# Community Standards Proposal for Commit-and-Prove Zero-Knowledge Proof Systems

Daniel Benarroch[1], Matteo Campanelli[2], and Dario Fiore[2]

[1] QED-it, Tel Aviv, Israel
daniel@qed-it.com
[2] IMDEA Software Institute, Madrid, Spain
matteo.campanelli@imdea.org, dario.fiore@imdea.org

## 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 propose and start 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.* In particular, we believe that the starting point will be discussing a community standard for the definition of commitment schemes. In spite of being mentioned in [BCM+18], a more extended discussion on terminology, syntax and definitions of commitments is currently lacking. A second step that, we believe, is worth discussing is the notion of commit-and-prove ZKPs: several different notions are used in the literature, and it would be important to agree on agree on a single definition definition or outlining a taxonomy of the existing variants.

## 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 paradigm has several interesting features:

- 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).
- 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.
- 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, CFQ19]), or simply because the public parameters of the ZKP systems are generated in different points in time or by different organizations.

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.

## 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,

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

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, CFQ19], 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.

## 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 [CFQ19]. 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 $1^\lambda$ to denote its unary representation. Throughout the paper we assume that all the algorithms of the cryptographic schemes take as input $1^\lambda$, 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 $\{1, \dots, n\}$ and by $[: n]$ the set $\{0, 1, \dots, n-1\}$. We use $(u_j)_{j\in[\ell]}$ to denote the tuple of elements $(u_1, \dots, u_\ell)$.

### 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 $\ell$ arbitrary domains $(\mathcal{D}_1 \times \cdots \times \mathcal{D}_\ell)$ for some arity $\ell$. In our security definitions we assume relations to be generated by a *relation generator* $\mathcal{RG}(1^\lambda)$ that, on input the security parameter $1^\lambda$, outputs $R$ together with some side information, an auxiliary input $\mathsf{aux}_R$, 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 ($\mathsf{NIZK}$, for short).

**Definition 4.1** ($\mathsf{NIZK}$)**.** *A* $\mathsf{NIZK}$ *for* $\{\mathcal{R}_\lambda\}_{\lambda\in\mathbb{N}}$ *is a tuple of three 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* $\lambda$ *and a relation* $R \in \mathcal{R}_\lambda$*, and outputs a common reference string consisting of an evaluation and a verification key.*
- $\mathsf{Prove}(\mathsf{ek}, x, w) \to \pi$ *takes an evaluation key for a relation* $R$*, a statement* $x$*, and a witness* $w$ *such that* $R(x, w)$ *holds, and returns a proof* $\pi$*.*
- $\mathsf{VerProof}(\mathsf{vk}, x, \pi) \to b$ *takes a verification key, a statement* $x$*, and either accepts* ($b = 1$) *or rejects* ($b = 0$) *the proof* $\pi$*.*

**Completeness.** *For any* $\lambda \in \mathbb{N}$*,* $R \in \mathcal{R}_\lambda$ *and* $(x, w)$ *such that* $R(x, w)$*, 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.** $\Pi$ *has knowledge soundness 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* $\Pi$ *is knowledge sound if there exists benign* $\mathcal{RG}$ *and* $\mathcal{Z}$ *such that* $\Pi$ *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* $\Pi$ *satisfies 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

$$\Pr\begin{bmatrix}(R, \mathsf{aux}_R) \leftarrow \mathcal{RG}(1^\lambda);\\ \mathsf{crs} \leftarrow \mathsf{KeyGen}(R)\\ \mathcal{A}(\mathsf{crs}, \mathsf{aux}_R) = 1\end{bmatrix} \approx \Pr\begin{bmatrix}(R, \mathsf{aux}_R) \leftarrow \mathcal{RG}(1^\lambda);\\ (\mathsf{crs}, \mathsf{td}_\mathsf{k}) \leftarrow \mathcal{S}_{\mathsf{kg}}(R)\\ \mathcal{A}(\mathsf{crs}, \mathsf{aux}_R) = 1\end{bmatrix}$$

PROOF INDISTINGUISHABILITY *For all* $(x, w)$ *such that* $R(x, w) = 1$*,*

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

*Remark 4.1 (On auxiliary inputs).* In the notion of knowledge soundness defined above we consider two kinds of auxiliary inputs, $\mathsf{aux}_R$ generated together with the relation by $\mathcal{RG}$, and $\mathsf{aux}_Z$ that is generated from some distribution $\mathcal{Z}$ that may depend on the common reference string that in turns depends on $R$. Notice that although this notion 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 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.** *A commitment scheme is a tuple 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 and outputs a commitment key* $\mathsf{ck}$*. 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* $\mathsf{ck}$ *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:

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

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

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

**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 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 $x$ (or $\perp$ when appropriate).

### 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 $\ell$ subdomains. We call these subdomains *commitment slots* following [CFQ19]. 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 $\ell$ 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 $(\mathsf{ck}, R)$ 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*

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

*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 $((\mathsf{ck}, R), \mathsf{aux}_R)$.*

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

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

*Remark 4.5 (Comparing with existing definitions).* To define the Geppetto scheme [CFH+15] the authors define a notion of commit-and-prove SNARKs. Here we highlight the main differences between their definition and ours. 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. Our notion is closer to the one given by Lipmaa [Lip16] (with the exception that again we do not need commitments to be trapdoor), and is in fact a specialization of the SNARK notion when considering specific families of relations that include verifying openings of commitments.

## References


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.

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.

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.

CFQ19. Matteo Campanelli, Dario Fiore, and Anaïs Querol. LegoSNARK: Modular Design and Composition of Succinct Zero-Knowledge Proofs. Cryptology ePrint Archive, Report 2019/142, 2019. `http://eprint.iacr.org/`.

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.

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.

Kil89. J. Kilian. Uses of randomness in algorithms and protocols. PhD Thesis. Massachusetts Institute of Technology, 1989.

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.

Mic00. Silvio Micali. Computationally sound proofs. *SIAM Journal on Computing*, 30(4):1253–1298, 2000.

## A Other Flavors of Commit-and-Prove NIZKs

In this section we describe a variant of NIZKs (proposed in [CFQ19]) that lies in between standard NIZKs and CP-NIZKs. We call these schemes NIZK*s 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 [CFQ19] 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* ccΠ *of algorithms working as follows:*

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

– $\mathsf{VerCommit}(\mathsf{ck}, c, u, o) \to b$: *the commitment verification algorithm takes as input a commitment key, a commitment $c$, a message $u$ and an opening $o$ and accepts ($b = 1$) or rejects ($b = 0$).*

**Completeness.** *For any $\lambda \in \mathbb{N}$, $R \in \mathcal{R}_\lambda$ and $(x, w)$ such that $R(x, w)$, it holds*

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

**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 $\mathsf{ccKSND}(\mathcal{RG}, \mathcal{Z})$.*

$\mathsf{Game}^{\mathsf{ccKSND}}_{\mathcal{RG},\mathcal{Z},\mathcal{A},\mathcal{E}} \to b \in \{0, 1\}$

$(R, \mathsf{aux}_R) \leftarrow \mathcal{RG}(1^\lambda)$

$\mathsf{crs} := (\mathsf{ck}, \mathsf{ek}, \mathsf{vk}) \leftarrow \mathsf{KeyGen}(R)$

$\mathsf{aux}_Z \leftarrow \mathcal{Z}(R, \mathsf{aux}_R, \mathsf{crs})$

$(x, c, \pi) \leftarrow \mathcal{A}(R, \mathsf{crs}, \mathsf{aux}_R, \mathsf{aux}_Z)$

$(u, o, \omega) \leftarrow \mathcal{E}^{\mathcal{A}}(R, \mathsf{crs}, \mathsf{aux}_R, \mathsf{aux}_Z)$

$b \leftarrow \mathsf{VerProof}(\mathsf{vk}, x, c, \pi) = 1 \wedge (\mathsf{VerCommit}(\mathsf{ck}, c, u, o) = 0 \vee R(x, u, \omega) = 0)$

**Composable Zero-Knowledge.** *A scheme $\mathsf{cc\Pi}$ has 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.

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

Proof Indistinguishability. *For all $(x, w)$,*

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

**Binding.** *For every polynomial-time adversary $\mathcal{A}$ the following probability is* $\mathsf{negl}(\lambda)$*:*

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

*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 $\omega$.