Homomorphic Polynomial Commitments with Efficiently Verifiable Evaluation Proofs

Alan Szepieniec

Nervos Foundation
alan@nervos.org

1 Introduction

interactive proof systems for arbitrary computation
- zero-knowledge proofs - arithmetization - polynomial relations
circuit model: QAP and Groth16 NILP
- construction - used algebra - advantages and disadvantages
Turing model: STARK
- construction - used algebra - advantages and disadvantages
Class group
- groups of unknown order - vdfs / accumulators
Our contribution:
- polynomial commitments with groups of unknown order - efficiently verifiable evaluation proofs - interactive proofs for arbitrary computation

Notation.

- Let $f(x) \in \mathbb{F}_p[x]$ be a polynomial of degree at most $N-1$ where N is a power of two. The coefficients of $f(x)$ are denoted by f_i such that $f(x) \stackrel{\triangle}{=} \sum_{i=0}^{N-1} f_i x^i$.
- We work in a group $\mathbb{G}$ of unknown order (*e.g.* an ideal class group) with a designated base element $g \in \mathbb{G}$ with unknown order. (It might be tempting refer to this element as the *generator* but that terminology would imply that $\mathbb{G}$ is cyclic, which is not necessarily true.) We use multiplicative notation.
- Let $q \in \mathbb{N}$ be an integer with $q \gg p$.

2 Polynomial Commitment and Evaluation Proof

2.1 Commitment Scheme.

To commit to a polynomial $f(x) \in \mathbb{F}_p[x]$, the committer raises the designated group element g to the the power $\sum_{i=0}^{N-1} f_i q^i$; to open the commitment he produces this integer. Formally, the commitment scheme is described by the following functionalities:

- $\mathsf{com} : \mathbb{F}_p[x] \to \mathbb{G}\,,\ f(x) = \sum_{i=0}^{N-1} f_i x^i \mapsto g^{\sum_{i=0}^{N-1} f_i q^i}$

– $\mathsf{open} : \mathbb{G} \times \mathbb{Z} \to \{\perp\} \cup \mathbb{F}_p[x], (c, z) \mapsto \begin{cases} \text{if } g^z \neq c: & \perp \\ \text{else:} & f(x) = \sum_{i=0}^{N-1} f_i x^i \end{cases}$,
where $\forall i \in \{0, \ldots, M-1\} : f_i \equiv z_i \bmod p$ and where $(z_i)_{i=0}^{M-1}$ is the base-q expansion of z, *i.e.*, $z = \sum_{i=0}^{M-1} z_i q^i$ and $z_i \in \{0, \ldots, q-1\}$.

This commitment scheme is not meant to be hiding. It does have to be binding. The following theorem establishes this fact.

Theorem 1 (binding). *The commitment scheme* $(\mathsf{com}, \mathsf{open})$ *is computationally binding under the adaptive root assumption.*

Proof. Let $z_0 \neq z_1 \in \mathbb{Z}$ be two different valid openings of the commitment c then $g^{z_0} = g^{z_1} = c$, then $z_0 - z_1$ is a multiple of $\mathsf{ord}(g)$, which can be used in combination with the extended Euclidean algorithm to compute the dth root of g, for any d. This establishes that the ability to produce multiple openings of a single commitment to different polynomials breaks the adaptive root assumption. □

This commitment scheme is sort of homomorphic. The group of $\mathbb{Z}, +$ integers acts naturally on the group of unknown order $\mathbb{G}$; we embed the group of polynomials $\mathbb{F}_p[x], +$ into the group of integers by identifying $f(x) \in \mathbb{F}_p$ with $\sum_{i=0}^{N-1} f_i q^i \in \mathbb{Z}$. This embedding implies that the coefficients of the polynomial are not reduced modulo p; nevertheless, from the point of view of the commitment scheme, their representatives modulo p are understood. If these coefficients grow through a sequence of homomorphic operations to be larger than q, then the homomorphism fails. Therefore, q has to be large enough such that there will never be any overflow. While the group $\mathbb{G}$ is written multiplicatively, the "group" of commitments is written additively.

2.2 Proof of Correct Evaluation.

What follows is a protocol that allows the prover to prove that e is the evaluation of $f(x)$ in the given point z, where $f(x)$ was committed to in $c = \mathsf{com}(f(x))$. In particular, the verifier knows $c = \mathsf{com}(f(x)), z$, and e; and the prover knows $f(x)$. Formally, the proof system establishes membership in the language $\mathcal{L}_c = \{(e, z) \in \mathbb{F}_p \times \mathbb{F}_p \,|\, f(z) = e \wedge c = \mathsf{com}(f(x))\}$. We denote the protocol by $\mathsf{PoCEv}\{e, z, c \,|\, e = f(z) \wedge c = \mathsf{com}(f(x))\}$.

PoCEv is defined recursively as follows:

– **If** $N > 1$**:**
  - Split $f(x)$ into two polynomials: let $f_{(0)}(x) \triangleq \sum_{i=0}^{N/2-1} f_i x^i$ and $f_{(1)}(x) \triangleq \sum_{i=0}^{N/2-1} f_{N/2+i} x^i$.
  - Prover sends to Verifier: $e_0 \triangleq f_{(0)}(z) \bmod p, e_1 \triangleq f_{(1)}(z) \bmod p, c_0 \triangleq \mathsf{com}(f_{(0)}(x)), c_1 \triangleq \mathsf{com}(f_{(1)}(x))$.

- Verifier checks that $z^{N/2}e_1 + e_0 \stackrel{?}{\equiv} e \bmod p$ and that $q^{N/2}c_1 + c_0 \stackrel{?}{=} c$.[1]
- Verifier produces two random scalars, $a_0, a_1 \stackrel{\$}{\leftarrow} \mathbb{F}_p$ and sends them to Prover.
- Prover and Verifier proceed with the protocol $\mathsf{PoCEv}\{e', z, c' \,|\, e' = f'(z) \wedge c' = \mathsf{com}(f'(x))\}$ where $e' \stackrel{\triangle}{\equiv} a_0e_0 + a_1e_1 \bmod p$, $c' \stackrel{\triangle}{=} a_0c_0 + a_1c_1$, and $f'(x) \stackrel{\triangle}{=} a_0f_{(0)}(x) + a_1f_{(1)}(x)$. (The prover does not reduce the coefficients of this polynomial modulo p.)

– **If** $N = 1$**:**
- Prover sends f_0 to Verifier. (Remember, at this point $f(x) = f_0$.)
- Verifier checks that $f_0 \stackrel{?}{\equiv} e \bmod p$ and $c \stackrel{?}{=} \mathsf{com}(f(x))$.

2.3 Complexity Analysis.

The protocol recurses to a depth of $\log_2 N$ before it halts it triggers the case $N = 1$. At this point, the constant f_0 was determined from the coefficients of the original $f(x)$ and via $\log_2 N$ multiplications by random elements from $\mathbb{F}_p$ but without reduction modulo p. So $|f_0| \approx \log_2 N \times \log_2 p$. A more exact description provides a lower bound on the size of q. This establishes that the communication cost is logarithmic in N.

The verifier's computation involves logarithmically many (as a function of N) exponentiations in $\mathbb{F}_p$ (by exponent $N/2$) and as many exponentations in $\mathbb{G}$ (by exponent $q^{N/2}$). The total complexity of the field exponentiations is $O((\log N)^2)$. The total complexity of the group exponentiations is $O(N(\log N)^2)$ naïvely; but probably more like $O((\log N)^2)$ with the workaround of footnote 1.

2.4 Security Analysis

We start with proving a simpler, and statement, namely that the evaluation proof is sound under the assumption that $(\mathsf{com}, \mathsf{open})$ is a perfecly binding commitment scheme. This ridiculous is ridiculous in light of the fact that $\mathbb{G}$ is a finite group while the domain of com is infinite. Nevertheless, this first step serves the purpose of introducing the proof technique and language, and conveying the intuition behind soundness, and highlighting the obstacle that a valid proof should circumvent. Later, we drop the perfect binding assumption in favor of working in the generic group model, and prove that soundness still holds in this model.

Theorem 2 (soundness). *Suppose* $(\mathsf{com}, \mathsf{open})$ *is a perfectly binding commitment scheme. Then the proof system* $\mathsf{PoCEv}\{e, z, c \,|\, e = f(z) \wedge c = \mathsf{com}(f(x))\}$ *is statistically sound with respect to the language* $\mathcal{L}_c = \{(\boldsymbol{e}, \boldsymbol{z}) \in \mathbb{F}_p \times \mathbb{F}_p \,|\, \boldsymbol{f}(\boldsymbol{z}) = \boldsymbol{e} \wedge \mathsf{com}(\boldsymbol{f}(x)) = c\}$.

[1] Computing $q^{N/2}c_1$ naïvely is expensive. However, there is a workaround that avoids this expense, namely by having the prover compute this value and send it to the verifier. The two then engage in the proof of correct exponentiation PoE due to Wesolowski [**?**], or the batched variant thereof, PoHP [**?**].

Proof. By induction on N.

- **If $N = 1$:** The soundness of the proof system for $N = 1$ is a restatement of the binding property of the commitment scheme for $N = 1$.
- **If $N > 1$:** Suppose $(e, z) \notin \mathcal{L}_c$, meaning that $e \neq f(z)$.
  The binding property guarantees that $f_{(0)}(x)$ and $f_{(1)}(x)$ are uniquely defined as soon as c_0 and c_1 are sent. So the languages $\mathcal{L}_{c_0}$ and $\mathcal{L}_{c_1}$ are also uniquely defined.
  From $(e_0, z) \in \mathcal{L}_{c_0} \Leftrightarrow e_0 = f_{(0)}(z)$ and $(e_1, z) \in \mathcal{L}_{c_1} \Leftrightarrow e_1 = f_{(1)}(z)$, one derives $(e_0, z) \in \mathcal{L}_{c_0} \wedge (e_1, z) \in \mathcal{L}_{c_1} \Rightarrow e_0 + z^{N/2} e_1 = f_{(0)}(z) + z^{N/2} f_{(1)}(z)$ and this last implicand is equivalent to $(e, z) \in \mathcal{L}_c$ because $c = c_0 + q^{N/2} c_1$ and $e = e_0 + x^{N/2} e_1$. Negation therefore yields $(e, z) \notin \mathcal{L}_c \Rightarrow (e_0, z) \notin \mathcal{L}_{c_0} \vee (e_1, z) \notin \mathcal{L}_{c_1}$.
  Due to the binding commitment scheme, $f'(x) = a_0 f_{(0)}(x) + a_1 f_{(1)}(x)$ is unique as the opening of $c' = a_0 c_0 + a_1 c_1$. So $\mathcal{L}_{c'}$ is well defined and $(e', z) \in \mathcal{L}_{c'} \Leftrightarrow a_0 e_0 + a_1 e_1 = a_0 f_0(z) + a_1 f_1(z)$. When $e_0 \neq f_0(z)$ or $e_1 \neq f_1(z)$ this equation holds with probability $1/p$ for uniformly random a_0, a_1. With the complement probability, a_0, a_1 are sampled such that $(e', z) \notin \mathcal{L}_{c'}$. The inductive assumption states that the adversary must fail (except with negligible probability) to prove this membership. □

The reason why the proof of Theorem 2 fails when dropping the perfect binding assumption, is because a given commitment $c \in \mathbb{G}$ no longer uniquely determines a polynomial $f(x)$. As a result, the language $\mathcal{L}_c$ is ill-defined. Not only does the proof fail, the theorem statement becomes nonsensical.

It is possible to modify the definition by taking the union of all sets $\{(e, z) \in \mathbb{F}_p \times \mathbb{F}_p \mid f(z) = e\}$ for all $f(x)$ satisfying $\mathsf{com}(f(x)) = c$. The resulting language $\mathcal{L}_c^* = \{(e, z) \mid \exists f(x) \in \mathbb{F}_p[x] \,.\, c = \mathsf{com}(f(x)) \wedge f(z) = e\}$ is well defined, but hardly meaningful, because in general $\mathcal{L}_c^* = \mathbb{F}_p \times \mathbb{F}_p$. To see that any pair $(e, z) \in \mathcal{L}_c^*$, just find a $f(x)$ such that both $f(z) = e$ and $\mathsf{com}(f(x)) = c$ are satisfied — there are plenty of options for a generic group $\mathbb{G}$.

Instead, we want to rely on the fact that the prover P *knows* only one $f(x)$, even though c could be the proper commitment of any number of polynomials. Our strategy is motivated by the concept of *knowledge* in proofs of knowledge, where the prover does not establish that his claim is correct but furthermore that he *knows* a witness to this fact. In this context, "knowledge" is formalized by requiring the existence of an extractor machine E with the following properties:

- The extractor machine E has black box access to the prover P but only through the same interface as the verifier V. The difference between the extractor and the verifier is that the former gets to repeat interactions, record messages, and even rewind the prover. Additionally, the extractor controls any random oracles.
- If successful, the extractor machine terminates by outputting the information that the prover supposedly knows.
- The computational overhead of the extractor is small, under a suitable asymptotic or concrete definition of "small". This requirement guarantees

that the extractor is really extracting the knowledge from the prover, and not merely computing it from scratch while ignoring the prover.

The existence of E is a property of the proof system; the extractor should be successful for any successful prover, *i.e.*, one that convinces the verifier.

Note 1. We note that this intuitive definition omits the probabilistic expressions necessary for a rigorous definition.

At the present level of abstraction proof-of-knowledge is not a fitting property. A slight adaptation of the protocol, whereby the verifier determines the point z and the prover provides the evaluation $e = f(z)$, is trivially a proof of knowledge because the extractor can repeat the protocol N times and then fit a polynomial of degree $N - 1$ to the resulting points — provided that any commitment c binds the prover down to only one polynomial $f(x)$. The concept of knowledge is involved in formalizing this strong binding property. Once $f(x)$ is uniquely determined, the soundness of $\mathsf{PoCEv}\{e, z, c\}$ can be articulated and proved in relation to the language $\{(e, z) \in \mathbb{F}_p \times \mathbb{F}_p \mid f(z) = e\}$.

Rather than considering *black box access* to the prover, we leverage the generic group model and provide the extractor only with read access to the queries and responses made to and provided by the group oracle. Let $Q \triangleq ((q_i, r_i))_i$ be this list of queries and responses, including the queries and matching responses that were made before the commitment c was produced. In order for the commitment scheme to be interchangeable with one that is perfectly binding, we require two things: first, no computationally bounded algorithm can produce a commitment with two conflicting openings; and second, the extractor computes a valid opening for c from (g, c, Q), provided that c is a valid commitment. This second requirement captures the intuition that the list of queries and responses to and from the group oracle uniquely determines the value that the given commitment c commits to. The first requirement stipulates that this value is indeed unique. We call commitment schemes that satisfy both properties *relative binding*. In fact, the group hides the structure of the group and therefore the extractor's running time and memory are immaterial next to the number of queries provided in its input. Moreover, by quantifying only over valid commitment c, we guarantee that for every such commitment there is a matching opening. These observations motivate a functional, rather than algorithmic, characterization of the extractor.

Definition 1 (relative binding). *Let* $(\mathsf{com}, \mathsf{open})$ *be a commitment scheme defined relative to some oracle* O. *Then* $(\mathsf{com}, \mathsf{open})$ *is* relative binding *if both conditions are satisfied:*

1. *For all algorithms* C *making at most a polynomial number (as a function of the security parameter) of queries to the oracle, the probability that* C^{O} *outputs a triple of values* (c, o_0, o_1) *such that* $\mathsf{open}(c, o_0) \neq \bot$ *and* $\mathsf{open}(c, o_1) \neq \bot$ *and* $o_1 \neq o_0$*, is negligible (in the security parameter).*
2. *There is a function* E *that satisfies the following. Let* C *be any algorithm making at most a polynomial number (as a function of the security parameter)*

of queries to the oracle and terminates by outputting a valid commitment $c \leftarrow \mathsf{C}^{\mathsf{O}}()$. *And let* $Q = ((q_i, r_i))_i$ *be the corresponding list of queries and responses. Then* $\mathsf{open}(c, \mathsf{E}(c, Q)) \neq \bot$.

Armed with this notion, it is possible to restate and fix Theorem 2 and its proof. Let C be the portion of the prover-side algorithm that outputs the original commitment c but before engaging in the proof protocol; and let P be the portion that plays the part of the prover in the protocol. The list Q consists of the queries and responses originating from C, and is oblivious of the queries of P (or of V, for that matter). Then we consider the language $\mathcal{L}_{\mathsf{C}} \triangleq \{(e, z) \in \mathbb{F}_p \times \mathbb{F}_p \,|\, \mathsf{open}(c, \mathsf{E}(c, Q)) = f(x) \wedge e = f(z))\}$, and state and prove soundness with respect to this language.

Theorem 3. *Let* $(\mathsf{com}, \mathsf{open})$ *be a relative knowledge binding commitment scheme. Let* (C, P) *be a pair of algorithms that share a state and such that* $c \leftarrow \mathsf{C}$ *and such that* P *is a prover in the proof system* $\mathsf{PoCEv}\{e, z, c\}$. *Then this proof system is statistically sound with respect to the language* $\mathcal{L}_{\mathsf{C}} \triangleq \{(e, z) \in \mathbb{F}_p \times \mathbb{F}_p \,|\, \mathsf{open}(c, \mathsf{E}(c, Q)) = f(x) \wedge e = f(z))\}$.

Proof. The proof is identical to that of Theorem 2, except that the language $\mathcal{L}_c$ is replaced by $\mathcal{L}_{\mathsf{C}} \triangleq \{(e, z) \in \mathbb{F}_p \times \mathbb{F}_p \,|\, \mathsf{open}(c, \mathsf{E}(c, Q)) = f(x) \wedge e = f(z))\}$, and Q is updated at every recursion step to also contain the list of queries and responses made up until that point. □

What remains to be proven then, is that the commitment scheme $(\mathsf{com}, \mathsf{open})$ is relative binding. We show that this holds in the generic group model.

Theorem 4. *Let* $(\mathsf{com}, \mathsf{open})$ *be the commitment scheme with*

- $\mathsf{com} : \mathbb{F}_p[x] \to \mathbb{G}, f(x) = \sum_{i=0}^{N-1} f_i x^i \mapsto g^{\sum_{i=0}^{N-1} f_i q^i}$
- $\mathsf{open} : \mathbb{G} \times \mathbb{N} \to \mathbb{F}_p \cup \{\bot\}, (c, n) \mapsto \begin{cases} f(x) = \sum_{i=0}^{N-1} f_i x^i & \text{if } g^n = c\,; \\ \quad \text{where } f_i = n_i \bmod p & \\ \quad \text{and } \sum_{i=0}^{N-1} n_i q^i = n & \\ \bot & \text{else}\,, \end{cases}$

with the group operations being computed via the group oracle. Then $(\mathsf{com}, \mathsf{open})$ *is relative binding in the generic group model.*

Proof. Suppose the negation of the first item that defines relative binding, namely that there is an algorithm that makes a polynomial number (in the security parameter) of queries to the group oracle and outputs a triple (c, o_0, o_1) such that $\mathsf{open}(c, o_0) \neq \bot$ and $\mathsf{open}(c, o_1) \neq \bot$ and $o_1 \neq o_0$. This implies that $g^{o_0} = g^{o_1}$, meaning that $o_0 - o_1$ is a multiple of the order of g. Consequently, C, together with the extended Euclidean algorithm, can be made into a an algorithm that solves the adaptive root problem. Since this problem is impossible (with overwhelming probability) from only a polynomial number of queries in the generic

group model, this ability provides the necessary contradiction to make the contrapositive argument complete. So the first item that defines relative binding holds in the generic group model.

As for the second item, consider the table of discrete logarithms where every row and every column represents a group element, and the cell at (i, j) either holds an integer $n_{i,j}$ indicating that group element j raised to the power $n_{i,j}$ gives group element i; or the cell is empty, indicating that no such discrete logarithm exists. While the list of queries Q does not fix the entire table, it does fix some elements (modulo the group order), namely at most $\#Q$ rows and at most $\#Q$ columns. Moreover, this partial filling out (with concrete representatives for the equivalence classes modulo the group order) can be computed by starting from a table with one element containing the value 1, and iteratively expanding and updating this table as queries from the list are processed.

This table contains a discrete logarithm base g for c, because otherwise c cannot be a valid commitment. The output of E is the value of the cell whose indices match those of g and c in the list of group elements. Since $\mathsf{E}(c, Q)$ is a discrete logarithm of c base g, we have $\mathsf{open}(c, \mathsf{E}(c, Q)) \neq \bot$. □