Proposal: Commit-and-Prove Zero-Knowledge Proof Systems and Extensions

Daniel Benarroch[1], Matteo Campanelli[2], Dario Fiore[3], Jihye Kim[4], Jiwon Lee[5], Hyunok Oh[5], and Anaïs Querol[3,6]

[1] QED-it, Tel Aviv, Israel
daniel@qed-it.com
[2] Aarhus University, Denmark
matteo@cs.au.dk
[3] IMDEA Software Institute, Madrid, Spain
{dario.fiore,anais.querol}@imdea.org
[4] Kookmin University, Seoul, Korea
jihyek@kookmin.ac.kr
[5] Hanyang University, Seoul, Korea
{jiwonlee,hoh}@hanyang.ac.kr
[6] Universidad Politécnica de Madrid, Spain

Abstract. Commit-and-Prove Zero-Knowledge Proof systems (CP-ZKPs) [Kil89, CLOS02] generalize zero-knowledge proofs where we prove statements about values that are committed. In this document we propose this notion and its variants. It can be useful as a lingua franca framework because: it emerges in a wide variety of practical applications; it may unify abstractions and simplify proofs in cryptographic technical reports; although the notion is defined as a special case of NIZKs, some of its *efficient* constructions are non-trivial. While previous editions of the ZKProof workshop have confirmed interest in the formalization in this document, there is still significant work ahead in terms of: refining the content of the proposal, and describing existing candidate constructions as well as general design approaches.

1 Scope

Commit-and-Prove Zero-Knowledge Proof systems (CP-ZKPs) [Kil89, CLOS02] are a generalization of zero-knowledge proofs in which the prover proves statements about values that are committed.

The aim of this document is to stimulate a discussion on the formalization of CP-ZKPs. As we detail in the next section, the motivation for using CP-ZKPs is both theoretical and practical.

Following the ongoing standardization effort in the context of ZKPs [GKV$^+$18], *our goal here is to address terminology and definitions for the notions of commitments and CP-ZKPs and mention next steps on candidate constructions.*

Status progress and next steps. This is an updated version of the proposal submitted and discussed at the 2nd and 3rd ZKP workshop. This proposal is currently a work-in-progress with an official working group charter [1]. Previous editions of the workshop have confirmed that: there exist relevant applications that would benefit from this approach; that practitioners should have guidelines about how to align applications to this notion; that formalizing the latter would be advantageous and that the one in this document may be the right approach for it.

While there has been progress from the last edition, there is still significant work ahead. With respect to the last edition of the workshop, one milestone we achieved at the time of writing of this

[1] Available here: https://hackmd.io/@dariofiore/rkXo8EBp8

document is that of merging the vision between the commit-and-prove and SAVER working groups (see URL in Footnote 1 for resources linking to these working groups): this document describes both the formalization for commit-and-prove as well as for encrypt-and-prove.

Together with other members of the community we also started discussing concrete candidates that could be standardized. We discuss some highlights in Section 6.

The next milestones involve:

- Agree on a preliminary set of goals and open the discussion to the community.
- Refine the content of the standardization draft proposal.
- Describe generic construction methodologies and specify concrete schemes for both commit-and-prove and encrypt-and-prove proof systems.

2 Motivation

As observed in the ZKProof proceedings (Applications Track) [BCM$^+$18], most applications of zero knowledge proofs require the use of some commitment scheme in order to ensure the privacy and confidentiality of the users and their data, by proving knowledge of the opening. The use of the Commit-and-Prove (CP) paradigm has several interesting features:

- **CP as a lens for authenticated data structures.** If the commitment is compressing one can distribute succinct and private representations of data that significantly reduce communication complexity and input size for verifiers (as well as their running time)[2]. An interesting observation is that a special case of this scenario are authenticated data structures (in their succinct variant) [Tam03], and in particular cryptographic primitives such as vector commitments [CF13, Mer88], polynomial commitments [KZG10] and accumulators [Bd94] which are already in use (or considered to be put in use) in combination with ZKP systems in a variety of applications, such as for example [HBHW16, But18] . We can see these authenticated data structures as commit-and-prove systems where we first compress a large data structure (e.g. a vector or a set) and then prove something about its content (that a certain position has a certain value or that does not contain a certain string).
- **CP is agnostic to relations and proof systems.** One can publish commitments to data previous to generating proofs about them. For example, the identity scheme in [BCM$^+$18] requires that some issuer publishes a credential for the user, which will later be used to prove some attribute; in some cases even before the statement to be proven for a specific attribute is established. This means that, ideally, we would like to have some flexibility as to what statements are proven on the opening of the commitment, as well as to which ZKP schemes are used for the different statements.
- **CP promotes interoperability.** One can use commitments to make different proof systems *interoperable*. For example, one can prove two different statements about the same commitment using two distinct ZKP systems. This can be advantageous in order to exploit the different efficiency tradeoffs of existing systems (see e.g., [AGM18, CFQ19a]), or simply because the public parameters of the ZKP systems are generated in different points in time or by different organizations. Interoperability *through time* is particularly important for legacy systems: commitments produced in the past and through a specific (legacy) commitment scheme can still be used as inputs to proof systems that will be developed in the future.

[2] We point out that if this commitment is compressing we may benefit from its use even if it is not hiding. Indeed the verifier now just needs a "compact placeholder" to the input and this can be sufficient in some applications.

It is important to stress that not all efficient constructions of this notion are straightforward. Naturally, one can see proving "commitment opening and some other property" simply as a "larger property" we are proving (and indeed the main notion we propose is defined this way modulo some technical care). On the other hand, encoding this property directly as a larger circuit may be highly inefficient. Related discussions and efficient solutions can be found for example in [FFG+16, CFQ19b].

We find thus motivating to have a framework for properly building such applications and instantiating the commitment schemes with the corresponding ZKP schemes. We believe that the starting point of such a framework should be a formal definition of commitments and commit-and-prove zero-knowledge proof systems.

Finally, we believe that standardizing the definition for CP-ZKP (and, at a later time, its framework) offers yet one more advantage related to how we can program constraint systems. Today there are several high-level languages that can be used to express constraints. In our experience, and analogously to other programming languages, it would seem beneficial to have pre-defined types for constraint system variables. These types would be common among most or all applications. The current proposal could lead to defining the variable type *commitment* and *opening*, which could lead to better security assurances and engineering practices.

2.1 Extensions to Encryption

So far we discussed and motivated a framework that decompose commitments and proof systems that take commitments as inputs to a verification procedure. Similar observations lead to decomposing encryption and proof systems. One particular application of this is in electronic voting [LCKO19], where parties should be able to prove knowledge of a plaintext in a rerandomizable encryption scheme.

3 Background

The commit-and-prove approach in zero-knowledge proofs dates back to the works of Kilian [Kil89] and Canetti et al. [CLOS02], and has been used extensively, implicitly or explicitly, in plenty of works.

As we detail in Section 4.4, the notion of CP-ZKPs could be seen as a specialization of ZKPs by considering languages that are parametrized by the commitment key, e.g., using informal notation,

$$\mathcal{L}_{\mathsf{ck}} = \{c_x : c_x \text{ opens to } x \text{ and } x \in \mathcal{L}\}$$

When focusing on non-interactive zero-knowledge proof systems (NIZK), in which one generates a language-dependent CRS, there is a variety of CP-ZKP notions used in the literature, such as those where *the commitment key is generated together with the CRS*, e.g., [CFH+15], and those where *the commitment key is taken as an input in the NIZK CRS generation* [Lip16, EG14, CFQ19a], which in turn include systems where the commitment key is the CRS itself (in which case the commitment must admit a trapdoor, e.g., [EG14, Lip16]).

Given the theoretical and practical relevance of the commit-and-prove approach, we believe it is important for the community to either agree on one or at least provide a taxonomy of the variants.

Other related work. Many of the advantages of a commit-and-prove formalization have to do with some level of uniform representation among cryptographic primitives (e.g., all proof systems should be able to interact through the same type of commitment) and efficient operations over that representation (e.g., it should be efficient to prove/verify over committed data). A different line of work that has an analogous vision is that of *structure-preserving cryptography* [AFG$^+$16] which proposes a framework for modular cryptographic building blocks (signatures, commitments, etc.) over bilinear groups that are interoperable and efficient.

While the proposed commit-and-prove (and encrypt-and-prove) framework in this document formalizes a *hiding encoding* of the data, there have been similar formalizations of *hash-and-prove systems* [FFG$^+$16]; here we do not require zero-knowledge and we provide a binding (but not hiding) digest of the data. We observe this is a direct special case of the framework in this document.

4 Definitions

In this section we give definitions for commitments, zero-knowledge proofs and commit-and-prove zero-knowledge proofs. We focus on the non-interactive setting although these definitions can be adapted to the interactive setting.

The definitions of CP-ZKPs proposed in this document are based on the ones recently used in [CFQ19a]. Although this definition still needs to be perfectly aligned with those in the rest of the ZKProof standardization effort, we do not necessarily mean this to be *the* definition but rather to serve as a starting point for a related discussion.

Notation. We use $\lambda \in \mathbb{N}$ to denote the security parameter, and 1^λ to denote its unary representation. Throughout the paper we assume that all the algorithms of the cryptographic schemes take as input 1^λ, and thus we omit it from the list of inputs. For a distribution D, we denote by $x \leftarrow D$ the fact that x is being sampled according to D. We remind the reader that an ensemble $\mathcal{X} = \{X_\lambda\}_{\lambda \in \mathbb{N}}$ is a family of probability distributions over a family of domains $\mathcal{D} = \{D_\lambda\}_{\lambda \in \mathbb{N}}$. We say two ensembles $\mathcal{D} = \{D_\lambda\}_{\lambda \in \mathbb{N}}$ and $\mathcal{D}' = \{D'_\lambda\}_{\lambda \in \mathbb{N}}$ are statistically indistinguishable (denoted by $\mathcal{D} \approx_s \mathcal{D}'$) if $\frac{1}{2}\sum_x |D_\lambda(x) - D'_\lambda(x)| < \mathsf{negl}(\lambda)$. If $\mathcal{A} = \{\mathcal{A}_\lambda\}$ is a (possibly non-uniform) family of circuits and $\mathcal{D} = \{D_\lambda\}_{\lambda \in \mathbb{N}}$ is an ensemble, then we denote by $\mathcal{A}(\mathcal{D})$ the ensemble of the outputs of $\mathcal{A}_\lambda(x)$ when $x \leftarrow D_\lambda$. We say two ensembles $\mathcal{D} = \{D_\lambda\}_{\lambda \in \mathbb{N}}$ and $\mathcal{D}' = \{D'_\lambda\}_{\lambda \in \mathbb{N}}$ are computationally indistinguishable (denoted by $\mathcal{D} \approx_c \mathcal{D}'$) if for every non-uniform polynomial time distinguisher $\mathcal{A}$ we have $\mathcal{A}(\mathcal{D}) \approx_s \mathcal{A}(\mathcal{D}')$. We denote by $[n]$ the set of integers $\{1, \ldots, n\}$ and by $[: n]$ the set $\{0, 1, \ldots, n-1\}$. We use $(u_j)_{j \in [\ell]}$ to denote the tuple of elements $(u_1, \ldots, u_\ell)$.

4.1 Relations

Let $\{\mathcal{R}_\lambda\}_{\lambda \in \mathbb{N}}$ be a family of polynomial-time decidable relations R on pairs (x, w) where $x \in \mathcal{D}_x$ is called the *statement* or *input*, and $w \in \mathcal{D}_w$ the *witness*. We write $R(x, w) = 1$ to denote that R holds on (x, w), else we write $R(x, w) = 0$. When discussing schemes that prove statements on committed values we assume that $\mathcal{D}_w$ can be split in two subdomains $\mathcal{D}_u \times \mathcal{D}_\omega$. Finally we sometimes use an even finer grained specification of $\mathcal{D}_u$ assuming we can split it over ℓ arbitrary domains $(\mathcal{D}_1 \times \cdots \times \mathcal{D}_\ell)$ for some arity ℓ. In our security definitions we assume relations to be generated by a *relation generator* $\mathcal{RG}(1^\lambda)$ that, on input the security parameter 1^λ, outputs R together with some side information, an auxiliary input aux_R, that is given to the adversary. We define $\mathcal{RG}_\lambda$ as the set of all relations that can be returned by $\mathcal{RG}(1^\lambda)$.

4.2 NIZKs

We recall the definition of non-interactive zero-knowledge arguments of knowledge (NIZK, for short).

Definition 4.1 (NIZK). *A NIZK for $\{\mathcal{R}_\lambda\}_{\lambda\in\mathbb{N}}$ is a triple of algorithms $\Pi = (\mathsf{KeyGen}, \mathsf{Prove}, \mathsf{VerProof})$ that work as follows and satisfy the notions of* completeness, knowledge soundness *and* zero-knowledge *defined below:*

- $\mathsf{KeyGen}(R) \to (\mathsf{ek}, \mathsf{vk})$*: takes the security parameter λ and a relation $R \in \mathcal{R}_\lambda$, and outputs a common reference string consisting of an evaluation* ek *and a verification key* vk*.*
- $\mathsf{Prove}(\mathsf{ek}, x, w) \to \pi$*: takes an evaluation key* ek *for a relation R, a statement x, and a witness w such that $R(x, w)$ holds, and returns a proof π.*
- $\mathsf{VerProof}(\mathsf{vk}, x, \pi) \to b$*: takes a verification key* vk*, a statement x, and either accepts ($b = 1$) or rejects ($b = 0$) the proof π.*

Completeness. *Π is complete if for any $\lambda \in \mathbb{N}$, $R \in \mathcal{R}_\lambda$ and (x, w) such that $R(x, w) = 1$, it holds:*

$$\Pr[(\mathsf{ek}, \mathsf{vk}) \leftarrow \mathsf{KeyGen}(R), \pi \leftarrow \mathsf{Prove}(\mathsf{ek}, x, w) : \mathsf{VerProof}(\mathsf{vk}, x, \pi) = 1] = 1$$

Knowledge Soundness. *A scheme Π is knowledge sound for $\mathcal{RG}$ and auxiliary input distribution $\mathcal{Z}$, denoted $\mathsf{KSND}(\mathcal{RG}, \mathcal{Z})$ for brevity, if for every (non-uniform) efficient adversary $\mathcal{A}$ there exists a (non-uniform) efficient extractor $\mathcal{E}$ such that $\Pr[\mathsf{Game}^{\mathsf{KSND}}_{\mathcal{RG},\mathcal{Z},\mathcal{A},\mathcal{E}} = 1] = \mathsf{negl}$. We say that Π is knowledge sound if there exists benign $\mathcal{RG}$ and $\mathcal{Z}$ such that Π is $\mathsf{KSND}(\mathcal{RG}, \mathcal{Z})$.*

$\underline{\mathsf{Game}^{\mathsf{KSND}}_{\mathcal{RG},\mathcal{Z},\mathcal{A},\mathcal{E}} \to b}$

$(R, \mathsf{aux}_R) \leftarrow \mathcal{RG}(1^\lambda)$
$\mathsf{crs} := (\mathsf{ek}, \mathsf{vk}) \leftarrow \mathsf{KeyGen}(R)$
$\mathsf{aux}_Z \leftarrow \mathcal{Z}(R, \mathsf{aux}_R, \mathsf{crs})$
$(x, \pi) \leftarrow \mathcal{A}(R, \mathsf{crs}, \mathsf{aux}_R, \mathsf{aux}_Z)$
$w \leftarrow \mathcal{E}(R, \mathsf{crs}, \mathsf{aux}_R, \mathsf{aux}_Z)$
$b \leftarrow \mathsf{VerProof}(\mathsf{vk}, x, \pi) = 1 \ \wedge R(x, w) = 0$

Composable Zero-Knowledge. *A scheme Π is composable zero-knowledge for a relation generator $\mathcal{RG}$ if there exists a simulator $\mathcal{S} = (\mathcal{S}_{\mathsf{kg}}, \mathcal{S}_{\mathsf{prv}})$ such that both following conditions hold for all adversaries $\mathcal{A}$:*

Keys Indistinguishability

$$\begin{aligned} &\Pr\left[\ (R, \mathsf{aux}_R) \leftarrow \mathcal{RG}(1^\lambda)\ ,\quad \mathsf{crs} \leftarrow \mathsf{KeyGen}(R)\ :\ \mathcal{A}(\mathsf{crs}, \mathsf{aux}_R) = 1\ \right] \\ \approx &\Pr\left[\ (R, \mathsf{aux}_R) \leftarrow \mathcal{RG}(1^\lambda)\ ,\ (\mathsf{crs}, \mathsf{td}_\mathsf{k}) \leftarrow \mathcal{S}_{\mathsf{kg}}(R)\ :\ \mathcal{A}(\mathsf{crs}, \mathsf{aux}_R) = 1\ \right] \end{aligned}$$

Proof Indistinguishability *For all (x, w) such that $R(x, w) = 1$,*

$$\begin{aligned} &\Pr\left[\ (R, \mathsf{aux}_R) \leftarrow \mathcal{RG}(1^\lambda)\ ,\ (\mathsf{crs}, \mathsf{td}_\mathsf{k}) \leftarrow \mathcal{S}_{\mathsf{kg}}(R)\ ,\ \pi \leftarrow \mathsf{Prove}(\mathsf{ek}, x, w)\ :\ \mathcal{A}(\mathsf{crs}, \mathsf{aux}_R, \pi) = 1\ \right] \\ \approx &\Pr\left[\ (R, \mathsf{aux}_R) \leftarrow \mathcal{RG}(1^\lambda)\ ,\ (\mathsf{crs}, \mathsf{td}_\mathsf{k}) \leftarrow \mathcal{S}_{\mathsf{kg}}(R)\ ,\ \pi \leftarrow \mathcal{S}_{\mathsf{prv}}(\mathsf{crs}, \mathsf{td}_\mathsf{k}, x)\ :\ \mathcal{A}(\mathsf{crs}, \mathsf{aux}_R, \pi) = 1\ \right] \end{aligned}$$

Remark 4.1 (On auxiliary inputs). In the notion of knowledge soundness defined above we consider two kinds of auxiliary inputs, aux_R generated together with the relation by $\mathcal{RG}$, and aux_Z that is generated from some distribution $\mathcal{Z}$ that may depend on the common reference string that in turn depends on R. Notice that although knowledge soundness is implied by a notion where auxiliary inputs can be arbitrary, our aim is a precise formalization of auxiliary inputs; this is useful to justify why certain auxiliary inputs should be considered benign, as required to avoid known impossibility results [BCPR14, BP15]. Finally, we also note that our notion is also implied by SNARKs that admit black-box extractors (as may be the case for those relying on random oracles [Mic00]).

Remark 4.2 (On definition of zkSNARKs). One can define zero-knowledge succinct non-interactive arguments (zkSNARKs) as NIZKs enjoying an additional property, *succinctness*, i.e. if the running time of $\mathsf{VerProof}$ is $\mathsf{poly}(\lambda + |x| + \log |w|)$ and the proof size is $\mathsf{poly}(\lambda + \log |w|)$.

Remark 4.3 (On notions of knowledge-soundness). Above we use a non black-box definition of extractability. Although this is virtually necessary in the case of zkSNARKs, NIZKs can also satisfy stronger notions of (knowledge) soundness.

4.3 Commitment Schemes

We recall the notion of non-interactive commitment schemes.

Definition 4.2 (Commitment Scheme). *A* commitment scheme *is a triple of algorithms* $\mathsf{Com} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{VerCommit})$ *that work as follows and satisfy the notions of* correctness*,* binding *and* hiding *defined below.*

- $\mathsf{Setup}(1^\lambda) \to \mathsf{ck}$*: takes the security parameter* λ *and outputs a commitment key* ck*. This key includes descriptions of the input space* $\mathcal{D}$*, commitment space* $\mathcal{C}$ *and opening space* $\mathcal{O}$*.*
- $\mathsf{Commit}(\mathsf{ck}, u) \to (c, o)$*: takes the commitment key* ck *and a value* $u \in \mathcal{D}$*, and outputs a commitment* c *and an opening* o*.*
- $\mathsf{VerCommit}(\mathsf{ck}, c, u, o) \to b$*: takes as input a commitment* c*, a value* u *and an opening* o*, and accepts (*$b = 1$*) or rejects (*$b = 0$*).*

A commitment scheme $\mathsf{Com} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{VerCommit})$ must satisfy the following properties:

Correctness. For all $\lambda \in \mathbb{N}$ and any input $u \in \mathcal{D}$ we have:

$$\Pr\left[\ \mathsf{ck} \leftarrow \mathsf{Setup}(1^\lambda)\ ,\ (c, o) \leftarrow \mathsf{Commit}(\mathsf{ck}, u)\ :\ \mathsf{VerCommit}(\mathsf{ck}, c, u, o) = 1\ \right] = 1$$

Binding. For every polynomial-time adversary $\mathcal{A}$

$$\Pr\left[\begin{array}{r} \mathsf{ck} \leftarrow \mathsf{Setup}(1^\lambda) \\ (c, u, o, u', o') \leftarrow \mathcal{A}(\mathsf{ck}) \end{array} : \begin{array}{r} \mathsf{VerCommit}(\mathsf{ck}, c, u, o) = 1 \\ \mathsf{VerCommit}(\mathsf{ck}, c, u', o') = 1 \\ u \neq u' \end{array}\right] = \mathsf{negl}(\lambda)$$

Hiding. For $\mathsf{ck} \leftarrow \mathsf{Setup}(1^\lambda)$ and every values $u, u' \in \mathcal{D}$, the following two distributions are statistically close: $\mathsf{Commit}(\mathsf{ck}, u) \approx \mathsf{Commit}(\mathsf{ck}, u')$.

Remark 4.4. In the literature there exists more than one syntax for commitment schemes. For example, some definitions replace the predicate $\mathsf{VerCommit}$ above with a procedure that, given a commitment c and some opening information o, outputs the committed value u (or $\perp$ when appropriate); or other definitions define $\mathsf{Commit}(\mathsf{ck}, u; r)$ as a deterministic function where the input r is the randomness, which also acts as opening information, and checking that c is a commitment for message u and opening r consists simply in recomputing $\mathsf{Commit}(\mathsf{ck}, u; r)$.

4.4 Definition of Commit-and-Prove NIZKs

In a nutshell, a *commit-and-prove* NIZK (CP-NIZK) is a NIZK that can prove knowledge of (x, w) such that $R(x, w)$ holds with respect to a witness $w = (u, \omega)$ such that u opens a commitment c_u. Our formal definitions below essentially add some syntactic sugar to this idea in order to explicitly handle relations in which the input domain $\mathcal{D}_u$ is more fine grained and splits over ℓ subdomains. We call these subdomains *commitment slots* following [CFQ19a]. Intuitively, each item in the commitment slot represents an input (or a vector of inputs) to the relation which the prover has previously committed to. We assume that the description of the splitting is part of R's description.

Definition 4.3 (CP-NIZK). *Let $\{\mathcal{R}_\lambda\}_{\lambda\in\mathbb{N}}$ be a family of relations R over $\mathcal{D}_x \times \mathcal{D}_u \times \mathcal{D}_\omega$ such that $\mathcal{D}_u$ splits over ℓ arbitrary domains $(\mathcal{D}_1 \times \cdots \times \mathcal{D}_\ell)$ for some arity parameter $\ell \geq 1$. Let* $\mathsf{Com} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{VerCommit})$ *be a commitment scheme (as per Definition 4.2) whose input space $\mathcal{D}$ is such that $\mathcal{D}_i \subset \mathcal{D}$ for all $i \in [\ell]$. A* commit-and-prove NIZK *for* Com *and $\{\mathcal{R}_\lambda\}_{\lambda\in\mathbb{N}}$ is a* NIZK *for a family of relations $\{\mathcal{R}^{\mathsf{Com}}_\lambda\}_{\lambda\in\mathbb{N}}$ such that:*

- *every $\mathbf{R} \in \mathcal{R}^{\mathsf{Com}}$ is represented by a pair (ck, R) where $\mathsf{ck} \in \mathsf{Setup}(1^\lambda)$ and $R \in \mathcal{R}_\lambda$;*
- *$\mathbf{R}$ is over pairs $(\mathbf{x}, \mathbf{w})$ where the statement is $\mathbf{x} := (x, (c_j)_{j\in[\ell]}) \in \mathcal{D}_x \times \mathcal{C}^\ell$, the witness is $\mathbf{w} := ((u_j)_{j\in[\ell]}, (o_j)_{j\in[\ell]}, \omega) \in \mathcal{D}_1 \times \cdots \times \mathcal{D}_\ell \times \mathcal{O}^\ell \times \mathcal{D}_\omega$, and the relation $\mathbf{R}$ holds iff*

$$\bigwedge\nolimits_{j\in[\ell]} \mathsf{VerCommit}(\mathsf{ck}, c_j, u_j, o_j) = 1 \wedge R(x, (u_j)_{j\in[\ell]}, \omega) = 1$$

Furthermore, when we say that CP *is knowledge-sound for a relation generator $\mathcal{RG}$ and auxiliary input generator $\mathcal{Z}$ (denoted $\mathsf{KSND}(\mathcal{RG}, \mathcal{Z})$, for short) we mean it is a knowledge-sound* NIZK *for the relation generator $\mathcal{RG}_{\mathsf{Com}}(1^\lambda)$ that runs $\mathsf{ck} \leftarrow \mathsf{Setup}(1^\lambda)$ and $(R, \mathsf{aux}_R) \leftarrow \mathcal{RG}(1^\lambda)$, and returns $((\mathsf{ck}, R), \mathsf{aux}_R)$.*

Explicit simplified syntax for a CP-NIZK. We denote a CP-NIZK as a triple of algorithms $\mathsf{CP} = (\mathsf{KeyGen}, \mathsf{Prove}, \mathsf{VerProof})$. For ease of exposition, an explicit syntax for CP's algorithms is as follows.

$\mathsf{KeyGen}(\mathsf{ck}, R)$	$\mathsf{Prove}(\mathsf{ek}, x, (c_j)_{j\in[\ell]}, (u_j)_{j\in[\ell]}, (o_j)_{j\in[\ell]}, \omega)$	$\mathsf{VerProof}(\mathsf{vk}, x, (c_j)_{j\in[\ell]}, \pi)$
returns $\mathsf{crs} := (\mathsf{ek}, \mathsf{vk})$	**returns** π	**returns** $b \in \{0, 1\}$

Comparing with existing definitions In the context of non-interactive proof systems, a variety of notions of commit-and-prove schemes have appeared explicitly in a few works in the literature [Fuc11, EG14, CFH$^+$15, Lip16, CFQ19a]. The $\mathsf{CP\text{-}NIZK}$ notion given above is the one that aims to make the proof system and the commitment scheme as decoupled as possible (and in particular it makes less requirements on the commitment). It is in fact a specialization of the NIZK notion when considering specific families of relations that include verifying openings of commitments.

One difference with all the other definitions [Fuc11, EG14, CFH$^+$15, Lip16] is that ours does not require the commitment to be a trapdoor commitment. In addition to this, when comparing with the definitions of Fuchsbauer [Fuc11] and Escala and Groth [EG14], we leave the possibility that the proof system has its own common reference string and that the relation to be proven takes other inputs in addition to those that are explicitly committed. Our notion is closer to the one given by Lipmaa [Lip16] (with the exception that again we do not need commitments to be trapdoor).

Finally, in comparison to the notion of commit-and-prove SNARKs defined in [CFH$^+$15] for the Geppetto scheme, the main differences are the following. First, our commitment key can be generated without fixing a priori a relation (or a set of relations, e.g., a multi-QAP). Second, in the model of [CFH$^+$15] one needs to commit to data using a commitment key corresponding to a specific portion of the input (in their lingo a "bank"), whereas in our model one can just commit to a vector of data, and only at proving time one assigns that data to a specific input slot. Third, in our notion the commitment scheme does not need to be a trapdoor commitment.

5 Encrypt-and-Prove

Here we describe a variant of commit-and-prove where we can prove/verify over encrypted data.

5.1 Motivation

Encrypt-and-Prove Zero-Knowledge Proofs (EP-ZKP) combine encryption and (non-interactive) zero-knowledge proofs; it was proposed originally in the 3rd ZKProof Workshop [LCKO19]. In this framework one can encrypt a message u to a ciphertext c and prove that the message belongs to a language $\mathcal{L}$ while also proving knowledge of the randomness that encrypts that same message to the ciphertext. This can be checked by a verifier that only holds a ciphertext c.

The formalization shares much with that of commit-and-prove (as well much of its motivation), but there are a few points of divergence. Although none of them on its own is major, together they justify a separate definition:

– its setup involves secret keys and is performed by a specific user, in contrast to a single setup with a public commitment key shared by everybody;
– while commitments are required to be binding, we may not require ciphertexts to be. From the notion of encryption above we require only indistinguishability under chosen plaintext attack (IND-CPA). Binding can be seen an extra property. We stress, however, that: this property is necessary in the case in which we use two $\mathsf{EP\text{-}NIZK}$s to compose proofs on the same data; many common encryption schemes, ElGamal among others, are binding.
– in some protocols, a public-key infrastructure may give substantially different properties than commitments. This is better explained with an example where using encrypt-and-prove in a protocol makes it immediately obvious who has access to the data and who does not: if we broadcast an encryption of data x through the public key of user U—together with a proof over

the encryption of the data—we know this will leak the property to everybody as well as the fact that the data is revealed to U. We can achieve this through a commitment scheme but less immediately and perhaps more inefficiently.

5.2 Preliminaries on Public-Key Encryption

We recall the definition of public-key encryption. To synchronize the notation, we denote the message as u, same as in the definition of commitment scheme.

Definition 5.1 (Public-Key Encryption Scheme). *A* public-key encryption scheme *is a triple of algorithms* $\Pi_{\mathsf{enc}} = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ *that work as follows satisfying the notions of* correctness *and* indistinguishability *under chosen plaintext attack (IND-CPA) defined below.*

- $\mathsf{KeyGen}(1^\lambda) \to (\mathsf{sk}, \mathsf{pk})$*: takes the security parameter* λ *as an input and outputs a secret key* sk *and a public key* pk *including descriptions of the input space* $\mathcal{D}$ *and ciphertext space* $\mathcal{CT}$*.*
- $\mathsf{Encrypt}(\mathsf{pk}, u; r) \to ct$*: takes the public key* pk *and a message* $u \in \mathcal{D}$ *as inputs and outputs a corresponding ciphertext* ct *with respect to an auxiliary random* r*.*
- $\mathsf{Decrypt}(\mathsf{sk}, ct) \to u$*: takes the secret key* sk *and a ciphertext* ct *as inputs and outputs a corresponding message* u*.*

Correctness. For all $\lambda \in \mathbb{N}$ and any input $u \in \mathcal{D}$ we have:

$$\Pr\left[\ (\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda)\ ,\ ct \leftarrow \mathsf{Encrypt}(\mathsf{pk}, u; r)\ :\ \mathsf{Decrypt}(\mathsf{sk}, ct) = u\ \right] = 1$$

Indistinguishability (IND-CPA). For every polynomial-time adversary $\mathcal{A}$:

$$\begin{aligned} &\Pr\left[\ (\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda)\ ,\ (u_0, u_1) \leftarrow \mathcal{A}(\mathsf{pk})\ ,\ ct \leftarrow \mathsf{Encrypt}(\mathsf{pk}, u_0; r)\ :\ \mathcal{A}(\mathsf{pk}, ct, u_0, u_1) = 0\ \right] \\ \approx &\Pr\left[\ (\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda), (u_0, u_1) \leftarrow \mathcal{A}(\mathsf{pk})\ ,\ ct \leftarrow \mathsf{Encrypt}(\mathsf{pk}, u_1; r)\ :\ \mathcal{A}(\mathsf{pk}, ct, u_0, u_1) = 0\ \right] \end{aligned}$$

5.3 Definition of Encrypt-and-Prove NIZKs

Similar to the CP-NIZK in Definition 4.3, an *encrypt-and-prove* NIZK (EP-NIZK) is a NIZK that can prove knowledge of (x, w) such that $R(x, w)$ holds with respect to a statement $x = (y, ct)$ and a witness $w = (u, \omega)$ such that ct decrypts to a message u. The input domain $\mathcal{D}_u$ splits over ℓ subdomains, which we call as *encryption slots*; each item in the encryption slot represents an input (or a vector of inputs) to the relation which the prover has previously encrypted.

Definition 5.2 (EP-NIZK)**.** *Let* $\{\mathcal{R}_\lambda\}_{\lambda \in \mathbb{N}}$ *be a family of relations* R *over* $\mathcal{D}_x \times \mathcal{D}_u \times \mathcal{D}_\omega$ *such that* $\mathcal{D}_u$ *splits over* ℓ *arbitrary domains* $(\mathcal{D}_1 \times \cdots \times \mathcal{D}_\ell)$ *for some arity parameter* $\ell \geq 1$*. Let* $\Pi_{\mathsf{enc}} = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ *be an encryption scheme (as per Definition 5.2) whose input space* $\mathcal{D}$ *is such that* $\mathcal{D}_i \subset \mathcal{D}$ *for all* $i \in [\ell]$*. An* encrypt-and-prove NIZK *for* Π_{enc} *and* $\{\mathcal{R}_\lambda\}_{\lambda \in \mathbb{N}}$ *is a* NIZK *for a family of relations* $\{\mathcal{R}_\lambda^{\Pi_{\mathsf{enc}}}\}_{\lambda \in \mathbb{N}}$ *such that:*

- *every* $\mathbf{R} \in \mathcal{R}^{\Pi_{\mathsf{enc}}}$ *is represented by a pair* (pk, R) *where* $\mathsf{pk} \in \mathsf{KeyGen}(1^\lambda)$ *and* $R \in \mathcal{R}_\lambda$*;*
- $\mathbf{R}$ *is over pairs* $(\mathbf{x}, \mathbf{w})$ *where the statement is* $\mathbf{x} := (x, (ct_j)_{j \in [\ell]}) \in \mathcal{D}_x \times \mathcal{CT}^\ell$*, the witness is* $\mathbf{w} := ((u_j)_{j \in [\ell]}, \omega) \in \mathcal{D}_1 \times \cdots \times \mathcal{D}_\ell \times \mathcal{D}_\omega$*, and the relation* $\mathbf{R}$ *holds iff*

$$\bigwedge\nolimits_{j \in [\ell]} \mathsf{Encrypt}(\mathsf{pk}, u; r) = ct \wedge R(x, (u_j)_{j \in [\ell]}, \omega) = 1$$

Furthermore, when we say that EP *is knowledge-sound for a relation generator* $\mathcal{RG}$ *and auxiliary input generator* $\mathcal{Z}$ *(denoted* $\mathsf{KSND}(\mathcal{RG}, \mathcal{Z})$*, for short) we mean it is a knowledge-sound* NIZK *for the relation generator* $\mathcal{RG}_{\Pi_{\mathsf{enc}}}(1^\lambda)$ *that runs* $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ *and* $(R, \mathsf{aux}_R) \leftarrow \mathcal{RG}(1^\lambda)$*, and returns* $((\mathsf{pk}, R), \mathsf{aux}_R)$*.*

Explicit simplified syntax for a EP-NIZK. We denote a EP-NIZK as a tuple of algorithms $\mathsf{EP} = (\mathsf{KeyGen}, \mathsf{Prove}, \mathsf{VerProof})$. For ease of exposition, an explicit syntax for EP's algorithms is as follows.

$\underline{\mathsf{KeyGen}(\mathsf{pk}, R)}$	$\underline{\mathsf{Prove}(\mathsf{ek}, x, (ct_j)_{j\in[\ell]}, (u_j)_{j\in[\ell]}, \omega)}$	$\underline{\mathsf{VerProof}(\mathsf{vk}, x, (ct_j)_{j\in[\ell]}, \pi)}$
returns $\mathsf{crs} := (\mathsf{ek}, \mathsf{vk})$	**returns** π	**returns** $b \in \{0, 1\}$

Beyond Commit-(or Encrypt-)and-Prove. We can think of commit-and-prove and encrypt-and-prove as a framework for zero-knowledge over augmented relations. We augment these relations with cryptographic primitives (commitment and encryption) that have a somewhat special role. One could push a similar line of formalization even further:

- one could have proof systems that support both commitments and encryptions as having a special role (roughly, a "(commit-or-encrypt)-and-prove");
- one could augment relations with cryptographic primitives beyond encryption and commitments, such as signatures. One could have a "sign-and-prove" framework able to efficiently prove $\mathsf{VerifySignature}(\mathsf{pk}, \sigma, m)$ over some committed (or encrypted) private data m. Something similar could be done for non-hiding cryptographic primitives such as accumulators and variants of vector commitments.

The observations above should be considered parenthetical; the formalizations we mention are currently out of the scope of this proposal.

6 Potential Candidates for the Reference Standard

Some candidate commitment schemes to standardize are the following:

- Pedersen commitments over different types of distributions (e.g. uniform distribution, Lagrange polynomials evaluated in a point);
  - It will possibly be required to standardize preliminaries necessary for Pedersen commitments: groups, basic cryptographic properties we require in them, how to sample binding commitment keys safely.
- SNARK-friendly commitment schemes, for example the JubJub curve used in ZCash;
- hashing through random-oracle or arbitrary collision-resistant hash functions;
- Merkle trees and some vector commitments in general.

Some candidate commit-and-prove schemes to standardize are the following:

- Sigma protocols;
- Groth-Sahai [GS08];
- Groth16 [Gro16] with SNARK-friendly commitment schemes;
- LegoGroth16 [CFQ19b].

References

AFG$^+$16. Masayuki Abe, Georg Fuchsbauer, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. Structure-preserving signatures and commitments to group elements. *Journal of Cryptology*, 29(2):363–421, 2016.

AGM18. Shashank Agrawal, Chaya Ganesh, and Payman Mohassel. Non-interactive zero-knowledge proofs for composite statements. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 643–673. Springer, Heidelberg, August 2018.

BCM$^+$18. Daniel Benarroch, Ran Canetti, Andrew Miller, Shashank Agrawal, Tony Arcieri, Vipin Bharathan, Josh Cincinnati, Joshua Daniel, Anuj Das Gupta, Angelo De Caro, Michael Dixon, Maria Dubovitskaya, Nathan George, Brett Hemenway Falk, Hugo Krawczyk, Jason Law, Anna Lysyanskaya, Zaki Manian, Eduardo Morais, Neha Narula, Gavin Pacini, Jonathan Rouach, Kartheek Solipuram, Mayank Varia, Douglas Wikstrom, and Aviv Zohar. Applications track proceeding. Technical report, ZKProof Standards, Berkeley, CA, May 2018. `https://zkproof.org/documents.html`.

BCPR14. Nir Bitansky, Ran Canetti, Omer Paneth, and Alon Rosen. On the existence of extractable one-way functions. In David B. Shmoys, editor, *46th ACM STOC*, pages 505–514. ACM Press, May / June 2014.

Bd94. Josh Cohen Benaloh and Michael de Mare. One-way accumulators: A decentralized alternative to digital sinatures (extended abstract). In Tor Helleseth, editor, *EUROCRYPT'93*, volume 765 of *LNCS*, pages 274–285. Springer, Heidelberg, May 1994.

BP15. Elette Boyle and Rafael Pass. Limits of extractability assumptions with distributional auxiliary input. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part II*, volume 9453 of *LNCS*, pages 236–261. Springer, Heidelberg, November / December 2015.

But18. Vitalik Buterin. On-chain scaling to potentially 500 tx/sec through mass tx validation, 2018. `https://ethresear.ch/t/on-chain-scaling-to-potentially-500-tx-sec-through-mass-tx-validation/3477`.

CF13. Dario Catalano and Dario Fiore. Vector commitments and their applications. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 55–72. Springer, Heidelberg, February / March 2013.

CFH$^+$15. Craig Costello, Cédric Fournet, Jon Howell, Markulf Kohlweiss, Benjamin Kreuter, Michael Naehrig, Bryan Parno, and Samee Zahur. Geppetto: Versatile verifiable computation. In *2015 IEEE Symposium on Security and Privacy*, pages 253–270. IEEE Computer Society Press, May 2015.

CFQ19a. Matteo Campanelli, Dario Fiore, and Anaïs Querol. LegoSNARK: Modular Design and Composition of Succinct Zero-Knowledge Proofs. Cryptology ePrint Archive, Report 2019/142, 2019. `https://eprint.iacr.org/2019/142.pdf`.

CFQ19b. Matteo Campanelli, Dario Fiore, and Anaïs Querol. LegoSNARK: Modular design and composition of succinct zero-knowledge proofs. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 2075–2092. ACM Press, November 2019.

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

EG14. Alex Escala and Jens Groth. Fine-tuning Groth-Sahai proofs. In Hugo Krawczyk, editor, *PKC 2014*, volume 8383 of *LNCS*, pages 630–649. Springer, Heidelberg, March 2014.

FFG$^+$16. Dario Fiore, Cédric Fournet, Esha Ghosh, Markulf Kohlweiss, Olga Ohrimenko, and Bryan Parno. Hash first, argue later: Adaptive verifiable computations on outsourced data. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1304–1316, 2016.

Fuc11. Georg Fuchsbauer. Commuting signatures and verifiable encryption. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 224–245. Springer, Heidelberg, May 2011.

GKV$^+$18. Jens Groth, Yael Kalai, Muthu Venkitasubramaniam, Nir Bitansky, Ran Canetti, Henry Corrigan-Gibbs, Shafi Goldwasser, Charanjit Jutla, Yuval Ishai, Rafail Ostrovsky, Omer Paneth, Tal Rabin, Mariana Raykova, Ron Rothblum, Alessandra Scafuro, Eran Tromer, and Douglas Wikström. Security track proceeding. Technical report, ZKProof Standards, Berkeley, CA, May 2018. `https://zkproof.org/documents.html`.

Gro16. Jens Groth. On the size of pairing-based non-interactive arguments. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 305–326. Springer, Heidelberg, May 2016.

GS08. Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 415–432. Springer, 2008.

HBHW16. Daira Hopwood, Sean Bowe, Taylor Hornby, and Nathan Wilcox. Zcash protocol specification. *Tech. rep. 2016–1.10. Zerocoin Electric Coin Company, Tech. Rep.*, 2016.
Kil89. J. Kilian. Uses of randomness in algorithms and protocols. PhD Thesis. Massachusetts Institute of Technology, 1989.
KZG10. Aniket Kate, Gregory M Zaverucha, and Ian Goldberg. Constant-size commitments to polynomials and their applications. In *International conference on the theory and application of cryptology and information security*, pages 177–194. Springer, 2010.
LCKO19. Jiwon Lee, Jaekyoung Choi, Jihye Kim, and Hyunok Oh. Saver: Snark-friendly, additively-homomorphic, and verifiable encryption and decryption with rerandomization. *IACR Cryptol. ePrint Arch.*, 2019:1270, 2019.
Lip16. Helger Lipmaa. Prover-efficient commit-and-prove zero-knowledge SNARKs. In David Pointcheval, Abderrahmane Nitaj, and Tajjeeddine Rachidi, editors, *AFRICACRYPT 16*, volume 9646 of *LNCS*, pages 185–206. Springer, Heidelberg, April 2016.
Mer88. Ralph C. Merkle. A digital signature based on a conventional encryption function. In Carl Pomerance, editor, *CRYPTO'87*, volume 293 of *LNCS*, pages 369–378. Springer, Heidelberg, August 1988.
Mic00. Silvio Micali. Computationally sound proofs. *SIAM Journal on Computing*, 30(4):1253–1298, 2000.
Tam03. Roberto Tamassia. Authenticated data structures. In Giuseppe Di Battista and Uri Zwick, editors, *Algorithms - ESA 2003*, pages 2–5, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

A Other Flavors of Commit/Encrypt-and-Prove NIZKs

In this section we describe a variant of NIZKs (proposed in [CFQ19a]) that lies in between standard NIZKs and CP-NIZKs. We believe it is worth mentioning since it captures a class of existing schemes that are not explicitly commit-and-prove but they implicitly have a weak form of the commit flavor.

This class of schemes are called NIZKs *with commit-carrying proofs* (or commit-carrying NIZKs, cc-NIZKs for short). In a nutshell, a cc-NIZK is like a NIZK in which the proof contains a commitment to the portion u of the witness. Formalizing this idea requires to make explicit the commitment scheme associated to the NIZK, as well as the commitment key that is part of the common reference string. In [CFQ19a] Campanelli, Fiore and Querol discuss how many of the existing NIZK constructions satisfy this property. In particular, this holds for popular zkSNARKs like [Gro16] They also show how cc-NIZKs can be lifted to become full fledged, composable, CP-NIZKs.

Definition A.1 (cc-NIZK). *A cc-NIZK is a tuple* $\mathsf{cc}\Pi$ *of algorithms working as follows:*

- $\mathsf{KeyGen}(R) \to (\mathsf{ck}, \mathsf{ek}, \mathsf{vk})$*: the key generation takes as input the security parameter* λ *and a relation* $R \in \mathcal{R}_\lambda$*, and outputs a common reference string that includes a commitment key* ck*, an evaluaton key* ek *and a verification key* vk*.*
- $\mathsf{Prove}(\mathsf{ek}, x, w) \to (\pi, c; o)$*: the proving algorithm takes as input an evaluation key* ek*, a statement* x *and a witness* $w := (u, \omega)$ *such that the relation* $R(x, u, \omega)$ *holds, and it outputs a proof* π*, a commitment* c *and an opening* o *such that* $\mathsf{VerCommit}(\mathsf{ck}, c, u, o) = 1$*.*
- $\mathsf{VerProof}(\mathsf{vk}, x, c, \pi) \to b$*: the verification algorithm takes a verification key* vk*, a statement* x*, a commitment* c*, and either accepts* $(b = 1)$ *or rejects* $(b = 0)$ *the proof* π*.*
- $\mathsf{VerCommit}(\mathsf{ck}, c, u, o) \to b$*: the commitment verification algorithm takes as input a commitment key* ck*, a commitment* c*, a message* u *and an opening* o *and accepts* $(b = 1)$ *or rejects* $(b = 0)$*.*

Completeness. $\mathsf{cc}\Pi$ *is* complete *if for any* $\lambda \in \mathbb{N}$*,* $R \in \mathcal{R}_\lambda$ *and* (x, w) *such that* $R(x, w)$*, it holds:*

$$\Pr\left[\ (\mathsf{ck}, \mathsf{ek}, \mathsf{vk}) \leftarrow \mathsf{KeyGen}(R)\ ,\ (c, \pi; o) \leftarrow \mathsf{Prove}(\mathsf{ek}, x, w)\ :\ \mathsf{VerProof}(\mathsf{vk}, x, c, \pi)\ \right] = 1$$

Knowledge Soundness. *Let* $\mathcal{RG}$ *be a relation generator such that* $\mathcal{RG}_\lambda \subseteq \mathcal{R}_\lambda$*.* $\mathsf{cc}\Pi$ *satisfies* *knowledge soundness for $\mathcal{RG}$ and auxiliary input distribution $\mathcal{Z}$, or* $\mathsf{ccKSND}(\mathcal{RG}, \mathcal{Z})$*, if there exists a (non-uniform) efficient extractor $\mathcal{E}$ that for every (non-uniform) efficient adversary $\mathcal{A}$ is such that* $\Pr[\mathsf{Game}^{\mathsf{ccKSND}}_{\mathcal{RG},\mathcal{Z},\mathcal{A},\mathcal{E}} = 1] = \mathsf{negl}$*. We say that* $\mathsf{cc}\Pi$ *is* knowledge sound *if there exist benign $\mathcal{RG}$ and $\mathcal{Z}$ such that* $\mathsf{cc}\Pi$ *is* $\mathsf{ccKSND}(\mathcal{RG}, \mathcal{Z})$.

$$
\begin{array}{l}
\underline{\mathsf{Game}^{\mathsf{ccKSND}}_{\mathcal{RG},\mathcal{Z},\mathcal{A},\mathcal{E}} \to b \in \{0,1\}} \\
(R, \mathsf{aux}_R) \leftarrow \mathcal{RG}(1^\lambda) \\
\mathsf{crs} := (\mathsf{ck}, \mathsf{ek}, \mathsf{vk}) \leftarrow \mathsf{KeyGen}(R) \\
\mathsf{aux}_Z \leftarrow \mathcal{Z}(R, \mathsf{aux}_R, \mathsf{crs}) \\
\big(x, c, \pi\big) \leftarrow \mathcal{A}(R, \mathsf{crs}, \mathsf{aux}_R, \mathsf{aux}_Z) \\
\big(u, o, \omega\big) \leftarrow \mathcal{E}^{\mathcal{A}}(R, \mathsf{crs}, \mathsf{aux}_R, \mathsf{aux}_Z) \\
b \leftarrow \mathsf{VerProof}(\mathsf{vk}, x, c, \pi) = 1 \ \wedge (\mathsf{VerCommit}(\mathsf{ck}, c, u, o) = 0 \ \vee R(x, u, \omega) = 0)
\end{array}
$$

Composable Zero-Knowledge. *A scheme* $\mathsf{cc}\Pi$ *is* composable zero-knowledge *for a relation generator $\mathcal{RG}$ if for every adversary $\mathcal{A}$ there exists a simulator $\mathcal{S} = (\mathcal{S}_{\mathsf{kg}}, \mathcal{S}_{\mathsf{prv}})$ such that both following conditions hold for all adversaries $\mathcal{A}$:*

Keys Indistinguishability.

$$
\begin{aligned}
&\Pr\big[\ (R, \mathsf{aux}_R) \leftarrow \mathcal{RG}(1^\lambda)\ ,\quad \mathsf{crs} \leftarrow \mathsf{KeyGen}(R)\ :\ \mathcal{A}(\mathsf{crs}, \mathsf{aux}_R) = 1\ \big] \\
\approx &\Pr\big[\ (R, \mathsf{aux}_R) \leftarrow \mathcal{RG}(1^\lambda)\ ,\ (\mathsf{crs}, \mathsf{td}_{\mathsf{k}}) \leftarrow \mathcal{S}_{\mathsf{kg}}(R)\ :\ \mathcal{A}(\mathsf{crs}, \mathsf{aux}_R) = 1\ \big]
\end{aligned}
$$

Proof Indistinguishability. *For all (x, w) such that $R(x, w) = 1$,*

$$
\begin{aligned}
&\Pr\big[(R, \mathsf{aux}_R) \leftarrow \mathcal{RG}(1^\lambda),\ (\mathsf{crs}, \mathsf{td}_{\mathsf{k}}) \leftarrow \mathcal{S}_{\mathsf{kg}}(R),\ (c, \pi; o) \leftarrow \mathsf{Prove}(\mathsf{ek}, x, w)\ :\ \mathcal{A}(\mathsf{crs}, \mathsf{aux}_R, c, \pi) = 1\big] \\
\approx &\Pr\big[(R, \mathsf{aux}_R) \leftarrow \mathcal{RG}(1^\lambda),\ (\mathsf{crs}, \mathsf{td}_{\mathsf{k}}) \leftarrow \mathcal{S}_{\mathsf{kg}}(R),\ (c, \pi) \leftarrow \mathcal{S}_{\mathsf{prv}}(\mathsf{crs}, \mathsf{td}_{\mathsf{k}}, x)\ :\ \mathcal{A}(\mathsf{crs}, \mathsf{aux}_R, c, \pi) = 1\big]
\end{aligned}
$$

Binding. $\mathsf{cc}\Pi$ *is* binding *if for every polynomial-time adversary $\mathcal{A}$ the following holds:*

$$
\Pr\left[\begin{array}{rcr}
(R, \mathsf{aux}_R) \leftarrow \mathcal{RG}(1^\lambda) & & \mathsf{VerCommit}(\mathsf{ck}, c, u, o) = 1 \\
(\mathsf{ck}, \mathsf{ek}, \mathsf{vk}) \leftarrow \mathsf{KeyGen}(R) & : & \mathsf{VerCommit}(\mathsf{ck}, c, u', o') = 1 \\
(c, u, o, u', o') \leftarrow \mathcal{A}(R, \mathsf{crs}, \mathsf{aux}_R) & & u \neq u'
\end{array}\right] = \mathsf{negl}(\lambda)
$$

Remark A.1. While our definitions consider the case where the proof contains a commitment to a portion u of the witness $w = (u, \omega)$, notice that this partition of the witness is arbitrary and thus this notion also captures those constructions where the commitment is to the entire witness if one thinks of a void ω.

Remark A.2. It is possible to define analogous notions for encrypt-and-prove: encryption-carrying NIZKs, (i.e. ec-NIZK).