Stretching Groth-Sahai: NIZK Proofs of Partial Satisfiability

Carla Ràfols

Horst-Görtz Institut für IT Sicherheit, Ruhr-Universität Bochum, Germany
carla.rafols@rub.de

Abstract. Groth, Ostrovsky and Sahai constructed a non-interactive Zap for NP-languages by observing that the common reference string of their proof system for circuit satisfiability admits what they call *correlated key generation*. The latter means that it is possible to create from scratch two common reference strings in such a way that it can be publicly verified that at least one of them guarantees perfect soundness while it is computationally infeasible to tell which one. Their technique also implies that it is possible to have NIWI Groth-Sahai proofs for certain types of equations over bilinear groups in the plain model. We extend the result of Groth, Ostrovsky and Sahai in several directions. Given as input some predicate P computable by some monotone span program over a finite field, we show how to generate a set of common reference strings in such a way that it can be publicly verified that the subset of them which guarantees perfect soundness is accepted by the span program. We give several different flavors of the technique suitable for different applications scenarios and different equation types. We use this to stretch the expressivity of Groth-Sahai proofs and construct NIZK proofs of partial satisfiability of sets of equations in a bilinear group and more efficient Groth-Sahai NIWI proofs without common reference string for a larger class of equation types. Finally, we apply our results to significantly reduce the size of the signatures of the ring signature scheme of Chandran, Groth and Sahai or to have a more efficient proof in the standard model that a commitment opens to an element of a public list.

1 Introduction

Zero-knowledge proofs have played a significant role both in the theory and the practice of cryptographic protocols. Although non-interactive zero-knowledge proofs are in principle more useful for practical purposes, for roughly twenty years after their invention [4] their prohibitive costs made their interactive counterparts the only real alternative. For example, although the connection between NIZKs and signatures was early recognized [2], in practice a widely used technique was to build schemes in the random oracle based on a special kind of interactive proof of knowledge, a Σ-protocol ([24,13]). This approach turned out to be quite fruitful and it was used to build a number of schemes even with complex functionalities, like for instance the numerous kinds of distributed signature schemes based on the work of Cramer *et al.* [9] on interactive proofs of partial

knowledge. In such a proof, the prover convinces the verifier that it knows a subset of the witnesses of a set of statements. Again, although De Santis *et al.* [10] had achieved similar results before for the non-interactive case, they were considered of theoretical interest only and went unnoticed by protocol designers.

The situation changed radically when in 2008, after some promising advances towards making non-interactive proofs practical, ([5,14]), Groth and Sahai ([16]) gave efficient non-interactive proofs of membership in the language of satisfiable quadratic equations in bilinear groups. Compared to previous work, their proposal had the advantage of considering a language which is both very natural and very general. The great number of protocols which use GS proofs ([19,17,6], just to name a few) shows the strength and flexibility of their framework.

More expressive is more efficient. The fact that practical instantiations of GS proofs are in bilinear groups imposes some limitations on the type of equations for which satisfiability can be proven. For instance, each equation should be at most quadratic, and further, in asymmetric bilinear groups, it should have degree at most one in each variable. This can be circumvented at the cost of adding additional variables and equations but this might significantly increase the proof size. Although GS proofs can be considered practical, they remain expensive and even proofs of simple statements might require several dozens of group elements. Therefore, the design goal of obtaining proofs for a more expressive language goes hand in hand with obtaining efficiency improvements.

For instance, suppose one wants to prove a statement of the type which is informally expressed as:

$$\text{“}\hat{\mathbf{c}} \text{ is a commitment to some value } X \in \{1, \ldots, L\}\text{,”} \tag{1}$$

for some $L \in \mathbb{N}$. This statement is naturally encoded as “$\hat{\mathbf{c}}$ opens to some value X which satisfies one of the equations $\{X-1=0, \ldots, X-L=0\}$”. However, as GS proofs do not support this kind of statements, following [6,14], the strategy is to add auxiliary variables $b_1, \ldots, b_L$, prove that $\sum_{i=1}^{L} b_i = 1$ and that, for all $i \in \{1, \ldots, L\}$, $b_i \in \{0,1\}$ and $(X-i)b_i = 0$. Further, in the instantiation of GS proofs in asymmetric bilinear groups, to prove each one of the statements $b_i \in \{0,1\}$ we must add a new additional variable $\overline{b}_i$, and prove satisfiability of the equations $\{b_i - \overline{b}_i = 0, b_i(\overline{b}_i - 1) = 0\}$. The question remains if we can encode these statements in some alternative, more efficient way by *stretching* GS proofs so that they support this sort of statements directly.

Partial Proofs. We would like a result close in spirit to [10,9] for GS proofs. In all these works, the main idea is to use as a building block a proof system $\mathcal{PS}_1$ that allows to prove a certain atomic statement x, and modify it to construct a proof system $\mathcal{PS}_2$ for statements of the kind “Given the atomic statements $x_1, \ldots, x_L$, there exists a subset of indexes $A \subset \{1, \ldots, L\}$ in some family of sets $\Omega \subset \mathcal{P}([L])$ such that all the atomic statements x_i, $i \in A$ are true”.

The prover in $\mathcal{PS}_2$ must construct a proof given only the witnesses for the statements x_i, $i \in A$ and the proof must leak no information about the actual set A, other than it is in Ω. The common strategy of these works is to construct the prover of $\mathcal{PS}_2$ using as building block both the prover of $\mathcal{PS}_1$ — for the

statements x_i, $i \in A$ — and the simulator — for the statements x_i, $i \notin A$. Since real proofs are (computationally) indistinguishable from simulated ones, the final proof output by the prover of $\mathcal{PS}_2$, which consists of proofs for all the statements $x_1, \ldots, x_L$, will reveal nothing about A. The main challenge is then to ensure that the soundness condition is met — guaranteeing that the prover cannot simulate all the proofs—, while making sure that the simulator gets a properly distributed input.

In all these works — including ours — this is done by means of secret sharing techniques, however the challenges that arise are specific to each proof system. For instance, in the work of Cramer *et al.* [9], both $\mathcal{PS}_1$ and $\mathcal{PS}_2$ are Σ-protocols, which are 3 round interactive protocols. In this case, the key difference between a prover and a simulator which outputs an accepting transcript is that the latter creates the transcript by altering the order in which the real protocol is executed and letting the information sent in the first round depend on the challenge, which is the information sent in the second round. In [9], the prover of $\mathcal{PS}_2$ receives a challenge c, from which it creates L challenges $c_1, \ldots, c_L$ and uses c_i as a challenge to prove the atomic statement x_i with $\mathcal{PS}_1$. The secret sharing techniques ensure that the prover of $\mathcal{PS}_2$ has the right amount of freedom in choosing these challenges: namely, they guarantee that there exists some $A \in \Omega$ such that the prover can choose all c_i, $i \in A^c$ (i.e. it can simulate the proofs of x_i, $i \in A^c$), while it is unable to guess the value of c_i, $i \in A$ (i.e. soundness must hold for x_i, $i \in A$).

The GS Proof System. Clearly, the techniques of Cramer *et al.* are specific to Σ-Protocols. On the other hand, the techniques of de Santis *et al.* [10] for the non-interactive case are specific to statements related to quadratic residuosity. Neither of them does apply in any obvious way to GS proofs as the conditions which guarantee soundness or allow to simulate proofs are quite different there.

Indeed, let us recall some basics about GS proofs. They allow to prove satisfiability of several equation types over a bilinear group $gk = (q, \hat{\mathbb{G}}, \check{\mathbb{H}}, \mathbb{T}, e, \hat{g}, \check{h})$, where $\hat{\mathbb{G}}, \check{\mathbb{H}}, \mathbb{T}$ are groups of prime order q in additive notation, the elements $\hat{g}, \check{h}$ are generators of $\hat{\mathbb{G}}, \check{\mathbb{H}}$ respectively, and $e : \hat{\mathbb{G}} \times \check{\mathbb{H}} \to \mathbb{T}$ is an efficiently computable, non-degenerate bilinear map. The witness of satisfiability is a solution to the equation which consists of several elements in $\mathbb{Z}_q$, $\hat{\mathbb{G}}$ or $\check{\mathbb{H}}$. The proof is constructed in a two-step process: first, the prover commits to each element of the witness, then it shows that the committed values satisfy the equation. The common reference string (essentially) consists of some commitment keys. These keys can be generated in one of two indistinguishable modes: in the soundness mode these keys define perfectly binding commitments, and even an unbounded prover cannot convince a verifier of a false statement, and in the witness indistinguishable mode they define perfectly hiding commitments. Further, in the latter case, there exists some trapdoor which allows to simulate proofs which are identically distributed to real proofs (computed as in the WI mode). Additionally, a key is binding or hiding depending on whether or not it satisfies certain linear relations. For instance, the commitment key in the group $\hat{\mathbb{G}}$ consists of three vectors $\hat{\mathbf{u}}, \hat{\mathbf{v}}, \hat{\mathbf{w}} \in \hat{\mathbb{G}}^2$. The commitment to a scalar $x \in \mathbb{Z}_q$ is $\hat{\mathbf{c}} = x\hat{\mathbf{w}} + r\hat{\mathbf{v}}$,

for some random $r \in \mathbb{Z}_q$, and to a group element $\hat{\mathbf{x}} \in \hat{\mathbb{G}}$ is $\hat{\mathbf{c}} = (\hat{\mathbf{0}}, \hat{\mathbf{x}})^\top + r\hat{\mathbf{v}} + s\hat{\mathbf{u}}$, for some random $r, s \in \mathbb{Z}_q$. For scalars the commitment is perfectly binding if $\hat{\mathbf{v}}, \hat{\mathbf{w}} \in \hat{\mathbb{G}}^2$ are linearly independent and perfectly hiding otherwise. For group elements the opposite is true of $\hat{\mathbf{v}}, \hat{\mathbf{u}}$: the commitment is perfectly hiding if $\hat{\mathbf{v}}, \hat{\mathbf{u}}$ are linearly independent and perfectly binding otherwise. To construct partial proofs for the GS proof system, the ideas of [10,9] must be adapted to the inner workings of GS proofs we have just described.

The Non-Interactive Zap of Groth, Ostrovsky and Sahai (GOS). On the other hand, the authors of [15], observe that the common reference string of GS proofs admits what we call Or-Verifiable Correlated Key Generation: namely, that a prover can create *from scratch*, without common reference string, a pair of two keys in such a way that it can be publicly verified that at least one of the two keys is binding for quadratic equations.

More specifically, the observation of GOS is that given any vector $\hat{\mathbf{v}} \in \hat{\mathbb{G}}^2$ such that $\{\hat{\mathbf{v}}, (\hat{0}, \hat{g})^\top\}$ are linearly independent vectors (this condition can be obviously publicly verified by checking if the first component of $\hat{\mathbf{v}}$ is $\hat{\mathbf{0}}$), and two vectors $\hat{\mathbf{z}}_1, \hat{\mathbf{z}}_2 \in \hat{\mathbb{G}}^2$ such that $\hat{\mathbf{z}}_1 + \hat{\mathbf{z}}_2 = (\hat{0}, \hat{g})^\top$, it holds that at least one of $\hat{\mathbf{z}}_1$, $\hat{\mathbf{z}}_2$ is independent of $\hat{\mathbf{v}}$ (otherwise their sum could not be independent of $\hat{\mathbf{v}}$). This means that given only some bilinear group gk (and no common reference string) and a tuple $(\hat{\mathbf{v}}, \hat{\mathbf{z}}_1, \hat{\mathbf{z}}_2)$ with the above constraints, it can be publicly verified that at least one of the pairs of correlated keys $\{\hat{\mathbf{v}}, \hat{\mathbf{z}}_1\}$, $\{\hat{\mathbf{v}}, \hat{\mathbf{z}}_2\}$ is a binding GS key for committing to scalars.

Since GS proofs have perfect soundness, if one of the keys is binding the prover cannot cheat even if it knows additional information about the common reference string (e.g. the discrete logarithm). Thus, if the prover chooses $(\hat{\mathbf{v}}, \hat{\mathbf{z}}_1, \hat{\mathbf{z}}_2)$ and then it proves a statement x with both pairs of correlated keys, the statement must be true. Further, the prover is free to choose one of the keys to be hiding for commitments to scalars and this can be done in such a way that it is computationally infeasible to tell which key is the hiding one. This implies that the proof reveals no information about the witness.

Thus, more technically if we prove the same statement x with both pairs of keys we have given a non-interactive Zap (NI Zap) for x, i.e. a witness indistinguishable proof in *the plain model*. On the other hand, if we take two different statements x_1, x_2 and we give a real proof for one of them with the binding key and we simulate the other one with the hiding key, we have given a NI Zap that at least one of x_1, x_2 is true. This seems to be exactly the right starting point for adapting the techniques for partial proofs to the GS setting. In this work we want to fully develop the approach of GOS, and we address several of its limitations:

1. The technique of GOS needs to be adapted to prove not only witness indistinguishability but also zero-knowledge. Indeed, in the construction above, one of the keys is always binding and to prove "x_1 or x_2", we always need the witness of at least one of the statements.
2. One of the pairs $\{\hat{\mathbf{v}}, \hat{\mathbf{z}}_1\}$, $\{\hat{\mathbf{v}}, \hat{\mathbf{z}}_2\}$ is a binding key to commit to scalars, but not to group elements. For which equation types does this approach allow to

gain efficiency? Can we find a similar technique for commitments to group elements?
3. Can we extend the techniques to other predicates other than the OR of two equations?

In summary, the question is if we can extend the notion of Verifiable Correlated Key Generation to incorporate all these aspects and in such a way that it is useful to construct more efficient NIZK proofs of partial satisfiability for a large class of predicates.

1.1 Our results

Labeled Commit-and-Prove schemes. Recently, Escala and Groth [11] gave a complete formulation of the GS proof system as a labeled Commit-and-Prove (CaP) scheme. The labels are meant to deal with different variable and equation types. This formulation is really convenient for our purposes, as it allows to define in a precise way which equation types admit verifiable correlated key generation and which do not. One of our contributions (section 3) is to slightly modify the definition of labeled CaP of [11] so that it can accommodate both the GS proof system and our new proof system for partial satisfiability.

Extending the definition of verifiable correlated key generation. In section 5, we extend the ideas of GOS in several directions, to use them as a building block to construct proofs of partial satisfiability.

- First, we give a new definition of verifiable correlated key generation — VCKG for short — to adapt it to more general predicates, namely to any predicate P computable by a monotone span program $\mathcal{SP}$. We also modify the definition to explicitly take as input a set of labels so that it fits with the GS CaP formulation of [11]. The motivation for doing so is that the same common reference string might be binding for some equation types and hiding for others. For instance, it is unclear how to extend the result of GOS to prove that an OR of two pairing product equations is satisfied without trusted setup. By this we do not mean that one could not use other to results to prove this (at worst we could prove this by reduction to circuit satisfiability). Our point is rather that introducing labels allows to clearly identify that the construction of GOS *only* ensures that one of the two commitment keys is binding for scalars, but not for group elements.
- Second, we define Simulatable VCKG (SVCKG), which is essentially the same as VCKG except for the fact that the generation algorithm takes as input a common reference string instead of creating the keys from scratch. The keys can be generated in two indistinguishable ways: in such a way that the indexes of the binding keys are a valid assignment of the predicate P (as in VCKG) or in such a way that they are hiding for every index. This definition is introduced with the aim of constructing NIZK proofs of partial satisfiability, and not only NIWI proofs.

In section 6 we show how to combine SVCKG (resp. VCKG) for a predicate P and vector of labels $\mathbf{T} = (\mathtt{t}_1, \ldots, \mathtt{t}_L)$ with the GS proof system to obtain a NIZK proof (resp. NI Zap) that some sets of quadratic equations $\mathcal{S}_1, \ldots, \mathcal{S}_L$ (compatible with $\mathbf{T}$) are partially satisfiable for the predicate P.

Constructions. In section 7 we give several explicit constructions of (S)VCKG for different equation types and predicates P. Essentially, all we require from P is that it should be computable by a monotone span program and the equation types should all admit what we call left-simulation (or all admit right simulation). The construction of GOS (and our extension for other P) guarantees that some keys are binding for committing to scalars and this limits the equation types for which one can prove partial satisfiability. Therefore, in appendix C, we also show how to do correlated key generation for commitment keys to group elements, at some efficiency cost. These explicit constructions of SVCKG (resp. VCKG) together with the GS CaP give a quite expressive realization of NIZK proofs (resp. NI Zap) for partial satisfiability of equations in bilinear groups.

Efficiency. In section 7.4 we discuss what is the size of our proofs of partial satisfiability. We then compare it with the size of the proof that 1-out-of-L sets of equations is satisfiable which results from the approach suggested by Groth ([6,14]).

Applications. In section 9 we discuss some applications. For instance, we show how to save $O(\sqrt{N})$ group elements — where N is the size of the ring — in the signature size of the ring signature scheme of Chandran *et al.* [8], which is the most efficient ring signature in the standard model.

2 Preliminaries

Given some $n \in \mathbb{N}$, $\mathbf{v} \in \mathbb{Z}_q^n$ denotes a column vector unless specifically stated. Given a set of vectors $\{\mathbf{v}_1, \ldots, \mathbf{v}_r\} \subset \mathbb{Z}_q^n$, $\langle\{\mathbf{v}_1, \ldots, \mathbf{v}_r\}\rangle$ denotes their linear span. Matrices are denoted in boldface and $\mathbf{0}_{m\times n}$ denotes the all-zero $m \times n$ matrix. Given some set $\mathcal{S}$ its cardinal is written as $|\mathcal{S}|$ and $s \leftarrow \mathcal{S}$ denotes the process of sampling an element uniformly at random. For an algorithm D, we write $z \leftarrow \mathsf{D}(x, y, \ldots)$ to indicate that D is a (probabilistic) algorithm that outputs z on input $(x, y, \ldots)$. Given a positive integer L, we denote by $[L]$ the set $\{1, \ldots, L\}$.

We identify a set $A \subset [L]$ in the natural way with a vector $\mathbf{v}_A \in \{0,1\}^L$ and we denote its complementary as $A^c := [L]\backslash A$. Given a family of sets $\Omega \subset \mathcal{P}([L])$, we denote $P_\Omega : \{0,1\}^L \to \{0,1\}$ the predicate such $P_\Omega(\mathbf{v}_A) = 1$ if and only if $A \in \Omega$.

2.1 Bilinear Groups

Let $\mathcal{G}$ be some probabilistic polynomial time algorithm which on input 1^λ, where λ is the security parameter, returns the description of a bilinear group $gk = (q, \hat{\mathbb{G}}, \check{\mathbb{H}}, \mathbb{T}, e, \hat{g}, \check{h})$, where $\hat{\mathbb{G}}, \check{\mathbb{H}}$ and $\mathbb{T}$ are groups of prime order q, the elements

$\hat{g}, \check{h}$ are generators of $\hat{\mathbb{G}}, \check{\mathbb{H}}$ respectively, and $e : \hat{\mathbb{G}} \times \check{\mathbb{H}} \to \mathbb{T}$ is an efficiently computable, non-degenerate bilinear map.

Essentially, we take up the notation of Escala and Groth [11] for elements and operations in the bilinear group. Namely, $\hat{\mathbb{G}}, \check{\mathbb{H}}, \mathbb{T}$ are written additively, elements $\hat{x} \in \hat{\mathbb{G}}$ are written with a hat and elements in $\check{y} \in \check{\mathbb{H}}$ with an inverted hat and $\hat{0}, \check{0}$ and $0_{\mathbb{T}}$ denote the neutral elements in the respective groups. For any $\hat{x} \in \hat{\mathbb{G}}, \check{y} \in \check{\mathbb{H}}$, multiplication refers to the pairing operation, *i.e*, $\hat{x}\check{y} := e(\hat{x}, \check{y})$. Matrix/vector or matrix/matrix multiplication of elements in $\hat{\mathbb{G}}$ and $\check{\mathbb{H}}$ are done in the natural way via the pairing operation, for example, given $\hat{\mathbf{X}} \in \hat{\mathbb{G}}^{\ell \times m}$ and $\check{\mathbf{Y}} \in \check{\mathbb{H}}^{m \times n}$, $\hat{\mathbf{X}}\check{\mathbf{Y}} \in \mathbb{T}^{\ell \times n}$.

2.2 SXDH Assumption

Let $(q, \hat{\mathbb{G}}, \check{\mathbb{H}}, \mathbb{T}, e, \hat{g}, \check{h}) \leftarrow \mathcal{G}(1^\lambda)$ be a bilinear group. The Decision Diffie-Hellman Assumption in $\hat{\mathbb{G}}$ states that 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 $\xi, \rho, \kappa \leftarrow \mathbb{Z}_q$, are computationally indistinguishable. The DDH Assumption in $\check{\mathbb{H}}$ is defined in a similar way.

Definition 1. *(SXDH Assumption) The Symmetric eXternal Diffie-Hellman Assumption holds relative to the group generator algorithm $\mathcal{G}$ if the DDH Assumption holds in both $\hat{\mathbb{G}}$ and $\check{\mathbb{H}}$ for $(q, \hat{\mathbb{G}}, \check{\mathbb{H}}, \mathbb{T}, e, \hat{g}, \check{h}) \leftarrow \mathcal{G}(1^\lambda)$.*

2.3 Monotone Span Programs

Definition 2. *[18] A monotone span program (MSP) over a field $\mathbb{Z}_q$ consists of a tuple $\mathcal{SP} = (\mathbf{M}, \rho)$, where $\mathbf{M} \in \mathbb{Z}_q^{(m+1)\times d}$ is a matrix with row vectors $\{\mathbf{r}_0, \mathbf{r}_1, \ldots, \mathbf{r}_m\}$, $\rho : [m] \to [L]$ is a labeling function and $\mathbf{r}_0$ is called the target vector. $\mathcal{SP}$ is said to compute a predicate $P : \{0,1\}^L \to \{0,1\}$ if for any $\mathbf{v}_A \in \{0,1\}^L$, $P(\mathbf{v}_A) = 1$ if and only if $\mathbf{r}_0 \in \langle\{\mathbf{r}_j : j \in \rho^{-1}(A)\}\rangle$.*

Without loss of generality we assume that $m > d$ and that $\mathbf{M}$ has full rank. Specially for MSPs, sometimes it is more intuitive to talk about sets: in this case we say that $\mathcal{SP}$ accepts $A \subset [L]$ if and only if $P(\mathbf{v}_A) = 1$ and that $\mathcal{SP}$ computes $\Omega \subset \mathcal{P}([L])$ if it computes P_Ω.

It is well known that there is a connection between MSPs, secret sharing schemes (sss, [25,3]) and linear codes. Borrowing some terminology from sss, we refer to the family of sets Ω computed by $\mathcal{SP}$ as an *access structure*. Also if $A \in \Omega$, A is said to be *an authorized subset*. If no proper subset of A is authorized, then we say that A is a *minimal authorized subset*. The dual access structure $\Omega^\star$ is defined as $\Omega^\star := \{[L] \backslash A : A \notin \Omega\}$. The latter notion has found applicability and various scenarios, including the proofs of partial satisfiability of [10,9].

A classical example of a (monotone) span program is the threshold one.

Example 1. $\Omega^{(k,L)} := \{A \subset [L] : |A| \geq k\}$ is called a (k, L)-threshold access structure. A span program $\mathcal{SP}_{(k,L)}$ computing $\Omega_{(k,L)}$ can be defined by letting

ρ be the identity function, $\mathbf{r}_i^\top = (1, i, \ldots, i^{k-1})$ and $\mathbf{r}_0^\top = (1, 0, \ldots, 0)$. The dual access structure is $\Omega^*_{(k,L)} := \{A \subset [L] : |A| \geq L - k + 1\}$.

The key ingredient for our results is the following technical lemma, most specially part 1), which states some well-known or easy facts about (monotone) span programs:

Lemma 1. *Let $\mathcal{SP} = (\mathbf{M}, \rho)$ be a monotone span program which computes Ω.*

1) *If $\boldsymbol{\zeta} = (\zeta_0, \zeta_1, \ldots, \zeta_m)^\top \in \mathbb{Z}_q^{m+1}$ is such that $\boldsymbol{\zeta}^\top \mathbf{M} = \mathbf{0}_d$, $\zeta_0 \neq 0$, then $\rho(\{j : \zeta_j \neq 0\}) \in \Omega$.*
2) *If $A \in \Omega$, it is possible to sample $\boldsymbol{\zeta} \in \mathbf{Im}(\mathbf{M}^*)$ uniformly conditioned on a) $\zeta_0 = 1$, and b) $\zeta_j = 0$ for all $j \in \rho^{-1}(A^c)$.*
3) *Let $\{j_1, \ldots, j_\ell\} = \rho^{-1}(A^c)$. For any $(a_{j_1}, \ldots, a_{j_\ell}) \in \mathbb{Z}_q^\ell$, the probability that $(\tau_{j_1}, \ldots, \tau_{j_\ell}) = (a_{j_1}, \ldots, a_{j_\ell})$ is the same if a) $\boldsymbol{\tau} \leftarrow \mathbf{Im}(\mathbf{M}^*)$ or b) $\boldsymbol{\tau} \leftarrow \mathbf{Im}(\mathbf{M}^*)$ conditioned on $\tau_0 = 0$.*

Proof. 1) Observe that, since $\mathbf{M}$ is the transposed of the parity check matrix of $\mathbf{M}^*$, $\boldsymbol{\zeta}^\top \mathbf{M} = \mathbf{0}_d$ if and only if $\boldsymbol{\zeta} \in \mathbf{Im}(\mathbf{M}^*)$, that is $\boldsymbol{\zeta} = \mathbf{M}^* \boldsymbol{\omega}$ for some $\boldsymbol{\omega} \in \mathbb{Z}_q^{m+1-d}$ and in particular $\zeta_j = (\mathbf{r}_j^*)^\top \boldsymbol{\omega}$. Suppose that $B := \rho(\{j : \zeta_j \neq 0\}) \notin \Omega$. But then, by definition of Ω^*, $B^c := [L] \backslash B \in \Omega^*$, so $\mathbf{r}_0^* = \sum_{j \in \rho^{-1}(B^c)} a_j \mathbf{r}_j^*$ for some coefficients a_j. Multiplying on both sides by $\boldsymbol{\omega}$, $\zeta_0 = \sum_{j \in \rho^{-1}(B^c)} a_j \zeta_j = 0$, which contradicts the assumption $\zeta_0 \neq 0$. The rest of the proof is given in appendix A.

3 Commit-and-Prove Scheme

GS Proofs consist of a two step process: given some set of equations and a solution — which is a witness of satisfiability — a prover first commits to the solution and then proves that the committed values satisfy the set of equations. This corresponds to the notion of commit-and-prove (CaP) scheme ([20,7]), although the reformulation in these terms is not straightforward, since GS Proofs allow for a flexibility (different commitment/ equation types) which is not easily captured by the standard definition of a CaP scheme. To address this issue, Escala and Groth [11] write the GS proof system as a CaP scheme with labels. The labels are meant to specify the different commitment/equation types. For example, one might commit to the pair $(\mathsf{t}, m) = (\mathrm{sca}_{\hat{\mathbb{G}}}, m)$, which indicates that $m \in \mathbb{Z}_q$ is a variable whose commitment is in the group $\hat{\mathbb{G}}$.

Let $\mathcal{R}_\mathcal{L}$ be an efficiently verifiable ternary relation, which is described by tuples $(gk, x, W) \in \mathcal{R}_\mathcal{L}$ consisting of a group key, the statement x and the witness W. Define $\mathcal{L}_{gk}$ the language of all statements x for which there is a witness W such that $(gk, x, W) \in \mathcal{R}_\mathcal{L}$. This witness W is a set of pairs $(\mathsf{t}_i, m_i) \in \mathcal{T}_{gk} \times \tilde{\mathcal{M}}_{gk} \subset \mathcal{M}_{gk}$, where $\mathcal{T}_{gk}$ is the label space and $\mathcal{M}_{gk}$ is the labeled message space. For instance, $(\mathrm{sca}_{\hat{\mathbb{G}}}, m) \in \mathcal{M}_{gk}$, $\mathrm{sca}_{\hat{\mathbb{G}}} \in \mathcal{T}_{gk}$.

We assume that the statement x unambiguously defines some vector T_x of elements of $\mathcal{T}_{gk}$. One should think of T_x as describing the labels which define the correct format of a witness of x.

Definition 3. *(Labeled CaP scheme (modified from [11])) A commit-and-prove scheme* $\mathsf{CaP} = (\mathsf{G}, \mathsf{LabGen}, \mathsf{Com}, \mathsf{P}, \mathsf{V})$ *for* $\mathcal{L}_{gk}$ *consists of five PPT algorithms.*

- $\mathsf{G}(1^\lambda)$*: This algorithm runs in two steps. On input the security parameter* λ*,* $\mathsf{G}_0(1^\lambda)$ *outputs a group key* gk *which includes the description of a group, a space* $\mathcal{K}_{gk}$ *of valid commitment keys, a message space* $\mathcal{M}_{gk}$*, a randomness space* $\mathcal{R}_{gk}$ *and a commitment space* $\mathcal{C}_{gk}$*. Algorithm* $\mathsf{G}_1(gk)$ *outputs the pair* (gk, ck) *where* $ck \in \mathcal{K}_{gk}$ *is a commitment key.*
- $\mathsf{LabGen}(gk, ck, x, W)$*: This algorithm, on input* gk*,* $ck \in \mathcal{K}_{gk}$*, a pair* (x, W) *such that* $(gk, x, W) \in \mathcal{R}_{\mathcal{L}}$*, outputs a public label,* $\mathsf{k}^p = (\mathsf{t}, \tilde{\mathsf{t}})$*, and a secret label,* k^s*, for each* $\mathsf{t} \in \mathsf{T}_x$*.*
- $\mathsf{Com}(gk, ck, (\mathsf{k}^p, \mathsf{k}^s, m))$*: On input* gk*,* $ck \in \mathcal{K}_{gk}$*, a public label* $\mathsf{k}^p = (\mathsf{t}, \tilde{\mathsf{t}})$ *such that* $(\mathsf{t}, m) \in \mathcal{M}_{gk}$ *and a secret label* k^s*, this algorithm picks randomness* $(\mathsf{t}, r) \in \mathcal{R}_{gk}$ *and returns a commitment* c *with label* k^p *such that* $(\mathsf{k}^p, c) \in \mathcal{C}_{gk}$*.*
- $\mathsf{P}(gk, ck, x, Op, C)$*: On input* gk*,* $ck \in \mathcal{K}_{gk}$*,* $x \in \mathcal{L}_{gk}$ *and sets* $Op = \{(\mathsf{k}_i^p = (\mathsf{t}_i, \tilde{\mathsf{t}}_i), \mathsf{k}_i^s, m_i, r_i) : i \in \mathcal{I}\}$*,* $C = \{(\mathsf{k}_i^p, c_i) : i \in \mathcal{I}\}$ *such that for each* $i \in \mathcal{I}$*,* $(\mathsf{k}_i^p, \mathsf{k}_i^s, m_i, r_i)$ *is a valid opening of* (k_i^p, c_i)*, and such that* $(gk, x, \{(\mathsf{t}_i, m_i) : i \in \mathcal{I}\}) \in \mathcal{R}_{\mathcal{L}}$*, this algorithm outputs a proof* π*.*
- $\mathsf{V}(gk, ck, x, C, \pi)$*: Given the group key* gk*, a commitment key* ck*, a statement* x*, a proof* π *and commitments* $(\tilde{\mathsf{t}}_i, c_i) \in \mathcal{C}_{gk}$*, algorithm* V *returns* 1 *if the proof is accepted and* 0 *otherwise.*

Compared to [11], in our definition commitments admit an extra label pair $(\mathsf{k}^p, \mathsf{k}^s)$ and an algorithm which generates this label LabGen. For the Groth-Sahai CaP scheme CaP_{GS}, we ignore these additional labels. This extra label will be necessary when we construct a CaP scheme CaP_{par} for partial satisfiability of quadratic equations based on CaP_{GS}, as we implicitly use different GS commitment keys ck_j to prove a single statement with CaP_{par}. The keys used for each commitment can be seen as public label of the commitment (but not as part of the witness (t_i, m_i)) and the secret label k^s as the trapdoor (when it exists, else it is some special symbol). That is why we also say in the definition of P "$(\mathsf{k}_i^p, \mathsf{k}_i^s, m_i, r_i)$ is a valid opening of (k_i^p, c_i)", as the opening might depend on k_i^p. Although the simulation trapdoor is not necessary to compute the commitments, we assume that the algorithm $\mathsf{Com}(gk, ck, (\mathsf{k}^p, \mathsf{k}^s, m))$ takes also as input k^s, because k^s might not only contain the simulation trapdoor but also additional information which allows to speed up the computation, like the discrete logarithms of the commitment keys given in k_i^p.[1]

Another difference with [11] is that, we explicitly define a keyspace $\mathcal{K}_{gk}$. We assume that membership in $\mathcal{K}_{gk}$ is efficiently decidable for all gk. Further, we distinguish between the group key and the commitment keys. The language for which we define the proof system depends on gk but not of ck, i.e. it is a group dependent language[2] This is done with the purpose of precisely defining the sets

[1] Escala and Groth observed that if the prover knows the discrete logarithm of ck, computation is sped up significantly, as then most exponentiations can be replaced by operations in $\mathbb{Z}_q$.

[2] As in the original definition of GS Proofs, [16].

of hiding and binding keys to define verifiable correlated key generation for the GS CaP. For the following definition, we restrict ourselves to some CaP scheme in which algorithm LabGen is trivial (so that we can omit $(\mathsf{k}^p, \mathsf{k}^s)$).

Definition 4. *Given some* CaP *scheme, we define* $\mathcal{K}^{\mathsf{t}}_{gk,\mathsf{bind}}$ *(resp.* $\mathcal{K}^{\mathsf{t}}_{gk,\mathsf{hid}}$*) as the set of* $ck \in \mathcal{K}_{gk}$ *such that* $\mathsf{Com}(gk, ck, \mathsf{t}, m)$ *is perfectly binding (resp. perfectly hiding) for all* $(\mathsf{t}, m) \in \mathcal{M}_{gk}$*.*

We defer the definitions of perfect completeness and perfect soundness to appendix B.1. Roughly speaking, completeness guarantees that correctly generated proofs are accepted, perfect soundness that a proof of a false statement is never accepted. Both the GS proof system and our scheme satisfy a strong notion of security, namely *composable zero-knowledge*.

Definition 5. *The commit-and-prove system* CaP *is (computationally) composable zero-knowledge if there exist PPT algorithms* $\mathsf{SimGen}, \mathsf{SimCom}, \mathsf{SimProve}, \mathsf{SimLabGen}$ *such that for all non-uniform polynomial time stateful interactive adversaries* $\mathcal{A}$*,*

$$\Pr\left[(gk, ck) \leftarrow \mathsf{G}(1^\lambda) : \mathcal{A}(gk, ck) = 1\right] \approx \Pr\left[(gk, ck, tk) \leftarrow \mathsf{SimGen}(1^\lambda) : \mathcal{A}(gk, ck) = 1\right] \text{ and}$$

$$\Pr\left[(gk, ck, tk) \leftarrow \mathsf{SimGen}(1^\lambda); (x, \mathcal{I}) \leftarrow \mathcal{A}^{\mathsf{Com}(gk,ck,\cdot),\widetilde{\mathsf{LabGen}}(gk,ck,\cdot,\cdot)}(gk, ck, tk); \pi \leftarrow \mathsf{P}(gk, ck, x, \{(\mathsf{k}^p_i, \mathsf{k}^s_i, m_i, r_i) : i \in \mathcal{I}\}, \{(\mathsf{k}^p_i, c_i) : i \in \mathcal{I}\}) : \mathcal{A}(\pi) = 1\right] =$$

$$\Pr\left[(gk, ck, tk) \leftarrow \mathsf{SimGen}(1^\lambda); (x, \mathcal{I}) \leftarrow \mathcal{A}^{\mathsf{SimCom}(gk,ck,\cdot),\widetilde{\mathsf{SimLabGen}}(gk,ck,\cdot)}(gk, ck, tk); \pi \leftarrow \mathsf{SimProve}(gk, ck, tk, x, \{(\mathsf{k}^p_i, \mathsf{k}^s, s_i) : i \in \mathcal{I}\}, \{(\mathsf{k}^p_i, c_i) : i \in \mathcal{I}\}) : \mathcal{A}(\pi) = 1\right],$$

where a) $\mathsf{SimLabGen}(gk, ck, tk, x)$ *returns* $(\mathsf{k}^p_i = (\mathsf{t}_i, \tilde{\mathsf{t}}_i), \mathsf{k}^s_i)$ *for each* $\mathsf{t}_i \in \mathsf{T}_x$*, b)* $\widetilde{\mathsf{SimLabGen}}(gk, ck, tk, x)$ *(resp.* $\widetilde{\mathsf{LabGen}}(gk, ck, x)$*) returns* $\mathsf{k}^p_i = (\mathsf{t}_i, \tilde{\mathsf{t}}_i)$ *for each* $\mathsf{t}_i \in \mathsf{T}_x$ *with the distribution induced by running* $\mathsf{SimLabGen}$ *(resp. by* LabGen*) with the same input, c)* $\mathsf{SimCom}(gk, ck, \cdot)$ *on* $(\mathsf{k}^p_i, \mathsf{k}^s_i)$ *outputs* $(\mathsf{k}^p_i, c_i) \in \mathcal{C}_{gk}$*, d)* $\mathcal{A}$ *picks* $(x, \mathcal{I})$ *such that* $(gk, x, \{(\mathsf{t}_i, m_i) : i \in \mathcal{I}\}) \in \mathcal{R}_{\mathcal{L}}$*.*

Finally a non-interactive Zap (NI Zap) is a (computationally) witness indistinguishable proof in the plain model, i.e. without common reference string. Informally, computational WI just requires that two proofs generated by the prover on a statement x with different witnesses W_0, W_1 should be computationally indistinguishable even for an adversary who chooses (x, W_0, W_1).

Definition 6. *We say that the commit-and-prove system* CaP *is a NI Zap if the algorithm* $G_1(gk)$ *is trivial (i.e.* $(gk, ck = \perp) \leftarrow \mathsf{G}_1(gk)$*) and the CaP has perfect completeness, perfect soundness and computational witness indistinguishability.*

$\mathsf{G}(1^\lambda)$	$\mathsf{SimGen}(1^\lambda)$
$gk \leftarrow (q, \hat{\mathbb{G}}, \check{\mathbb{H}}, \mathbb{T}, e, \hat{g}, \check{h}) \leftarrow \mathcal{G}(1^\lambda)$	$gk \leftarrow (q, \hat{\mathbb{G}}, \check{\mathbb{H}}, \mathbb{T}, e, \hat{g}, \check{h}) \leftarrow \mathcal{G}(1^\lambda)$
$\omega, \sigma, \xi, \psi \leftarrow \mathbb{Z}_q^*$	$\omega, \sigma, \xi, \psi, \leftarrow \mathbb{Z}_q^*$
$\hat{\mathbf{v}} \leftarrow (\xi\hat{g}, \hat{g})^\top$, $\check{\mathbf{v}} \leftarrow (\psi\check{h}, \check{h})^\top$	$\hat{\mathbf{v}} \leftarrow (\xi\hat{g}, \hat{g})^\top$, $\check{\mathbf{v}} \leftarrow (\psi\check{h}, \check{h})^\top$
$\hat{\mathbf{u}} \leftarrow \omega\hat{\mathbf{v}}$, $\check{\mathbf{u}} \leftarrow \sigma\check{\mathbf{v}}$	$\hat{\mathbf{u}} \leftarrow \omega\hat{\mathbf{v}} + (\hat{0}, \hat{g})^\top$, $\check{\mathbf{u}} \leftarrow \sigma\check{\mathbf{v}} + (\check{0}, \check{h})^\top$
$\hat{\mathbf{w}} \leftarrow \hat{\mathbf{u}} - (\hat{0}, \hat{g})^\top$, $\check{\mathbf{w}} \leftarrow \check{\mathbf{u}} - (\check{0}, \check{h})^\top$	$\hat{\mathbf{w}} \leftarrow \hat{\mathbf{u}} - (\hat{0}, \hat{g})^\top$, $\check{\mathbf{w}} \leftarrow \check{\mathbf{u}} - (\check{0}, \check{h})^\top$
$ck \leftarrow (\hat{\mathbf{u}}, \hat{\mathbf{v}}, \hat{\mathbf{w}}, \check{\mathbf{u}}, \check{\mathbf{v}}, \check{\mathbf{w}})$	$ck \leftarrow (gk, \hat{\mathbf{u}}, \hat{\mathbf{v}}, \hat{\mathbf{w}}, \check{\mathbf{u}}, \check{\mathbf{v}}, \check{\mathbf{w}})$
Return (gk, ck)	$tk \leftarrow (ck, \omega, \sigma)$
	Return (gk, ck, tk)

Label t	Message	Randomness	Commitment	$\mathcal{K}^{\mathsf{t}}_{gk,\mathsf{bind}}$	$\mathcal{K}^{\mathsf{t}}_{gk,\mathsf{hid}}$
$\mathrm{sca}_{\hat{\mathbb{G}}}$	$(\mathrm{sca}_{\hat{\mathbb{G}}}, x)$	$(\mathrm{sca}_{\hat{\mathbb{G}}}, r)$	$\hat{\mathbf{c}} \leftarrow \hat{\mathbf{w}}x + \hat{\mathbf{v}}r$	$ck : \hat{\mathbf{w}} \notin \langle\hat{\mathbf{v}}\rangle$	$ck : \hat{\mathbf{w}} \in \langle\hat{\mathbf{v}}\rangle$
$\mathrm{com}_{\hat{\mathbb{G}}}$	$(\mathrm{com}_{\hat{\mathbb{G}}}, \hat{x})$	$(\mathrm{com}_{\hat{\mathbb{G}}}, r, s)$	$\hat{\mathbf{c}} \leftarrow \mathbf{e}_2\hat{x} + \hat{\mathbf{v}}r + \hat{\mathbf{u}}s$	$ck : \hat{\mathbf{u}} \in \langle\hat{\mathbf{v}}\rangle$	$ck : \hat{\mathbf{u}} \notin \langle\hat{\mathbf{v}}\rangle$

Fig. 1. Generator algorithms of the CaP scheme of [11] and table describing most important commitment types.

4 Groth-Sahai NIZK Proofs

In this section we give a high-level description of the GS proof system in terms of a commit-and-prove scheme as in [11]. We concentrate on the key generation and commit phase, which are the ones necessary to understand our construction, for the full description we refer the reader to the original paper.

The GS proof system allows to prove that x is satisfiable, where x encodes some set of quadratic equations in a bilinear group of the following form:

$$\sum_{j=1}^{n} f(\alpha_j, y_j) + \sum_{i=1}^{m} f(x_i, \beta_i) + \sum_{i=1}^{m}\sum_{j=1}^{n} f(x_i, \gamma_{ij} y_j) = t, \tag{2}$$

where A_1, A_2, A_T are $\mathbb{Z}_q$-vector spaces equipped with some bilinear map $f : A_1 \times A_2 \to A_T$, $\boldsymbol{\alpha} \in A_1^n$, $\boldsymbol{\beta} \in A_2^m$, $\boldsymbol{\Gamma} = (\gamma_{ij}) \in \mathbb{Z}_q^{m\times n}$, $t \in A_T$. The modules and the map f can be defined in different ways as: (a) in pairing-product equations (PPEs), $A_1 = \hat{\mathbb{G}}$, $A_2 = \check{\mathbb{H}}$, $A_T = \mathbb{T}$, $f(\hat{x}, \check{y}) = \hat{x}\check{y} \in \mathbb{T}$, (b1) in multi-scalar multiplication equations in $\hat{\mathbb{G}}$ (MMEs), $A_1 = \hat{\mathbb{G}}$, $A_2 = \mathbb{Z}_q$, $A_T = \hat{\mathbb{G}}$, $f(\hat{x}, y) = y\hat{x} \in \hat{\mathbb{G}}$, b2) MMEs in $\check{\mathbb{H}}$ (MMEs), $A_1 = \mathbb{Z}_q$, $A_2 = \check{\mathbb{H}}$, $A_T = \check{\mathbb{H}}$, $f(x, \check{y}) = x\check{y} \in \check{\mathbb{H}}$, and (c) in quadratic equations in $\mathbb{Z}_q$ (QEs), $A_1 = A_2 = A_T = \mathbb{Z}_q$, $f(x, y) = xy \in \mathbb{Z}_q$. Each element describing an equation receives a label t_i and each equation a label L_{eq}, for instance $\mathsf{L}_{eq} = \mathrm{QE}$ is a quadratic equation, or $\mathsf{L}_{eq} = \mathrm{MLin}_{\hat{\mathbb{G}}}$ is a linear multi-scalar multiplication equation with variables in $\hat{\mathbb{G}}$. The classification of Escala and Groth of equation types (see [11], figure 6) is very fine grained with the objective of augmenting the class of equations which admit zero-knowledge proofs (softening the requirement $t = 0_{\mathbb{T}}$ for PPEs given in [16]) and also of describing efficiency improvements which only apply to a specific equation type.

Given some equation of the form (2), the first step of the prover is to commit to all elements describing the equation according to their label, where "commit"

is used in a wide sense as an equivalent to embed the elements in the right space. That is, for instance, the equation $\hat{x}_1\check{b}_1 + \hat{x}_2\check{b}_2 = 0_{\mathbb{T}}$ is described by $\hat{x}_1, \hat{x}_2, \check{b}_1, \check{b}_2$. As $\hat{x}_1, \hat{x}_2$ are variables in $\hat{\mathbb{G}}$, they have the label $\mathrm{com}_{\hat{\mathbb{G}}}$, while the constants $\check{b}_1, \check{b}_2$ have the labels $(\mathsf{pub}_{\check{\mathbb{H}}}, \check{b}_1)$ and $(\mathsf{pub}_{\check{\mathbb{H}}}, \check{b}_2)$. The commitment to an element with label $\mathrm{com}_{\hat{\mathbb{G}}}$ is described in figure 1, and the one to $\check{b}_i$ is simply $(\check{0}, \check{b}_i)^\top \in \check{\mathbb{H}}^2$. The latter deviates from the usual definition of commitment in the sense that it is not computationally hiding.

The vector of labels T_x associated to some statement x, to which we referred in the syntactic definition of Labeled CaP, is the specification (in some fixed order) of the label type of all the elements describing the equation, for instance, in the example above $\mathsf{T}_x = (\mathrm{com}_{\hat{\mathbb{G}}}, \mathrm{com}_{\hat{\mathbb{G}}}, (\mathsf{pub}_{\check{\mathbb{H}}}, \check{b}_1), (\mathsf{pub}_{\check{\mathbb{H}}}, \check{b}_2))$. Of course this vector of labels must be consistent with the equation type L_{eq}.

Recall that GS CaP uses the *parameter switching technique* of [15]. This means that the common reference string can be generated in two different, computationally indistinguishable ways: in the soundness setting, not even a computationally unbounded adversary can convince a verifier of a false statement, while in the witness indistinguishability (WI) setting, the keys are generated with a trapdoor which allows to construct simulated proofs.

In the CaP scheme for partial satisfiability we will let the prover choose some keys ck_j and some trapdoors tk_j, $j = 1, \ldots, m$. Each key ck_j will be used to prove/simulate a different atomic statement x_i (i.e. satisfiability of some equation set $\mathcal{S}_i$). It is fundamental to define precisely for which type of equations a key ck_j defines perfectly sound proofs or when it allows to simulate them, so that we can prove meaningful statements about partial satisfiability.

For this, although we do not describe all possible equation types or all possible labels in $\mathcal{T}_{gk}$ (or their corresponding commitments), we must specify how to commit to variables. The four possible label types for variables are $\mathrm{sca}_{\hat{\mathbb{G}}}, \mathrm{sca}_{\check{\mathbb{H}}}, \mathrm{com}_{\hat{\mathbb{G}}}, \mathrm{com}_{\check{\mathbb{H}}}$, which correspond, respectively, to elements 1) in $A_1 = \mathbb{Z}_q$, 2) in $A_2 = \mathbb{Z}_q$, 3) in $A_1 = \hat{\mathbb{G}}$ or 4) in $A_2 = \check{\mathbb{H}}$. The interesting thing about these commitments (see figure 1) is that they are binding or hiding depending on the way we generate the keys. Very roughly, simulation in GS Proofs works by opening "commitments" to more than one element. Thus, a necessary condition to simulate proofs for some equation of the form (2) with a certain ck is that ck defines a hiding commitment for the variables in one of the modules A_i. In summary, it is essential to discuss which keys ck are binding/hiding for each variable type.

Key space and commitments. The space of keys $\mathcal{K}_{gk}$ consists of all tuples $(gk, \hat{\mathbf{u}}, \hat{\mathbf{v}}, \hat{\mathbf{w}}, \check{\mathbf{u}}, \check{\mathbf{v}}, \check{\mathbf{w}})$ such that gk is a valid description of an asymmetric bilinear group, $\hat{\mathbf{u}}, \hat{\mathbf{v}} \in \hat{\mathbb{G}}^2$, $\check{\mathbf{u}}, \check{\mathbf{v}} \in \check{\mathbb{H}}^2$ and $\hat{\mathbf{w}} = \hat{\mathbf{u}} - (\hat{0}, \hat{g})^\top$, $\check{\mathbf{w}} = \check{\mathbf{u}} - (\check{0}, \check{h})^\top$. We define $\mathbf{e}_2 := (0,1)^\top$ and $\hat{\mathbf{e}}_2 := (\hat{0}, \hat{g})^\top$. Commitments to scalars and group elements are described in the table below (for the group $\check{\mathbb{H}}$ they are defined analogously). Note that in the soundness setting the algorithm $\mathsf{G}(1^\lambda)$ outputs keys $ck \in \mathcal{K}^{\mathrm{sca}_{\hat{\mathbb{G}}}}_{gk,\mathsf{bind}} \cap \mathcal{K}^{\mathrm{com}_{\hat{\mathbb{G}}}}_{gk,\mathsf{bind}}$ (see Definition 4) and in the WI setting, SimGen outputs $ck \in \mathcal{K}^{\mathrm{sca}_{\hat{\mathbb{G}}}}_{gk,\mathsf{hid}} \cap \mathcal{K}^{\mathrm{com}_{\hat{\mathbb{G}}}}_{gk,\mathsf{hid}}$, but in general a key might be binding/hiding only for one

label t, this is why the CaP formulation of GS Proofs is really convenient to define correlated key generation.

Right vs Left-Simulatable. The simulation trapdoor $tk = (ck, \omega, \sigma)$ allows to double-open some commitments. This trapdoor allows to simulate all the considered equations, but it will be convenient to be more precise. We say that an equation is left-simulatable if there exists an efficient algorithm $\mathsf{SimProve}$ which takes as input $tk = \omega$ (right-simulatable if the same holds for σ). Roughly speaking, an equation x of the form (2) is left (resp. right) simulatable if it is possible to equivocate enough commitments to elements of A_1 (resp. to A_2) to $\hat{0}$ (resp. $\check{0}$) so that the equation admits the trivial solution. In any case, for our purposes it is enough to know that there are equations which can only be simulated on one side and that this can be made precise.

Admissible simulation labels. Further, we say an equation type L_{eq} admits label $\mathsf{t}_{sim} = \mathrm{com}_{\hat{\mathbb{G}}}$ if $A_1 = \hat{\mathbb{G}}$ and it is left-simulatable or label $\mathsf{t}_{sim} = \mathrm{sca}_{\hat{\mathbb{G}}}$ if $A_1 = \mathbb{Z}_q$ and it is left-simulatable (the same w.r.t. $\check{\mathbb{H}}$ and right-simulatable). For instance, QEs are both left and right-simulatable and admit both labels $\{\mathrm{sca}_{\hat{\mathbb{G}}}, \mathrm{sca}_{\check{\mathbb{H}}}\}$, while linear MMEs in $\hat{\mathbb{G}}$ admit the label $\mathrm{sca}_{\hat{\mathbb{G}}}$ if the variables are in $\mathbb{Z}_q$ but only admit the label $\mathrm{com}_{\hat{\mathbb{G}}}$ if the variables are in $\hat{\mathbb{G}}$ and the equation is homogeneous.

5 (Simulatable) Verifiable Correlated Key Generation: Definitions

Let $\mathsf{CaP} = (\mathsf{G} = (\mathsf{G}_0, \mathsf{G}_1), \mathsf{Com}, \mathsf{P}, \mathsf{V})$ be a commit-and-prove scheme with perfect soundness, perfect completeness and composable zero-knowledge and let $\mathsf{SimGen}, \mathsf{SimCom}, \mathsf{SimProve}$ be the corresponding simulation algorithms.

The definitions in this section are meant to capture the necessary properties that a CaP scheme must satisfy so that we can extend it to give proofs for partial relations. Given a monotone span program $\mathcal{SP} = (\mathbf{M}, \rho)$, we will require the existence of an algorithm $\mathsf{K}_{\mathsf{corr}}$ (or $\mathsf{K}_{\mathsf{scorr}}$ for the simulatable case) which outputs a set of correlated keys $\Sigma = \{ck_1, \ldots, ck_m\} \subset \mathcal{K}_{gk}$. These keys should be such that it can be publicly verified that the (unknown) subset of binding keys corresponds to some satisfying assignment of the predicate computed by $\mathcal{SP}$. Further, $\mathsf{K}_{\mathsf{corr}}$ ($\mathsf{K}_{\mathsf{scorr}}$) should also output a trapdoor for the non-binding keys.

When P is the predicate OR of two variables, the first definition (of VCKG) matches the original one of GOS. In this definition, we require the keys to be created from scratch, given only the group key gk.

On the other hand, for the second definition (SVCKG), our algorithms take as input a common reference string ck. In this case, we require the existence of an algorithm $\mathsf{K}_{\mathsf{scorr}}$ which outputs some set of keys with the same properties as in VCKG when ck is binding. We also require the existence of another algorithm which outputs only hiding keys with their simulation trapdoor. When ck is hiding, both should have identically distributed output.

To construct NIZK proofs of partial satisfiability, we will use as a building block a SVCKG scheme, while a construction of VCKG combined with the GS CaP will allow us to derive NIWI proofs of partial satisfiability in the plain model.

In both definitions, the vector $\mathbf{T}$ specifies a vector of labels. With these labels we can define precisely what we mean by "hiding key" or "binding key", as this depends on the label type. Later, when we use (S)VCKG to prove partial satisfiability, we will use the key ck_j to prove statement $x_{\rho(j)}$. These labels will guarantee that the key ck_j matches the equation type of $x_{\rho(j)}$.

Definition 7. CaP *admits P-Verifiable Correlated Key Generation for the predicate* $P_\Omega : \{0,1\}^L \to \{0,1\}$ *computed by a span program* $\mathcal{SP} = (\mathbf{M}, \rho)$ *and some label vector* $\mathbf{T} = (\mathsf{t}_1, \mathsf{t}_2, \ldots, \mathsf{t}_L)$*, if there exist two probabilistic polynomial time algorithms* $(\mathsf{K}_{\mathsf{corr}}, \mathsf{V}_{\mathsf{corr}})$*, with the following properties:*

a) Given any $\mathbf{v}_A \in \{0,1\}^L$ *such that* $P(\mathbf{v}_A) = 1$*, and* $gk \leftarrow \mathsf{G}_0(1^\lambda)$*, algorithm* $\mathsf{K}_{\mathsf{corr}}(gk, \mathcal{SP}, \mathbf{v}_A, \mathbf{T})$ *outputs* $(\Sigma, \mathsf{TK}_{A^c})$*, where* $\Sigma = \{ck_1, \ldots, ck_m\}$ *is such that for all* $j \in \rho^{-1}(A^c)$*,* $ck_j \in \mathcal{K}_{gk,\mathsf{hid}}^{\mathsf{t}_{\rho(j)}}$*, and* $\mathsf{TK}_{A^c} := \{tk_j : j \in \rho^{-1}(A^c)\}$ *is the set of the corresponding (valid) trapdoors.*

b) For all PPT adversaries $\mathsf{D} = (\mathsf{D}_0, \mathsf{D}_1)$ *and if* $\mathbf{v}_A, \mathbf{v}_B$ *are such that* $P(\mathbf{v}_A) = P(\mathbf{v}_B) = 1$*,*

$$\begin{aligned}\Pr\Big[gk \leftarrow \mathsf{G}_0(1^\lambda); (\Sigma, \mathsf{TK}_{A^c}) \leftarrow \mathsf{K}_{\mathsf{corr}}(gk, \mathcal{SP}, \mathbf{v}_A, \mathbf{T});\\ (\mathbf{v}_A, \mathbf{v}_B, st) \leftarrow \mathsf{D}_0(gk, \mathcal{SP}, \mathbf{T}) : \mathsf{D}_1(\Sigma, \mathsf{TK}_{A^c \cap B^c}, st) = 1\Big]\\ \approx \Pr\Big[gk \leftarrow \mathsf{G}_0(1^\lambda); (\Sigma, \mathsf{TK}_{B^c}) \leftarrow \mathsf{K}_{\mathsf{corr}}(gk, \mathcal{SP}, \mathbf{v}_B, \mathbf{T});\\ (\mathbf{v}_A, \mathbf{v}_B, st) \leftarrow \mathsf{D}_0(gk, \mathcal{SP}, \mathbf{T}) : \mathsf{D}_1(\Sigma, \mathsf{TK}_{A^c \cap B^c}, st) = 1\Big]\end{aligned}$$

c) Given as input $(gk, \mathcal{SP}, \Sigma, \mathbf{T})$*,* $\mathsf{V}_{\mathsf{corr}}$ *outputs a bit* b *such that, for all PPT adversaries* D*:*

$$\Pr\Big[(\Sigma, \mathbf{T}) \leftarrow \mathsf{D}(gk, \mathcal{SP}, \mathbf{T}); \mathsf{V}_{\mathsf{corr}}(gk, \mathcal{SP}, \Sigma, \mathbf{T}) = 1 : \rho(\{j : ck_j \in \mathcal{K}_{gk,\mathsf{bind}}^{\mathsf{t}_{\rho(j)}}\}) \notin \Omega\Big] = 0.$$

Definition 8. CaP *admits P-Simulatable Verifiable Correlated Key Generation for the predicate* $P_\Omega : \{0,1\}^L \to \{0,1\}$ *computed by a span program* $\mathcal{SP} = (\mathbf{M}, \rho)$ *and some label vector* $\mathbf{T} = (\mathsf{t}_1, \ldots, \mathsf{t}_L)$*, if there exist three probabilistic polynomial time algorithms* $(\mathsf{K}_{\mathsf{scorr}}, \mathsf{V}_{\mathsf{scorr}}, \mathsf{SimCorr})$*, with the following properties:*

a) as in point a) of definition 7 except that $\mathsf{K}_{\mathsf{scorr}}$ *receives* $(gk, ck) \leftarrow \mathsf{G}(1^\lambda)$ *as part of the input.*

b) Given some $(gk, ck, tk) \leftarrow \mathsf{SimGen}(1^\lambda)$*, algorithm* $\mathsf{SimCorr}(gk, ck, tk, \mathcal{SP}, \mathbf{T})$ *outputs* $(ck, \Sigma, \mathsf{TK})$ *such that* $\Sigma = \{ck_1, \ldots, ck_m\}$ *is a set of commitment keys with* $ck_j \in \mathcal{K}_{gk,\mathsf{hid}}^{\mathsf{t}_{\rho(j)}}$ *for all* $j \in [m]$ *and a set* $\mathsf{TK} := \{tk_j : j \in [m]\}$ *such that* (ck_j, tk_j)*,* $j \in [m]$ *is a valid pair of commitment key and trapdoor. Further,*

$$\Pr\Big[(gk, ck, tk) \leftarrow \mathsf{SimGen}(1^\lambda); (\Sigma, \mathsf{TK}_{A^c}) \leftarrow \mathsf{K}_{\mathsf{scorr}}(gk, ck, \mathcal{SP}, \mathbf{v}_A, \mathbf{T}) : \mathsf{D}(gk, ck, \mathcal{SP}, \Sigma, \mathbf{T}) = 1\Big]$$
$$= \Pr\Big[(gk, ck, tk) \leftarrow \mathsf{SimGen}(1^\lambda); (\Sigma, \mathsf{TK}) \leftarrow \mathsf{SimCorr}(gk, ck, tk, \mathcal{SP}, \mathbf{T}) : \mathsf{D}(gk, ck, \mathcal{SP}, \Sigma, \mathbf{T}) = 1\Big]$$

c) Given $(gk, ck, \mathcal{SP}, \Sigma, \mathbf{T})$, $\mathsf{V}_{\mathsf{scorr}}$ *outputs a bit* b *such that, for all PPT adversaries* D*:*

$$\Pr\Big[(gk, ck) \leftarrow \mathsf{G}(1^\lambda); (\Sigma, \mathbf{T}) \leftarrow \mathsf{D}(gk, ck, \mathcal{SP}, \mathbf{T}); \mathsf{V}_{\mathsf{scorr}}(gk, ck, \mathcal{SP}, \Sigma, \mathbf{T}) = 1 : \rho(\{j : ck_j \in \mathcal{K}^{\mathsf{t}_{\rho(j)}}_{gk,\mathsf{bind}}\}) \notin \Omega\Big] = 0.$$

6 NIZK Proofs and NI Zap of Partial Satisfiability

In this section we formally put together all the pieces of the puzzle: the GS CaP, the new definition of labeled CaP and the notion of SVCKG (resp. VCKG) to construct NIZK proofs (resp. a NI Zap) for partial satisfiability.

More specifically, starting from the GS CaP described in section 4, $\mathsf{CaP}_{GS} = (\mathsf{G}, \mathsf{Com}, \mathsf{P}, \mathsf{V})$ for the language defined by relation $\mathcal{R}_{\mathcal{L}}$, and any construction of (S)VCKG, we build a CaP scheme $\mathsf{CaP}_{par} = (\mathsf{G}_p, \mathsf{LabGen}_p, \mathsf{Com}_p, \mathsf{P}_p, \mathsf{V}_p)$ and a NI Zap for the relation $\mathcal{R}_{par}$.

Relations of partial satisfiability. Formally, $\mathcal{R}_{par}$ consists of the tuples (gk, X, W) such that:

a) X consists of $\{\{x_i : i \in [L]\}, \mathbf{T}, \mathcal{SP}\}$, where x_i is a quadratic equation in the group described by gk and $\mathcal{SP}$ is a monotone span program which computes some predicate $P : \{0,1\}^L \to \{0,1\}$ such that CaP admits simulatable verifiable correlated key generation for P and the vector of labels $\mathbf{T}$,
b) Each statement x_i admits the simulation label t_i,
c) W is of the form $\{\tilde{W}_i : i \in [L]\}$, where for all i, $\tilde{W}_i = \{(\mathsf{t}_{i\ell}, \tilde{m}_{i\ell}) : \ell \in [n_i], \tilde{m}_{i\ell} = (b_i, m_{i\ell})\}$ is such that a) if $b_i = 0$, $m_{i\ell} = 0$ for all $\ell \in [n_i]$ and b) if $b_i = 1$, $W_i := \{(\mathsf{t}_{i\ell}, m_{i\ell}) : \ell \in [n_i]\}$ is such that $(gk, x_i, W_i) \in \mathcal{R}_{\mathcal{L}}$,
d) If $A := \{i \in [L] : b_i = 1\}$, then $P(\mathbf{v}_A) = 1$.

6.1 NIZK Proofs of Partial Satisfiability

The main idea of the construction is the following: the prover of the proof system for partial satisfiability runs $\mathsf{K}_{\mathsf{scorr}}$ on input the span program $\mathcal{SP}$ (of size m) encoded in X and some set A accepted by $\mathcal{SP}$. Algorithm $\mathsf{K}_{\mathsf{scorr}}$ returns a set of commitment keys $ck_1, \ldots, ck_m$ and the trapdoors for all ck_j, $j \in \rho^{-1}(A^c)$. The set A should correspond to the statements for which the prover has a real

witness. The proof of the rest of the statements can be simulated using the trapdoors output by $\mathsf{K}_{\mathsf{scorr}}$. For zero-knowledge, the simulator will run $\mathsf{SimCorr}$ to generate only hiding keys with their respective trapdoors. The trapdoor tk_j will be used to simulate the proof of the statement $x_{\rho(j)}$, which is possible because the statement $x_{\rho(j)}$ admits the simulation label $\mathsf{t}_{\rho(j)}$.

- $\mathsf{G}_p(1^\lambda)$: Runs $(gk, ck) \leftarrow \mathsf{G}(1^\lambda)$.
- $\mathsf{LabGen}_p(gk, ck, X, W)$: Runs $(ck, \Sigma, \mathsf{TK}_{A^c}) \leftarrow \mathsf{K}_{\mathsf{scorr}}(gk, ck, \mathcal{SP}, \mathbf{v_A}, \mathbf{T})$, it parses Σ as $\{ck_1, \ldots, ck_m\}$, and for each pair (i, ℓ), it outputs $(\mathsf{k}^p_{i\ell}, \mathsf{k}^s_{i\ell})$, where $\mathsf{k}^p_{i\ell} = (\mathsf{t}_{i\ell}, \tilde{\mathsf{t}}_{i\ell})$ and

$$(\tilde{\mathsf{t}}_{i\ell}, \mathsf{k}^s_{i\ell}) = \begin{cases} \tilde{\mathsf{t}}_{i\ell} = \{ck_j : j \in \rho^{-1}(i)\}, \mathsf{k}^s_{i\ell} = \{tk_j : j \in \rho^{-1}(i)\} & \text{if } i \in A^c, \\ \tilde{\mathsf{t}}_{i\ell} = \{ck_j : j \in \rho^{-1}(i)\}, \mathsf{k}^s_{i\ell} = 0 & \text{if } i \in A. \end{cases}$$

- $\mathsf{Com}_p(gk, ck, (\mathsf{k}^p_{i\ell}, \mathsf{k}^s_{i\ell}, m_{i\ell}))$: Parse $\mathsf{k}^p_{i\ell}$ as $(\mathsf{t}_{i\ell}, \{ck_j : j \in \rho^{-1}(i)\})$ and for each $i \in [L]$ and each $j \in \rho^{-1}(i)$, it defines

$$c_{i\ell j} := \begin{cases} c_{i\ell j} \leftarrow \mathsf{SimCom}(gk, ck_j, tk_j, \mathsf{t}_{i\ell}) & \text{if } i \in A^c, \\ c_{i\ell j} \leftarrow \mathsf{Com}(gk, ck_j, \mathsf{t}_{i\ell}, m_{i\ell}) & \text{if } i \in A. \end{cases}$$

  It outputs $(\mathsf{k}^p_{i\ell}, \bigcup_{j \in \rho^{-1}(i)} c_{i\ell j})$.
- $\mathsf{P}_p(gk, ck, X, Op, C)$: Receives as input the statement X, some set $C = \bigcup_{i \in [L]} \bigcup_{j \in \rho^{-1}(i)} C_{ij}$ which is the union of sets of commitments $C_{ij} = \{c_{i\ell j} : \ell \in [n_i]\}$, and a set $Op = \bigcup_{i \in [L]} \bigcup_{j \in \rho^{-1}(i)} Op_{ij}$ which is the union of the sets $Op_{ij} := \{(\mathsf{t}_{i\ell}, ck_j, tk_j, m_{i\ell}, r_{i\ell}) : \ell \in [n_i]\}$, where each Op_{ij} is a valid opening of C_{ij} (we assume that for simulated commitments $m_{i\ell}$ is just set to 0). For each $i \in [L]$ and for each $j \in \rho^{-1}(i)$,

$$\pi_j := \begin{cases} \pi_j \leftarrow \mathsf{SimProve}(gk, ck_j, tk_j, x_i, Op_{ij}) & \text{if } i \in A^c, \\ \pi_j \leftarrow \mathsf{P}(gk, ck_j, x_i, Op_{ij}) & \text{if } i \in A. \end{cases}.$$

  Let $\Pi_i := \{\pi_j : j \in \rho^{-1}(i)\}$ and output $\Pi = \bigcup_{i \in [L]} \Pi_i$.
- $\mathsf{V}_p(gk, ck, X, C, \Pi)$: Given the group key gk, a commitment key ck, a statement X (which includes a description of $\mathbf{T}$), a proof Π and a set of commitments C, algorithm V_p proceeds as follows:
  - From the public types of the commitments in C, it derives a list of commitment keys $\Sigma = \{ck_1, \ldots, ck_m\}$ (or outputs failure if this is not possible). This is done by checking that for each $i \in [L]$ and each $\ell \in [n_i]$, the public types $\mathsf{k}^p_{i\ell} = (\mathsf{t}_{i\ell}, \tilde{\mathsf{t}}_{i\ell})$ are consistently assigned. That is, for each $i \in [L]$, $\tilde{\mathsf{t}}_{i\ell}$ should encode the same set of cardinal $|\rho^{-1}(i)|$ of commitment keys $\{ck_j : j \in \rho^{-1}(i)\} \subset \mathcal{K}_{gk}$, regardless of ℓ.
  - It runs $b \leftarrow \mathsf{V}_{\mathsf{scorr}}(gk, ck, \Sigma, \mathbf{T})$ (the set of labels $\mathbf{T}$ is encoded in X). If $b = 0$, halts and outputs 0, else it proceeds.
  - For each $i \in [L]$, and each $j \in \rho^{-1}(i)$, it verifies that each of the proofs π_j of statement x_i is satisfied individually by running $\mathsf{V}(gk, ck_j, x_i, C_{ij}, \pi_j)$.

- It outputs 0 if any of these checks fails, else it outputs 1.

– $\mathsf{SimGen}_p(1^\lambda)$: Runs $(gk, ck, tk) \leftarrow \mathsf{SimGen}(1^\lambda)$.
– $\mathsf{SimLabGen}_p(gk, ck, tk, X)$: Runs $(ck, \Sigma, \mathsf{TK}) \leftarrow \mathsf{SimCorr}(gk, ck, tk, \mathbf{T})$ and for every $\mathsf{t}_{i\ell}$, it returns $\mathsf{k}^p_{i\ell} := (\mathsf{t}_{i\ell}, \{ck_j : j \in \rho^{-1}(i)\})$ and $\mathsf{k}^s_{i\ell} := \{tk_j : j \in \rho^{-1}(i)\}$.
– $\mathsf{SimCom}_p(gk, ck, (\mathsf{k}^p_{i\ell}, \mathsf{k}^s_{i\ell}))$: This algorithm parses $\mathsf{k}^p_{i\ell}$ as $(t_{i\ell}, \{ck_j : j \in \rho^{-1}(i)\})$ and $\mathsf{k}^s_{i\ell}$ as $\mathsf{k}^s_{i\ell} = \{tk_j : j \in \rho^{-1}(i)\}$. It outputs $\{(\mathsf{k}^p_{i\ell}, \bigcup_{j\in\rho^{-1}(i)} c_{i\ell j})\}$, where $c_{i\ell j} \leftarrow \mathsf{SimCom}(gk, ck_j, tk_j, \mathsf{t}_{i\ell})$.
– $\mathsf{SimProve}_p(gk, ck, tk, X, Op)$: For all $i \in [L]$, and all $j \in \rho^{-1}(i)$, and a set of commitment openings $Op = \bigcup_{i\in[L]} \bigcup_{j\in\rho^{-1}((i)} Op_{ij}$, this algorithm computes:

$$\pi_j \leftarrow \mathsf{SimProve}(gk, ck_j, tk_j, x_i, Op_{ij}).$$

It outputs $\Pi = \bigcup_{i\in[L]} \Pi_i$, where $\Pi_i := \{\pi_j : j \in \rho^{-1}(i)\}$.

Theorem 1. *CaP_{par} is a CaP scheme with perfect completeness, perfect soundness and composable zero-knowledge for $\mathcal{L}_{par}$.*

Proof. Perfect completeness follows from the completeness of the GS CaP and the fact, for all $i \in [L]$, x_i admits the simulation label t_i (else the prover could fail to compute a simulated proof for x_i). Perfect soundness follows from the perfect soundness of the GS CaP and the properties of SVCKG. Indeed, by property c) of SVCKG, if the verifier accepts the keys $\Sigma = \{ck_1, \ldots, ck_m\}$ (after running $\mathsf{V}_{\mathsf{scorr}}$), then $A := \rho(\{j : ck_j \in \mathcal{K}^{\mathsf{t}_{\rho(j)}}_{gk,\mathsf{bind}}\}) \in \Omega$. That is, there is some set $A \in \Omega$, such that for every $i \in A$ at least one $j \in \rho^{-1}(i)$ is a binding key for the label t_i. Therefore, for every $i \in A$, at least one of the proofs in the set Π_i (there are $|\rho^{-1}(i)|$ proofs of x_i) is generated with a binding key. By the perfect soundness of GS Proofs, this means $(gk, x_i, W_i) \in \mathcal{R}_{\mathcal{L}}$. Composable zero-knowledge holds because, by property b) of SVCKG, the keys output by $\mathsf{SimCorr}$ and by $\mathsf{K}_{\mathsf{scorr}}$ on a simulated key ck are identically distributed. The composable zero-knowledge property of GS Proofs guarantees that if x_i is a satisfiable quadratic equation (that is, if x_i is in the language accepted by GS Proofs), then a real proof computed with a simulated key has the same distribution as a simulated proof. On the other hand, if x_i is not satisfiable, then both in a real proof (computed with the output of $\mathsf{K}_{\mathsf{scorr}}$) or in a fake proof, the proof is simulated, so in both cases it follows the same distribution.

6.2 Non-Interactive Zap for Partial Satisfiability

The NI Zap for Satisfiability is constructed in a very similar way as the NIZK proofs of partial satisfiability, except that one uses as a building block the algorithms for VCKG (instead of SVCKG) and of course, the fact that $ck = \perp$. It is obvious that the resulting construction is complete and soundness follows from the same arguments as before, namely from property c) of the definition of VCKG. We next sketch the proof for computational WI.

Suppose an adversary B against the WI of the Zap outputs $(X, W_0, W_1) \in \mathcal{R}_{par}$. Each W_b encodes a set $A_b \in \Omega$ which specifies for which sets of equations W_b contains a real witness. If $A_0 = A_1$, then the adversary will not be able to distinguish a proof computed with W_0 or W_1 unless it breaks the composable zero-knowledge property of GS proofs, which implies that real proofs with different witnesses are computationally indistinguishable and that simulated proofs are independent of the witness. Therefore, we can assume that $A_0 \neq A_1$. But in this case, we can use B to construct an adversary D that breaks property b) of the VCKG scheme. Indeed, D gives to its challenger $(\mathbf{v}_{A_0}, \mathbf{v}_{A_1})$ and receives $(\Sigma, \mathsf{TK}_{A_0^c \cap A_1^c})$, with Σ generated from A_b, for $b \leftarrow \{0,1\}$. Even if b is unknown to D, it can compute a proof of the statement X. Indeed, for all $i \in [L]$, and all $j \in \rho^{-1}(i)$, D can compute a proof π_j of the statement x_i, as follows:

- if $i \in A_0 \cup A_1$, π_j is a real proof as it can extract a witness for x_i from W_0 or W_1,
- if $i \in (A_0 \cup A_1)^c = A_0^c \cap A_1^c$, π_j is a simulated proof computed with $\mathsf{TK}_{A_0^c \cap A_1^c}$.

Finally, D gives the keys Σ and the proof of X to B, who outputs a bit b', and D forwards this bit to its challenger.

Since in the GS CaP, real proofs with a simulated key have the same distribution as simulated proofs, the proof given to B follows the same distribution as a proof generated with W_b. Thus, $|\Pr[b' = b] - 1/2|$ is non-negligible, so D is successful with non-negligible probability.

7 (Simulatable) Verifiable Correlated Key Generation: Constructions

We give some constructions of verifiable correlated key generation in different flavors. Let $\mathcal{SP}$ be a monotone span program computing $\Omega \subset \mathcal{P}([L])$. From the construction of proofs of partial satisfiability of last section, we know that to prove that there is a set of indexes $A \in \Omega$ such that all $\mathcal{S}_i$, for $i \in A$ are satisfiable, the GS CaP must admit (S)VCKG for a vector $\mathbf{T} = (\mathsf{t}_1, \ldots, \mathsf{t}_L)$ such that each equation $\mathcal{S}_i$ admits t_i as a simulation label. Therefore, we are interested in constructing P-verifiable correlated key generation for as many possible types of vectors $\mathbf{T}$ and general predicates P, since this means that our CaP for partial satisfiability will admit a wider class of languages.

7.1 SVCKG for $\mathrm{sca}_{\hat{\mathbb{G}}}$ and MSPs

This construction of SVCK works for any $P : \{0,1\}^L \to \{0,1\}$ computed by a a monotone span program $\mathcal{SP}$, but only for the vector of labels $\mathbf{T} = (\mathrm{sca}_{\hat{\mathbb{G}}}, \ldots, \mathrm{sca}_{\hat{\mathbb{G}}})$ (the case $\mathbf{T} = (\mathrm{sca}_{\check{\mathbb{H}}}, \ldots, \mathrm{sca}_{\check{\mathbb{H}}})$ is defined in a similar way in $\check{\mathbb{H}}$).

- $\mathsf{K}_{\mathsf{scorr}}(gk, ck, \mathcal{SP}, A, \mathbf{T})$: The algorithm receives as input gk, $ck = (\hat{\mathbf{u}}, \hat{\mathbf{v}}, \hat{\mathbf{w}}, \check{\mathbf{u}}, \check{\mathbf{v}}, \check{\mathbf{w}})$, the description of a MSP $\mathcal{SP}$, a set of indexes $A \subset [L]$ such that $\mathcal{SP}$ accepts A and a vector of labels $\mathbf{T}$. It proceeds as follows:

1 It samples $\boldsymbol{\tau}, \boldsymbol{\zeta} \in \mathbf{Im}(\mathbf{M}^*)$ uniformly at random conditioned on a) $\zeta_0 = 1$, $\zeta_j = 0$ for all $j \in \rho^{-1}(A^c)$ and b) $\tau_0 = 0$ (as in lemma 1.)

2 It defines $\hat{\mathbf{z}}_j := \tau_j \hat{\mathbf{v}} + \zeta_j \hat{\mathbf{w}}$, $j \in [m]$ and outputs $(ck, \Sigma, \mathsf{TK}_{A^c})$, where $\Sigma = \{ck_1, \ldots, ck_m\}$, $ck_j := (\hat{\mathbf{u}}_j, \hat{\mathbf{v}}, \hat{\mathbf{z}}_j, \check{\mathbf{u}}, \check{\mathbf{v}}, \check{\mathbf{w}})$, $\hat{\mathbf{u}}_j := \hat{\mathbf{z}}_j + (\hat{0}, \hat{g})^\top$ and $\mathsf{TK}_{A^c} := \{\tau_j : j \in \rho^{-1}(A^c)\}$.

– $\mathsf{V}_{\mathsf{scorr}}(gk, ck, \mathcal{SP}, \Sigma, \mathbf{T})$: Parse each key as $ck_j := (\hat{\mathbf{u}}_j, \hat{\mathbf{v}}_j, \hat{\mathbf{w}}_j, \check{\mathbf{u}}_j, \check{\mathbf{v}}_j, \check{\mathbf{w}}_j)$, and reject if, for some $j \in [m]$, $ck_j \notin \mathcal{K}_{gk}$, $\hat{\mathbf{v}}_j \neq \hat{\mathbf{v}}$, $\check{\mathbf{u}}_j \neq \check{\mathbf{u}}$ or $\check{\mathbf{v}}_j \neq \check{\mathbf{v}}$. Else, define $\hat{\mathbf{z}}_0 := \hat{\mathbf{w}}$ and $\hat{\mathbf{Z}} := (\hat{\mathbf{z}}_0 || \hat{\mathbf{z}}_1 || \ \ldots \ || \hat{\mathbf{z}}_m)$. Output 1 if $\hat{\mathbf{Z}}\mathbf{M} = \hat{\mathbf{0}}_{2\times d}$ holds, else output 0.

– $\mathsf{SimCorr}(gk, ck, tk, \mathcal{SP}, \mathbf{T})$: The algorithm receives $(gk, ck, tk) \leftarrow \mathsf{SimGen}(1^\lambda)$, with $tk = (\omega, \sigma)$. It samples a uniform vector in $\boldsymbol{\mu}^\top = (\mu_0, \ldots, \mu_m) \in \mathbf{Im}(\mathbf{M}^*)$ subject to the the restriction $\mu_0 = \omega$. For all $j \in [m]$, it defines $\hat{\mathbf{z}}_j := \mu_j \hat{\mathbf{v}}$, $\hat{\mathbf{u}}_j := \hat{\mathbf{z}}_j + (\hat{0}, \hat{g})^\top$ and $ck_j := (\hat{\mathbf{u}}_j, \hat{\mathbf{v}}, \hat{\mathbf{z}}_j, \check{\mathbf{u}}, \check{\mathbf{v}}, \check{\mathbf{w}})$. It outputs $(ck, \Sigma, \mathsf{TK})$, where $\Sigma = \{ck_1, \ldots, ck_m\}$ and $\mathsf{TK} := \{\mu_j : j \in [m]\}$.

Lemma 2. *The GS CaP scheme described in section 4 admits simulatable verifiable correlated key generation for labels* $\mathbf{T} = (sca_{\hat{\mathbb{G}}}, \ldots, sca_{\hat{\mathbb{G}}})$.

Proof. We prove that the algorithms described above satisfy points a), b), c) of definition 8.

By definition of $\boldsymbol{\zeta}$, $\zeta_j = 0$ for all $j \in \rho^{-1}(i)$, $i \in A^c$. Therefore, for all $i \in A^c$, $\hat{\mathbf{z}}_j = \tau_j \hat{\mathbf{v}}$, so according to figure (1), $ck_j \in\in \mathcal{K}^{\mathrm{sca}_{\hat{\mathbb{G}}}}_{gk,\mathsf{hid}}$ and the corresponding trapdoor is $tk_j = \tau_j$.

To see b), note that all the keys output by $\mathsf{SimCorr}$ are obviously in $\mathcal{K}^{\mathrm{sca}_{\hat{\mathbb{G}}}}_{gk,\mathsf{hid}}$ and the trapdoor is valid. We just have to argue that the output of the algorithm $\mathsf{SimCorr}$ has the same distribution as the output of $\mathsf{K}_{\mathsf{scorr}}$ when $(gk, ck, tk) \leftarrow \mathsf{SimGen}(1^\lambda)$. In that case, $\hat{\mathbf{w}} = \omega \hat{\mathbf{v}}$ and $\mathsf{K}_{\mathsf{scorr}}$ outputs $\hat{\mathbf{z}}_j = \tau_j \hat{\mathbf{v}} + \zeta_j \hat{\mathbf{w}} = (\tau_j + \omega\zeta_j)\hat{\mathbf{v}}$, for all $j \in [m] \cup \{0\}$. Let $\boldsymbol{\nu} := \boldsymbol{\tau} + \omega\boldsymbol{\zeta}$. The constraints imposed on $\boldsymbol{\tau}, \boldsymbol{\zeta}$ imply that $\boldsymbol{\nu}$ is uniform conditioned on a) $\boldsymbol{\nu} \in \mathbf{Im}(\mathbf{M}^*)$, b) $\nu_0 = \omega$ and c) $\nu_j = \tau_j$ for all $j \in \rho^{-1}(A^c)$. Because of part 3) of lemma 1, if $\{j_1, \ldots, j_\ell\} = \rho^{-1}(A^c)$, the distribution of $(\tau_{j_1}, \ldots, \tau_{j_\ell})$, is the uniform one conditioned on $\boldsymbol{\tau} \leftarrow \mathbf{Im}(\mathbf{M}^*)$. We conclude that $\boldsymbol{\nu}$ is a uniformly random vector in $\mathbf{Im}(\mathbf{M}^*)$ conditioned to $\nu_0 = \omega$, so the outputs of $\mathsf{SimCorr}$ and $\mathsf{K}_{\mathsf{scorr}}$ are identically distributed, which proves b).

Finally, for c), let $\Sigma = \{ck_1, \ldots, ck_m\}$ be some set of keys accepted by the verifier, i.e. some set such that $\Sigma \subset \mathcal{K}_{gk}$ and $\hat{\mathbf{Z}}\mathbf{M} = \hat{\mathbf{0}}_{2\times d}$. Since if $(gk, ck) \leftarrow \mathsf{G}(1^\lambda)$, the vectors $\hat{\mathbf{v}}, \hat{\mathbf{w}}$ are a basis of $\hat{\mathbb{G}}^2$, we can write each column of $\mathbf{Z}$ (numbered from 0 to m) as $\hat{\mathbf{z}}_j = \hat{\mathbf{v}}\tau_j + \hat{\mathbf{w}}\zeta_j$, for some arbitrary values τ_j, ζ_j. In this notation, $\hat{\mathbf{Z}} = \hat{\mathbf{v}}\boldsymbol{\tau}^\top + \hat{\mathbf{w}}\boldsymbol{\zeta}^\top$. Replacing in the verification equation, we have that $(\hat{\mathbf{v}}\boldsymbol{\tau}^\top + \hat{\mathbf{w}}\boldsymbol{\zeta}^\top)\mathbf{M} = \hat{\mathbf{0}}_{2\times d}$. But since $\hat{\mathbf{v}}, \hat{\mathbf{w}}$ are linearly independent, the equation can only hold if $\boldsymbol{\zeta}^\top\mathbf{M} = \mathbf{0}_{1\times d}$. We can now apply lemma 1, part 1), to conclude that $A := \rho(\{j : \zeta_j \neq 0\}) \in \Omega$. But if $\zeta_j \neq 0$, then ck_j is a binding key for the label $\mathrm{sca}_{\hat{\mathbb{G}}}$, which proves c).

7.2 VCKG for $\mathrm{sca}_{\hat{\mathbb{G}}}$ and MSPs

This construction is almost identical to the previous one except that for the non-simulatable case. For the OR predicate of two variables, it matches exactly the GOS construction.

- $\mathsf{K}_{\mathsf{corr}}(gk, \mathcal{SP}, A, \mathbf{T})$: The algorithm first runs the key generation algorithm of the CaP scheme obtaining $gk \leftarrow \mathsf{G}_0(1^\lambda)$. Then it proceeds as in the $\mathsf{K}_{\mathsf{scorr}}$ algorithm, except that the vectors $\hat{\mathbf{z}}_j$ are now defined by $\hat{\mathbf{z}}_j = \tau_j \hat{\mathbf{v}} + \zeta_j (\hat{0}, \hat{g})^\top$.
- $\mathsf{V}_{\mathsf{corr}}(gk, \Sigma, \mathbf{T})$: The algorithm proceeds as algorithm $\mathsf{K}_{\mathsf{scorr}}$ but with $\hat{\mathbf{z}}_0 := (\hat{0}, \hat{g})^\top$.

The proof follows the same lines as the previous one, the only relevant difference is that to prove point b) of definition 7, we use the DDH Assumption in $\hat{\mathbb{G}}$. The argument is very similar to [15] and is omitted.

7.3 Other Labels

We could not construct (S)VCKG for $\mathbf{T} = (\mathrm{com}_{\hat{\mathbb{G}}}, \ldots, \mathrm{com}_{\hat{\mathbb{G}}})$ for the original GS CaP based on SXDH. The core of our construction is to use secret sharing techniques to guarantee that at least a certain subset of the vectors $\hat{\mathbf{z}}_1, \ldots, \hat{\mathbf{z}}_m$ is linearly independent of $\hat{\mathbf{v}}$. The problem is that for group elements, the soundness condition is exactly the opposite, namely it requires linear dependency of $\hat{\mathbf{v}}, \hat{\mathbf{u}}$ (see Fig. 1). In appendix C.1 we extend the GS CaP based on SXDH to admit new labels, which allow to commit to group elements and to scalars in a different way, with respective labels $\mathrm{com}^{par}_{\hat{\mathbb{G}}}$ and $\mathrm{sca}^{par}_{\hat{\mathbb{G}}}$ ($\mathrm{sca}^{par}_{\hat{\mathbb{H}}}, \mathrm{com}^{par}_{\hat{\mathbb{H}}}$). This new instantiation of GS proofs $\tilde{\mathsf{CaP}}_{GS}$ admits (S)VCKG for these new label types, i.e. for any vector $\mathbf{T} = (\mathsf{t}_1, \ldots, \mathsf{t}_L)$, for $\mathsf{t}_i \in \{\mathrm{com}^{par}_{\hat{\mathbb{G}}}, \mathrm{sca}^{par}_{\hat{\mathbb{G}}}\}$. This means that we can apply our approach to many more sets of equations $\mathcal{S}_1, \ldots, \mathcal{S}_L$, but at some efficiency cost, because $\tilde{\mathsf{CaP}}_{GS}$ is less efficient that the original GS CaP.

7.4 Efficiency Discussion

Proof size. The proof that some sets of equations $\mathcal{S}_i, i \in [L]$ are partially satisfiable requires the prover to send the keys Σ and then a proof (real or simulated) of satisfiability of each $\mathcal{S}_i$. The size of the proofs depends thus on the equations in $\mathcal{S}_i$. Therefore, to understand the performance of our proof system for partial satisfiability, the best thing is to analyze its *overhead*, which is the difference between the size of our proof and the sum of the sizes of a simulated proof of $\mathcal{S}_i$, for all $i \in [L]$. The number of elements necessary to commit to all the variables in $\mathcal{S}_i$ is also counted as part of the proof of $\mathcal{S}_i$. That is, the overhead is the difference between proving partial satisfiability and proving that all of $\mathcal{S}_i, i \in [L]$ hold, using independent variables for each of the $\mathcal{S}_i$.

Efficient encoding of Σ. It is quite important for the efficiency comparison to note that the set Σ of keys output by the correlated key generation algorithm admit a more efficient encoding. In all our constructions, the description of Σ

requires to give m vectors $\hat{\mathbf{z}}_1, \ldots, \hat{\mathbf{z}}_m$. Instead of letting the prover choose Σ and then verifying if the keys are valid with the verification algorithm $\mathsf{V}_{\mathsf{corr}}$ (or $\mathsf{V}_{\mathsf{scorr}}$) by checking whether $\hat{\mathbf{Z}}\mathbf{M} = \mathbf{0}$ (where the last m columns of $\hat{\mathbf{Z}}$ are $\hat{\mathbf{z}}_1, \ldots, \hat{\mathbf{z}}_m$ and the first is $\hat{\mathbf{z}}_0 := \hat{\mathbf{w}}$), it is enough to let the prover output only $\hat{\mathbf{z}}_1, \ldots, \hat{\mathbf{z}}_{m-d}$. Indeed, let $\mathbf{M}_0^*$ be the $(m+1-d) \times (m+1-d)$ minor formed by the first $m+1-d$ rows of $\mathbf{M}^*$ and $\mathbf{M}_1^*$ the minor formed by the rest of the rows. Reordering if necessary, we can assume that $\mathbf{M}_0^*$ is invertible. Then, the following holds:

Lemma 3. *Let $\hat{\mathbf{Z}}$ be an arbitrary matrix in $\hat{\mathbb{G}}^{2\times(m+1)}$, with columns $\hat{\mathbf{z}}_0, \hat{\mathbf{z}}_1, \ldots, \hat{\mathbf{z}}_m$, then:*

$$\hat{\mathbf{Z}}\mathbf{M} = \mathbf{0} \iff \left(\hat{\mathbf{z}}_{m+1-d} || \ldots || \hat{\mathbf{z}}_m\right) = \left(\hat{\mathbf{z}}_0 || \hat{\mathbf{z}}_1 || \ldots || \hat{\mathbf{z}}_{m-d}\right) \left(\mathbf{M}_1^* (\mathbf{M}_0^*)^{-1}\right)^\top. \quad (3)$$

Proof. Denote as $\mathbf{f}_1, \mathbf{f}_2$ the rows of $\mathbf{Z}$. Note that $\hat{\mathbf{Z}}\mathbf{M} = \mathbf{0}$ if and only if $\mathbf{f}_1, \mathbf{f}_2 \in \mathbf{Im}(\mathbf{M}^*)$, since the columns of $\mathbf{M}^*$ are a basis of all vectors $\mathbf{f}$ such that $\mathbf{f}^\top \mathbf{M} = \mathbf{0}$ (by definition of the parity check matrix). On the other hand, a vector $\mathbf{f} \in \mathbf{Im}(\mathbf{M}^*)$ if and only if there exists some $\mathbf{w} \in \mathbb{Z}_q^2$ such that $\mathbf{f}^\top = (\mathbf{M}_0^* \mathbf{w} || \mathbf{M}_1^* \mathbf{w})^\top$. Since $\mathbf{M}_0^*$ has full rank this is equivalent to $\mathbf{f} \in \mathbf{Im}(\mathbf{M}^*)$ if and only if $(f_{m+1-d}, \ldots, f_m)^\top = \mathbf{M}_1^* (\mathbf{M}_0^*)^{-1} (f_0, f_1, \ldots, f_{m-d})^\top$. The statement follows from applying this reasoning to $\mathbf{f}_1, \mathbf{f}_2$.

The lemma implies that we can eliminate the test of algorithms $\mathsf{V}_{\mathsf{corr}}$, $\mathsf{V}_{\mathsf{scorr}}$, and instead let the verifier compute the last d columns of $\hat{\mathbf{Z}}$ on its own using $\hat{\mathbf{z}}_1, \ldots, \hat{\mathbf{z}}_{m-d}$. This means that sending Σ requires only $2(m-d)$ group elements.

Comparison with previous work. We compare our results with the approach of Groth [14] (simplified by Camenisch *et al.* [6]) for the statement "1-out-of-L sets of equations $\mathcal{S}_1, \ldots, \mathcal{S}_L$ are satisfiable". They construct a "compiler" which takes some sets of satisfiable equations $\mathcal{S}_1, \ldots, \mathcal{S}_L$ and turns them into a single set of equations which is only satisfiable if one of the $\mathcal{S}_i$'s is. The compiler works by (renaming if necessary) assuming the $\mathcal{S}_1, \ldots, \mathcal{S}_L$ have independent variables and adding variables $b_1, \ldots, b_{L-1}$, $b_i \in \{0,1\}$ and defining $b_L := 1 - b_1 - \ldots - b_{L-1}$. For each $i \in [L]$, b_i modifies the equations in $\mathcal{S}_i$ so that they admit the trivial solution if $b_i = 0$ and that they remain unchanged if $b_i = 1$. The overhead is the cost of proving $b_i \in \{0,1\}$ for all $i \in [L-1]$, which is $(L-1)(6|\hat{\mathbb{G}}| + 6|\check{\mathbb{H}}|)$. Our solution is notably more efficient (only $2|L-1||\hat{\mathbb{G}}|$) when the vector of admissible simulation labels of $\mathcal{S}_1, \ldots, \mathcal{S}_L$ is $(\mathrm{sca}_{\hat{\mathbb{G}}}, \ldots, \mathrm{sca}_{\hat{\mathbb{G}}})$, although it is not clear what happens for other $\mathbf{T}$ (see the efficiency discussion in appendix C.1).

8 Examples

In this section we give two examples of P-Simulatable Verifiable Correlated Key Generation. Throughout this section, given a set $S \subset \mathbb{Z}_q$ and some $i \in S$, $\lambda_i^S(X) := \prod_{j \in S \setminus \{i\}} \frac{X-j}{i-j}$.

Example 2. (Or of two equations).

$$\mathbf{M} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \qquad \mathbf{M}^* = \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ -2 & -1 \end{pmatrix}$$

$$\mathbf{M}_0^* = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \qquad \mathbf{M}_1^* = \begin{pmatrix} -2 & -1 \end{pmatrix} \qquad \mathbf{T}_{\mathbf{M}^*} = \mathbf{M}_1^*(\mathbf{M}_0^*)^{-1} = \begin{pmatrix} -1 & -1 \end{pmatrix}.$$

$(\mathbf{M}, \rho)$ is a monotone span program, where $\rho(1) = 1$, $\rho(2) = 2$ which computes the predicate Or of two variables, $(\mathbf{M}^*, \rho)$ computes the dual predicate. The algorithm $\mathsf{K}_{\mathsf{scorr}}$ (or $\mathsf{K}_{\mathsf{corr}}$), receives a vector $\mathbf{v}_A \in \{0,1\}^2$ such that $P(\mathbf{v}_A) = 1$. Since P is monotone, alternatively we can say that it receives a set $A \subset \{1,2\}$ such that $|A| \geq 1$. It then generates $\boldsymbol{\zeta}, \boldsymbol{\tau}$ according to one of these two possibilities:

1. If $A = \{1\}$, it sets $\boldsymbol{\zeta} = (1,-1,0) = \mathbf{M}^* \begin{pmatrix} 1 \\ -2 \end{pmatrix}$, $\boldsymbol{\tau} = (0,r,-r) = \mathbf{M}^* \begin{pmatrix} 0 \\ -r \end{pmatrix}$, for $r \leftarrow \mathbb{Z}_q$, that is, $\boldsymbol{\zeta}, \boldsymbol{\tau} \in \mathbf{Im}(\mathbf{M}^*)$ are uniform conditioned on $\zeta_0 = 1$, $\zeta_2 = 0$ and $\tau_0 = 0$.
2. If $A = \{2\}$, $\boldsymbol{\zeta} = (1,0,-1) = \mathbf{M}^* \begin{pmatrix} 1 \\ -1 \end{pmatrix}$, $\boldsymbol{\tau} = (0,r,-r) = \mathbf{M}^* \begin{pmatrix} 0 \\ -r \end{pmatrix}$, for $r \leftarrow \mathbb{Z}_q$, that is, $\boldsymbol{\zeta}, \boldsymbol{\tau} \in \mathbf{Im}(\mathbf{M}^*)$ are uniform conditioned on $\zeta_0 = 1$, $\zeta_1 = 0$ and $\tau_0 = 0$.

If $A = \{1,2\}$, it can choose one of the previous alternatives arbitrarily, so we can assume w.l.o.g. that $|A| = 1$. Then, according to the type of labels which it receives, it proceeds as follows:

– If $\mathbf{T} = (\mathrm{sca}_{\hat{\mathbb{G}}}, \mathrm{sca}_{\hat{\mathbb{G}}})$, it sets $\hat{\mathbf{z}}_1 = \zeta_1 \hat{\mathbf{w}} + \tau_1 \hat{\mathbf{v}}$, $\hat{\mathbf{z}}_2 = \zeta_2 \hat{\mathbf{w}} + \tau_2 \hat{\mathbf{v}}$, the trapdoor for the key indexed by A^c is $\pm r$. That is, we have:
  1. If $A = \{1\}$, $\hat{\mathbf{z}}_1 = -\hat{\mathbf{w}} + r\hat{\mathbf{v}}$, $\hat{\mathbf{z}}_2 = -r\hat{\mathbf{v}}$, $tk_2 = -r$.
  2. If $A = \{2\}$, $\hat{\mathbf{z}}_1 = r\hat{\mathbf{v}}$, $\hat{\mathbf{z}}_2 = -\hat{\mathbf{w}} - r\hat{\mathbf{v}}$, $tk_1 = r$.

As we explained in section 6, instead of letting the prover of CaP_{par} run $\mathsf{K}_{\mathsf{scorr}}$ and output $(\hat{\mathbf{z}}_0 := \hat{\mathbf{w}}, \hat{\mathbf{z}}_1, \hat{\mathbf{z}}_2)$ and then let the verifier run $\mathsf{V}_{\mathsf{corr}}$ to see if the keys are properly generated, one can gain some efficiency by letting the prover only send $\hat{\mathbf{z}}_1$, and recover $\hat{\mathbf{z}}_2$ as:

$$\hat{\mathbf{z}}_2 = \hat{\mathbf{Z}}_0 \mathbf{T}_{\mathbf{M}^*}^{\top},$$

where $\hat{\mathbf{Z}}_0 = \begin{pmatrix} \hat{\mathbf{z}}_0 || \hat{\mathbf{z}}_1 \end{pmatrix}$.

Example 3. k-out-of-L equations.

$$\mathbf{M} = \begin{pmatrix} 1 & 0 & \dots & 0 \\ 1 & 1 & \dots & 1 \\ \vdots & \vdots & & \vdots \\ 1 & L & \dots & L^{k-1} \end{pmatrix} \qquad \mathbf{M}^* = \begin{pmatrix} 1 & 0 & \dots & 0 \\ -\lambda_1^{[L]}(0) & -\lambda_1^{L}(0) & \dots & -\lambda_1^{[L]}(0) \\ -\lambda_2^{[L]}(0) & -2\lambda_1^{[L]}(0) & \dots & -2^{L-k}\lambda_2^{[L]}(0) \\ \vdots & \vdots & & \vdots \\ -\lambda_L^{[L]}(0) & -L\lambda_L^{[L]}(0) & \dots & -L^{L-k}\lambda_L^{[L]}(0) \end{pmatrix}.$$

This is the definition of $\mathbf{M}$, $\mathbf{M}^*$ which is consistent with the definition given in section 2.3 but as in the more efficient version of our protocol, the matrix

$\mathbf{M}$ does not play any role, we can choose a more efficient encoding of $\mathbf{M}^*$ (this allows to save in computation for the prover), namely:

$$\mathbf{M}^* = \begin{pmatrix} 1 & 0 & \dots & 0 \\ 1 & 1 & \dots & 1 \\ \vdots & \vdots & & \vdots \\ 1 & L & \dots & L^{L-k} \end{pmatrix}.$$

It is obvious that with both definitions $\mathbf{M}^*$ computes the same span program as in one case we just have replaced each row by some scalar multiple of itself, and this does not change the linear dependencies among the rows. It can be easily verified that:

$$\mathbf{T}_{\mathbf{M}^*} = \begin{pmatrix} \lambda_0^S(L-k+1) & \lambda_1^S(L-k+1) & \dots & \lambda_{L-k}^S(L-k+1) \\ \lambda_1^S(L-k+2) & \lambda_0^S(L-k+2) & \dots & \lambda_{L-k}^S(L-k+2) \\ \vdots & \vdots & & \vdots \\ \lambda_0^S(L) & \lambda_1^S(L) & \dots & \lambda_{L-k}^S(L) \end{pmatrix},$$

where $S = \{0, 1, \dots, L-k\}$. That is, if $\hat{\mathbf{z}}_i^\top = (\hat{z}_{i1}, \hat{z}_{i2})$, then both $(\hat{z}_{01}, \hat{z}_{11}, \dots, \hat{z}_{L1})$ and $(\hat{z}_{02}, \hat{z}_{12}, \dots, \hat{z}_{L2})$ are evaluations of some univariate polynomial of degree at most $L-k$ in the points $0, 1, \dots, L$ and for any $j = 1, 2$, the transformation matrix $\mathbf{T}_{\mathbf{M}^*}$ which allows to compute $(\hat{z}_{L-k+1j}, \dots, \hat{z}_{Lj})$ from $(\hat{z}_{0j}, \dots, \hat{z}_{L-kj})$ is simply a polynomial interpolation matrix. Further, given some $A \in \Omega_{(k,L)}$, the vectors $\boldsymbol{\zeta}, \boldsymbol{\tau}$ can be defined as the evaluation in $0, 1, \dots, L$ of two uniformly random polynomials $\zeta(x), \tau(x)$ of degree at most $L-k$ conditioned on 1) $\zeta(0) = 1$ and $\zeta(i) = 0$ for all $i \in A^c$ (ζ always exists since $|A^c| \leq L-k$ and is unique if $|A| = k$) and 2) $\tau(0) = 0$.

Another paradigmatic example of access structure realizable by a monotone span program is the threshold hierarchical one, see Tassa [26]. Although we did not include any example, recall that our construction is also for non-ideal sss, that is, the monotone span program might have more than one row with the same label (this is important since there are not that many known instances of ideal sss for interesting access structures).

9 Applications

Next we discuss some applications of our results, but we expect that many more can be found, for instance, in the design of signature schemes with complex functionalities in the standard model in bilinear groups like attribute-based signatures. Another interesting direction to explore is the application to anonymous credentials.

Proving that some commitments open to $\mathbf{b} \in \{0,1\}^L$*, and* $\mathbf{wt}(\mathbf{b}) = 1$. Given some group key gk and some commitment keys $\hat{\mathbf{w}}, \hat{\mathbf{v}}$, our results allow to give more efficient proofs that each of the commitments in $\{\hat{\mathbf{c}}_i : i \in [L]\} \subset \hat{\mathbb{G}}^2$ opens to a bit $b_i \in \{0,1\}$, and that $\sum_{i \in [L]} b_i = 1$.

Alternatively, if we let $\hat{\mathbf{c}}_L := \hat{\mathbf{w}} - \sum_{i\in[L-1]} \hat{\mathbf{c}}_i$, it is enough to prove that each of the commitments in $\{\hat{\mathbf{c}}_i : i \in [L-1]\}$ opens to a bit $b_i \in \{0,1\}$. In the asymmetric instantiation of bilinear groups of GS proofs, this requires $(L-1)(4|\hat{\mathbb{G}}| + 6|\check{\mathbb{H}}|)$ elements for the proofs and $2(L-1)|\hat{\mathbb{G}}|$ for the description of $\hat{\mathbf{c}}_1, \ldots, \hat{\mathbf{c}}_{L-1}$. On the other hand, we can encode the statement as a partial satisfiability statement as:

$$\text{“}(L-1)\text{-out-of-}L \text{ of } (\{\exists r_1 \in \mathbb{Z}_q : \hat{\mathbf{c}}_1 = r_1\hat{\mathbf{v}}\}, \ldots, \{\exists r_L \in \mathbb{Z}_q : \hat{\mathbf{c}}_L = r_L\hat{\mathbf{v}}\}) \text{ hold.}'' \tag{4}$$

Each statement $x_i = \{\exists r_i \in \mathbb{Z}_q : \hat{\mathbf{c}}_i = r_i\hat{\mathbf{v}}\}$ can be encoded as two linear equations (with equation label $\mathsf{MLin}_{\check{\mathbb{H}}}$), and they both admit the simulation label $\mathrm{sca}_{\check{\mathbb{H}}}$. The size of the proof is thus $2|\check{\mathbb{H}}|$ for the description of the correlated keys Σ (see section 8), $L(2|\check{\mathbb{H}}| + 2|\hat{\mathbb{G}}|)$ for the proof (real or simulated) of x_i and $2(L-1)|\hat{\mathbb{G}}|$ for the description of $\hat{\mathbf{c}}_1, \ldots, \hat{\mathbf{c}}_{L-1}$. In conclusion, our approach saves $O(L)$ elements in the proof size.

We note that if $\hat{\mathbf{w}}, \hat{\mathbf{v}}$ are part of some common reference string generated by a trusted party, we can prove (4) in zero-knowledge, but we can also take $\hat{\mathbf{w}} = (\hat{0}, \hat{g})^\top$ and let the prover generate $\hat{\mathbf{v}}$ so that no other party knows its discrete logarithm. Then using the NI Zap for partial satisfiability, the prover can create a NIWI proof of (4) without a trusted setup.

Proving membership in a list. Chandran, Groth and Sahai [8] showed how to prove that a committed value is in some public list $\{\lambda_1, \ldots, \lambda_N\} \subset \check{\mathbb{H}}$ with proof size $O(\sqrt{N})$. The main idea is to write the list elements in a matrix $\mathbf{R}$ of size $L \times L$, $L := \sqrt{N}$ and then give two sets of commitments $\{\hat{\mathbf{c}}_i : i \in [L]\}$, $\{\hat{\mathbf{d}}_i : i \in [L]\} \subset \hat{\mathbb{G}}^2$, each opening to a different bit string of weight 1. Without going into details, using some homomorphic properties of the commitments, the prover uses one of the bit strings to (privately) select a row i of the matrix $\mathbf{R}$ and the other to select a column j. With some additional checks, this convinces the verifier that a commitment $\hat{\mathbf{c}}$ opens to some (secret) position i, j of the matrix $\mathbf{R}$. In summary, for the proof of membership in a list of size N we need to prove twice a statement of the type 4, so our results allow to save $O(\sqrt{N})$ group elements.

Ring signatures. Ring signatures [22] allow a signer to sign on behalf of an ad-hoc group to which it belongs, anonymously. The proof of membership in a list of size $O(\sqrt{N})$ was designed by Chandran, Groth and Sahai [8] with the objective of designing more efficient ring signatures. Their scheme (with a signature size of $O(\sqrt{N})$ when the ring size is N) has the shortest signature size of all the schemes known in the standard model. Our savings for the proof of membership translate directly into savings for this construction.

Simulation-sound NIZKs. Simulation-sound NIZKs [21,23] are non-interactive zero-knowledge proofs with a stronger soundness requirement. More specifically, no prover should be able to construct a false proof which is accepted by the verifier even after seeing several simulated proofs of false statements. This notion

is useful to construct IND-CCA2 encryption schemes following the Naor-Yung paradigm [21].

One technique to build simulation-sound proofs of satisfiability of some set of equations $\mathcal{S}$ over a bilinear group suggested by Groth [14] and subsequently explored by several papers with small variations [6,17], is to give a GS proof of the statement: "$\mathcal{S}$ is satisfiable" or "$\hat{\mathbf{c}}$ is a commitment to some signature". Real proofs will use a witness for $\mathcal{S}$, while simulated proofs will prove the other branch of the statement, using as simulation trapdoor the secret key of the signature.

In general, the technique of Groth for constructing simulation-sound proofs might not be the most efficient for all equation types $\mathcal{S}$, (for instance if $\mathcal{S}$ encodes membership in a linear space of $\hat{\mathbb{G}}^n$, see [1]) but when it is, one should check if our improvements apply. For instance, in the simulation sound proof of Camenisch *et al.* [6], one has to prove the OR of two equations which admit the label $(\mathrm{sca}_{\hat{\mathbb{G}}}, \mathrm{sca}_{\hat{\mathbb{G}}})$, and this has an overhead of only $2|\hat{\mathbb{G}}|$, as opposed to the $6|\hat{\mathbb{G}}|+6|\check{\mathbb{H}}|$ elements originally computed in [6]. On the other hand, our techniques do not seem to help for the simulation sound proof of [17].

References

1. M. Abdalla, F. Benhamouda, and D. Pointcheval. Disjunctions for hash proof systems: New constructions and applications. Cryptology ePrint Archive, Report 2014/483, 2014. `http://eprint.iacr.org/2014/483`. 25
2. M. Bellare and S. Goldwasser. New paradigms for digital signatures and message authentication based on non-interative zero knowledge proofs. In G. Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 194–211, Santa Barbara, CA, USA, Aug. 20–24, 1989. Springer, Berlin, Germany. 1
3. G. Blakley. Safeguarding cryptographic keys. *Proceedings of the National Computer Conference, American Federation of Information*, Processing Societies Proceedings(48):313–317, 1979. 7
4. M. Blum, P. Feldman, and S. Micali. Non-interactive zero-knowledge and its applications. *STOC'88*, pages 103–112, 1988. 1
5. X. Boyen and B. Waters. Compact group signatures without random oracles. In S. Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 427–444, St. Petersburg, Russia, May 28 – June 1, 2006. Springer, Berlin, Germany. 2
6. J. Camenisch, N. Chandran, and V. Shoup. A public key encryption scheme secure against key dependent chosen plaintext and adaptive chosen ciphertext attacks. In A. Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 351–368, Cologne, Germany, Apr. 26–30, 2009. Springer, Berlin, Germany. 2, 6, 21, 25
7. R. Canetti, Y. Lindell, R. Ostrovsky, and A. Sahai. Universally composable two-party and multi-party secure computation. In *34th ACM STOC*, pages 494–503, Montréal, Québec, Canada, May 19–21, 2002. ACM Press. 8
8. N. Chandran, J. Groth, and A. Sahai. Ring signatures of sub-linear size without random oracles. In L. Arge, C. Cachin, T. Jurdzinski, and A. Tarlecki, editors, *ICALP 2007*, volume 4596 of *LNCS*, pages 423–434, Wroclaw, Poland, July 9–13, 2007. Springer, Berlin, Germany. 6, 24
9. R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In Y. Desmedt, editor, *CRYPTO'94*,

volume 839 of *LNCS*, pages 174–187, Santa Barbara, CA, USA, Aug. 21–25, 1994. Springer, Berlin, Germany. 1, 2, 3, 4, 7
10. A. De Santis, G. Di Crescenzo, and G. Persiano. Secret sharing and perfect zero knowledge. In D. R. Stinson, editor, *CRYPTO'93*, volume 773 of *LNCS*, pages 73–84, Santa Barbara, CA, USA, Aug. 22–26, 1993. Springer, Berlin, Germany. 2, 3, 4, 7
11. A. Escala and J. Groth. Fine-tuning Groth-Sahai proofs. In H. Krawczyk, editor, *PKC 2014*, volume 8383 of *LNCS*, pages 630–649, Buenos Aires, Argentina, Mar. 26–28, 2014. Springer, Berlin, Germany. 5, 7, 8, 9, 11, 27, 28
12. A. Escala, G. Herold, E. Kiltz, C. Ràfols, and J. Villar. An algebraic framework for Diffie-Hellman assumptions. In R. Canetti and J. A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 129–147, Santa Barbara, CA, USA, Aug. 18–22, 2013. Springer, Berlin, Germany. 28
13. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In A. M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194, Santa Barbara, CA, USA, Aug. 1986. Springer, Berlin, Germany. 1
14. J. Groth. Simulation-sound NIZK proofs for a practical language and constant size group signatures. In X. Lai and K. Chen, editors, *ASIACRYPT 2006*, volume 4284 of *LNCS*, pages 444–459, Shanghai, China, Dec. 3–7, 2006. Springer, Berlin, Germany. 2, 6, 21, 25
15. J. Groth, R. Ostrovsky, and A. Sahai. New techniques for noninteractive zero-knowledge. *J. ACM*, 59(3):11, 2012. 4, 12, 20
16. J. Groth and A. Sahai. Efficient noninteractive proof systems for bilinear groups. *SIAM J. Comput.*, 41(5):1193–1232, 2012. 2, 9, 11, 28
17. D. Hofheinz and T. Jager. Tightly secure signatures and public-key encryption. In R. Safavi-Naini and R. Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 590–607, Santa Barbara, CA, USA, Aug. 19–23, 2012. Springer, Berlin, Germany. 2, 25
18. M. Karchmer and A. Wigderson. On span programs. *In Structure in Complexity Theory Conference*, pages 102 – 111, 1993. 7
19. J. Katz and V. Vaikuntanathan. Round-optimal password-based authenticated key exchange. In Y. Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 293–310, Providence, RI, USA, Mar. 28–30, 2011. Springer, Berlin, Germany. 2
20. J. Kilian. Use of randomness on algorithms and protocols. *MIT Press*, 1990. 8
21. M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *22nd ACM STOC*, pages 427–437, Baltimore, Maryland, USA, May 14–16, 1990. ACM Press. 24, 25
22. R. L. Rivest, A. Shamir, and Y. Tauman. How to leak a secret. In C. Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 552–565, Gold Coast, Australia, Dec. 9–13, 2001. Springer, Berlin, Germany. 24
23. A. Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *40th FOCS*, pages 543–553, New York, New York, USA, Oct. 17–19, 1999. IEEE Computer Society Press. 24
24. C.-P. Schnorr. Efficient identification and signatures for smart cards. In G. Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 239–252, Santa Barbara, CA, USA, Aug. 20–24, 1989. Springer, Berlin, Germany. 1
25. A. Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979. 7
26. T. Tassa. Hierarchical threshold secret sharing. In M. Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 473–490, Cambridge, MA, USA, Feb. 19–21, 2004. Springer, Berlin, Germany. 23

A Proof of lemma 1

We give the proof of parts 2),3) of lemma 1.

Proof. (Lemma 1) Since $A \in \Omega$, $A^c \notin \Omega^*$. Thus, if B is a basis of the vectors $\{\mathbf{r}_j^* : j \in \rho^{-1}(A^c)\}$, the set $\{\mathbf{r}_j^* : j \in B \cup \{0\}\}$ is a set of linearly independent vectors. Find a set of indexes $C \subset [m]$ such that $B \cup \{0\} \subset C$ and the vectors $\{\mathbf{r}_j^* : j \in C\}$ are a basis the space spanned by the rows of $\mathbf{M}^*$. Note that $\boldsymbol{\zeta} = \mathbf{M}^*\boldsymbol{\omega}_1$ and $\boldsymbol{\tau} = \mathbf{M}^*\boldsymbol{\omega}_2$, if and only if $\zeta_j = (\mathbf{r}_j^*)^\top\boldsymbol{\omega}_1$, $\tau_j = (\mathbf{r}_j^*)^\top\boldsymbol{\omega}_2$. Because the rows indexed by C are a basis of the rows of $\mathbf{M}^*$, ζ_j, τ_j, $j \in C$, uniquely define $\boldsymbol{\zeta}, \boldsymbol{\tau}$ and further, if we sample a vector $\boldsymbol{\nu} \leftarrow \mathbf{Im}(\mathbf{M}^*)$, $\{\nu_j : j \in C\}$ is a uniform set of values in $\mathbb{Z}_q$. This shows that there is always one and only one vector $\boldsymbol{\nu}$ which is compatible with some fixed set of values $\{\nu_j : j \in C\}$. This proves part 2), as it implies that if we set $\zeta_0 = 1$ and $\zeta_j = 0$ for all $j \in B$, and $\zeta_j \leftarrow \mathbb{Z}_q$ for all $j \in C\backslash(B \cup \{0\})$, this defines a unique vector $\boldsymbol{\zeta}$ which is uniform conditioned on satisfying the constraints specified in 2). To see 3), just note that, regardless of whether we sample $\boldsymbol{\tau} \leftarrow \mathbf{Im}(\mathbf{M}^*)$, or $\boldsymbol{\tau} \leftarrow \mathbf{Im}(\mathbf{M}^*)$ conditioned on $\tau_0 = 0$, the same arguments used so far guarantee that $\{\tau_j : j \in C\backslash\{0\}\}$ is a uniform set of values in $\mathbb{Z}_q$, and by construction of B and C, $\{\tau_j : j \in \rho^{-1}(A^c)\}$ are completely determined by a subset of this set, namely by $\{\tau_j : j \in B\}$ and therefore independent of τ_0.

B Security definitions

B.1 Commit-and-prove schemes

Below we give the remaining security definitions for commit-and-prove schemes as taken from [11] and adapted to our modifications.

Definition 9 (Perfect completeness). *The commit-and-prove system* CaP *is (perfectly) correct if for all adversaries* $\mathcal{A}$

$$\Pr\Big[(gk, ck) \leftarrow \mathsf{G}(1^\lambda); (x, W = \{(\mathsf{t}_i, m_i) : i \in \mathcal{I}\}) \leftarrow \mathcal{A}(gk, ck); \\ \{(\mathsf{k}_i^p, \mathsf{k}_i^s) : i \in \mathcal{I}\} \leftarrow \mathsf{LabGen}(gk, ck, x, W); \\ C = \{(\mathsf{k}_i^p, c_i) \leftarrow \mathsf{Com}(gk, ck, (\mathsf{k}_i^p, \mathsf{k}_i^s, m_i)) : i \in \mathcal{I}\}; \\ \pi \leftarrow \mathsf{P}(gk, ck, x, Op, C) : \mathsf{V}(gk, ck, x, C, \pi) = 1\Big] = 1,$$

where $\mathcal{A}$ *outputs* (x, W) *such that* $(gk, x, W) \in \mathcal{R}_\mathcal{L}$ *and* Op *is a valid set of openings of* C*.*

A commit-and-prove scheme is sound if it is impossible to prove a false statement.

Definition 10 (Perfect soundness). *The commit-and-prove system* CaP *is (perfectly) sound if there exists a deterministic (unbounded) opening algorithm* Open *such that for all adversaries* $\mathcal{A}$

$$\Pr\Big[(gk,ck)\leftarrow \mathsf{G}(1^\lambda); (x,\{(\mathsf{k}_i^p,c_i): i\in\mathcal{I}\},\pi)\leftarrow\mathcal{A}(gk,ck);$$
$$\{(\mathsf{t}_i,m_i):(\mathsf{t}_i,m_i)\leftarrow \mathrm{Open}(gk,ck,(\mathsf{k}_i^p,c_i))\}:$$
$$\mathsf{V}(gk,ck,x,\{(\mathsf{k}_i^p,c_i):i\in\mathcal{I}\},\pi)=0 \ \vee \ (gk,x,\{(\mathsf{t}_i,m_i):i\in\mathcal{I}\})\in\mathcal{R}_{\mathcal{L}}\Big]=1.$$

C Verifiable Correlated Key Generation For Other Equation Types

Extending the Groth-Sahai CaP Based on SXDH. We describe an alternative instantiation of the GS CaP scheme based on SXDH. Recall that the original one does not admit correlated key generation for labels $\mathrm{com}_{\hat{\mathbb{G}}}$ (or mixed labels $\mathrm{com}_{\hat{\mathbb{G}}}, \mathrm{sca}_{\hat{\mathbb{G}}}$), while this one does. We just specify how to to generate the real commitment keys and the simulated keys with the simulation trapdoor, the rest of the algorithms of the CaP are easy to derive from the original paper of Groth and Sahai [16] or from the specification of GS proofs for any matrix assumption [12].

Essentially, the new instantiation introduces new label types so that one can commit to group elements and to scalars in two different ways. The labels $\mathrm{sca}_{\hat{\mathbb{G}}}, \mathrm{sca}_{\check{\mathbb{H}}}, \mathrm{com}_{\hat{\mathbb{G}}}, \mathrm{com}_{\check{\mathbb{H}}}$ indicate that one should commit to a group element as in the original instantiation of [11]. With these labels we can do what we described before, namely, we can prove that some equation admitted by the GS proof system is satisfiable, or that a set of equations with admissible simulation labels $(\mathrm{sca}_{\hat{\mathbb{G}}},\ldots,\mathrm{sca}_{\hat{\mathbb{G}}})$ is partially satisfiable.

The new labels are $\mathrm{sca}^{par}_{\hat{\mathbb{G}}}, \mathrm{com}^{par}_{\hat{\mathbb{G}}}$ ($\mathrm{sca}^{par}_{\check{\mathbb{H}}}, \mathrm{com}^{par}_{\check{\mathbb{H}}}$ in $\check{\mathbb{H}}$). The new instantiation of GS proofs we give below admits verifiable correlated key generation for $\mathbf{T}=(\mathsf{t}_1,\ldots,\mathsf{t}_L)$ where, for all $i\in[L]$, $\mathsf{t}_i\in\{\mathrm{com}^{par}_{\hat{\mathbb{G}}},\mathrm{sca}^{par}_{\hat{\mathbb{G}}}\}$ (or for all $i\in[L]$, $\mathsf{t}_i\in\{\mathrm{sca}^{par}_{\check{\mathbb{H}}},\mathrm{com}^{par}_{\check{\mathbb{H}}}\}$). The table in figure 2 describes how to commit with these new label types, where $\mathbf{e}_3=(0,0,1)^\top$ and $\hat{\mathbf{e}}_3=(\hat{0},\hat{0},\hat{g})^\top$.

For a complete description of the new CaP, we would need to specify new equation types L_{eq} and define which types of commitments are compatible with each equation type, since we are dealing with vectors of potentially different sizes. Given a quadratic equation written in the form given in equation (2) (section 4), it is enough that the commitments to all the elements involved in the equation in the same module A_i are in the same space. For instance we can define $\mathsf{L}_{eq}=\mathrm{MLin}_{\hat{\mathbb{G}},par}$ as a linear multi-scalar multiplication equation in which the variables in $A_1=\hat{\mathbb{G}}$ are committed with label $\mathrm{com}^{par}_{\hat{\mathbb{G}}}$ and the constants in $A_2=\mathbb{Z}_q$ are in the usual space $\check{\mathbb{H}}^2$. This is cumbersome to specify but straightforward, and we omit any further details.

A commitment to an element with any of these labels is a vector of dimension 3. Further, with these new labels, we essentially commit to scalars and group

$\mathsf{G}(1^\lambda)$
$gk \leftarrow (q, \hat{\mathbb{G}}, \check{\mathbb{H}}, \mathbb{T}, e, \hat{g}, \check{h}) \leftarrow \mathcal{G}(1^\lambda)$
$\omega, \sigma, \xi, \chi, \psi, \phi \leftarrow \mathbb{Z}_q^*$
$\hat{\mathbf{v}} \leftarrow (\xi\hat{g}, \hat{g})^\top$, $\check{\mathbf{v}} \leftarrow (\psi\check{h}, \check{h})^\top$
$\hat{\mathbf{u}} \leftarrow \omega\hat{\mathbf{v}}$, $\check{\mathbf{u}} \leftarrow \sigma\check{\mathbf{v}}$
$\hat{\mathbf{w}} \leftarrow \hat{\mathbf{u}} - (\hat{0}, \hat{g})^\top$, $\check{\mathbf{w}} \leftarrow \check{\mathbf{u}} - (\check{0}, \check{h})^\top$
$\hat{\mathbf{a}} \leftarrow (\chi\hat{g}, \xi\hat{g}, \hat{g})^\top$, $\check{\mathbf{a}} \leftarrow (\phi\check{h}, \psi\check{h}, \check{h})^\top$
$\hat{\mathbf{b}} \leftarrow \omega\hat{\mathbf{a}} + (\hat{g}, \hat{0}, \hat{0})^\top$, $\check{\mathbf{b}} \leftarrow \sigma\check{\mathbf{a}} + (\check{h}, \check{0}, \check{0})^\top$
$ck \leftarrow (\hat{\mathbf{u}}, \hat{\mathbf{v}}, \hat{\mathbf{w}}, \check{\mathbf{u}}, \check{\mathbf{v}}, \check{\mathbf{w}}, \hat{\mathbf{a}}, \hat{\mathbf{b}}, \check{\mathbf{a}}, \check{\mathbf{b}})$
Return (gk, ck)

$\mathsf{SimGen}(1^\lambda)$
$gk \leftarrow (q, \hat{\mathbb{G}}, \check{\mathbb{H}}, \mathbb{T}, e, \hat{g}, \check{h}) \leftarrow \mathcal{G}(1^\lambda)$
$\rho, \omega, \xi, \chi, \psi, \phi \leftarrow \mathbb{Z}_q^*$
$\hat{\mathbf{v}} \leftarrow (\xi\hat{g}, \hat{g})^\top$, $\check{\mathbf{v}} \leftarrow (\psi\check{h}, \check{h})^\top$
$\hat{\mathbf{u}} \leftarrow \omega\hat{\mathbf{v}} + (\hat{0}, \hat{g})^\top$, $\check{\mathbf{u}} \leftarrow \sigma\check{\mathbf{v}} + (\check{0}, \check{h})^\top$
$\hat{\mathbf{w}} \leftarrow \hat{\mathbf{u}} - (\hat{0}, \hat{g})^\top$, $\check{\mathbf{w}} \leftarrow \check{\mathbf{u}} - (\check{0}, \check{h})^\top$
$\hat{\mathbf{a}} \leftarrow (\chi\hat{g}, \xi\hat{g}, \hat{g})^\top$, $\check{\mathbf{a}} \leftarrow (\phi\check{h}, \psi\check{h}, \check{h})^\top$
$\hat{\mathbf{b}} \leftarrow \omega\hat{\mathbf{a}} + (\hat{0}, \hat{0}, \hat{g})^\top$, $\check{\mathbf{b}} \leftarrow \sigma\check{\mathbf{a}} + (\check{0}, \check{0}, \check{h})^\top$
$ck \leftarrow (\hat{\mathbf{u}}, \hat{\mathbf{v}}, \hat{\mathbf{w}}, \check{\mathbf{u}}, \check{\mathbf{v}}, \check{\mathbf{w}}, \hat{\mathbf{a}}, \hat{\mathbf{b}}, \check{\mathbf{a}}, \check{\mathbf{b}})$
$tk \leftarrow (ck, \sigma, \omega)$
Return (gk, ck, tk)

Label t	Message	Randomness	Commitment	$\mathcal{K}^{\mathsf{t}}_{gk,\mathsf{bind}}$	$\mathcal{K}^{\mathsf{t}}_{gk,\mathsf{hid}}$
$\mathrm{sca}_{\hat{\mathbb{G}}}$	$(\mathrm{sca}_{\hat{\mathbb{G}}}, x)$	$(\mathrm{sca}_{\hat{\mathbb{G}}}, r)$	$\hat{\mathbf{c}} \leftarrow \hat{\mathbf{w}}x + \hat{\mathbf{v}}r$	$ck : \hat{\mathbf{w}} \notin \langle\hat{\mathbf{v}}\rangle$	$ck : \hat{\mathbf{w}} \in \langle\hat{\mathbf{v}}\rangle$
$\mathrm{com}_{\hat{\mathbb{G}}}$	$(\mathrm{com}_{\hat{\mathbb{G}}}, \hat{x})$	$(\mathrm{com}_{\hat{\mathbb{G}}}, r, s)$	$\hat{\mathbf{c}} \leftarrow \mathbf{e}_2\hat{x} + \hat{\mathbf{v}}r + \hat{\mathbf{u}}s$	$ck : \hat{\mathbf{e}}_2 \notin \langle\hat{\mathbf{u}}, \hat{\mathbf{v}}\rangle$	$ck : \hat{\mathbf{e}}_2 \in \langle\hat{\mathbf{u}}, \hat{\mathbf{v}}\rangle$
$\mathrm{sca}^{par}_{\hat{\mathbb{G}}}$	$(\mathrm{sca}^{par}_{\hat{\mathbb{G}}}, x)$	$(\mathrm{sca}^{par}_{\hat{\mathbb{G}}}, r, s)$	$\hat{\mathbf{c}} \leftarrow \hat{\mathbf{e}}_3x + \hat{\mathbf{a}}r + \hat{\mathbf{b}}s$	$ck : \hat{\mathbf{e}}_3 \notin \langle\hat{\mathbf{a}}, \hat{\mathbf{b}}\rangle$	$ck : \hat{\mathbf{e}}_3 \in \langle\hat{\mathbf{a}}, \hat{\mathbf{b}}\rangle$
$\mathrm{com}^{par}_{\hat{\mathbb{G}}}$	$(\mathrm{com}^{par}_{\hat{\mathbb{G}}}, \hat{x})$	$(\mathrm{com}^{par}_{\hat{\mathbb{G}}}, r, s)$	$\hat{\mathbf{c}} \leftarrow \ \mathbf{e}_3\hat{x} + \hat{\mathbf{a}}r + \hat{\mathbf{b}}s$	$ck : \hat{\mathbf{e}}_3 \notin \langle\hat{\mathbf{a}}, \hat{\mathbf{b}}\rangle$	$ck : \hat{\mathbf{e}}_3 \in \langle\hat{\mathbf{a}}, \hat{\mathbf{b}}\rangle$

Fig. 2. Generator algorithms. The two last coordinates of $\hat{\mathbf{a}}$ (resp. of $\hat{\mathbf{b}}, \check{\mathbf{a}}, \check{\mathbf{b}}$) correspond to the vector $\hat{\mathbf{v}}$ (resp. to $\hat{\mathbf{u}}, \check{\mathbf{v}}, \check{\mathbf{u}}$). Table describing the most important commitment types.

elements in the same way (so that a key can be binding/hiding for scalars and group elements at the same time). Therefore, there is no longer an efficiency advantage in the proof size for equations involving scalars over equations involving group elements[3]) This has an impact on efficiency, that is why one should only use these labels to prove statements which are too expensive to prove with the normal instantiation. For instance, if one just wants to prove satisfiability of one PPE, one should use a standard commitment to group elements.

C.1 VCKG for Group Elements

To prove soundness in the constructions of section 7, we used in a fundamental way that $\hat{\mathbf{w}}$ (or $(\hat{0}, \hat{g})^\top$) and $\hat{\mathbf{v}}$ are linearly independent. By giving this new instantiation with an additional dimension, the same arguments follow in a relative straightforward way. Indeed, the main reason why this new scheme admits verifiable correlated key generation for these new label types is that *both* for binding and hiding keys, $\hat{\mathbf{b}} \notin \langle\hat{\mathbf{a}}\rangle$. Intuitively, the point is that the secret sharing techniques we are using "work well" with linear independence relations and they "fail" with linear dependence relations. When we tried to construct correlated

[3] In the original GS proof instantiation, equations involving $\mathrm{sca}_{\hat{\mathbb{G}}}, \mathrm{sca}_{\check{\mathbb{H}}}$ are more efficient than (similar) equations with $\mathrm{com}_{\hat{\mathbb{G}}}, \mathrm{com}_{\check{\mathbb{H}}}$. For instance, the equation $\hat{x}a = \hat{t}$ requires one extra proof element compared to $\hat{a}x = \hat{t}$, but this is no longer true for the new labels.

key generation for the label types $(\mathrm{com}_{\hat{\mathbb{G}}}, \ldots, \mathrm{com}_{\hat{\mathbb{G}}})$, we did not know how to force the prover to choose binding keys for group elements, i.e. keys such that $\hat{\mathbf{u}} \in \langle \hat{\mathbf{v}} \rangle$.

We sketch the construction of section 7.2 for the vector of labels $(\mathrm{com}^{par}_{\hat{\mathbb{G}}}, \ldots, \mathrm{com}^{par}_{\hat{\mathbb{G}}})$. Algorithm $\mathsf{K}_{\mathsf{scorr}}$ samples two vectors $\boldsymbol{\tau} \in \mathbb{Z}_q^{m+1}$, $\boldsymbol{\kappa} \in \mathbb{Z}_q^{m+1}$ uniformly at random conditioned on $\tau_0 = 0, \kappa_0 = 0$. It also samples $\boldsymbol{\zeta}$ as usual, namely, as a uniform vector conditioned on $\zeta_0 = 1$ and $\zeta_j = 0$ for all $j \in \rho^{-1}(i)$, $i \in A^c$. The vectors $\hat{\mathbf{z}}_j$ are defined as $\hat{\mathbf{z}}_j = \tau_j \hat{\mathbf{a}} + \zeta_j \hat{\mathbf{b}} + \kappa_j (\hat{0}, \hat{0}, \hat{g})^\top$ and $\Sigma = \{ck_1, \ldots, ck_m\}$, where $ck_j = (\hat{\mathbf{u}}_j, \hat{\mathbf{v}}, \hat{\mathbf{w}}_j, \check{\mathbf{u}}, \check{\mathbf{v}}, \check{\mathbf{w}}, \hat{\mathbf{a}}, \hat{\mathbf{z}}_j, \check{\mathbf{a}}, \check{\mathbf{b}})$ (and $\hat{\mathbf{u}}_j, \hat{\mathbf{w}}_j$ are changed according to $\hat{\mathbf{z}}_j$ to guarantee that $ck_j \in \mathcal{K}_{gk}$ (that is, $\hat{\mathbf{u}}_j$ should match the last two coordinates of $\hat{\mathbf{z}}_j$, $\hat{\mathbf{w}}_j = \hat{\mathbf{u}}_j - (\hat{0}, \hat{g})^\top$). The construction also works for $\mathbf{T} = (\mathsf{t}_1, \ldots, \mathsf{t}_L)$, $\mathsf{t}_i \in \{\mathrm{com}^{par}_{\hat{\mathbb{G}}}, \mathrm{sca}^{par}_{\hat{\mathbb{G}}}\}$, since the set of binding/hiding keys for $\mathrm{com}^{par}_{\hat{\mathbb{G}}}$ and $\mathrm{sca}^{par}_{\hat{\mathbb{G}}}$ is the same.

The proof is identical to the one of lemma 2. Indeed, the key observation is that if $\zeta_j = 0$, then $(\hat{0}, \hat{0}, \hat{g})^\top \in \langle \hat{\mathbf{a}}, \hat{\mathbf{z}}_j \rangle$ (unless $\kappa_j = 0$, which occurs only with negligible probability). This means that the key ck_j is hiding, and further the simulation trapdoor is (τ_j, κ_j). On the other hand, if $\zeta_j \neq 0$, since $(\hat{0}, \hat{0}, \hat{g})^\top \notin \langle \hat{\mathbf{a}}, \hat{\mathbf{z}}_j \rangle$ the key is binding. Therefore, we are in the same situation as in lemma 2. The rest of the algorithms/ proof are also straightforward. Now the matrix $\hat{\mathbf{Z}} \in \hat{\mathbb{G}}^{3 \times (m+1)}$ and the verifier checks if $\mathbf{ZM} = \hat{\mathbf{0}}_{3 \times d}$, where $\mathbf{M}$ is the matrix associated to the span program.

Example. We retake the example of the OR of two equations as defined in section 8 but for the labels $\mathbf{T} = (\mathrm{com}^{par}_{\hat{\mathbb{G}}}, \mathrm{com}^{par}_{\hat{\mathbb{G}}})$. The vectors $\boldsymbol{\zeta}, \boldsymbol{\tau}$ are defined as explained in section 8. Additionally, one chooses another vector $\boldsymbol{\kappa} \in \mathbf{Im}(\mathbf{M}^*)$ uniformly conditioned on $\kappa_0 = 0$, i.e. $\boldsymbol{\kappa} = (0, s, -s)$, $s \leftarrow \mathbb{Z}_q$. Let's see why the approach works. For instance, assume that $A = \{1\}$ was the set used to compute the keys. In this case, $\hat{\mathbf{z}}_1 = r\hat{\mathbf{a}} - \hat{\mathbf{b}} + s(\hat{0}, \hat{0}, \hat{g})^\top$, $\hat{\mathbf{z}}_2 = -r\hat{\mathbf{a}} - s(\hat{0}, \hat{0}, \hat{g})^\top$, $tk_2 = (r, s)$. The reason why the key $(\hat{0}, \hat{0}, \hat{g})^\top \notin \langle \hat{\mathbf{a}}, \hat{\mathbf{z}}_1 \rangle$ is because $\hat{\mathbf{b}}$ is linearly independent of $\hat{\mathbf{a}}$ in the soundness setting.

Efficiency. For the same span program $\mathcal{SP}$, the description of Σ for $\mathbf{T} = (\mathsf{t}_1, \ldots, \mathsf{t}_L) = (\mathrm{sca}_{\hat{\mathbb{G}}}, \ldots, \mathrm{sca}_{\hat{\mathbb{G}}})$ is more efficient than when $\mathsf{t}_i \in \{\mathrm{com}^{par}_{\hat{\mathbb{G}}}, \mathrm{sca}^{par}_{\hat{\mathbb{G}}}\}$, because for the latter we need to send $3(m-d)|\hat{\mathbb{G}}|$ elements. Additionally, for this construction there is an overhead that depends on the number of variables and the equation type. This is because for each variable that we commit to using one of the labels $\mathsf{t}_i \in \{\mathrm{com}^{par}_{\hat{\mathbb{G}}}, \mathrm{sca}^{par}_{\hat{\mathbb{G}}}\}$ we need $3|\hat{\mathbb{G}}|$, as opposed to $2|\hat{\mathbb{G}}|$ in the normal instantiation of GS proofs. For quadratic equations (but not for linear ones), this also results in a larger proofs. For each equation type, one should evaluate if the approach is competitive, but for simple statements it looks like an interesting alternative (for instance, if one wants to prove OR of two linear equations in $\hat{\mathbb{G}}$, each with one variable in $\hat{\mathbb{G}}$).