Fine-Tuning Groth-Sahai Proofs

Alex Escala[1] and Jens Groth[2]

[1] Universitat Autònoma de Barcelona
[2] University College of London

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 the truth of a statement without revealing any other information but the truth of the statement. 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 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, i.e., they enable a prover to convince a verifier that a statement is true in a way such that the verifier cannot tell which witness the prover has used. 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, i.e., they 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. Recently, [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.

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. The commitment key can be set up to be perfectly binding, in which case it is impossible to cheat in the proof (perfect soundness). Alternatively, the commitment key can be set up to be perfectly hiding, in which case the NIZK proof can be simulated without knowledge of the witness (perfect zero-knowledge).

In our scheme we allow the prover to encrypt variables using ElGamal encryption as an alternative to the commitment scheme. ElGamal encryption requires two scalar multiplications and commitments in Groth-Sahai proofs require four scalar multiplications so this reduces the prover's computation. Moreover, equations that use ElGamal ciphertexts instead of commitments have simpler proofs and we save two group elements for each of those equations.

We must be careful when using ElGamal encryption though since it will always be perfectly binding regardless of the key. We can therefore not set up the common reference string to give us perfect zero-knowledge any more. Instead, 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. Groth-Sahai proofs are zero-knowledge if all pairing-product equations have $t_{\mathbb{T}} = 0_{\mathbb{T}}$ and no public constants are paired with each other. Groth and Sahai gave a technique to rewrite the equations to enable zero-knowledge proofs when $t_{\mathbb{T}} = \sum_{i=1}^{n} \hat{t}_i \cdot \check{t}_i$, however, this comes at a cost of $O(n)$ group elements.

We observe that the commitment keys can be set up to allow simulation in the special case where the public constants are the base elements $\hat{g}$ or $\check{h}$. In the context of Groth-Sahai proofs this makes zero-knowledge simulation possible when $t_{\mathbb{T}} = \hat{a} \cdot \check{h} + \hat{g} \cdot \check{b}$ 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 to get zero-knowledge.

In addition, 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 constants consist of 6 group elements unless $\hat{t} = \hat{0}$ or $\check{t} = \check{0}$, in which case the size of the proof is reduced to 2 group elements. We observe that if $\hat{t} = \hat{g}$ or $\check{t} = \check{h}$ then we can still have zero-knowledge proofs which consist of 2 group elements.

Using commitment keys with known discrete logarithms. In Groth-Sahai proofs, the prover uses a common reference string shared between the prover and the verifier to create NIZK proofs. This common reference string is created by a trusted entity and, in particular, the discrete logarithms of the commitment keys with respect to $\hat{g}$ and $\check{h}$ are unknown to both the prover and the verifier.

In this paper 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.

The verifier does of course need a guarantee that the prover's public key is formed as a perfectly binding common reference string, otherwise the proofs would not be sound any more. This can be accomplished by letting the prover give a Groth-Sahai proof, using a common reference string the verifier does trust, for the public key being correct. We show that the prover can communicate her public key and prove it is correct using 12 group elements in total, which is a one-off cost as the public key can be used for many commitments and proofs.

Seeing the common reference string as the prover's public key also 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 the setup assumption is relaxed to assuming 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 commit-and-prove scheme [Kil90,CLOS02] is a natural generalization of a zero-knowledge proof, where the prover can commit to values and prove statements about the committed values. Commit-and-prove schemes provide extra flexibility and reduce communication; the same commitments can for instance be used in different proofs for adaptively chosen statements about the committed values.

Groth-Sahai proofs can be used to build a non-interactive commit-and-prove scheme in a natural way. Fuchsbauer [Fuc11] defined a witness-indistinguishable Groth-Sahai based commit-and-prove scheme and used it in the construction of delegatable anonymous credentials.

We will formulate our construction as a non-interactive commit-and-prove scheme because it allows for greater flexibility. It is for instance possible to choose values to be committed to in an adaptive fashion that depends on previous commitments or proofs, while the traditional definition of NIZK proofs would require the prover to make an entirely new set of commitments for each statement to be proven.

Our results are natural extensions of Groth-Sahai proofs and 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. We could resolve this by using multiple commitment schemes, i.e., define and construct a many-commitment-scheme-and-prove system. However, the definition would quickly become unwieldy as the number of commitment schemes grows.

To give a generally applicable definition of non-interactive commit-and-prove schemes, we instead 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. One example of a type could for instance be $t = \text{enc}_{\hat{\mathbb{G}}}$ meaning the value m should be encrypted (as opposed to using the more expensive commitment operation) and it should be done in group $\hat{\mathbb{G}}$. Type-based commitments resemble tag-based commitments, where a message m can be committed under an arbitrary tag

tag, but a crucial difference is that while tags can be arbitrary bit-strings, types may be restricted. More precisely, we require that the type and message pair (t, m) belong to a message space $\mathcal{M}_{ck}$. We consider the type as being part of the message, so another way of looking at a type-based commitment scheme is to say a message consists of public part t and a private part m. This increases the flexibility of the commitment scheme, we can for instance create a type $(\text{pub}_{\hat{\mathbb{G}}}, x)$ 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.

Armed with type-based commitments we provide a formal definition of commit-and-prove schemes, where the prover may commit to values and prove statements about the values. We stress that commitments and proofs can be arbitrarily interleaved so at any given point a new value or statement may depend on previous commitments and proofs, which makes them more flexible than standard Groth-Sahai proofs. The commit-and-prove scheme is defined to be sound if it is impossible to prove a false statement about the committed values and it is defined to be zero-knowledge if there exists a simulator that can simulate commitments and proofs.

Applications. To illustrate the advantages of our fine-tuned Groth-Sahai proofs we will 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

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

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 $(\hat{v} + m\hat{g}) \cdot \check{\sigma} - \hat{g}\check{y} = 0_{\mathbb{T}}$. This work-around 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 $\check{\sigma}$ 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 still 2 group elements just as the commitment in a Groth-Sahai proof 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 of the proof by 63% from 16 group elements to 6 group elements.

In Sec. 8 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 $\mathrm{CP} = (\mathrm{Gen}, \mathrm{Com}, \mathrm{Prove}, \mathrm{Verify})$ consists of four polynomial time algorithms. The algorithms $\mathrm{Gen}, \mathrm{Prove}$ are probabilistic and the algorithms $\mathrm{Com}, \mathrm{Verify}$ 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 knowledge as the ability 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, 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-knowledge). *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 *has F-knowledge for a function F if for all adversaries* $\mathcal{A}$

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

We say 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 already leaked by the public parts. This is defined as the possibility 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 about the commitment key ck.

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* $\mathrm{SimGen}, \mathrm{SimCom}, \mathrm{SimProve}$ *such that for all non-uniform polynomial time stateful interactive adversaries* $\mathcal{A}$ [3]

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

and

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

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* $(ck, x, w_{i_1}, \ldots, w_{i_n}) \in R_L$

Comparison to standard zero-knowledge proofs. The commit-and-prove functionality defined above is a generalization of standard zero-knowledge proofs. The commitment key corresponds to a common reference string. To give a proof for $x \in L_{ck}$ the prover could commit to the witness and then give a proof. The correctness, soundness and zero-knowledge properties given above imply that this would yield a non-interactive zero-knowledge proof system for R_L.

The generalization to a commit-and-prove functionality is quite useful for efficiency reasons. Using a traditional zero-knowledge proof system the prover has to start from scratch every time a new statement has to be proved. In a commit-and-prove system, commitments to values can be reused in multiple proofs, which reduces both computation and communication.

There are also conceptual advantages to using a commit-and-prove functionality. One is that it is convenient that the same committed values can be used across multiple proofs. Another one is that we capture in a natural way that statements may be related to commitments and proofs that have been made before.

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.

[3] 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 $f(k) \approx 1$.

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 $0_{\mathbb{T}}$.

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

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

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 $(p, \hat{\mathbb{G}}, \check{\mathbb{H}}, \mathbb{T}, e, \hat{g}, \check{h}) \leftarrow \mathcal{G}(1^k)$ 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 $\varGamma = \{\gamma_{ij}\}_{i=1,j=1}^{m,n} \in \mathbb{Z}_p^{m\times n}$ and $t_{\mathbb{T}} \in \mathbb{T}$ be public values. We can now write the pairing product equation simply as

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

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

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

for suitable choices of $m, n \in \mathbb{N}, \mathit{\Gamma} \in \mathbb{Z}_p^{m\times n}, \hat{\boldsymbol{x}} \in \hat{\mathbb{G}}^{1\times m}, \check{\boldsymbol{y}} \in \check{\mathbb{H}}^{n\times 1}, \boldsymbol{x} \in \mathbb{Z}_p^{1\times m}, \boldsymbol{y} \in \mathbb{Z}_p^{n\times 1}, \hat{t} \in \hat{\mathbb{G}}, \check{t} \in \check{\mathbb{H}}$ 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. For multi-scalar multiplication equations this has an implicit cost, though: we need to treat the field element 1 as a variable, as the simulator will need to equivocate it to the field element 0. This means that if all the field elements that appear in the multi-scalar multiplication equation are constants, the equation will not be treated as a linear equation any more, which have witness-indistinguishable proofs which consist of 2 group elements. Instead, the zero-knowledge proof will consist of 6 group elements. However, in the special case where $\hat{t} = \hat{g}$ or $\check{t} = \check{h}$ we will show that we can still treat the equation as a linear equation, which have zero-knowledge proofs consisting of 2 group elements.

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 $\mathit{\Gamma}$ 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 SXDH assumption extraction keys and simulation keys are computationally indistinguishable.[4]

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

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

Fig. 1: Generator algorithms

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

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

[4] 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. 9. 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].

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.

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

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

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

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 allow commitments of type $t = \text{enc}_{\hat{\mathbb{G}}}$ where $s = 0$. A type $t = \text{enc}_{\hat{\mathbb{G}}}$ commitment to a group element $\hat{x}$ is $\hat{\boldsymbol{c}} \leftarrow \boldsymbol{e}^\top \hat{x} + \hat{\boldsymbol{v}} r$, 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 $t = \text{base}_{\hat{\mathbb{G}}}$ 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 the 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 $\hat{g}$ and $\check{h}$.

Scalars have the type $t = \text{sca}_{\hat{\mathbb{G}}}$ and we use the type $t = \text{unit}_{\hat{\mathbb{G}}}$ for a commitment to the scalar 1 using $r = s = 0$. Please note that $t = \text{unit}_{\hat{\mathbb{G}}}$ suffices to incorporate any public value $a \in \mathbb{Z}_p$ into our equations by multiplying the corresponding row in the matrix $\mathit{\Gamma}$ 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.

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

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

Fig. 3: Commitment algorithm. The commitments marked with $(\star)$ were not defined in [GS12].

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

$\mathrm{Ext}_{xk}(t, \hat{\boldsymbol{c}})$ 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.

The simulated commitment algorithm $\mathrm{SimCom}_{tk}(t)$ is described in Fig. 5. Essentially it 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.

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

Fig. 5: Simulation algorithm for commitments.

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 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

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

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

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

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

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

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

Soundness and $\boldsymbol{F}$-knowledge. 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-knowledge, 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 $\gamma = \delta = 0$. This gives us a proof that consists of 4 elements in $\hat{\mathbb{G}}$ and 2 elements in $\check{\mathbb{H}}$ instead of 4 elements both in $\hat{\mathbb{G}}$ and $\check{\mathbb{H}}$. 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 Fig. 9 for the 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. 6, which provides an algorithm for checking that the statement format is correct.

The relation R_L is defined in Fig. 7, 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. 6 and then whether the relevant pairing product, multi-scalar multiplication or quadratic equation is satisfied.

The prover and verifier are given in Fig. 8. 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.

As described earlier there are some types of equations where parts of the proof are just 0 or can be compressed. In Fig. 9 we show the parts of the proofs that allow reduced communication for each type of equation and, if the proof is shortened, we explain how it is done.

Let F be given by

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

```
$\mathsf{CheckFormat}_{ck}(T, \Gamma, \{t_{x_i}\}_{i=1}^{m}, \{t_{y_j}\}_{j=1}^{n})$
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. 6: Equation - message types check

```
$R_L(ck, (T, \Gamma), (\{(t_{x_i}, x_i)\}_{i=1}^{m}, \{(t_{y_j}, y_j)\}_{j=1}^{n}))$
$\mathsf{CheckFormat}_{ck}(T, \Gamma, \{t_{x_i}\}_{i=1}^{m}, \{t_{y_j}\}_{j=1}^{n})$
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. 7: Relation that defines the key-dependent languages for our proofs

```
$\mathsf{Prove}_{ck}(T, \Gamma, \{(t_{x_i}, x_i, (r_{x_i}, s_{x_i}))\}_{i=1}^{m}, \{(t_{y_j}, y_j, (r_{y_j}, s_{y_j}))\}_{j=1}^{n})$
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 $T = \mathrm{PPE}$ pick $\alpha, \beta, \gamma, \delta \leftarrow \mathbb{Z}_p$
If $T \in \{\mathrm{PEnc}_{\hat{\mathbb{G}}}, \mathrm{ME}_{\check{\mathbb{H}}}\}$ pick $\alpha, \beta \leftarrow \mathbb{Z}_p$
If $T \in \{\mathrm{PEnc}_{\check{\mathbb{H}}}, \mathrm{ME}_{\hat{\mathbb{G}}}\}$ pick $\alpha, \gamma \leftarrow \mathbb{Z}_p$
If $T \in \{\mathrm{MEnc}_{\hat{\mathbb{G}}}, \mathrm{MEnc}_{\check{\mathbb{H}}}, \mathrm{QE}\}$ 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 $\pi = (\check{\boldsymbol{\pi}}_{\hat{v}}, \check{\boldsymbol{\pi}}_{\hat{w}}, \hat{\boldsymbol{\pi}}_{\check{v}}, \hat{\boldsymbol{\pi}}_{\check{w}})$
```

```
$\mathsf{Verify}_{ck}(T, \Gamma, \{(t_{x_i}, \hat{\boldsymbol{c}}_i)\}_{i=1}^{m}, \{(t_{y_j}, \check{\boldsymbol{d}}_j)\}_{j=1}^{n}, \pi)$
$\mathsf{CheckFormat}_{ck}(T, \Gamma, \{t_{x_i}\}_{i=1}^{m}, \{t_{y_j}\}_{j=1}^{n})$
Check $\hat{C} = (\hat{\boldsymbol{c}}_1 \cdots \hat{\boldsymbol{c}}_m) \in \hat{\mathbb{G}}^{2 \times m}$ and $\check{D} = \left(\check{\boldsymbol{d}}_1 \cdots \check{\boldsymbol{d}}_n\right)^\top \in \check{\mathbb{H}}^{n \times 2}$
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. 8: Prover and verifier algorithms

T	$\check{\boldsymbol{\pi}}_{\hat{v}}$	$\check{\boldsymbol{\pi}}_{\hat{w}}$	$\hat{\boldsymbol{\pi}}_{\check{v}}$	$\hat{\boldsymbol{\pi}}_{\check{w}}$	$\hat{\mathbb{G}}$	$\check{\mathbb{H}}$	$\mathbb{Z}_p$
PPE	$\check{\boldsymbol{\pi}}_{\hat{v}}$	$\check{\boldsymbol{\pi}}_{\hat{w}}$	$\hat{\boldsymbol{\pi}}_{\check{v}}$	$\hat{\boldsymbol{\pi}}_{\check{w}}$	4	4	0
PEnc$_{\hat{\mathbb{G}}}$ $(\star)$	$\check{\boldsymbol{\pi}}_{\hat{v}}$	$\check{\mathbf{0}}$	$\hat{\boldsymbol{\pi}}_{\check{v}}$	$\hat{\boldsymbol{\pi}}_{\check{w}}$	4	2	0
PConst$_{\hat{\mathbb{G}}}$	$\check{\mathbf{0}}$	$\check{\mathbf{0}}$	$\boldsymbol{e}^\top(\hat{\boldsymbol{x}}\Gamma\boldsymbol{r}_y)$	$\boldsymbol{e}^\top(\hat{\boldsymbol{x}}\Gamma\boldsymbol{s}_y)$	2	0	0
PEnc$_{\check{\mathbb{H}}}$ $(\star)$	$\check{\boldsymbol{\pi}}_{\hat{v}}$	$\check{\boldsymbol{\pi}}_{\hat{w}}$	$\hat{\boldsymbol{\pi}}_{\check{v}}$	$\hat{\mathbf{0}}$	2	4	0
PConst$_{\check{\mathbb{H}}}$	$(\boldsymbol{r}_x\Gamma\check{\boldsymbol{y}})\boldsymbol{e}$	$(\boldsymbol{s}_x\Gamma\check{\boldsymbol{y}})\boldsymbol{e}$	$\hat{\mathbf{0}}$	$\hat{\mathbf{0}}$	0	2	0
ME$_{\hat{\mathbb{G}}}$	$\check{\boldsymbol{\pi}}_{\hat{v}}$	$\check{\boldsymbol{\pi}}_{\hat{w}}$	$\hat{\boldsymbol{\pi}}_{\check{v}}$	$\hat{\mathbf{0}}$	2	4	0
MEnc$_{\hat{\mathbb{G}}}$ $(\star)$	$\check{\boldsymbol{\pi}}_{\hat{v}}$	$\hat{\mathbf{0}}$	$\hat{\boldsymbol{\pi}}_{\check{v}}$	$\hat{\mathbf{0}}$	2	2	0
MConst$_{\hat{\mathbb{G}}}$	$\check{\mathbf{0}}$	$\check{\mathbf{0}}$	$\boldsymbol{e}^\top(\hat{\boldsymbol{x}}\Gamma\boldsymbol{r}_y)$	$\hat{\mathbf{0}}$	1	0	0
MLin$_{\hat{\mathbb{G}}}$	$(\boldsymbol{r}_x\Gamma\boldsymbol{y})\check{\boldsymbol{u}}$	$(\boldsymbol{s}_x\Gamma\boldsymbol{y})\check{\boldsymbol{u}}$	$\hat{\mathbf{0}}$	$\hat{\mathbf{0}}$	0	0	2
ME$_{\check{\mathbb{H}}}$	$\check{\boldsymbol{\pi}}_{\hat{v}}$	$\check{\mathbf{0}}$	$\hat{\boldsymbol{\pi}}_{\check{v}}$	$\hat{\boldsymbol{\pi}}_{\check{w}}$	4	2	0
MEnc$_{\check{\mathbb{H}}}$ $(\star)$	$\check{\boldsymbol{\pi}}_{\hat{v}}$	$\check{\mathbf{0}}$	$\hat{\boldsymbol{\pi}}_{\check{v}}$	$\hat{\mathbf{0}}$	2	2	0
MConst$_{\check{\mathbb{H}}}$	$(\boldsymbol{r}_x\Gamma\check{\boldsymbol{y}})\boldsymbol{e}$	$\check{\mathbf{0}}$	$\hat{\mathbf{0}}$	$\hat{\mathbf{0}}$	0	1	0
MLin$_{\check{\mathbb{H}}}$	$\check{\mathbf{0}}$	$\check{\mathbf{0}}$	$\hat{\boldsymbol{u}}(\boldsymbol{x}\Gamma\boldsymbol{r}_y)$	$\hat{\boldsymbol{u}}(\boldsymbol{x}\Gamma\boldsymbol{s}_y)$	0	0	2
QE	$\check{\boldsymbol{\pi}}_{\hat{v}}$	$\check{\mathbf{0}}$	$\hat{\boldsymbol{\pi}}_{\check{v}}$	$\hat{\mathbf{0}}$	2	2	0
QConst$_{\hat{\mathbb{G}}}$	$\check{\mathbf{0}}$	$\check{\mathbf{0}}$	$\hat{\boldsymbol{u}}(\boldsymbol{x}\Gamma\boldsymbol{r}_y)$	$\hat{\mathbf{0}}$	0	0	1
QConst$_{\check{\mathbb{H}}}$	$(\boldsymbol{r}_x\Gamma\boldsymbol{y})\check{\boldsymbol{u}}$	$\check{\mathbf{0}}$	$\hat{\mathbf{0}}$	$\hat{\mathbf{0}}$	0	0	1

Fig. 9: Size of an NIZK proof for each type of equation. The size of the NIZK proofs are the same as in [GS12], except for the equation types marked with $(\star)$, which were not defined in [GS12].

Theorem 1. *The commit-and-prove scheme given in Figs. 1,3,4 and 8 has perfect correctness, perfect soundness and F-knowledge for the function F defined above, and computational composable zero-knowledge if the SXDH assumption holds relative to $\mathcal{G}$.*

The description of the zero-knowledge simulator is given in Sec. 6. The proof of the theorem follows from Lemmas 1, 2 and 3 given in Sec. 7.

6 Zero-Knowledge simulator

Simulation. The simulator gets as input a well-formed statement where the equation and message types match. Unlike the prover, the simulator does not know the witness. Some of the simulator's inputs x_i and y_j will correspond to public values, base elements and units, while others have been generated with the simulated commitments and are 0. Having the secret simulation key the simulator can equivocate the base$_{\hat{\mathbb{G}}}$, unit$_{\hat{\mathbb{G}}}$ and base$_{\check{\mathbb{H}}}$, unit$_{\check{\mathbb{H}}}$ type commitments to 0. This enables it to get a satisfying witness for the statement and using this simulated witness it can now create the proof as an honest prover would do. The simulator algorithm is detailed in Fig. 10.

7 Security proofs for the commit-and-prove scheme

Lemma 1. *The commit-and-prove scheme has perfect correctness.*

Proof. The prover gets a statement and valid witness $(ck, (T, \Gamma), \{(t_{x_i}, x_i)\}_{i=1}^m, \{(t_{y_n}, y_n)\}_{j=1}^n) \in R_L$, which guarantees the statement is correctly formatted, $(t_{x_i}, x_i), (t_{y_j}, y_j) \in \mathcal{M}_{ck}$ and the vectors $\boldsymbol{x} = (x_1, \ldots, x_m)$ and $\boldsymbol{y} = (y_1, \ldots, y_n)^\top$ fall into one out of four cases given below.

Case	Implication
$\boldsymbol{x} \in \hat{\mathbb{G}}^{1\times m}, \boldsymbol{y} \in \check{\mathbb{H}}^{n\times 1}$ and $\boldsymbol{x}\Gamma\boldsymbol{y} = 0_{\mathbb{T}}$	$\boldsymbol{e}^\top\boldsymbol{x}\Gamma\boldsymbol{y}\boldsymbol{e} = 0_{\mathbb{T}}$
$\boldsymbol{x} \in \hat{\mathbb{G}}^{1\times m}, \boldsymbol{y} \in \mathbb{Z}_p^{n\times 1}$ and $\boldsymbol{x}\Gamma\boldsymbol{y} = \hat{0}$	$\boldsymbol{e}^\top\boldsymbol{x}\Gamma\boldsymbol{y}\check{\boldsymbol{u}} = 0_{\mathbb{T}}$
$\boldsymbol{x} \in \mathbb{Z}_p^{1\times m}, \boldsymbol{y} \in \check{\mathbb{H}}^{n\times 1}$ and $\boldsymbol{x}\Gamma\boldsymbol{y} = \check{0}$	$\hat{\boldsymbol{u}}\boldsymbol{x}\Gamma\boldsymbol{y}\boldsymbol{e} = 0_{\mathbb{T}}$
$\boldsymbol{x} \in \mathbb{Z}_p^{1\times m}, \boldsymbol{y} \in \mathbb{Z}_p^{n\times 1}$ and $\boldsymbol{x}\Gamma\boldsymbol{y} = 0$	$\hat{\boldsymbol{u}}\boldsymbol{x}\Gamma\boldsymbol{y}\check{\boldsymbol{u}} = 0_{\mathbb{T}}$

The prover's input also satisfies $(t_{x_i}, (r_{x_i}, s_{x_i})), (t_{y_j}, (r_{y_j}, s_{y_j})) \in \mathcal{R}_{ck}$. For $\boldsymbol{x} \in \mathbb{Z}_p^m$ this gives us $\boldsymbol{s}_x = \mathbf{0}$ and we can write $\hat{C} = \hat{\boldsymbol{u}}\boldsymbol{x} + \hat{\boldsymbol{v}}\boldsymbol{r}_x + \hat{\boldsymbol{w}}\boldsymbol{s}_x$. Similarly, for $\boldsymbol{y} \in \mathbb{Z}_p^n$ it gives us $\boldsymbol{s}_y = \mathbf{0}$ and we can write $\check{D} = \boldsymbol{y}\check{\boldsymbol{u}} + \boldsymbol{r}_y\check{\boldsymbol{v}} + \boldsymbol{s}_y\check{\boldsymbol{w}}$.

This means that for each of the four cases

$$(\hat{C} - \hat{\boldsymbol{v}}\boldsymbol{r}_x - \hat{\boldsymbol{w}}\boldsymbol{s}_x)\Gamma(\check{D} - \boldsymbol{r}_y\check{\boldsymbol{v}} - \boldsymbol{s}_y\check{\boldsymbol{w}}) = 0_{\mathbb{T}}.$$

$\mathrm{SimProve}_{tk}(T, \mathit{\Gamma}, \{(t_{x_i}, (x_i, r_{x_i}, s_{x_i}))\}_{i=1}^m, \{(t_{y_j}, (y_j, r_{y_j}, s_{y_j}))\}_{j=1}^n)$

Modify some values according to the following tables

T	t_{x_i}	x_i, r_{x_i}, s_{x_i}
PPE, $\mathrm{PEnc}_{\check{\mathbb{H}}}$, $\mathrm{PConst}_{\check{\mathbb{H}}}$, $\mathrm{MLin}_{\hat{\mathbb{G}}}$	$\mathrm{base}_{\hat{\mathbb{G}}}$	$x_i \leftarrow \hat{0}, r_{x_i} \leftarrow \rho, s_{x_i} \leftarrow -1$
$\mathrm{ME}_{\check{\mathbb{H}}}$, $\mathrm{MEnc}_{\check{\mathbb{H}}}$, $\mathrm{MConst}_{\check{\mathbb{H}}}$, QE, $\mathrm{QConst}_{\check{\mathbb{H}}}$	$\mathrm{unit}_{\hat{\mathbb{G}}}$	$x_i \leftarrow 0, r_{x_i} \leftarrow \rho, s_{x_i} \leftarrow 0$

T	t_{y_j}	y_j, r_{y_j}, s_{y_j}
PPE, $\mathrm{PEnc}_{\hat{\mathbb{G}}}$, $\mathrm{PConst}_{\hat{\mathbb{G}}}$, $\mathrm{MLin}_{\check{\mathbb{H}}}$	$\mathrm{base}_{\check{\mathbb{H}}}$	$y_j \leftarrow \check{0}, r_{y_j} \leftarrow \sigma, s_{y_j} \leftarrow -1$
$\mathrm{ME}_{\hat{\mathbb{G}}}$, $\mathrm{MEnc}_{\hat{\mathbb{G}}}$, $\mathrm{MConst}_{\hat{\mathbb{G}}}$, QE, $\mathrm{QConst}_{\hat{\mathbb{G}}}$	$\mathrm{unit}_{\check{\mathbb{H}}}$	$y_j \leftarrow 0, r_{y_j} \leftarrow \sigma, s_{y_j} \leftarrow 0$

```
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 = $ PPE pick $\alpha, \beta, \gamma, \delta \leftarrow \mathbb{Z}_p$
If $T \in \{\mathrm{PEnc}_{\hat{\mathbb{G}}}, \mathrm{ME}_{\check{\mathbb{H}}}\}$ pick $\alpha, \beta \leftarrow \mathbb{Z}_p$
If $T \in \{\mathrm{PEnc}_{\check{\mathbb{H}}}, \mathrm{ME}_{\hat{\mathbb{G}}}\}$ pick $\alpha, \gamma \leftarrow \mathbb{Z}_p$
If $T \in \{\mathrm{MEnc}_{\hat{\mathbb{G}}}, \mathrm{MEnc}_{\check{\mathbb{H}}}, \mathrm{QE}\}$ pick $\alpha \leftarrow \mathbb{Z}_p$
    $\check{\boldsymbol{\pi}}_{\hat{v}} \leftarrow \boldsymbol{r}_x \mathit{\Gamma} \check{D} + \alpha\check{\boldsymbol{v}} + \beta\check{\boldsymbol{w}}$    $\hat{\boldsymbol{\pi}}_{\check{v}} \leftarrow (\hat{C} - \hat{\boldsymbol{v}}\boldsymbol{r}_x - \hat{\boldsymbol{w}}\boldsymbol{s}_x)\mathit{\Gamma}\boldsymbol{r}_y - \hat{\boldsymbol{v}}\alpha - \hat{\boldsymbol{w}}\gamma$
    $\check{\boldsymbol{\pi}}_{\hat{w}} \leftarrow \boldsymbol{s}_x \mathit{\Gamma} \check{D} + \gamma\check{\boldsymbol{v}} + \delta\check{\boldsymbol{w}}$    $\hat{\boldsymbol{\pi}}_{\check{w}} \leftarrow (\hat{C} - \hat{\boldsymbol{v}}\boldsymbol{r}_x - \hat{\boldsymbol{w}}\boldsymbol{s}_x)\mathit{\Gamma}\boldsymbol{s}_y - \hat{\boldsymbol{v}}\beta - \hat{\boldsymbol{w}}\delta$
Return $\pi = (\check{\boldsymbol{\pi}}_{\hat{v}}, \check{\boldsymbol{\pi}}_{\hat{w}}, \hat{\boldsymbol{\pi}}_{\check{v}}, \hat{\boldsymbol{\pi}}_{\check{w}})$
```

Fig. 10: Simulator algorithm

This fact gives us that the verification equation used by the verifier holds.

$$\begin{aligned}
\hat{C}\mathit{\Gamma}\check{D} &= (\hat{C} - \hat{\boldsymbol{v}}\boldsymbol{r}_x - \hat{\boldsymbol{w}}\boldsymbol{s}_x)\mathit{\Gamma}\check{D} + \hat{\boldsymbol{v}}\boldsymbol{r}_x\mathit{\Gamma}\check{D} + \hat{\boldsymbol{w}}\boldsymbol{s}_x\mathit{\Gamma}\check{D} \\
&= (\hat{C} - \hat{\boldsymbol{v}}\boldsymbol{r}_x - \hat{\boldsymbol{w}}\boldsymbol{s}_x)\mathit{\Gamma}(\check{D} - \boldsymbol{r}_y\check{\boldsymbol{v}} - \boldsymbol{s}_y\check{\boldsymbol{w}}) + (\hat{C} - \hat{\boldsymbol{v}}\boldsymbol{r}_x - \hat{\boldsymbol{w}}\boldsymbol{s}_x)\mathit{\Gamma}\boldsymbol{r}_y\check{\boldsymbol{v}} + (\hat{C} - \hat{\boldsymbol{v}}\boldsymbol{r}_x - \hat{\boldsymbol{w}}\boldsymbol{s}_x)\mathit{\Gamma}\boldsymbol{s}_y\check{\boldsymbol{w}} \\
&\quad + \hat{\boldsymbol{v}}\boldsymbol{r}_x\mathit{\Gamma}\check{D} + \hat{\boldsymbol{w}}\boldsymbol{s}_x\mathit{\Gamma}\check{D} + \hat{\boldsymbol{v}}(\alpha - \alpha)\check{\boldsymbol{v}} + \hat{\boldsymbol{v}}(\beta - \beta)\check{\boldsymbol{w}} + \hat{\boldsymbol{w}}(\gamma - \gamma)\check{\boldsymbol{v}} + \hat{\boldsymbol{w}}(\delta - \delta)\check{\boldsymbol{w}} \\
&= 0_{\mathbb{T}} + \Big((\hat{C} - \hat{\boldsymbol{v}}\boldsymbol{r}_x - \hat{\boldsymbol{w}}\boldsymbol{s}_x)\mathit{\Gamma}\boldsymbol{r}_y - \hat{\boldsymbol{v}}\alpha - \hat{\boldsymbol{w}}\gamma\Big)\check{\boldsymbol{v}} + \Big((\hat{C} - \hat{\boldsymbol{v}}\boldsymbol{r}_x - \hat{\boldsymbol{w}}\boldsymbol{s}_x)\mathit{\Gamma}\boldsymbol{s}_y - \hat{\boldsymbol{v}}\beta - \hat{\boldsymbol{w}}\delta\Big)\check{\boldsymbol{w}} \\
&\quad + \hat{\boldsymbol{v}}(\boldsymbol{r}_x\mathit{\Gamma}\check{D} + \alpha\check{\boldsymbol{v}} + \beta\check{\boldsymbol{w}}) + \hat{\boldsymbol{w}}(\boldsymbol{s}_x\mathit{\Gamma}\check{D} + \gamma\check{\boldsymbol{v}} + \delta\check{\boldsymbol{w}}) \\
&= \hat{\boldsymbol{\pi}}_{\check{v}}\check{\boldsymbol{v}} + \hat{\boldsymbol{\pi}}_{\check{w}}\check{\boldsymbol{w}} + \hat{\boldsymbol{v}}\check{\boldsymbol{\pi}}_{\hat{v}} + \hat{\boldsymbol{w}}\check{\boldsymbol{\pi}}_{\hat{w}}.
\end{aligned}$$

□

Lemma 2. *The commit-and-prove scheme has perfect soundness and perfect F-knowledge.*

Proof. The verifier checks that the statement is correctly formatted, the commitments are valid and the proof is well-formed. Finally, the verifier checks that the commitments organized into matrices $\hat{C}$ and $\check{D}$ satisfy

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

Recall ExtGen returns $\boldsymbol{\xi}, \boldsymbol{\psi}$ 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 by $\boldsymbol{\xi}$ and $\boldsymbol{\psi}$ on the left and right side respectively, we get

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

Observe, $\hat{\boldsymbol{x}} = \boldsymbol{\xi}\hat{C}$ are the values the extractor Ext_{xk} gets from the commitments $(t_{x_1}, \hat{\boldsymbol{c}}_1), \ldots, (t_{x_m}, \hat{\boldsymbol{c}}_n)$ and $\check{\boldsymbol{y}} = \check{D}\boldsymbol{\psi}$ are the values the extractor Ext_{xk} gets from the commitments $(t_{y_1}, \check{\boldsymbol{d}}_1), \ldots, (t_{y_n}, \check{\boldsymbol{d}}_n)$. The extracted values from the commitments therefore satisfy

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

If the equation is a pairing product equation, this directly gives us both perfect soundness and perfect F-knowledge.

If the equation is a multi-scalar multiplication equation in $\check{\mathbb{H}}$, then the committed value is $\boldsymbol{x} = \log_{\hat{g}} \hat{\boldsymbol{x}}$. Then the opened values satisfy $\boldsymbol{x}\mathit{\Gamma}\check{\boldsymbol{y}} = \check{0}$, which gives us perfect soundness and perfect F-knowledge. The case of a multi-scalar multiplication in $\hat{\mathbb{G}}$ is similar. Finally, for quadratic equations over $\mathbb{Z}_p$ the committed values are $\boldsymbol{x} = \log_{\hat{g}} \hat{\boldsymbol{x}}$, $\boldsymbol{y} = \log_{\check{h}} \check{\boldsymbol{y}}$ and they satisfy $\boldsymbol{x}\mathit{\Gamma}\boldsymbol{y} = 0$, which gives us perfect soundness and perfect F-knowledge. □

Lemma 3. *The commit-and-prove scheme is composable zero-knowledge if the SXDH assumption holds relative to $\mathcal{G}$.*

Proof. We first recall what the Gen and SimGen algorithms are. Both algorithms output a commitment key which specifies a prime order bilinear group $(p, \hat{\mathbb{G}}, \check{\mathbb{H}}, \mathbb{T}, e, \hat{g}, \check{h})$ and elements $\hat{\boldsymbol{v}}, \hat{\boldsymbol{w}}, \hat{\boldsymbol{u}}, \check{\boldsymbol{v}}, \check{\boldsymbol{w}}, \check{\boldsymbol{u}}$. The only difference between Gen and SimGen (besides the fact that SimGen outputs a simulation trapdoor) is that in Gen, the elements are of the form $\hat{\boldsymbol{v}} = (\hat{g}, \xi\hat{g})^\top$, $\check{\boldsymbol{v}} = (\check{h}, \check{h}\psi)$, $\hat{\boldsymbol{w}} = \rho\hat{\boldsymbol{v}}$, $\check{\boldsymbol{w}} = \sigma\check{\boldsymbol{v}}$, $\hat{\boldsymbol{u}} = \rho\hat{\boldsymbol{v}} + \boldsymbol{e}^\top\hat{g}$, $\check{\boldsymbol{u}} = \sigma\check{\boldsymbol{v}} + \boldsymbol{e}\check{h}$, whereas in SimGen the elements are of the form $\hat{\boldsymbol{v}} = (\hat{g}, \xi\hat{g})^\top$, $\check{\boldsymbol{v}} = (\check{h}, \psi\check{h})$, $\hat{\boldsymbol{w}} = \rho\hat{\boldsymbol{v}} - \boldsymbol{e}^\top\hat{g}$, $\check{\boldsymbol{w}} = \sigma\hat{\boldsymbol{v}} - \boldsymbol{e}\check{h}$, $\hat{\boldsymbol{u}} = \rho\hat{\boldsymbol{v}}$, $\check{\boldsymbol{u}} = \sigma\check{\boldsymbol{v}}$.

Under the SXDH assumption, the distribution of the elements $\hat{\boldsymbol{v}}, \hat{\boldsymbol{w}}, \hat{\boldsymbol{u}}, \check{\boldsymbol{v}}, \check{\boldsymbol{w}}, \check{\boldsymbol{u}}$ generated by Gen is computationally indistinguishable from the distribution of the elements $\hat{\boldsymbol{v}}, \hat{\boldsymbol{w}}, \hat{\boldsymbol{u}}, \check{\boldsymbol{v}}, \check{\boldsymbol{w}}, \check{\boldsymbol{u}}$ generated by SimGen. This shows that the first requirement of composable zero-knowledge is satisfied.

For the second requirement, we first give a high level description of the simulator in Fig. 10.

- For elements in the witness such that $t \in \{\mathrm{com}_{\hat{\mathbb{G}}}, \mathrm{sca}_{\hat{\mathbb{G}}}, \mathrm{com}_{\check{\mathbb{H}}}, \mathrm{sca}_{\check{\mathbb{H}}}\}$, the simulator will create commitments to $m = \hat{0}, m = \check{0}$ or $m = 0$ according to the type. These commitments are perfectly hiding under the commitment key output by SimGen.
- For elements in the witness such that $t \in \{\mathrm{base}_{\hat{\mathbb{G}}}, \mathrm{unit}_{\hat{\mathbb{G}}}, \mathrm{base}_{\check{\mathbb{H}}}, \mathrm{unit}_{\check{\mathbb{H}}}\}$, the simulator will use the simulation trapdoor tk to be able to treat them as commitments to $\hat{0}, \check{0}$ or 0 according to t.
- For elements in the witness such that $t \in \{\mathrm{enc}_{\hat{\mathbb{G}}}, \mathrm{enc}_{\check{\mathbb{H}}}\}$, the commitments are computed as encryptions of $\hat{0}$ and $\check{0}$ respectively.
- Having obtained satisfying openings of the commitments the simulator creates the proof as an honest prover would do with those values.

We now argue that a simulated proof is computationally indistinguishable from an honestly generated proof when the commitment key is output using SimGen. To show that, we define three games and use a hybrid argument.

- Game 1 is the game where the commitment key is output using SimGen and the proof is done using a valid witness.
- Game 2 is as Game 1, but instead of using a valid witness, commitments with $t \in \{\mathrm{com}_{\hat{\mathbb{G}}}, \mathrm{sca}_{\hat{\mathbb{G}}}, \mathrm{com}_{\check{\mathbb{H}}}, \mathrm{sca}_{\check{\mathbb{H}}}\}$ are done to $m = \hat{0}, m = \check{0}$ or $m = 0$, and commitments with $t \in \{\mathrm{base}_{\hat{\mathbb{G}}}, \mathrm{unit}_{\hat{\mathbb{G}}}, \mathrm{base}_{\check{\mathbb{H}}}, \mathrm{unit}_{\check{\mathbb{H}}}\}$ are equivocated as commitments to $\hat{0}, \check{0}$ or 0, according to t and T as described in Fig. 10.
- Game 3 is as Game 2, but commitments with $t \in \{\mathrm{enc}_{\hat{\mathbb{G}}}, \mathrm{enc}_{\check{\mathbb{H}}}\}$ are changed for encryptions of $\hat{0}, \check{0}$ according to t. Note that Game 3 is equivalent to the simulation.

Game 1 and Game 2 are perfectly indistinguishable due to the fact that commitments with $t \in \{\mathrm{com}_{\hat{\mathbb{G}}}, \mathrm{sca}_{\hat{\mathbb{G}}}, \mathrm{com}_{\check{\mathbb{H}}}, \mathrm{sca}_{\check{\mathbb{H}}}\}$ are perfectly hiding and the uniformity of the randomized proofs, which we prove in Lemma 4. Equivocating the commitments with $t \in \{\mathrm{base}_{\hat{\mathbb{G}}}, \mathrm{unit}_{\hat{\mathbb{G}}}, \mathrm{base}_{\check{\mathbb{H}}}, \mathrm{unit}_{\check{\mathbb{H}}}\}$ as commitments to $\hat{0}, \check{0}$ or 0 only changes the proofs and not the commitments themselves, so this case is also indistinguishable by Lemma 4.

Game 2 and Game 3 are computationally indistinguishable due to the IND-CPA security of ElGamal. Note that, in Game 2 and Game 3, the equation is satisfied for any value of the elements in the witness such that $t \in \{\mathrm{enc}_{\hat{\mathbb{G}}}, \mathrm{enc}_{\check{\mathbb{H}}}\}$. Actually, for each element $\hat{x}_i$ and $\check{y}_j$ we can just get the corresponding ciphertext $\hat{\boldsymbol{c}}_i$ or $\check{\boldsymbol{d}}_j$ and build the proof on top of it. We can do this because all the terms of the proof which have to be computed using the ciphertext's randomness will be 0 as the element paired to the randomness is going to be 0 due to the restrictions on the equation[5]. This allows us to reduce the computational indistinguishability of Game 2 and Game 3 to the IND-CPA security of ElGamal. □

Lemma 4. *Proofs are uniformly random over solutions to the verification equations when the commitment key is generated using* SimGen.

Proof. In all games from Game 1 to Game 3, all witnesses will yield proofs satisfying the verification equation by perfect completeness. Let us now argue that these proofs are uniformly random over the possible solutions to the verification equation. Observe that, as the commitment key is generated using SimGen, we have that $\hat{\boldsymbol{v}}, \hat{\boldsymbol{w}}$ generate

[5] Strictly speaking, if the proofs are computed using the algorithm given in Fig. 10, then this argument is only true for encryptions of elements in $\check{\mathbb{H}}$. However, the proofs can be written in an alternative way by changing the role of $\hat{\mathbb{G}}$ and $\check{\mathbb{H}}$ so that the same argument holds for encryptions of elements in $\hat{\mathbb{G}}$

$\hat{\mathbb{G}}^{1\times 2}$, $\check{\boldsymbol{v}}, \check{\boldsymbol{w}}$ generate $\check{\mathbb{H}}^{2\times 1}$, $\hat{\boldsymbol{u}} \in \langle \hat{\boldsymbol{v}} \rangle$ and $\check{\boldsymbol{u}} \in \langle \check{\boldsymbol{v}} \rangle$. From now on, let us define the pre-proofs as those proofs that satisfy the verification equation but haven't been randomized yet, for instance, $\hat{\boldsymbol{\pi}}'_{\check{v}} = \boldsymbol{e}^\top \hat{\boldsymbol{x}} \varGamma \boldsymbol{r}_y$.

We will prove either that there is only one proof which is accepted by the verification equation or that when there are many possible proofs both honestly generated and simulated proofs will be uniformly distributed among all valid proofs, i.e., among all solutions to the verification equations. In the first case, by using the techniques used in [GS12] it is easy to see that when $\check{\boldsymbol{\pi}}_{\hat{v}}, \check{\boldsymbol{\pi}}_{\hat{w}}$ (resp. $\hat{\boldsymbol{\pi}}_{\check{v}}, \hat{\boldsymbol{\pi}}_{\check{w}}$) are 0 there is only one value for $\hat{\boldsymbol{\pi}}_{\check{v}}, \hat{\boldsymbol{\pi}}_{\check{w}}$ (resp. $\check{\boldsymbol{\pi}}_{\hat{v}}, \check{\boldsymbol{\pi}}_{\hat{w}}$) which satisfies the verification equation. In the latter case, we will show that $\check{\boldsymbol{\pi}}_{\hat{v}}, \check{\boldsymbol{\pi}}_{\hat{w}}$ (resp. $\hat{\boldsymbol{\pi}}_{\check{v}}, \hat{\boldsymbol{\pi}}_{\check{w}}$) are uniformly random, and using the techniques from [GS12] it is easy to see that for fixed values for $\check{\boldsymbol{\pi}}_{\hat{v}}, \check{\boldsymbol{\pi}}_{\hat{w}}$ (resp. $\hat{\boldsymbol{\pi}}_{\check{v}}, \hat{\boldsymbol{\pi}}_{\check{w}}$) there is only one value for $\hat{\boldsymbol{\pi}}_{\check{v}}, \hat{\boldsymbol{\pi}}_{\check{w}}$ (resp. $\check{\boldsymbol{\pi}}_{\hat{v}}, \check{\boldsymbol{\pi}}_{\hat{w}}$) which satisfies the verification equation. If some of $\check{\boldsymbol{\pi}}'_{\hat{v}}, \check{\boldsymbol{\pi}}'_{\hat{w}}, \hat{\boldsymbol{\pi}}'_{\check{v}}, \hat{\boldsymbol{\pi}}'_{\check{w}}$ are already 0 due to the type of equation, we will not randomize them. The reason is that due to the restrictions on the equation types, if for a honestly generated proof some of $\check{\boldsymbol{\pi}}'_{\hat{v}}, \check{\boldsymbol{\pi}}'_{\hat{w}}, \hat{\boldsymbol{\pi}}'_{\check{v}}, \hat{\boldsymbol{\pi}}'_{\check{w}}$ are 0 then the zero-knowledge simulator will be able to construct simulated proofs such that the same parts of the proof are 0. Therefore, all valid proofs will be such that the same parts of the proof are 0, so these components will not need to be randomized.

For each type of equation, we now detail whether $\check{\boldsymbol{\pi}}_{\hat{v}}, \check{\boldsymbol{\pi}}_{\hat{w}}$ (or $\hat{\boldsymbol{\pi}}_{\check{v}}, \hat{\boldsymbol{\pi}}_{\check{w}}$) are 0 or whether $\check{\boldsymbol{\pi}}_{\hat{v}}, \check{\boldsymbol{\pi}}_{\hat{w}}$ (or $\hat{\boldsymbol{\pi}}_{\check{v}}, \hat{\boldsymbol{\pi}}_{\check{w}}$) are uniformly random, where we assume that $\alpha, \beta, \gamma, \delta \leftarrow \mathbb{Z}_p$:

For $T = \text{PPE}$, $\check{\boldsymbol{\pi}}_{\hat{v}} = \check{\boldsymbol{\pi}}'_{\hat{v}} + \alpha\check{\boldsymbol{v}} + \beta\check{\boldsymbol{w}}$, $\check{\boldsymbol{\pi}}_{\hat{w}} = \check{\boldsymbol{\pi}}'_{\hat{w}} + \gamma\check{\boldsymbol{v}} + \delta\check{\boldsymbol{w}}$ are uniformly distributed. For $T = \text{PEnc}_{\hat{\mathbb{G}}}$, $\check{\boldsymbol{\pi}}_{\hat{v}} = \check{\boldsymbol{\pi}}'_{\hat{v}} + \alpha\check{\boldsymbol{v}} + \beta\check{\boldsymbol{w}}$ is uniformly distributed and $\check{\boldsymbol{\pi}}_{\hat{w}} = 0$ (similarly for $T = \text{PEnc}_{\check{\mathbb{H}}}$). For $T = \text{PConst}_{\hat{\mathbb{G}}}$, $\check{\boldsymbol{\pi}}_{\hat{v}} = \check{\boldsymbol{\pi}}_{\hat{w}} = 0$ (similarly for $T = \text{PConst}_{\check{\mathbb{H}}}$).

For $T = \text{ME}_{\hat{\mathbb{G}}}$, $\hat{\boldsymbol{\pi}}_{\check{v}} = \hat{\boldsymbol{\pi}}'_{\check{v}} + \alpha\hat{\boldsymbol{v}} + \beta\hat{\boldsymbol{w}}$ is uniformly distributed and $\hat{\boldsymbol{\pi}}_{\check{w}} = 0$. For $T = \text{MEnc}_{\hat{\mathbb{G}}}$, $\check{\boldsymbol{\pi}}_{\hat{v}} = \check{\boldsymbol{\pi}}'_{\hat{v}} + \alpha\check{\boldsymbol{v}}$ is uniformly distributed and $\check{\boldsymbol{\pi}}_{\hat{w}} = 0$. For $T = \text{MConst}_{\hat{\mathbb{G}}}$, $\check{\boldsymbol{\pi}}_{\hat{v}}, \check{\boldsymbol{\pi}}_{\hat{w}} = 0$. For $T = \text{MLin}_{\hat{\mathbb{G}}}$, $\hat{\boldsymbol{\pi}}_{\check{v}}, \hat{\boldsymbol{\pi}}_{\check{w}} = 0$. (Similarly for $T \in \{\text{ME}_{\check{\mathbb{H}}}, \text{MEnc}_{\check{\mathbb{H}}}, \text{MConst}_{\check{\mathbb{H}}}, \text{MLin}_{\check{\mathbb{H}}}\}$).

For $T = \text{QE}$, $\hat{\boldsymbol{\pi}}_{\check{v}} = \hat{\boldsymbol{\pi}}'_{\check{v}} + \alpha\hat{\boldsymbol{v}}$ is uniformly distributed and $\hat{\boldsymbol{\pi}}_{\check{w}} = 0$. For $T = \text{QConst}_{\hat{\mathbb{G}}}$, $\check{\boldsymbol{\pi}}_{\hat{v}} = \check{\boldsymbol{\pi}}_{\hat{w}} = 0$ (similarly for $T = \text{QConst}_{\check{\mathbb{H}}}$). □

8 Applications of our results

In [BCKL08] the authors give a construction of a P-Signature scheme in asymmetric bilinear groups. This construction uses Groth-Sahai proofs based on the SXDH instantiation. The statements that have to be proven are such that our techniques can be applied to have better efficiency. Concretely, in the construction given in the paper each P-signature proof consists of 18 elements in $\hat{\mathbb{G}}$ and 16 elements in $\check{\mathbb{H}}$. Using our techniques, we can get 16 elements in $\hat{\mathbb{G}}$ and 16 elements in $\check{\mathbb{H}}$. In the full version of [BCKL08], they give another construction in which our techniques could be applied too, bringing the 12 elements in $\hat{\mathbb{G}}$ and 10 elements in $\check{\mathbb{H}}$ that they need to 10 elements in $\hat{\mathbb{G}}$ and 10 elements in $\check{\mathbb{H}}$. In addition, in the equations that appear in their scheme, the constant terms in $\mathbb{T}$ that appear are of the form $e(\hat{g}, \check{h})$. Using Groth-Sahai proofs, to get NIZK proofs they would need to add extra commitments and proofs. This is why they do not use NIZK proofs but NIWI proofs. However, using our techniques the same proofs are actually NIZK proofs, without increasing the number of elements needed. It would be interesting to see if this would result in more efficient P-signature schemes.

Another example is [AFG$^+$10], which defines structure-preserving signatures, i.e., signature schemes where the messages, signatures and verification keys are group elements and where the verification predicate is a conjunction of pairing-product equations. This definition makes structure-preserving signatures particularly well-suited for the application of Groth-Sahai proofs. One of their applications of structure-preserving signatures given in [AFG$^+$10] is a blind signature scheme in which Groth-Sahai proofs are used. To obtain a blind signature, the user commits to several elements and proves in zero-knowledge that some relations are satisfied. These are relations in which we can apply our techniques to get better efficiency. Concretely, the output of the Obtain protocol of the blind signature scheme consists of 18 elements in $\hat{\mathbb{G}}$ and 16 elements in $\check{\mathbb{H}}$, and using our techniques the output of the protocol can be reduced to 16 elements in $\hat{\mathbb{G}}$ and 14 elements in $\check{\mathbb{H}}$.

9 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. The public key has to be created as a perfectly binding key, otherwise the soundness of the NIZK proof could be easily broken by the prover. The prover will then have to prove to the verifier that her public key has been created as a perfectly binding key. To do that the prover will give an NIZK proof using the shared common reference string. The definitions for commit-and-prove schemes with prover-chosen CRS are given in Sec. 9.1. In Secs. 9.2 to 9.4 we explain how the prover creates her public key, proves its well-formedness and we detail what the efficiency improvement obtained is. Finally, in Sec. 9.5 we prove the security of our scheme.

9.1 Definitions for prover-chosen CRS commit-and-prove schemes

The definitions given in Sec. 2 are not suitable for the scenario where the prover chooses her own common reference string . The reason is that the prover will only communicate her public key pk to the verifier once, and the public key pk will be used to generate and verify commitments and proofs.

Therefore, we define extended commit-and-prove schemes for a relation R_L, which capture the fact that the prover chooses her own public key. We will only consider the common reference string model, where the prover makes a well-formedness NIZK proof for her pk by using the shared common reference string between the prover and the verifier. In particular, we are not considering the multi-string model [GO07] nor the scenario where the prover gets a certificate on her public key. The definitions can be naturally extended to those settings.

The definitions share many elements with the definitions given in Sec. 2. 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 also divide each committed value into two parts $w_i = (t_i, m_i)$.

An extended commit-and-prove scheme $\mathrm{ECP} = (\mathrm{Gen}, \mathrm{CreatePK}, \mathrm{VerifyPK}, \mathrm{Com}, \mathrm{Prove}, \mathrm{Verify})$ consists of six algorithms. The algorithms $\mathrm{Gen}, \mathrm{CreatePK}, \mathrm{Prove}$ are probabilistic and the algorithms $\mathrm{VerifyPK}, \mathrm{Com}, \mathrm{Verify}$ 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.

$\mathrm{CreatePK}(ck)$**:** Given a commitment key ck, returns a public key pk, a secret key sk and a well-formedness proof π_{pk}.

$\mathrm{VerifyPK}_{ck}(pk, \pi_{pk})$**:** Given a commitment key ck, a public key pk and a well-formedness proof π_{pk} returns 1 (accept) or 0 (reject).

$\mathrm{Com}_{ck,sk}(t, m; r)$**:** Given a commitment key ck, a secret key sk, 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 $(t, c) \in \mathcal{C}_{ck}$.

$\mathrm{Prove}_{ck,sk}(x, (t_1, m_1, r_1), \ldots, (t_n, m_n, r_n))$**:** Given a commitment key ck, a secret key sk, a 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.

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

Definition 6 (Perfect correctness). *The extended commit-and-prove system* ECP *is (perfectly) correct if for all adversaries* $\mathcal{A}$

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

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* $(ck, x, w_1, \ldots, w_n) \in R_L$.

Definition 7 (Perfect soundness). *The extended commit-and-prove system* ECP *is (perfectly) sound if there exists a deterministic (unbounded) opening algorithm* Open *such that for all adversaries* $\mathcal{A}$

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

Perfect F-knowledge is defined exactly as for commit-and-prove schemes. This means that we will be using a global extraction key asscoiated with ck for all commitments regardless of which public keys are used to generate the commitments.

Definition 8 (Composable zero-knowledge). *The extended commit-and-prove system* ECP *is (computationally) composable zero-knowledge if there exist probabilistic polynomial time algorithms* SimGen, SimCreatePK, SimCom, SimProve *such that for all non-uniform polynomial time stateful interactive adversaries* $\mathcal{A}$

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

and

$$\begin{aligned} &\Pr\left[(ck, tk) \leftarrow \mathrm{SimGen}(1^k); (pk, sk, \pi_{pk}) \leftarrow \mathrm{CreatePK}(ck) : \mathcal{A}(ck, tk, pk, \pi_{pk}) = 1\right] \\ \approx\ &\Pr\left[(ck, tk) \leftarrow \mathrm{SimGen}(1^k); (pk, sk, \pi_{pk}) \leftarrow \mathrm{SimCreatePK}(tk) : \mathcal{A}(ck, tk, pk, \pi_{pk}) = 1\right], \end{aligned}$$

and

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

where

- $\mathrm{Com}_{ck,sk}(\cdot)$ *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* $c_i = \mathrm{Com}_{ck,sk}(w_i; r_i)$
- $\mathrm{SimCom}_{ck,sk}(\cdot)$ *on* $w_i = (t_i, m_i) \in \mathcal{M}_{ck}$ *runs* $(c_i, s_i) \leftarrow \mathrm{SimCom}_{ck,sk}(t_i)$ *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* $(ck, x, w_{i_1}, \ldots, w_{i_n}) \in R_L$

9.2 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)
$\rho_P \leftarrow \mathbb{Z}_p$                        $\sigma_P \leftarrow \mathbb{Z}_p$
$\hat{\boldsymbol{v}}_P \leftarrow \hat{\boldsymbol{v}}$                        $\check{\boldsymbol{v}}_P \leftarrow \check{\boldsymbol{v}}$
$\hat{\boldsymbol{w}}_P \leftarrow \rho_P \hat{\boldsymbol{v}}_P$                 $\check{\boldsymbol{w}}_P \leftarrow \sigma_P \check{\boldsymbol{v}}_P$
$\hat{\boldsymbol{u}}_P \leftarrow \hat{\boldsymbol{w}}_P + (\hat{0}, \hat{g})^\top$      $\check{\boldsymbol{u}}_P \leftarrow \check{\boldsymbol{w}}_P + (\check{0}, \check{h})$
$pk \leftarrow (\hat{\boldsymbol{u}}_P, \hat{\boldsymbol{v}}_P, \hat{\boldsymbol{w}}_P, \check{\boldsymbol{u}}_P, \check{\boldsymbol{v}}_P, \check{\boldsymbol{w}}_P)$
$sk \leftarrow (pk, \rho_P, \sigma_P)$
Return $(pk, sk)$
```

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

Fig. 11: Public key generator algorithms

As shown in Fig. 11, 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 $\mathrm{MConst}_{\hat{\mathbb{G}}}$ involving public elements in $\hat{\mathbb{G}}$ and a secret ρ_P committed in $\check{\mathbb{H}}$, and two equations of type $\mathrm{MConst}_{\check{\mathbb{H}}}$ 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 Fig. 12 we give the exact NIZK proofs that have to be computed. The total cost of communicating the public key, which is determined by $\hat{\boldsymbol{w}}_P, \check{\boldsymbol{w}}_P$, the two commitments to ρ_P and σ_P and the four 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.

9.3 Proving the well-formedness of a prover-chosen CRS

As explained in Sec. 9.2, once the prover has chosen its own CRS a proof of well-formedness has to be given in order to guarantee soundness. This is, the prover has to show that the values $\hat{\boldsymbol{v}}_P, \hat{\boldsymbol{w}}_P, \check{\boldsymbol{v}}_P, \check{\boldsymbol{w}}_P$ satisfying

$$\hat{\boldsymbol{w}}_P - \hat{\boldsymbol{v}}_P \rho_P = \hat{\mathbf{0}} \qquad \text{and} \qquad \check{\boldsymbol{w}}_P - \sigma_P \check{\boldsymbol{v}}_P = \check{\mathbf{0}},$$

for some secret values ρ_P, σ_P. To prove that these relations are satisfied, the prover can use Groth-Sahai proofs. In Fig. 12 we detail what proofs need to be done by the prover to prove the well-formedness of her pk, writing J^{ij} for the 2×2 matrix such that all its elements are 0 except the element in row i and column j, which takes the value 1. Note that these are ordinary Groth-Sahai proofs, which can be verified using the commitment key ck.

```
$\mathrm{ProvePublicKey}_{ck}(pk, sk)$
Parse $\hat{\boldsymbol{v}}_P, \hat{\boldsymbol{w}}_P, \check{\boldsymbol{v}}_P, \check{\boldsymbol{w}}_P$ as $((\hat{\boldsymbol{v}}_P)_1, (\hat{\boldsymbol{v}}_P)_2 = \hat{g}), ((\hat{\boldsymbol{w}}_P)_1, (\hat{\boldsymbol{w}}_P)_2), ((\check{\boldsymbol{v}}_P)_1, (\check{\boldsymbol{v}}_P)_2 = \check{h}), ((\check{\boldsymbol{w}}_P)_1, (\check{\boldsymbol{w}}_P)_2))$
Compute the commitments $\check{d}_\rho \leftarrow \rho_P \check{\boldsymbol{u}}_P + r_\rho \check{\boldsymbol{v}}; \hat{c}_\sigma \leftarrow \hat{\boldsymbol{u}}_P \sigma_P + \hat{\boldsymbol{v}} r_\sigma;$ where $r_\rho, r_\sigma \leftarrow \mathbb{Z}_p$
Define the following variables and types:
$r_{x_1}, s_{x_1}, r_{x_2}, s_{x_2}, r_{y_1}, s_{y_1}, s_{y_2} \leftarrow 0;\ r_{y_2} \leftarrow r_\rho$
$t_{x_1} = (\mathrm{pub}_{\hat{\mathbb{G}}}, (\hat{\boldsymbol{w}}_P)_1),\ x_1 = (\hat{\boldsymbol{w}}_P)_1,\ t_{x_2} = (\mathrm{pub}_{\hat{\mathbb{G}}}, (\hat{\boldsymbol{v}}_P)_1),\ x_2 = (\hat{\boldsymbol{v}}_P)_1$
$t_{y_1} = \mathrm{unit}_{\check{\mathbb{H}}},\quad y_1 = 1,\quad t_{y_2} = \mathrm{sca}_{\check{\mathbb{H}}},\quad y_2 = \rho_P$
$\pi_1 \leftarrow \mathrm{Prove}_{ck}(\mathrm{MConst}_{\hat{\mathbb{G}}}, (J^{11} - J^{22}), \{(t_{x_i}, x_i, (r_{x_i}, s_{x_i}))\}_{i=1}^2, \{(t_{y_j}, y_j, (r_{y_j}, s_{y_j}))\}_{j=1}^2)$
Redefine the following variables and types:
$r_{x_1}, s_{x_1}, r_{x_2}, s_{x_2}, r_{y_1}, s_{y_1}, s_{y_2} \leftarrow 0;\ r_{y_2} \leftarrow r_\rho$
$t_{x_1} = (\mathrm{pub}_{\hat{\mathbb{G}}}, (\hat{\boldsymbol{w}}_P)_2),\ x_1 = (\hat{\boldsymbol{w}}_P)_2,\ t_{x_2} = \mathrm{base}_{\hat{\mathbb{G}}},\ x_2 = \hat{g}$
$t_{y_1} = \mathrm{unit}_{\check{\mathbb{H}}},\quad y_1 = 1,\quad t_{y_2} = \mathrm{sca}_{\check{\mathbb{H}}},\ y_2 = \rho_P$
$\pi_2 \leftarrow \mathrm{Prove}_{ck}(\mathrm{MConst}_{\hat{\mathbb{G}}}, (J^{11} - J^{22}), \{(t_{x_i}, x_i, (r_{x_i}, s_{x_i}))\}_{i=1}^2, \{(t_{y_j}, y_j, (r_{y_j}, s_{y_j}))\}_{j=1}^2)$
Redefine the following variables and types:
$r_{x_1}, s_{x_1}, s_{x_2}, r_{y_1}, s_{y_1}, r_{y_2}, s_{y_2} \leftarrow 0;\ r_{x_2} \leftarrow r_\sigma$
$t_{x_1} = \mathrm{unit}_{\hat{\mathbb{G}}},\quad x_1 = 1,\quad t_{x_2} = \mathrm{sca}_{\hat{\mathbb{G}}},\quad x_2 = \sigma_P$
$t_{y_1} = (\mathrm{pub}_{\check{\mathbb{H}}}, (\check{\boldsymbol{w}}_P)_1),\ y_1 = (\check{\boldsymbol{w}}_P)_1,\ t_{y_2} = (\mathrm{pub}_{\check{\mathbb{H}}}, (\check{\boldsymbol{v}}_P)_1),\ y_2 = (\check{\boldsymbol{v}}_P)_1$
$\pi_3 \leftarrow \mathrm{Prove}_{ck}(\mathrm{MConst}_{\check{\mathbb{H}}}, (J^{11} - J^{22}), \{(t_{x_i}, x_i, (r_{x_i}, s_{x_i}))\}_{i=1}^2, \{(t_{y_j}, y_j, (r_{y_j}, s_{y_j}))\}_{j=1}^2)$
Redefine the following variables and types:
$r_{x_1}, s_{x_1}, s_{x_2}, r_{y_1}, s_{y_1}, r_{y_2}, s_{y_2} \leftarrow 0;\ r_{x_2} \leftarrow r_\sigma$
$t_{x_1} = \mathrm{unit}_{\hat{\mathbb{G}}},\quad x_1 = 1,\quad t_{x_2} = \mathrm{sca}_{\hat{\mathbb{G}}},\ x_2 = \sigma_P$
$t_{y_1} = (\mathrm{pub}_{\check{\mathbb{H}}}, (\check{\boldsymbol{w}}_P)_2),\ y_1 = (\check{\boldsymbol{w}}_P)_2,\ t_{y_2} = \mathrm{base}_{\check{\mathbb{H}}},\ y_2 = \check{h}$
$\pi_4 \leftarrow \mathrm{Prove}_{ck}(\mathrm{MConst}_{\check{\mathbb{H}}}, (J^{11} - J^{22}), \{(t_{x_i}, x_i, (r_{x_i}, s_{x_i}))\}_{i=1}^2, \{(t_{y_j}, y_j, (r_{y_j}, s_{y_j}))\}_{j=1}^2)$
Return $\pi_{pk} = (\hat{c}_\rho, \check{d}_\sigma, \pi_1, \pi_2, \pi_3, \pi_4)$
```

Fig. 12: Algorithm to prove well-formedness of the public key

9.4 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. 8, 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.

As shown in Fig. 13, 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%[6]. Furthermore, in most applications found in the literature there are only a few variables in the equations, which makes our improvements more significant.

	GS commitments		This work	
t	$\hat{\mathbb{G}}$	$\check{\mathbb{H}}$	$\hat{\mathbb{G}}$	$\check{\mathbb{H}}$
$\mathrm{com}_{\hat{\mathbb{G}}}$	4	0	2	0
$\mathrm{com}_{\check{\mathbb{H}}}$	0	4	0	2
$\mathrm{enc}_{\hat{\mathbb{G}}}$	Undefined		2	0
$\mathrm{enc}_{\check{\mathbb{H}}}$	Undefined		0	2
$\mathrm{sca}_{\hat{\mathbb{G}}}$	4	0	2	0
$\mathrm{sca}_{\check{\mathbb{H}}}$	0	4	0	2

	GS proofs		This work	
T	$\hat{\mathbb{G}}$	$\check{\mathbb{H}}$	$\hat{\mathbb{G}}$	$\check{\mathbb{H}}$
PPE	$4\lvert\hat{\boldsymbol{x}}\rvert + 8$	$2\lvert\check{\boldsymbol{y}}\rvert + 8$	$4\lvert\hat{\boldsymbol{x}}\rvert + 4$	$2\lvert\check{\boldsymbol{y}}\rvert + 4$
$\mathrm{PEnc}_{\hat{\mathbb{G}}}$	Undefined		$4\lvert\hat{\boldsymbol{x}}\rvert + 4$	$\lvert\check{\boldsymbol{y}}\rvert + 2$
$\mathrm{PConst}_{\hat{\mathbb{G}}}$	$2\lvert\hat{\boldsymbol{x}}\rvert$	0	$2\lvert\hat{\boldsymbol{x}}\rvert$	0
$\mathrm{PEnc}_{\check{\mathbb{H}}}$	Undefined		$2\lvert\hat{\boldsymbol{x}}\rvert + 2$	$2\lvert\check{\boldsymbol{y}}\rvert + 4$
$\mathrm{PConst}_{\check{\mathbb{H}}}$	0	$2\lvert\check{\boldsymbol{y}}\rvert$	0	$2\lvert\check{\boldsymbol{y}}\rvert$
$\mathrm{ME}_{\hat{\mathbb{G}}}$	$\lvert\hat{\boldsymbol{x}}\rvert + 4$	8	$\lvert\hat{\boldsymbol{x}}\rvert + 2$	4
$\mathrm{MEnc}_{\hat{\mathbb{G}}}$	Undefined		$\lvert\hat{\boldsymbol{x}}\rvert + 2$	2
$\mathrm{MConst}_{\hat{\mathbb{G}}}$	$\lvert\hat{\boldsymbol{x}}\rvert$	0	$\lvert\hat{\boldsymbol{x}}\rvert$	0
$\mathrm{MLin}_{\hat{\mathbb{G}}}$	0	0	0	0
$\mathrm{ME}_{\check{\mathbb{H}}}$	8	$\lvert\check{\boldsymbol{y}}\rvert + 4$	4	$\lvert\check{\boldsymbol{y}}\rvert + 2$
$\mathrm{MEnc}_{\check{\mathbb{H}}}$	Undefined		2	$\lvert\check{\boldsymbol{y}}\rvert + 2$
$\mathrm{MConst}_{\check{\mathbb{H}}}$	0	$\lvert\check{\boldsymbol{y}}\rvert$	0	$\lvert\check{\boldsymbol{y}}\rvert$
$\mathrm{MLin}_{\check{\mathbb{H}}}$	0	0	0	0
QE	4	4	2	2
$\mathrm{QConst}_{\hat{\mathbb{G}}}$	0	0	0	0
$\mathrm{QConst}_{\check{\mathbb{H}}}$	0	0	0	0

Fig. 13: Scalar multiplications needed for computing commitments and NIZK proofs

9.5 Security proofs for the prover-chosen CRS commit-and-prove scheme

Lemma 5. *The prover-chosen CRS commit-and-prove scheme has perfect correctness.*

Lemma 6. *The prover-chosen CRS commit-and-prove scheme has perfect soundness and perfect F-knowledge.*

Perfect correctness, perfect soundness and perfect F-knowledge can be proven in a similar way to how they are proven for our commit-and-prove scheme.

Lemma 7. *The prover-chosen CRS commit-and-prove scheme is composable zero-knowledge if the SXDH assumption holds relative to $\mathcal{G}$.*

Proof. (Sketch) The zero-knowledge simulator is quite straightforward given a zero-knowledge simulator for our commit-and-prove scheme. First, the simulator will create the public key using the algorithm $\mathrm{SimProverGen}$ given in Fig. 11. Then, instead of making NIZK proofs to show that the generated public key is binding, it will simulate these proofs using the trapdoor key tk associated to ck. After simulating the well-formedness of pk, the simulator is able simulate any proof by using the secret key sk.

It should be noted that the simulator is simulating an NIZK proof for a false statement as the public key generated is no longer perfectly binding. In principle, the definition of zero-knowledge only guarantees that simulated proofs are indistinguishable from honestly generated proofs as long as the statement is true. However, in our case the public key and the simulated proof of well-formedness are computationally indistinguishable from a binding public key generated with the algorithm $\mathrm{ProverGen}$ and an honestly generated well-formedness proof.

To see that, consider the following hybrid proof: the first game corresponds to the setting where the public key is generated using $\mathrm{ProverGen}$ and the well-formedness proof is not simulated. The second game corresponds to the same setting as the first game, but the well-formedness proof is simulated. Both games are computationally indistinguishable due to the zero-knowledge property of Groth-Sahai proofs. In the third game, the public key is generated using $\mathrm{SimProverGen}$ and the well-formedness proof is simulated. The second game and the third game are computationally indistinguishable because the two types of public key are indistinguishable under the SXDH assumption. □

[6] 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.

References

[AFG$^+$10] Masayuki Abe, Georg Fuchsbauer, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. Structure-preserving signatures and commitments to group elements. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 209–236. Springer, August 2010.

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

[BCC$^+$09] Mira Belenkiy, Jan Camenisch, Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Hovav Shacham. Randomizable proofs and delegatable anonymous credentials. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 108–125. Springer, August 2009.

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

[BFI$^+$10] Olivier Blazy, Georg Fuchsbauer, Malika Izabachène, Amandine Jambert, Hervé Sibert, and Damien Vergnaud. Batch Groth-Sahai. In Jianying Zhou and Moti Yung, editors, *ACNS 10*, volume 6123 of *LNCS*, pages 218–235. Springer, June 2010.

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, STOC '88, pages 103–112, New York, NY, USA, 1988. ACM.

[BW06] Xavier Boyen and Brent Waters. Compact group signatures without random oracles. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 427–444. Springer, May / June 2006.

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

[CHP07] Jan Camenisch, Susan Hohenberger, and Michael Østergaard Pedersen. Batch verification of short signatures. In Moni Naor, editor, *EUROCRYPT 2007*, volume 4515 of *LNCS*, pages 246–263. Springer, May 2007.

[CKLM12] Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Sarah Meiklejohn. Malleable proof systems and applications. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 281–300. Springer, April 2012.

[CLOS02] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *34th ACM STOC*, pages 494–503. ACM Press, May 2002.

[Dam92] Ivan Damgård. Non-interactive circuit based proofs and non-interactive perfect zero-knowledge with preprocessing. In Rainer A. Rueppel, editor, *EUROCRYPT'92*, volume 658 of *LNCS*, pages 341–355. Springer, May 1992.

[DFMS04] Ivan Damgård, Serge Fehr, Kirill Morozov, and Louis Salvail. Unfair noisy channels and oblivious transfer. In Moni Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 355–373. Springer, February 2004.

[EG84] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *CRYPTO 1984*, pages 10–18. Springer Berlin Heidelberg, 1985.

[Fuc11] Georg Fuchsbauer. Commuting Signatures and Verifiable Encryption. In Kenneth G.Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 224–245. Springer, May 2011.

[FLS99] Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple non-interactive zero knowledge proofs under general assumptions. *SIAM Journal on Computing*, 29(1):1–28, 1999.

[FPV09] Georg Fuchsbauer, David Pointcheval, and Damien Vergnaud. Transferable Constant-Size Fair E-Cash. In Juan A. Garay, Atsuko Miyaji and Akira Otsuka, editors, *CANS 2009*, volume 5888 of *LNCS*, pages 226–247. Springer, December 2009.

[GMPY06] Juan A. Garay, Philip D. MacKenzie, Manoj Prabhakaran, and Ke Yang. Resource fairness and composability of cryptographic protocols. In Shai Halevi and Tal Rabin, editors, *TCC 2006*, volume 3876 of *LNCS*, pages 404–428. Springer, March 2006.

[GO07] Jens Groth and Rafail Ostrovsky. Cryptography in the Multi-string Model. In Alfred Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 323–341. Springer, August 2007.

[GOS12] Jens Groth, Rafail Ostrovsky, and Amit Sahai. New techniques for noninteractive zero-knowledge. *J. ACM*, 59(3):11, 2012.

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

[Gro06] Jens Groth. Simulation-sound NIZK proofs for a practical language and constant size group signatures. In Xuejia Lai and Kefei Chen, editors, *ASIACRYPT 2006*, volume 4284 of *LNCS*, pages 444–459. Springer, December 2006.

[Gro07] Jens Groth. Fully Anonymous Group Signatures Without Random Oracles. In Kaoru Kurosawa, editor, *ASIACRYPT 2007*, volume 4833 of *LNCS*, pages 164–180. Springer, December 2007.

[Gro10] Jens Groth. Short non-interactive zero-knowledge proofs. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 341–358. Springer, December 2010.

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

[GSW10] Essam Ghadafi, Nigel P. Smart, and Bogdan Warinschi. Groth-Sahai proofs revisited. In Phong Q. Nguyen and David Pointcheval, editors, *PKC 2010*, volume 6056 of *LNCS*, pages 177–192. Springer, May 2010.

[Kil90] Joe Kilian. *Uses of randomness in algorithms and protocols.* MIT Press, 1990.

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

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

[OPV08] Rafail Ostrovsky, Giuseppe Persiano, and Ivan Visconti. Constant-round concurrent non-malleable zero knowledge in the bare public-key model. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *ICALP 2008, Part II*, volume 5126 of *LNCS*, pages 548–559. Springer, July 2008.

[Seo12] Jae Hong Seo On the (Im)possibility of Projecting Property in Prime-Order Setting. In Xiaoyun Wang and Kazue Sako, editors, *ASIACRYPT 2012*, volume 7658 of *LNCS*, pages 61–79. Springer, December 2012.

[TY98] Yiannis Tsiounis and Moti Yung. On the Security of ElGamal Based Encryption In Hideki Imai and Yuliang Zheng, editors, *PKC 1998*, volume 1431 of *LNCS*, pages 117–134. Springer, February 1998.