitments and the randomness $r, s \in \mathbb{Z}_p$ we use will depend on the type. Fig. 2 summarizes the commitment types and describes the message, randomness and commitment spaces specified by the public key ck.

The type $t = (\text{pub}_{\hat{\mathbb{G}}}, \hat{x})$ corresponds to a commitment to a public value $\hat{x}$ using randomness $r = s = 0$. It is easy for the verifier to check whether a commitment $\hat{\boldsymbol{c}} = \boldsymbol{e}^\top \hat{x}$ is indeed a correct commitment to a public value $\hat{x}$. Explicitly allowing public values in the commitments simplifies the description of the proofs because we can now treat all elements $\hat{x}_1, \ldots, \hat{x}_m$ in a pairing product equation as committed values regardless of whether they are public or secret. Suppose some of the elements $\hat{\boldsymbol{x}} \in \hat{\mathbb{G}}^{1\times m}$ that appear in a pairing-product equation are committed as constant and others as Groth-Sahai commitments. The matrix consisting of all the commitments $\hat{C} = (\hat{\boldsymbol{c}}_1 \ \cdots \ \hat{\boldsymbol{c}}_m) \in \hat{\mathbb{G}}^{2\times m}$ can be written in a compact way as $\hat{C} = \boldsymbol{e}^\top \hat{\boldsymbol{x}} + \hat{\boldsymbol{v}} \boldsymbol{r}_x + \hat{\boldsymbol{w}} \boldsymbol{s}_x$, where for a constant $\hat{x}_i$ we just have $r_{x_i} = 0$ and $s_{x_i} = 0$.

In a standard Groth-Sahai proof, group element variables are committed as type $t = \text{com}_{\hat{\mathbb{G}}}$ using randomness $r, s \leftarrow \mathbb{Z}_p$. We will for greater efficiency also

[2] The commitment keys are not defined exactly as in [GS12]: by defining $\hat{\boldsymbol{v}}$ as $(\xi\hat{g}, \hat{g})^\top$ instead of $(\hat{g}, \xi\hat{g})^\top$ we will be able to reduce the computational cost of the prover, as explained in Sec. 6. Besides this small difference, the keys $\hat{\boldsymbol{v}}, \hat{\boldsymbol{w}}, \hat{\boldsymbol{u}}, \check{\boldsymbol{v}}, \check{\boldsymbol{w}}$ and $\check{\boldsymbol{u}}$ correspond to u_1, u_2, u, v_1, v_2 and v in [GS12].

t	m	(r,s)	$\boldsymbol{\hat{c}}$	t	m	(r,s)	$\boldsymbol{\check{d}}$
$(\mathrm{pub}_{\hat{\mathbb{G}}}, m)$	$\hat{m} \in \hat{\mathbb{G}}$	$r = s = 0$	$\boldsymbol{\hat{c}} = (\hat{0}, \hat{m})^\top$	$(\mathrm{pub}_{\check{\mathbb{H}}}, m)$	$\check{m} \in \check{\mathbb{H}}$	$r = s = 0$	$\boldsymbol{\check{d}} = (\check{0}, \check{m})$
$\mathrm{enc}_{\hat{\mathbb{G}}}$	$\hat{m} \in \hat{\mathbb{G}}$	$r \in \mathbb{Z}_p, s = 0$	$\boldsymbol{\hat{c}} \in \hat{\mathbb{G}}^{2\times 1}$	$\mathrm{enc}_{\check{\mathbb{H}}}$	$\check{m} \in \check{\mathbb{H}}$	$r \in \mathbb{Z}_p, s = 0$	$\boldsymbol{\check{d}} \in \check{\mathbb{H}}^{1\times 2}$
$\mathrm{com}_{\hat{\mathbb{G}}}$	$\hat{m} \in \hat{\mathbb{G}}$	$r, s \in \mathbb{Z}_p$	$\boldsymbol{\hat{c}} \in \hat{\mathbb{G}}^{2\times 1}$	$\mathrm{com}_{\check{\mathbb{H}}}$	$\check{m} \in \check{\mathbb{H}}$	$r, s \in \mathbb{Z}_p$	$\boldsymbol{\check{d}} \in \check{\mathbb{H}}^{1\times 2}$
$\mathrm{base}_{\hat{\mathbb{G}}}$	$\hat{m} = \hat{g}$	$r = s = 0$	$\boldsymbol{\hat{c}} = (\hat{0}, \hat{g})^\top$	$\mathrm{base}_{\check{\mathbb{H}}}$	$\check{m} = \check{h}$	$r = s = 0$	$\boldsymbol{\check{d}} = (\check{0}, \check{h})$
$\mathrm{sca}_{\hat{\mathbb{G}}}$	$m \in \mathbb{Z}_p$	$r \in \mathbb{Z}_p, s = 0$	$\boldsymbol{\hat{c}} \in \hat{\mathbb{G}}^{2\times 1}$	$\mathrm{sca}_{\check{\mathbb{H}}}$	$m \in \mathbb{Z}_p$	$r \in \mathbb{Z}_p, s = 0$	$\boldsymbol{\check{d}} \in \check{\mathbb{H}}^{1\times 2}$
$\mathrm{unit}_{\hat{\mathbb{G}}}$	$m = 1$	$r = s = 0$	$\boldsymbol{\hat{c}} = \boldsymbol{\hat{u}}$	$\mathrm{unit}_{\check{\mathbb{H}}}$	$m = 1$	$r = s = 0$	$\boldsymbol{\check{d}} = \boldsymbol{\check{u}}$

Fig. 2. $\mathcal{M}_{ck}, \mathcal{R}_{ck}$ and $\mathcal{C}_{ck}$

allow commitments of type $t = \mathrm{enc}_{\hat{\mathbb{G}}}$ where $s = 0$. A type $t = \mathrm{enc}_{\hat{\mathbb{G}}}$ commitment to a group element $\hat{x}$ is $\boldsymbol{\hat{c}} \leftarrow \boldsymbol{e}^\top \hat{x} + \boldsymbol{\hat{v}} r$, which is an ElGamal encryption of $\hat{x}$ as described in Sec. 3.3. Using encryption of variables instead of commitments reduces the computation and in some instances the size of the proofs. However, even on a simulation key the encryptions are only computationally hiding, so we must take care to ensure that it is possible to simulate proofs.

We also introduce the type $t = \mathrm{base}_{\hat{\mathbb{G}}}$ for a commitment to the base element $\hat{g}$ using $r = s = 0$. This type allows us to differentiate $\hat{g}$ from other public values, which is important because simulation becomes problematic when public values are paired with each other. However in the special case when $\hat{g}$ is paired with $\check{h}$ or public constants it is possible to simulate. In addition, one can get shorter zero-knowledge proofs for certain equations by using the special properties of commitments with types $t = \mathrm{base}_{\hat{\mathbb{G}}}$ and $t = \mathrm{base}_{\check{\mathbb{H}}}$.

Scalars have the type $t = \mathrm{sca}_{\hat{\mathbb{G}}}$ and we use the type $t = \mathrm{unit}_{\hat{\mathbb{G}}}$ for a commitment to the scalar 1 using $r = s = 0$. Please note that $t = \mathrm{unit}_{\hat{\mathbb{G}}}$ suffices to incorporate any public value $a \in \mathbb{Z}_p$ into our equations by multiplying the corresponding row in the matrix Γ with a. With these two types we can therefore commit to both variables and constants in $\mathbb{Z}_p$, which simplifies the description of the proofs.

We have now described the types of commitments in $\hat{\mathbb{G}}$ and similar types for commitments in $\check{\mathbb{H}}$ are given in Fig. 2. The commitment algorithm is described in Fig. 3.

The extraction key xk includes a vector $\boldsymbol{\xi}$ such that $\boldsymbol{\xi}\boldsymbol{\hat{v}} = \boldsymbol{\xi}\boldsymbol{\hat{w}} = \hat{0}$ and $\boldsymbol{\xi}\boldsymbol{e}^\top = 1, \boldsymbol{\xi}\boldsymbol{\hat{u}} = \hat{g}$. On a commitment to a group element $\boldsymbol{\hat{c}} = \boldsymbol{e}^\top \hat{x} + \boldsymbol{\hat{v}} r + \boldsymbol{\hat{w}} s$ or on an encryption to a group element $\boldsymbol{\hat{c}} = \boldsymbol{e}^\top \hat{x} + \boldsymbol{\hat{v}} r$ we can extract $\hat{x}$ by computing $\hat{x} = \boldsymbol{\xi}\boldsymbol{\hat{c}}$. On a commitment to a scalar $\boldsymbol{\hat{c}} = \boldsymbol{\hat{u}} x + \boldsymbol{\hat{v}} r$ we extract $\hat{g}x = \boldsymbol{\xi}\boldsymbol{\hat{c}}$, which uniquely determines the committed value x. The extraction algorithm is given in Fig. 4.

The simulated commitment algorithm $\mathrm{SimCom}_{tk}(t)$ commits honestly to public constants, base elements $\hat{g}, \check{h}$ and units 1, which is easy to verify using public information. For all other types it commits to 0. We refer to the full paper [EG13] for a detailed specification.

On a simulation key, the commitments of types $\mathrm{com}_{\hat{\mathbb{G}}}$ or $\mathrm{sca}_{\hat{\mathbb{G}}}$ are perfectly hiding. Commitments of types $(\mathrm{pub}_{\hat{\mathbb{G}}}, \hat{x})$ or $\mathrm{enc}_{\hat{\mathbb{G}}}$ on the other hand are perfectly binding. However, by the SXDH assumption commitments of type $\mathrm{enc}_{\hat{\mathbb{G}}}$ cannot

Input	Randomness	Output	Input	Randomness	Output
$(\mathrm{pub}_{\hat{\mathbb{G}}}, \hat{x}), \hat{x}$	$r \leftarrow 0, s \leftarrow 0$	$\hat{\boldsymbol{c}} \leftarrow \boldsymbol{e}^\top \hat{x}$	$(\mathrm{pub}_{\check{\mathbb{H}}}, \check{y}), \check{y}$	$r \leftarrow 0, s \leftarrow 0$	$\check{\boldsymbol{d}} \leftarrow \check{y}\boldsymbol{e}$
$\mathrm{enc}_{\hat{\mathbb{G}}}, \hat{x}$ $(\star)$	$r \leftarrow \mathbb{Z}_p, s \leftarrow 0$	$\hat{\boldsymbol{c}} \leftarrow \boldsymbol{e}^\top \hat{x} + \hat{\boldsymbol{v}}r$	$\mathrm{enc}_{\check{\mathbb{H}}}, \check{y}$ $(\star)$	$r \leftarrow \mathbb{Z}_p, s \leftarrow 0$	$\check{\boldsymbol{d}} \leftarrow \check{y}\boldsymbol{e} + r\check{\boldsymbol{v}}$
$\mathrm{com}_{\hat{\mathbb{G}}}, \hat{x}$	$r \leftarrow \mathbb{Z}_p, s \leftarrow \mathbb{Z}_p$	$\hat{\boldsymbol{c}} \leftarrow \boldsymbol{e}^\top \hat{x} + \hat{\boldsymbol{v}}r + \hat{\boldsymbol{w}}s$	$\mathrm{com}_{\check{\mathbb{H}}}, \check{y}$	$r \leftarrow \mathbb{Z}_p, s \leftarrow \mathbb{Z}_p$	$\check{\boldsymbol{d}} \leftarrow \check{y}\boldsymbol{e} + r\check{\boldsymbol{v}} + s\check{\boldsymbol{w}}$
$\mathrm{base}_{\hat{\mathbb{G}}}, \hat{g}$ $(\star)$	$r \leftarrow 0, s \leftarrow 0$	$\hat{\boldsymbol{c}} \leftarrow \boldsymbol{e}^\top \hat{g}$	$\mathrm{base}_{\check{\mathbb{H}}}, \check{h}$ $(\star)$	$r \leftarrow 0, s \leftarrow 0$	$\check{\boldsymbol{d}} \leftarrow \check{h}\boldsymbol{e}$
$\mathrm{sca}_{\hat{\mathbb{G}}}, x$	$r \leftarrow \mathbb{Z}_p, s \leftarrow 0$	$\hat{\boldsymbol{c}} \leftarrow \hat{\boldsymbol{u}}x + \hat{\boldsymbol{v}}r$	$\mathrm{sca}_{\check{\mathbb{H}}}, y$	$r \leftarrow \mathbb{Z}_p, s \leftarrow 0$	$\check{\boldsymbol{d}} \leftarrow y\check{\boldsymbol{u}} + r\check{\boldsymbol{v}}$
$\mathrm{unit}_{\hat{\mathbb{G}}}, 1$	$r \leftarrow 0, s \leftarrow 0$	$\hat{\boldsymbol{c}} \leftarrow \hat{\boldsymbol{u}}$	$\mathrm{unit}_{\check{\mathbb{H}}}, 1$	$r \leftarrow 0, s \leftarrow 0$	$\check{\boldsymbol{d}} \leftarrow \check{\boldsymbol{u}}$

Fig. 3. Commitment algorithm. [GS12] do not have the types marked with $(\star)$

$\mathrm{Ext}_{xk}(t, \hat{\boldsymbol{c}})$ where $\hat{\boldsymbol{c}} \in \hat{\mathbb{G}}^{2\times 1}$	$\mathrm{Ext}_{xk}(t, \check{\boldsymbol{d}})$ where $\check{\boldsymbol{d}} \in \check{\mathbb{H}}^{1\times 2}$
Return $\hat{x} \leftarrow \boldsymbol{\xi}\hat{\boldsymbol{c}}$	Return $\check{y} \leftarrow \check{\boldsymbol{d}}\boldsymbol{\psi}$

Fig. 4. Extraction algorithm

be distinguished from commitments to other elements. Commitments of type $(\mathrm{pub}_{\hat{\mathbb{G}}}, \hat{x})$ are public, so we do not require any hiding property.

Commitments to $\hat{g}$ and 1 of types $\mathrm{base}_{\hat{\mathbb{G}}}$ and $\mathrm{unit}_{\hat{\mathbb{G}}}$ are interesting. The secret simulation key specifies ρ such that $\hat{\boldsymbol{u}} = \rho\hat{\boldsymbol{v}}$ and $\boldsymbol{e}^\top\hat{g} = \rho\hat{\boldsymbol{v}} - \hat{\boldsymbol{w}}$. This means that commitments of types $\mathrm{base}_{\hat{\mathbb{G}}}$ and $\mathrm{unit}_{\hat{\mathbb{G}}}$ can be equivocated as either commitments to $\hat{g}$ and 1 or as commitments to $\hat{0}$ and 0. The zero-knowledge simulator will use the equivocations to simulate proofs involving the base element $\hat{g}$ or constants in $\mathbb{Z}_p$.

5 Proofs

We will first explain how the proofs work using the example of pairing product equations to give intuition. We want to prove that committed values $\hat{\boldsymbol{x}}, \check{\boldsymbol{y}}$ satisfy the equation

$$\hat{\boldsymbol{x}}\Gamma\check{\boldsymbol{y}} = 0_{\mathbb{T}}.$$

Assume that we have committed to $\hat{\boldsymbol{x}}, \check{\boldsymbol{y}}$ as $\hat{C} = \boldsymbol{e}^\top\hat{\boldsymbol{x}} + \hat{\boldsymbol{v}}\boldsymbol{r}_x + \hat{\boldsymbol{w}}\boldsymbol{s}_x$ and $\check{D} = \check{\boldsymbol{y}}\boldsymbol{e} + \boldsymbol{r}_y\check{\boldsymbol{v}} + \boldsymbol{s}_y\check{\boldsymbol{w}}$. We then have

$$\begin{aligned}\hat{C}\Gamma\check{D} =&(\boldsymbol{e}^\top\hat{\boldsymbol{x}} + \hat{\boldsymbol{v}}\boldsymbol{r}_x + \hat{\boldsymbol{w}}\boldsymbol{s}_x)\Gamma(\check{\boldsymbol{y}}\boldsymbol{e} + \boldsymbol{r}_y\check{\boldsymbol{v}} + \boldsymbol{s}_y\check{\boldsymbol{w}})\\ =&\boldsymbol{e}^\top\hat{\boldsymbol{x}}\Gamma\check{\boldsymbol{y}}\boldsymbol{e} + \hat{\boldsymbol{v}}\boldsymbol{r}_x\Gamma\check{D} + \hat{\boldsymbol{w}}\boldsymbol{s}_x\Gamma\check{D} + \boldsymbol{e}^\top\hat{\boldsymbol{x}}\Gamma\boldsymbol{r}_y\check{\boldsymbol{v}} + \boldsymbol{e}^\top\hat{\boldsymbol{x}}\Gamma\boldsymbol{s}_y\check{\boldsymbol{w}}\\ =&0_{\mathbb{T}} + \hat{\boldsymbol{v}}\check{\boldsymbol{\pi}}'_{\hat{v}} + \hat{\boldsymbol{w}}\check{\boldsymbol{\pi}}'_{\hat{w}} + \hat{\boldsymbol{\pi}}'_{\check{v}}\check{\boldsymbol{v}} + \hat{\boldsymbol{\pi}}'_{\check{w}}\check{\boldsymbol{w}}\end{aligned}$$

where $\check{\boldsymbol{\pi}}'_{\hat{v}} = \boldsymbol{r}_x\Gamma\check{D}$, $\check{\boldsymbol{\pi}}'_{\hat{w}} = \boldsymbol{s}_x\Gamma\check{D}$, $\hat{\boldsymbol{\pi}}'_{\check{v}} = \boldsymbol{e}^\top\hat{\boldsymbol{x}}\Gamma\boldsymbol{r}_y$, $\hat{\boldsymbol{\pi}}'_{\check{w}} = \boldsymbol{e}^\top\hat{\boldsymbol{x}}\Gamma\boldsymbol{s}_y$.

The prover randomizes $\check{\boldsymbol{\pi}}'_{\hat{v}}, \check{\boldsymbol{\pi}}'_{\hat{w}}, \hat{\boldsymbol{\pi}}'_{\check{v}}, \hat{\boldsymbol{\pi}}'_{\check{w}}$ as $\check{\boldsymbol{\pi}}_{\hat{v}} = \check{\boldsymbol{\pi}}'_{\hat{v}} + \alpha\check{\boldsymbol{v}} + \beta\check{\boldsymbol{w}}$, $\check{\boldsymbol{\pi}}_{\hat{w}} = \check{\boldsymbol{\pi}}'_{\hat{w}} + \gamma\check{\boldsymbol{v}} + \delta\check{\boldsymbol{w}}$, $\hat{\boldsymbol{\pi}}_{\check{v}} = \hat{\boldsymbol{\pi}}'_{\check{v}} - \hat{\boldsymbol{v}}\alpha - \hat{\boldsymbol{w}}\gamma$, $\hat{\boldsymbol{\pi}}_{\check{w}} = \hat{\boldsymbol{\pi}}'_{\check{w}} - \hat{\boldsymbol{v}}\beta - \hat{\boldsymbol{w}}\delta$. This gives us a randomized proof $\check{\boldsymbol{\pi}}_{\hat{v}}, \check{\boldsymbol{\pi}}_{\hat{w}}, \hat{\boldsymbol{\pi}}_{\check{v}}, \hat{\boldsymbol{\pi}}_{\check{w}}$ satisfying the verification equation

$$\hat{C}\Gamma\check{D} = \hat{\boldsymbol{v}}\check{\boldsymbol{\pi}}_{\hat{v}} + \hat{\boldsymbol{w}}\check{\boldsymbol{\pi}}_{\hat{w}} + \hat{\boldsymbol{\pi}}_{\check{v}}\check{\boldsymbol{v}} + \hat{\boldsymbol{\pi}}_{\check{w}}\check{\boldsymbol{w}}.$$

Soundness and $\boldsymbol{F}$-extractability. An extraction key xk contains $\boldsymbol{\xi}, \boldsymbol{\psi}$ such that $\boldsymbol{\xi}\hat{\boldsymbol{v}} = \boldsymbol{\xi}\hat{\boldsymbol{w}} = \hat{0}$ and $\check{\boldsymbol{v}}\boldsymbol{\psi} = \check{\boldsymbol{w}}\boldsymbol{\psi} = \check{0}$. Multiplying the verification equation by $\boldsymbol{\xi}$ and $\boldsymbol{\psi}$ on the left and right side respectively, we get

$$\boldsymbol{\xi}\hat{C}\Gamma\check{D}\boldsymbol{\psi} = \boldsymbol{\xi}\hat{\boldsymbol{v}}\check{\boldsymbol{\pi}}_{\hat{v}}\boldsymbol{\psi} + \boldsymbol{\xi}\hat{\boldsymbol{w}}\check{\boldsymbol{\pi}}_{\hat{w}}\boldsymbol{\psi} + \boldsymbol{\xi}\hat{\boldsymbol{\pi}}_{\check{v}}\check{\boldsymbol{v}}\boldsymbol{\psi} + \boldsymbol{\xi}\hat{\boldsymbol{\pi}}_{\check{w}}\check{\boldsymbol{w}}\boldsymbol{\psi} = 0_{\mathbb{T}}.$$

Observe, $\hat{\boldsymbol{x}} = \boldsymbol{\xi}\hat{C}$ are the values the extractor Ext_{xk} gets from the commitments $\hat{C}$ and $\check{\boldsymbol{y}} = \check{D}\boldsymbol{\psi}$ are the values the extractor Ext_{xk} gets from the commitments $\check{D}$. The extracted values from the commitments therefore satisfy $\hat{\boldsymbol{x}}\Gamma\check{\boldsymbol{y}} = 0_{\mathbb{T}}$. This gives us perfect soundness and perfect F-extractability, where F on group elements in $\hat{\mathbb{G}}$ and $\check{\mathbb{H}}$ is the identity function.

Zero-Knowledge. The simulator will simulate proofs by equivocating the commitments to values $\hat{\boldsymbol{x}}, \check{\boldsymbol{y}}$ that satisfy the equation $\hat{\boldsymbol{x}}\Gamma\check{\boldsymbol{y}} = 0_{\mathbb{T}}$. On a simulation key, commitments with types $\mathrm{com}_{\hat{\mathbb{G}}}, \mathrm{com}_{\check{\mathbb{H}}}$ are perfectly hiding. The simulator can therefore use $\hat{x}_i = \hat{0}$ or $\check{y}_j = \check{0}$. Commitments with types $\mathrm{base}_{\hat{\mathbb{G}}}, \mathrm{base}_{\check{\mathbb{H}}}$ are also equivocable to $\hat{0}$ or $\check{0}$ since on a simulation key $\boldsymbol{e}^{\top}\hat{g} = \hat{\boldsymbol{v}}\rho - \hat{\boldsymbol{w}}$ and $\check{h}\boldsymbol{e} = \sigma\check{\boldsymbol{v}} - \check{\boldsymbol{w}}$. By using equivocations to $\hat{0}$ and $\check{0}$ we can now ensure that $\hat{x}_i\gamma_{i,j}\check{y}_j = 0_{\mathbb{T}}$ whenever $t_{x_i} \in \{\mathrm{base}_{\hat{\mathbb{G}}}, \mathrm{com}_{\hat{\mathbb{G}}}\}$ or $t_{y_j} \in \{\mathrm{base}_{\check{\mathbb{H}}}, \mathrm{com}_{\check{\mathbb{H}}}\}$. Commitments of type $t_{x_i} \in \{(\mathrm{pub}_{\hat{\mathbb{G}}}, \hat{x}), \mathrm{enc}_{\hat{\mathbb{G}}}\}$ and $t_{y_j} \in \{(\mathrm{pub}_{\check{\mathbb{H}}}, \check{y}), \mathrm{enc}_{\check{\mathbb{H}}}\}$ cannot be equivocated and, to get zero-knowledge, we will therefore assume $\gamma_{i,j} = 0$ whenever such types are paired (as is also the case in [GS12]).

We now have that the simulator can equivocate commitments and base elements to $\hat{0}$ and $\check{0}$ such that the resulting $\hat{\boldsymbol{x}}, \check{\boldsymbol{y}}$ satisfy $\hat{\boldsymbol{x}}\Gamma\check{\boldsymbol{y}} = 0_{\mathbb{T}}$. The randomization of the proofs ensures that they will not leak information about whether we are giving a real proof or simulating. Recall the prover randomized $\check{\boldsymbol{\pi}}'_{\hat{v}}, \check{\boldsymbol{\pi}}'_{\hat{w}}, \hat{\boldsymbol{\pi}}'_{\check{v}}, \hat{\boldsymbol{\pi}}'_{\check{w}}$ as $\check{\boldsymbol{\pi}}_{\hat{v}} = \check{\boldsymbol{\pi}}'_{\hat{v}} + \alpha\check{\boldsymbol{v}} + \beta\check{\boldsymbol{w}}$, $\check{\boldsymbol{\pi}}_{\hat{w}} = \check{\boldsymbol{\pi}}'_{\hat{w}} + \gamma\check{\boldsymbol{v}} + \delta\check{\boldsymbol{w}}$, $\hat{\boldsymbol{\pi}}_{\check{v}} = \hat{\boldsymbol{\pi}}'_{\check{v}} - \hat{\boldsymbol{v}}\alpha - \hat{\boldsymbol{w}}\gamma$, $\hat{\boldsymbol{\pi}}_{\check{w}} = \hat{\boldsymbol{\pi}}'_{\check{w}} - \hat{\boldsymbol{v}}\beta - \hat{\boldsymbol{w}}\delta$. On a simulation key this means regardless of whether we are giving a real proof or a simulated proof $\check{\boldsymbol{\pi}}_{\hat{v}}, \check{\boldsymbol{\pi}}_{\hat{w}}$ are uniformly random and $\hat{\boldsymbol{\pi}}_{\check{v}}, \hat{\boldsymbol{\pi}}_{\check{w}}$ are the unique values that make the verification equation true. Finally, the encrypted elements are computationally hidden by the SXDH assumption, so here the simulator may use encryptions of $\hat{0}$ and $\check{0}$ instead of the witness and as we shall show the proofs can be constructed on top of the ciphertexts such that they do not reveal whether the underlying plaintext are part of a real witness or are set to zero by the simulator.

Optimizations. Now let us return to the prover. Observe that $\boldsymbol{r}_x, \boldsymbol{s}_x, \boldsymbol{r}_y, \boldsymbol{s}_y$ may have some zero elements. In particular, assume that all elements in $\boldsymbol{s}_x$ are 0. This happens if all $\hat{x}_i$ in the statement have types $\mathrm{enc}_{\hat{\mathbb{G}}}$, $(\mathrm{pub}_{\hat{\mathbb{G}}}, \hat{x}_i)$ or $\mathrm{base}_{\hat{\mathbb{G}}}$. Moreover, assume that all elements $\check{\boldsymbol{y}}$ have as types either $\mathrm{com}_{\check{\mathbb{H}}}$ or $\mathrm{base}_{\check{\mathbb{H}}}$ so that a simulator uses $\check{\boldsymbol{y}} = \check{\boldsymbol{0}}$ in the simulated proof. This sets $\check{\boldsymbol{\pi}}'_{\hat{w}} = \check{0}$. As $\check{\boldsymbol{\pi}}'_{\hat{w}}$ is the same for all witnesses, even for "simulated witnesses", we might as well set $\gamma = \delta = 0$. For such equations, we therefore save 2 group elements or 25% of the proof size compared to Groth and Sahai [GS12] where there is no $\mathrm{enc}_{\hat{\mathbb{G}}}$ or $\mathrm{enc}_{\check{\mathbb{H}}}$ types. We refer to the full paper [EG13] for a list of equation types and the corresponding proof sizes.

5.1 The Full Proof System

We divide the possible statements into 16 different types. They are summarized in Fig. 5, which provides an algorithm for checking that the statement format is correct. The relation R_L is defined in Fig. 6, which provides an algorithm to check whether a statement is true. The relation first checks that the types of the witnesses and the types of the equations match according to Fig. 5 and then whether the relevant pairing product, multi-scalar multiplication or quadratic equation is satisfied.

$\mathrm{CheckFormat}_{ck}(T, \Gamma, \{t_{x_i}\}_{i=1}^{m}, \{t_{y_j}\}_{j=1}^{n})$

Check $\Gamma \in \mathbb{Z}_p^{m \times n}$

Check that the equation and message types match each other according to the table below

T	$t_{x_1}, \ldots, t_{x_m}$	$t_{y_1}, \ldots, t_{y_n}$
PPE	$\mathrm{base}_{\hat{\mathbb{G}}}, (\mathrm{pub}_{\hat{\mathbb{G}}}, \hat{x}_i), \mathrm{enc}_{\hat{\mathbb{G}}}, \mathrm{com}_{\hat{\mathbb{G}}}$	$\mathrm{base}_{\check{\mathbb{H}}}, (\mathrm{pub}_{\check{\mathbb{H}}}, \check{y}_j), \mathrm{enc}_{\check{\mathbb{H}}}, \mathrm{com}_{\check{\mathbb{H}}}$
$\mathrm{PEnc}_{\hat{\mathbb{G}}}$	$\mathrm{base}_{\hat{\mathbb{G}}}, (\mathrm{pub}_{\hat{\mathbb{G}}}, \hat{x}_i), \mathrm{enc}_{\hat{\mathbb{G}}}$	$\mathrm{base}_{\check{\mathbb{H}}}, \mathrm{com}_{\check{\mathbb{H}}}$
$\mathrm{PConst}_{\hat{\mathbb{G}}}$	$\mathrm{base}_{\hat{\mathbb{G}}}, (\mathrm{pub}_{\hat{\mathbb{G}}}, \hat{x}_i)$	$\mathrm{base}_{\check{\mathbb{H}}}, \mathrm{com}_{\check{\mathbb{H}}}$
$\mathrm{PEnc}_{\check{\mathbb{H}}}$	$\mathrm{base}_{\hat{\mathbb{G}}}, \mathrm{com}_{\hat{\mathbb{G}}}$	$\mathrm{base}_{\check{\mathbb{H}}}, (\mathrm{pub}_{\check{\mathbb{H}}}, \check{y}_j), \mathrm{enc}_{\check{\mathbb{H}}}$
$\mathrm{PConst}_{\check{\mathbb{H}}}$	$\mathrm{base}_{\hat{\mathbb{G}}}, \mathrm{com}_{\hat{\mathbb{G}}}$	$\mathrm{base}_{\check{\mathbb{H}}}, (\mathrm{pub}_{\check{\mathbb{H}}}, \check{y}_j)$
$\mathrm{ME}_{\hat{\mathbb{G}}}$	$\mathrm{base}_{\hat{\mathbb{G}}}, (\mathrm{pub}_{\hat{\mathbb{G}}}, \hat{x}_i), \mathrm{enc}_{\hat{\mathbb{G}}}, \mathrm{com}_{\hat{\mathbb{G}}}$	$\mathrm{unit}_{\check{\mathbb{H}}}, \mathrm{sca}_{\check{\mathbb{H}}}$
$\mathrm{MEnc}_{\hat{\mathbb{G}}}$	$\mathrm{base}_{\hat{\mathbb{G}}}, (\mathrm{pub}_{\hat{\mathbb{G}}}, \hat{x}_i), \mathrm{enc}_{\hat{\mathbb{G}}}$	$\mathrm{unit}_{\check{\mathbb{H}}}, \mathrm{sca}_{\check{\mathbb{H}}}$
$\mathrm{MConst}_{\hat{\mathbb{G}}}$	$\mathrm{base}_{\hat{\mathbb{G}}}, (\mathrm{pub}_{\hat{\mathbb{G}}}, \hat{x}_i)$	$\mathrm{unit}_{\check{\mathbb{H}}}, \mathrm{sca}_{\check{\mathbb{H}}}$
$\mathrm{MLin}_{\hat{\mathbb{G}}}$	$\mathrm{base}_{\hat{\mathbb{G}}}, \mathrm{com}_{\hat{\mathbb{G}}}$	$\mathrm{unit}_{\check{\mathbb{H}}}$
$\mathrm{ME}_{\check{\mathbb{H}}}$	$\mathrm{unit}_{\hat{\mathbb{G}}}, \mathrm{sca}_{\hat{\mathbb{G}}}$	$\mathrm{base}_{\check{\mathbb{H}}}, (\mathrm{pub}_{\check{\mathbb{H}}}, \check{y}_j), \mathrm{enc}_{\check{\mathbb{H}}}, \mathrm{com}_{\check{\mathbb{H}}}$
$\mathrm{MEnc}_{\check{\mathbb{H}}}$	$\mathrm{unit}_{\hat{\mathbb{G}}}, \mathrm{sca}_{\hat{\mathbb{G}}}$	$\mathrm{base}_{\check{\mathbb{H}}}, (\mathrm{pub}_{\check{\mathbb{H}}}, \check{y}_j), \mathrm{enc}_{\check{\mathbb{H}}}$
$\mathrm{MConst}_{\check{\mathbb{H}}}$	$\mathrm{unit}_{\hat{\mathbb{G}}}, \mathrm{sca}_{\hat{\mathbb{G}}}$	$\mathrm{base}_{\check{\mathbb{H}}}, (\mathrm{pub}_{\check{\mathbb{H}}}, \check{y}_j)$
$\mathrm{MLin}_{\check{\mathbb{H}}}$	$\mathrm{unit}_{\hat{\mathbb{G}}}$	$\mathrm{base}_{\check{\mathbb{H}}}, \mathrm{com}_{\check{\mathbb{H}}}$
QE	$\mathrm{unit}_{\hat{\mathbb{G}}}, \mathrm{sca}_{\hat{\mathbb{G}}}$	$\mathrm{unit}_{\check{\mathbb{H}}}, \mathrm{sca}_{\check{\mathbb{H}}}$
$\mathrm{QConst}_{\hat{\mathbb{G}}}$	$\mathrm{unit}_{\hat{\mathbb{G}}}$	$\mathrm{unit}_{\check{\mathbb{H}}}, \mathrm{sca}_{\check{\mathbb{H}}}$
$\mathrm{QConst}_{\check{\mathbb{H}}}$	$\mathrm{unit}_{\hat{\mathbb{G}}}, \mathrm{sca}_{\hat{\mathbb{G}}}$	$\mathrm{unit}_{\check{\mathbb{H}}}$

If $T = \mathrm{PPE}$ check $\Gamma_{i,j} = 0$ for all (i, j) where $t_{x_i} \in \{(\mathrm{pub}_{\hat{\mathbb{G}}}, \hat{x}_i), \mathrm{enc}_{\hat{\mathbb{G}}}\}$ and $t_{y_j} \in \{(\mathrm{pub}_{\check{\mathbb{H}}}, \check{y}_j), \mathrm{enc}_{\check{\mathbb{H}}}\}$

Accept format if all checks pass, else abort

Fig. 5. Equation - message types check

The prover and verifier are given in Fig. 7. The prover constructs a proof for the relevant type of equation assuming the input is a correctly formatted statement with valid openings of commitments to a satisfying witness. The verifier uses the matching verification equation to check validity of a proof.

Let F be given by

$$\begin{array}{ll} F(ck, t, \hat{x}) = \hat{x} & \text{for } t \in \{(\mathrm{pub}_{\hat{\mathbb{G}}}, \hat{x}), \mathrm{enc}_{\hat{\mathbb{G}}}, \mathrm{com}_{\hat{\mathbb{G}}}, \mathrm{base}_{\hat{\mathbb{G}}}\} \\ F(ck, t, x) = \hat{g}x & \text{for } t \in \{\mathrm{sca}_{\hat{\mathbb{G}}}, \mathrm{unit}_{\hat{\mathbb{G}}}\} \\ F(ck, t, \check{y}) = \check{y} & \text{for } t \in \{(\mathrm{pub}_{\check{\mathbb{H}}}, \check{y}), \mathrm{enc}_{\check{\mathbb{H}}}, \mathrm{com}_{\check{\mathbb{H}}}, \mathrm{base}_{\check{\mathbb{H}}}\} \\ F(ck, t, y) = y\check{h} & \text{for } t \in \{\mathrm{sca}_{\check{\mathbb{H}}}, \mathrm{unit}_{\check{\mathbb{H}}}\} \end{array} .$$

```
$R_L(ck, (T, \Gamma), (\{(t_{x_i}, x_i)\}_{i=1}^m, \{(t_{y_j}, y_j)\}_{j=1}^n))$
$\text{CheckFormat}_{ck}(T, \Gamma, \{t_{x_i}\}_{i=1}^m, \{t_{y_j}\}_{j=1}^n)$
For all $i, j$ check $(t_{x_i}, x_i) \in \mathcal{M}_{ck}$ and $(t_{y_j}, y_j) \in \mathcal{M}_{ck}$
If $\boldsymbol{x} \in \hat{\mathbb{G}}^m$ and $\boldsymbol{y} \in \check{\mathbb{H}}^n$ check $\boldsymbol{x}\Gamma\boldsymbol{y} = 0_{\mathbb{T}}$
If $\boldsymbol{x} \in \hat{\mathbb{G}}^m$ and $\boldsymbol{y} \in \mathbb{Z}_p^n$ check $\boldsymbol{x}\Gamma\boldsymbol{y} = \hat{0}$
If $\boldsymbol{x} \in \mathbb{Z}_p^m$ and $\boldsymbol{y} \in \check{\mathbb{H}}^n$ check $\boldsymbol{x}\Gamma\boldsymbol{y} = \check{0}$
If $\boldsymbol{x} \in \mathbb{Z}_p^m$ and $\boldsymbol{y} \in \mathbb{Z}_p^n$ check $\boldsymbol{x}\Gamma\boldsymbol{y} = 0$
Accept if and only if all checks pass
```

Fig. 6. Relation that defines the key-dependent languages for our proofs

```
$\text{Prove}_{ck}(T, \Gamma, \{(t_{x_i}, x_i, (r_{x_i}, s_{x_i}))\}_{i=1}^m, \{(t_{y_j}, y_j, (r_{y_j}, s_{y_j}))\}_{j=1}^n)$
If $\boldsymbol{x} \in \hat{\mathbb{G}}^m$ define $\hat{C} = \boldsymbol{e}^\top \boldsymbol{x} + \hat{\boldsymbol{v}}\boldsymbol{r}_x + \hat{\boldsymbol{w}}\boldsymbol{s}_x$ else if $\boldsymbol{x} \in \mathbb{Z}_p^m$ define $\hat{C} = \hat{\boldsymbol{u}}\boldsymbol{x} + \hat{\boldsymbol{v}}\boldsymbol{r}_x$
If $\boldsymbol{y} \in \check{\mathbb{H}}^n$ define $\check{D} = \boldsymbol{y}\boldsymbol{e} + \boldsymbol{r}_y\check{\boldsymbol{v}} + \boldsymbol{s}_y\check{\boldsymbol{w}}$ else if $\boldsymbol{y} \in \mathbb{Z}_p^n$ define $\check{D} = \boldsymbol{y}\check{\boldsymbol{u}} + \boldsymbol{r}_y\check{\boldsymbol{v}}$
Set $\alpha = \beta = \gamma = \delta = 0$
If $T = \text{PPE}$ pick $\alpha, \beta, \gamma, \delta \leftarrow \mathbb{Z}_p$
If $T \in \{\text{PEnc}_{\hat{\mathbb{G}}}, \text{ME}_{\check{\mathbb{H}}}\}$ pick $\alpha, \beta \leftarrow \mathbb{Z}_p$
If $T \in \{\text{PEnc}_{\check{\mathbb{H}}}, \text{ME}_{\hat{\mathbb{G}}}\}$ pick $\alpha, \gamma \leftarrow \mathbb{Z}_p$
If $T \in \{\text{MEnc}_{\hat{\mathbb{G}}}, \text{MEnc}_{\check{\mathbb{H}}}, \text{QE}\}$ pick $\alpha \leftarrow \mathbb{Z}_p$
    $\check{\boldsymbol{\pi}}_{\hat{v}} \leftarrow \boldsymbol{r}_x\Gamma\check{D} + \alpha\check{\boldsymbol{v}} + \beta\check{\boldsymbol{w}}$        $\hat{\boldsymbol{\pi}}_{\check{v}} \leftarrow (\hat{C} - \hat{\boldsymbol{v}}\boldsymbol{r}_x - \hat{\boldsymbol{w}}\boldsymbol{s}_x)\Gamma\boldsymbol{r}_y - \hat{\boldsymbol{v}}\alpha - \hat{\boldsymbol{w}}\gamma$
    $\check{\boldsymbol{\pi}}_{\hat{w}} \leftarrow \boldsymbol{s}_x\Gamma\check{D} + \gamma\check{\boldsymbol{v}} + \delta\check{\boldsymbol{w}}$        $\hat{\boldsymbol{\pi}}_{\check{w}} \leftarrow (\hat{C} - \hat{\boldsymbol{v}}\boldsymbol{r}_x - \hat{\boldsymbol{w}}\boldsymbol{s}_x)\Gamma\boldsymbol{s}_y - \hat{\boldsymbol{v}}\beta - \hat{\boldsymbol{w}}\delta$
Return $\pi = (\check{\boldsymbol{\pi}}_{\hat{v}}, \check{\boldsymbol{\pi}}_{\hat{w}}, \hat{\boldsymbol{\pi}}_{\check{v}}, \hat{\boldsymbol{\pi}}_{\check{w}})$
```

```
$\text{Verify}_{ck}(T, \Gamma, \{(t_{x_i}, \hat{\boldsymbol{c}}_i)\}_{i=1}^m, \{(t_{y_j}, \check{\boldsymbol{d}}_j)\}_{j=1}^n, \pi)$
$\text{CheckFormat}_{ck}(T, \Gamma, \{t_{x_i}\}_{i=1}^m, \{t_{y_j}\}_{j=1}^n)$
Check $\hat{C} = (\hat{\boldsymbol{c}}_1 \cdots \hat{\boldsymbol{c}}_m) \in \hat{\mathbb{G}}^{2\times m}$ and $\check{D} = \big(\check{\boldsymbol{d}}_1 \cdots \check{\boldsymbol{d}}_n\big)^\top \in \check{\mathbb{H}}^{n\times 2}$
Check $\pi = (\check{\boldsymbol{\pi}}_{\hat{v}}, \check{\boldsymbol{\pi}}_{\hat{w}}, \hat{\boldsymbol{\pi}}_{\check{v}}, \hat{\boldsymbol{\pi}}_{\check{w}}) \in \check{\mathbb{H}}^{2\times 1} \times \check{\mathbb{H}}^{2\times 1} \times \hat{\mathbb{G}}^{1\times 2} \times \hat{\mathbb{G}}^{1\times 2}$
Check $\hat{C}\Gamma\check{D} = \hat{\boldsymbol{v}}\check{\boldsymbol{\pi}}_{\hat{v}} + \hat{\boldsymbol{w}}\check{\boldsymbol{\pi}}_{\hat{w}} + \hat{\boldsymbol{\pi}}_{\check{v}}\check{\boldsymbol{v}} + \hat{\boldsymbol{\pi}}_{\check{w}}\check{\boldsymbol{w}}$
Return 1 if all checks pass, else return 0
```

Fig. 7. Prover and verifier algorithms

Theorem 1. *The commit-and-prove scheme given in Figs. 1,3,4 and 7 has perfect correctness, perfect soundness and F-extractability for the function F defined above, and computational composable zero-knowledge if the SXDH assumption holds relative to $\mathcal{G}$.*

Due to lack of space, the description of the zero-knowledge simulator and the proof of the theorem is given in the full version [EG13].

6 NIZK Proofs with Prover-Chosen CRS

In Groth-Sahai proofs, the prover uses a common reference string shared between the prover and the verifier to construct NIZK proofs. We can improve efficiency by letting the prover choose her own common reference string, which we will refer to as her public key. To maintain the soundness of the NIZK proof, the prover will create its public key as a perfectly binding key and will make a NIZK proof using the shared common reference string to prove that the public key is binding. In this section we will explain how the prover creates her public key, proves its well-formedness and we explain what the efficiency improvement obtained is. In the full version of this paper [EG13] we give definitions for commit-and-prove schemes with prover-chosen CRS and we prove the security of our scheme.

6.1 Creating the Public Key

Like commitment keys, public keys can be created in two ways: they can either be perfectly binding or perfectly hiding. These two types of keys are computationally indistinguishable if the SXDH assumption holds. As we already argued, we will require the prover to create her public key in a perfectly binding way. However, the zero-knowledge simulator will create a perfectly hiding public key and simulate the NIZK proof for well-formedness.

ProverGen(ck)		SimProverGen(ck)	
$\rho_P \leftarrow \mathbb{Z}_p$	$\sigma_P \leftarrow \mathbb{Z}_p$	$\rho_P \leftarrow \mathbb{Z}_p$	$\sigma_P \leftarrow \mathbb{Z}_p$
$\hat{\boldsymbol{v}}_P \leftarrow \hat{\boldsymbol{v}}$	$\check{\boldsymbol{v}}_P \leftarrow \check{\boldsymbol{v}}$	$\hat{\boldsymbol{v}}_P \leftarrow \hat{\boldsymbol{v}}$	$\check{\boldsymbol{v}}_P \leftarrow \check{\boldsymbol{v}}$
$\hat{\boldsymbol{w}}_P \leftarrow \rho_P \hat{\boldsymbol{v}}_P$	$\check{\boldsymbol{w}}_P \leftarrow \sigma_P \check{\boldsymbol{v}}_P$	$\hat{\boldsymbol{w}}_P \leftarrow \rho_P \hat{\boldsymbol{v}}_P - (\hat{0}, \hat{g})^\top$	$\check{\boldsymbol{w}}_P \leftarrow \sigma_P \check{\boldsymbol{v}}_P - (\check{0}, \check{h})$
$\hat{\boldsymbol{u}}_P \leftarrow \hat{\boldsymbol{w}}_P + (\hat{0}, \hat{g})^\top$	$\check{\boldsymbol{u}}_P \leftarrow \check{\boldsymbol{w}}_P + (\check{0}, \check{h})$	$\hat{\boldsymbol{u}}_P \leftarrow \hat{\boldsymbol{w}}_P + (\hat{0}, \hat{g})^\top$	$\check{\boldsymbol{u}}_P \leftarrow \check{\boldsymbol{w}}_P + (\check{0}, \check{h})$
$pk \leftarrow (\hat{\boldsymbol{u}}_P, \hat{\boldsymbol{v}}_P, \hat{\boldsymbol{w}}_P, \check{\boldsymbol{u}}_P, \check{\boldsymbol{v}}_P, \check{\boldsymbol{w}}_P)$		$pk \leftarrow (\hat{\boldsymbol{u}}_P, \hat{\boldsymbol{v}}_P, \hat{\boldsymbol{w}}_P, \check{\boldsymbol{u}}_P, \check{\boldsymbol{v}}_P, \check{\boldsymbol{w}}_P)$	
$sk \leftarrow (pk, \rho_P, \sigma_P)$		$sk \leftarrow (pk, \rho_P, \sigma_P)$	
Return (pk, sk)		Return (pk, sk)	

Fig. 8. Public key generator algorithms

As shown in Fig. 8, the public key is created in a similar way to how the commitment key is created. The main difference is that the bilinear group $(p, \hat{\mathbb{G}}, \check{\mathbb{H}}, \mathbb{T}, e, \hat{g}, \check{h})$ is already fixed, and that we allow the prover to reuse the elements $\hat{\boldsymbol{v}}, \check{\boldsymbol{v}}$. This both reduces the size of the public key and also ensures that the prover's commitments are extractable even when using her own key.

Once the prover has created her pair of public key and secret key, she has to compute an NIZK proof to show that her pk is perfectly binding. A valid public key is defined by the existence of some ρ_P, σ_P such that $\hat{\boldsymbol{w}}_P = \rho_P \hat{\boldsymbol{v}}$ and $\check{\boldsymbol{w}}_P = \sigma_P \check{\boldsymbol{v}}$, which can be written as two equations of type MConst$_{\hat{\mathbb{G}}}$ involving public elements in $\hat{\mathbb{G}}$ and a secret ρ_P committed in $\check{\mathbb{H}}$, and two equations of type MConst$_{\check{\mathbb{H}}}$ involving public elements in $\check{\mathbb{H}}$ and a secret σ_P committed in $\hat{\mathbb{G}}$. These are simple statements that each have a proof consisting of a single group element. In the full version of this paper [EG13] we give the exact NIZK proofs that have to be computed. The total cost of communicating the public key, which is determined by the commitments to ρ_P and σ_P and the NIZK proofs is 12 group elements. Since we are using a commit-and-prove scheme we can consider this as a one-off cost for each verifier engaging with the prover after which the public key may be used for many commitments and proofs.

6.2 Computing Commitments and NIZK Proofs

Once the prover has created her public key *pk* and has proven its well-formedness, she can make commitments and prove statements using *pk* instead of *ck*. The commitments and proofs are created and verified in exactly the same way as described in Fig. 3 and Fig. 7, but the number of scalar multiplications needed to compute commitments and NIZK proofs can be reduced using her knowledge of the discrete logarithms in *sk*. We have for instance

$$\hat{\boldsymbol{c}} = \boldsymbol{e}^\top \hat{x} + \hat{\boldsymbol{v}} r + \hat{\boldsymbol{w}}_P s = \boldsymbol{e}^\top \hat{x} + \hat{\boldsymbol{v}}(r + \rho_P s),$$

so the prover can compute a commitment with 2 scalar multiplications instead of 4 scalar multiplications.

By using the secret key *sk* the prover can reduce the number of scalar multiplications by 50% for commitments to group elements and commitments to elements in $\mathbb{Z}_p$. Computing NIZK proofs is more complicated and there are many operations that cannot be avoided by using the secret key *sk*. However, in some cases the improvement is very noticeable as in the case of quadratic equations (T = QE) where the number of scalar multiplications is reduced by 50%[3]. Furthermore, in most applications found in the literature there are only a few variables in the equations, which makes our improvements more significant. The exact savings can be found in the full version of this paper [EG13].

References

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

[BFI⁺10] Blazy, O., Fuchsbauer, G., Izabachène, M., Jambert, A., Sibert, H., Vergnaud, D.: Batch Groth-Sahai. In: Zhou, J., Yung, M. (eds.) ACNS 2010. LNCS, vol. 6123, pp. 218–235. Springer, Heidelberg (2010)

[3] We assume that operations in $\check{\mathbb{H}}$ are more computationally expensive than operations in $\hat{\mathbb{G}}$, as usually $\hat{\mathbb{G}}$ is an elliptic curve over a prime order field and $\check{\mathbb{H}}$ is the same elliptic curve over an extension field [GPS08]. Therefore, we have tried to reduce the numbers of operations in $\check{\mathbb{H}}$ as much as possible. In addition, we have for simplicity assumed that the commitments that appear in the NIZK proof have as many randomization factors as possible conditioned to the equation type T.

[BFM88] Blum, M., Feldman, P., Micali, S.: Non-interactive zero-knowledge and its applications. In: Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing, STOC 1988, pp. 103–112. ACM, New York (1988)

[BW06] Boyen, X., Waters, B.: Compact group signatures without random oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 427–444. Springer, Heidelberg (2006)

[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)

[CHP07] Camenisch, J.L., Hohenberger, S., Pedersen, M.Ø.: Batch verification of short signatures. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 246–263. Springer, Heidelberg (2007)

[CKLM12] Chase, M., Kohlweiss, M., Lysyanskaya, A., Meiklejohn, S.: Malleable proof systems and applications. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 281–300. Springer, Heidelberg (2012)

[CLOS02] Canetti, R., Lindell, Y., Ostrovsky, R., Sahai, A.: Universally composable two-party and multi-party secure computation. In: 34th ACM STOC, pp. 494–503. ACM Press (2002)

[Dam92] Damgård, I.B.: Non-interactive circuit based proofs and non-interactive perfect zero-knowledge with preprocessing. In: Rueppel, R.A. (ed.) EUROCRYPT 1992. LNCS, vol. 658, pp. 341–355. Springer, Heidelberg (1993)

[EG13] Alex Escala and Jens Groth. Fine-Tuning Groth-Sahai Proofs. Cryptology ePrint Archive, Report 2013/662

[EG84] El Gamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 10–18. Springer, Heidelberg (1985)

[EHKRV13] Escala, A., Herold, G., Kiltz, E., Ràfols, C., Villar, J.: An Algebraic Framework for Diffie-Hellman Assumptions. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 129–147. Springer, Heidelberg (2013)

[Fuc11] Fuchsbauer, G.: Commuting Signatures and Verifiable Encryption. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 224–245. Springer, Heidelberg (2011)

[FLS99] Feige, U., Lapidot, D., Shamir, A.: Multiple non-interactive zero knowledge proofs under general assumptions. SIAM Journal on Computing 29(1), 1–28 (1999)

[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)

[GO07] Groth, J., Ostrovsky, R.: Cryptography in the Multi-string Model. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 323–341. Springer, Heidelberg (2007)

[GOS12] Groth, J., Ostrovsky, R., Sahai, A.: New techniques for noninteractive zero-knowledge. J. ACM 59(3), 11 (2012)

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

[Gro06] Groth, J.: Simulation-sound NIZK proofs for a practical language and constant size group signatures. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 444–459. Springer, Heidelberg (2006)

[Gro07] Groth, J.: Fully Anonymous Group Signatures Without Random Oracles. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 164–180. Springer, Heidelberg (2007)

[GS12] Groth, J., Sahai, A.: Efficient non-interactive proof systems for bilinear groups. SIAM Journal on Computing 41(5), 1193–1232 (2012)

[GSW10] Ghadafi, E., Smart, N.P., Warinschi, B.: Groth-Sahai proofs revisited. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 177–192. Springer, Heidelberg (2010)

[JR13] Jutla, C.S., Roy, A.: Shorter Quasi-Adaptive NIZK Proofs for Linear Subspaces. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013, Part I. LNCS, vol. 8269, pp. 1–20. Springer, Heidelberg (2013)

[Kil90] Kilian, J.: Uses of randomness in algorithms and protocols. MIT Press (1990)

[KP98] Kilian, J., Petrank, E.: An efficient noninteractive zero-knowledge proof system for NP with general assumptions. Journal of Cryptology 11(1), 1–27 (1998)

[Mei09] Meiklejohn, S.: An Extension of the Groth-Sahai Proof System. Master's thesis, Brown University, Providence, RI (2009)

[Seo12] Seo, J.H.: On the (Im)possibility of Projecting Property in Prime-Order Setting. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 61–79. Springer, Heidelberg (2012)

[TY98] Tsiounis, Y., Yung, M.: On the security of ELGamal based encryption. In: Imai, H., Zheng, Y. (eds.) PKC 1998. LNCS, vol. 1431, pp. 117–134. Springer, Heidelberg (1998)