On the Formal Verification of Polynomial Commitments: two KZG constructions and the Algebraic Group Model

Tobias Rothmann[0009−0000−2343−5793]

Technical University of Munich, Germany
tobias.rothmann@tum.de

Abstract. We formalize the notion of polynomial commitment schemes (PCSs) in the proof assistant Isabelle/HOL and formally verify the security proofs of two variants of the widely popular Kate, Zaverucha, and Goldberg (KZG) construction. Moreover, we formalize the Algebraic Group Model (AGM) by Fuchsbauer, Kiltz, and Loss using a novel constraint-programming-inspired approach. We formalize a reusable abstract definition of polynomial commitment schemes and define games for correctness, binding, hiding, and knowledge soundness/extractability. Based on this, we verify all applicable security proofs for two concrete PCS constructions: the standard (DL-)KZG and a batched KZG, using our AGM formalization in the knowledge-soundness proofs. Our proofs follow Shoup's sequence-of-games approach, with machine-checked transitions, and are carried out in the CryptHOL framework for formal verification of cryptography in Isabelle. To our knowledge, this work is the first formalization of polynomial commitment schemes, the first formalization of the AGM, and the first formal verification of the security proofs for any concrete polynomial commitment scheme. This work lays the foundation for the formal verification of advanced cryptographic constructions, such as pairing-based zero-knowledge proofs (ZKPs) and succinct arguments.

Keywords: Polynomial Commitments · Formal Verification · Algebraic Group Model · KZG

1 Introduction

In the past years, zero-knowledge proofs (ZKPs) [24] and, particularly, succinct non-interactive arguments of knowledge (SNARKs) [13] have gained widespread adoption in the blockchain industry, securing billions of (US) dollars in value[30, 31]. More recently, this adoption has expanded beyond the blockchain sphere, with Google open-sourcing *longfellow-zk*[25], a library for ZKP-based age verification[21], and the European Digital Identity (EUDI) Project considering ZKPs for the EU digital identity [18]. Formal verification of these cryptographic proof systems has become imperative to ensure their security and reliability.

Until now, primarily early constructions, such as sigma-protocols, have been formally verified [20, 3, 8, 16]. However, the ZKPs and SNARKs used in practice usually do not follow the sigma-protocol paradigm [42]. Typically, these

(ZK) proof systems are built from general-purpose (succinct) arguments, which are compiled to (succinct) non-interactive arguments ((S)NARGs) via the Fiat-Shamir transform [19, 42].

Our work is a major step towards the formal verification of pairing-based succinct arguments, which are still the most verifier-efficient [33]. The dominant paradigm combines an interactive oracle proof (IOP)[11] with a polynomial commitment scheme (PCS)[27]: the IOP+PCS paradigm underlies Sonic[34], Marlin[17], Halo[14], Plonk[23], and Polymath[33]—all using KZG variants with (knowledge) soundness proofs in the Algebraic Group Model (AGM) [22]. PCSs (especially the KZG) and the AGM are cornerstones of pairing-based succinct arguments. We lay the foundation for their formal verification by formalizing the PCS notion, verifying the security proofs of two KZG variants, and formalizing the AGM.

1.1 Contributions

We have three main contributions in this work:

1. the formal verification of the security proofs for two KZG versions (including *knowledge soundness*)
2. the formalization of the AGM and a novel, generally applicable abstraction for AGM-like models
3. an abstract formalization of polynomial commitment schemes with the standard security notions, reusable by future work—in particular, for modularly composing IOPs with PCSs

We formalize an abstract PCS interface with standard security properties (*correctness, binding, hiding, knowledge soundness*) following [27, 42]. We instantiate the "standard" KZG and a batched KZG from [27], developing rigorous Sequence-of-Games (SoG) proofs (based on [28]) for *correctness, polynomial binding, evaluation binding*, and *knowledge soundness* for both, plus *weak evaluation hiding* for the standard KZG. We strengthen the results from [27, 28] by reducing *polynomial binding* to t-DL (previously t-SDH) and proving a stronger *(weak) evaluation hiding* that lets the adversary choose evaluation points (Theorems 1, 3). All proofs are in the standard model except *knowledge soundness*, which uses our novel constraint-programming-inspired AGM formalization—the, to the best of our knowledge, first (general-purpose) AGM formalization.

All proofs and formalizations in Isabelle may be found in the Archive of Formal Proofs (AFP) [1] entry called "Polynomial Commitment Schemes" [39].

1.2 Related Work

There have been multiple formalizations of commitment schemes [16, 3, 35], all formalizing the Pedersen commitment. We are the first to formalize the more complicated PCS notion. Palak and Haines formalize the KZG in EasyCrypt[37], noting their work is subsequent to ours [40]. They do not formalize batched versions, the AGM, or *knowledge soundness* (*extractability*)—the central PCS

property for succinct arguments/ZKPs [42]. We formalize *knowledge soundness* for PCSs and prove it for both KZG versions in the AGM.

The ROM [9, 6, 38, 7] and GGM [6] have been formalized in various proof assistants, typically via oracle embeddings. Our constraint-programming approach for the AGM avoids oracle embeddings, reducing reasoning overhead. Bailey and Miller [4] formalize proofs within the AGM in Lean but provide only a problem-specific structure that abstracts the group away via known exponents. In contrast, we provide a generic AGM formalization that does not abstract the group away or assume knowledge of exponents.

2 Preliminaries

In this work, we will reason primarily about univariate polynomials and points of univariate polynomials. We make the following definitions: for points (x, y) where for a univariate polynomial $p, p(x) = y$, we will call x an *argument* and y an *evaluation*.

Definition 1 (pairings). *Let G and H be two groups of prime order p. A pairing is a function $e : G \times G \to H$, with the following two properties [15]:*
- ***Bilinearity:*** $\forall\, g, h \in G.\ \forall\, a, b \in \mathbb{Z}_p.\ e(g^a, h^b) = e(g, h)^{ab}$
- ***Non-degeneracy:*** $\neg(\forall\, g, h \in G.\ e(g, h) = 1)$

Definition 2 (algebraic algorithm). *Let G be a cyclic group of prime order p. An algebraic algorithm is a (probabilistic) algorithm whose input includes at least one group element, the generator of G. Furthermore, an algebraic algorithm is characterized by outputting a "creation" vector for each output group element, which describes how the respective group element was algebraically "created" from all seen group elements. Formally, for every group output $z \in G$, an algebraic algorithm must output $z = (z_0, \ldots, z_{n-1}) \in \mathbb{Z}_p$, such that $z = \prod_{i=0}^{n} g_i^{z_i}$ [22], where $g = (g_0, \ldots, g_{n-1})$ is the vector of all group elements the algebraic algorithm has already received [22].*

Definition 3. *The Algebraic Group Model (AGM) is a computational model in which all adversaries are algebraic algorithms [22].*

Definition 4. *Games are probabilistic algorithms with access to an adversary [15]. They output a boolean indicating whether the adversary has won or lost the game. Formally, we write games as a sequence of functions and adversary calls.*

Our proofs follow the sequence-of-games framework [41]:

Definition 5. *A sequence-of-games proof bounds the probability of one game to another via game-hops [41]. A **bridging step** alters definitions while preserving game probability; a **failure-event hop** bounds the difference by the probability of a failure event [41].*

Typically, we reduce a security game to a hardness assumption game via game hops, showing that breaking the security property is at least as hard as breaking the assumption [15].

Definition 6. *A Commitment Scheme [15] consists of three functions:* ***KeyGen*** *generates keys ck, vk for committer and verifier;* ***Commit***(ck, m) *computes a commitment c and trapdoor td; and* ***Verify***(vk, c, td, m) *checks whether c is a valid commitment to m.*

2.1 Isabelle/HOL and CryptHOL

Isabelle/HOL[36] is a foundational theorem prover based on Higher-Order Logic; we abbreviate it as *Isabelle*. We chose Isabelle for its foundational approach and its large ecosystem, including the Archive of Formal Proofs (AFP) [1], which we utilize heavily. CryptHOL[9] is an Isabelle framework for game-based cryptographic proofs following the sequence-of-games style (see 5). Games are modeled as monads over sub-probability mass functions (SPMFs) and can be forced to fail via *assert* statements; we refer to [9] for details. For readability, we take some liberties with Isabelle's syntax in the main text.

3 Polynomial Commitment Schemes

We formalize the notion of Polynomial Commitment Schemes, originally proposed in [27]. We deviate slightly from [27] to gain expressivity.

Definition 7. *A Polynomial Commitment Scheme is a Commitment Scheme as defined in 6, with its message space restricted to polynomials (i.e., messages are polynomials). Furthermore, a PCS has two more functions to allow point-wise (i.e. partially) opening a commitment to a polynomial:*

- ***Eval*** *takes the committer key ck, the trapdoor information td to the commitment c, the committed polynomial p, an argument i, and computes the evaluation of p at i, $p(i)$, with a witness w for the point $(i, p(i))$.*
- ***VerifyEval:*** *takes the verifier keys vk, a commitment c, a point $(i, p(i))$, a witness w, and checks that the point is consistent with the commitment using w and vk.*

We call the standard commitment scheme function Verify *in a PCS* VerifyPoly *to avoid ambiguity with* VerifyEval.

Most PCS definitions in the literature fit roughly our definition [27, 43, 32]; those using interactive evaluation protocols [14] can typically be compiled to ours via the Fiat-Shamir [19] or BCS [12] transform.

3.1 Abstract Definition

We formalize our PCS definition as an Isabelle locale [5], an abstract module that fixes the five PCS functions (*KeyGen, Commit, VerifyPoly, Eval, VerifyEval*) with generic type variables, plus three validity checks (*ValidPoly, ValidArgument, ValidEval*). We limit our formalization to the univariate case; multivariate PCSs are left for future work. Following Definition 7, our PCS extends the formalization of Commitment Schemes from [16], from which we automatically inherit games for *polynomial binding*, *polynomial hiding*, and *polynomial correctness* (see Appendix A.2 for the games).

3.2 Correctness & Security notions

We formally define the PCS security notions, games, and adversary advantages in Isabelle. We present the games below and point to the Isabelle formalization for the property definitions and adversary advantages (which will *always* be the probability that the adversary wins the game). We already inherit *polynomial binding*, *polynomial (IND-CPA) hiding*, and *polynomial correctness* from [16] (see Appendix A.2 for the games).

Correctness We formalize a *correctness* game for the evaluation phase, asserting that an honestly generated evaluation verifies for any polynomial p (see Appendix A.1), following [27]. Together with the *polynomial correctness* game from [16], these two notions capture correctness for a PCS.

Evaluation Binding We define the *evaluation binding* game for an adversary $\mathcal{A}$ following [27] in Figure 1 (with the added validity checks).

Hiding We define a hiding game for *Eval*, strengthening the notion in [27] in two ways: (1) the polynomial is an arbitrary parameter rather than uniformly sampled, and (2) the adversary picks the evaluation points. We formalize the game in Figure 1. The game takes a two-phase adversary $\mathcal{A} = (\mathcal{A}1, \mathcal{A}2)$, that passes state in between phases (σ from $\mathcal{A}1$ to $\mathcal{A}2$). In the first phase of the game, $\mathcal{A}1$ chooses the values to be opened (named I). In the second phase, it guesses the polynomial. The adversary wins the game if the polynomial guess is correct.

Knowledge Soundness In addition to the security properties defined in [27], we formalize the *knowledge soundness* property, also called *extractability* [34, 42], following [23] and [43]. This notion is widely used in succinct arguments and ZKPs [23, 34, 43, 17, 14] and thus a valuable property to formalize. Intuitively, *knowledge soundness* ensures that an adversary that can produce a valid evaluation proof for a point must "know" the polynomial that the commitment was computed for. More formally, there exists an extractor algorithm $\mathcal{E}$ that can extract the polynomial p from the commitment c, such that the opening from the adversary is consistent with p. We formalize this notion for an adversary $\mathcal{A}$ in Figure 1. The two-phase adversary $\mathcal{A} = (\mathcal{A}1, \mathcal{A}2)$ first outputs a commitment c; then the extractor $\mathcal{E}$ extracts a polynomial p; finally $\mathcal{A}2$ outputs an evaluation (i, p_i, w). $\mathcal{A}$ wins if the opening verifies but is inconsistent with the extracted p.

$\underline{\mathit{EvalBind}(\mathcal{A}):}$

$$
\left(\begin{array}{l}
(ck, vk) \leftarrow \mathit{KeyGen}, \\
(c, i, v, w, v', w') \leftarrow \mathcal{A}(ck), \\
\text{assert}(v \neq v' \\
\quad \wedge \text{ValidArgument}(i) \\
\quad \wedge \text{ValidEval}(v, w) \\
\quad \wedge \text{ValidEval}(v', w')), \\
b = \text{VerifyEval}(vk, c, i, (v, w)), \\
b' = \text{VerifyEval}(vk, c, i, (v', w')), \\
\text{return}(b \wedge b')
\end{array}\right)
$$

$\underline{\mathit{EvalHid}(p):}$

$$
\left(\begin{array}{l}
(ck, vk) \leftarrow \text{KeyGen}, \\
(c, d) \leftarrow \text{Commit}(ck, p), \\
(I, \sigma) \leftarrow \mathcal{A}1(vk), \\
\quad W = \{\text{Eval}(ck, d, p, i)\}_{i \in I}, \\
\quad p' \leftarrow \mathcal{A}2(vk, \sigma, c, I, W), \\
\quad \text{return}(p = p')
\end{array}\right)
$$

$\underline{\mathit{KSound}(\mathcal{A}, \mathcal{E}):}$

$$
\left(\begin{array}{l}
(ck, vk) \leftarrow \text{KeyGen}, \\
(c, \sigma) \leftarrow \mathcal{A}1(ck), \\
(p, d) \leftarrow \mathcal{E}(c), \\
(i, (p_i, w)) \leftarrow \mathcal{A}2(ck, \sigma), \\
\quad \text{assert}(\text{ValidArgument}(i) \\
\quad\quad \wedge \text{ValidEval}(p_i, w)), \\
(p_i', w') = \text{Eval}(ck, d, p, i), \\
\quad \text{return}(\text{VerifyEval}(vk, c, i, (p_i, w)) \\
\quad\quad \wedge p_i \neq p_i')
\end{array}\right)
$$

Fig. 1. The evaluation binding, evaluation hiding, and knowledge soundness games.

4 Formal Verification of the "standard" KZG

We now formalize the KZG Polynomial Commitment Scheme [27] as a PCS in Isabelle. We focus on the "standard" KZG, described in [27] as the DL version. Note that most proofs for the theorems in [27] are to be found only in [28], the extended version of [27].

4.1 Formalization as a PCS

Fix a group G of prime order p with generator g and a non-degenerate bilinear pairing $e : G \times G \to H$. Let t be the maximum degree the PCS must support. We fix these parameters throughout this section. The KZG is defined as follows.

- **KeyGen**(t)**:** generate a secret "toxic waste" $a \xleftarrow{\$} \mathbb{Z}_p$ and output the structured reference string (srs) $srs = \left(g,\ g^a,\ g^{a^2},\ \ldots,\ g^{a^t}\right) \in G^{t+1}$ as both committer and verifier keys: $(ck, vk) = (srs, srs)$.
- **Commit**$(ck, p(x))$**:** for a polynomial $p(x) = \sum_{j=0}^{\deg(p)} p_j x^j$ with $\deg(p) \leq t$, define the commitment $c = g^{p(a)} = \prod_{j=0}^{\deg(p)} (g^{a^j})^{p_j}$. The trapdoor is the polynomial $p(x)$ itself, so we return "unit" as the trapdoor.
- **VerifyPoly**$(vk, c, p(x))$**:** check the equality $c \stackrel{?}{=} g^{p(a)}$.
- **Eval**$(ck, p(x), i)$**:** compute the evaluation $p(i)$ and the quotient polynomial $q_i(x) = \frac{p(x)-p(i)}{x-i}$, (which has degree at most $t-1$), and return the opening $(p(i), w := g^{q_i(a)})$.
- **VerifyEval**$(vk, c, i, (p(i), w))$**:** accept iff. the following pairing equation holds: $e(c, g) = e(w,\ g^a/g^i) \cdot e(g, g)^{p(i)}$.
- **ValidPoly**(p)**:** asserts $\deg(p) \leq t$
- **ValidArgument**(i)**:** is *True* for all inputs
- **ValidEval**$(p(i), w)$**:** asserts $w \in G$

The above concrete definitions are formalized and instantiated in the abstract PCS in Isabelle in the *KZG_Def.thy* theory. Note that we also formally verify *perfect correctness* as defined in 3.2 for this formalization; the proof can be found in the *KZG_correct.thy* theory.

4.2 Hardness Assumptions

Before we address the security proofs, we introduce the hardness assumption games to which the proofs reduce. We denote uniform sampling from a set by $\xleftarrow{\$}$. We formalize the discrete logarithm (DL) and t-strong Diffie-Hellman (t-SDH) games following [27], and the t-DL game following [22] as in Figure 2.

4.3 Remark on the Formal Verification of "Paper" Proofs

The proofs in [27] are reductions in the "assume an adversary is successful" paradigm, which is not straightforward to express in proof assistants and less rigorous than sequence-of-games arguments [41]. We adapt all proofs from [28] to sequence-of-games proofs. A key tool is the following lemma.

$\underline{DL(\mathcal{A}')}:$

$$\begin{pmatrix} a \xleftarrow{\$} \mathbb{Z}_p, \\ a' \leftarrow \mathcal{A}'(g^a), \\ \text{return}(a' = a) \end{pmatrix}$$

$\underline{t\text{-}DL(\mathcal{A})}:$

$$\begin{pmatrix} a \xleftarrow{\$} \mathbb{Z}_p, \\ a' \leftarrow \mathcal{A}(g, g^a, \ldots, g^{a^t}), \\ \text{return}(a' = a) \end{pmatrix}$$

$\underline{t\text{-}SDH(\mathcal{A})}:$

$$\begin{pmatrix} a \xleftarrow{\$} \mathbb{Z}_p, \\ (c, h) \leftarrow \mathcal{A}(g, g^a, \ldots, g^{a^t}), \\ \text{return}(g^{\frac{1}{a+c}} = h) \end{pmatrix}$$

Fig. 2. The DL, t-DL, and t-SDH hardness assumption games.

Lemma 1 *(Implication Lemma)*
Given a game $\mathcal{G}$ that asserts A *(noted $\mathcal{G}^A$), if* A *implies* B*, then:*

$$(A \Longrightarrow B) \Longrightarrow Pr[\mathcal{G}^A = 1] \leq Pr[\mathcal{G}^B = 1]$$

We prove this in *CryptHOL_ext.thy* and use it to relate assertions of security and hardness games: we assemble facts from the security game into an assertion A, apply the Implication Lemma to obtain an assertion B with the hardness-game-breaking fact/value, then restructure the game into the hardness game played by a reduction adversary.

4.4 Polynomial Binding

We reduce *polynomial binding* to t-DL, which is a supposedly stronger result than the reduction to t-SDH in [27]. Nevertheless, we are able to reuse the methods from [28] (the extended version of [27]) to achieve this goal.

Theorem 1 *(KZG Polynomial Binding)*
For the KZG and any adversary $\mathcal{A}$, the following holds:

$$Pr[PolyBind_{KZG}(\mathcal{A})] \leq Pr[t\text{-}DL(redAdv)]$$

where redAdv *is a reduction adversary that uses $\mathcal{A}$ to break the t-DL assumption.*

Note that the t-DL instance is exactly the KZG's srs $(g, g^a, \ldots, g^{a^t})$ and thus can be trivially embedded as such. Following [28], we give the following reduction map: Given two distinct polynomials p and p' that break *polynomial binding*, i.e., $g^{p(a)} = g^{p'(a)} \Leftrightarrow p(a) = p'(a)$, one can recover the t-DL secret a by polynomial factorization as a root of the polynomial $p'' := p - p'$. In contrast to the proof in [28], we realize that the extraction of a is possible on the field value level and can thus go directly to t-DL instead of t-SDH.

Our sequence-of-games proof collapses the assertions (incl. winning conditions) in the *polynomial binding* game to obtain $p(a) = p'(a)$, applies the Implication Lemma to extract a as a root of $p - p'$, and reorders the game into a t-DL game with reduction adversary. We refer to the *KZG_poly_bind.thy* theory for the complete proof. Factoring polynomials is not straightforward to formalize and posed the biggest challenge in the formalization of this security property. We elaborate on this in section 8.4.

4.5 Evaluation Binding

The *evaluation binding* property reduces to t-SDH, following [28]:
Interpreting the *srs* as a t-SDH instance (with secret α), given two distinct evaluations $p(i), p(i)'$ with proofs w, w' that pass *VerifyEval*, unfolding yields:

$$e(w, g^{a-i})e(g,g)^{p(i)} = e(C,g) = e(w', g^{a-i})e(g,g)^{p(i)'}$$

which implies:

$$q_i(a)\cdot(a-i) + p(i) = q_i(a)'\cdot(a-i) + p(i)' \Leftrightarrow \frac{q_i(a) - q_i(a)'}{p(i)' - p(i)} = \frac{1}{a-i}$$

where $q_i(a) = log_g\ w$ and $q_i(a)' = log_g\ w'$. Hence:

$$\left(\frac{w}{w'}\right)^{\frac{1}{p(i)'-p(i)}} = \left(\frac{g^{q_i(a)}}{g^{q_i(a)'}}\right)^{\frac{1}{p(i)'-p(i)}} = g^{\frac{q_i(a)-q_i(a)'}{p(i)'-p(i)}} = g^{\frac{1}{a-i}}$$

Note, $g^{\frac{1}{a-i}}$ breaks the t-SDH assumption for $-i$. Thus, the map is a simple rearranging of the evaluation values into the term $\left(\frac{w}{w'}\right)^{\frac{1}{p(i)'-p(i)}}$ [28].

Theorem 2 *(KZG Evaluation Binding)*
For the KZG and any adversary $\mathcal{A}$, the following holds:

$$Pr[EvalBind_{KZG}(\mathcal{A})] \leq Pr[t\text{-}SDH(redAdv)]$$

where redAdv *is a reduction adversary that uses $\mathcal{A}$ to break the t-SDH assumption.*

The sequence-of-games proof exposes the *srs* as the t-SDH instance, collapses the winning condition and asserts in the *evaluation binding* game, implies via the Implication Lemma that the reduction value $\left(\frac{w}{w'}\right)^{\frac{1}{p(i)'-p(i)}}$ breaks t-SDH for $-i$, and encapsulates everything into a reduction adversary. See *KZG_eval_bind.thy* for the complete proof.

4.6 Weak Evaluation Hiding

We show that the KZG fulfills the *evaluation hiding* property (3.2) for uniformly sampled polynomials, which we call *weak evaluation hiding*. Notably, we allow the adversary to choose the evaluation points, thereby strengthening the property compared to [28]. This change is motivated by a flaw in the original proof; we discuss this in detail in Section 8.1. Formally, we define the *weak evaluation hiding* game for an adversary $\mathcal{A}$ as follows:

$$WeakEvalHid(\mathcal{A}) := \begin{pmatrix} p \xleftarrow{\$} \{p \in \mathbb{Z}_p[X] \mid \deg(p) \leq t\}, \\ b \leftarrow EvalHid(p, \mathcal{A}) \\ \text{return}(b) \end{pmatrix}$$

The key proof idea (following [28]) is to express a uniform polynomial as uniform points via Lagrange interpolation and interpret one evaluation as the DL secret b. Given all points' data except b, but the DL-instance g^b, one can emulate a commitment and evaluations to run the hiding adversary and recover b. We need two auxiliary lemmas:

Lemma 2 *(Interpolating Uniform Polynomials)*
For any $t \in \mathbb{N}, I \in \mathbb{Z}_p^{t+1}$ with distinct elements, the Lagrange interpolation of points (I_i, J_i) with $i \in [0;\ t]$ and $J \xleftarrow{\$} \mathbb{Z}_p^{t+1}$ is equivalent to a uniform random polynomial $p \in \mathbb{Z}_p[X]$ with $\deg(p) \leq t$.

Lemma 3 *(Interpolating on Groups)*
There exists a concrete function interpolation'*, such that for any list of points $(x, y)_i \in \mathbb{Z}_p^2$ with $i \in [0;\ t]$, the Lagrange interpolation of $(x, y)_i$ named $p \in \mathbb{Z}_p[X]^{\leq t}$, any $a \in \mathbb{Z}_p$, and group generator $g \in G$ of group G with prime order p, the following holds: interpolation'$((x, g^y)_i, a) = g^{p(a)}$.*

We formally prove both lemmas in Isabelle, formalizing a concrete version of *interpolation'* for which we prove Lemma 3 (see *CryptHOL_ext.thy* for Lemma 2 and *KZG_hiding.thy* for Lemma 3).

Theorem 3 *(Weak Evaluation Hiding)*
For the KZG and any adversary $\mathcal{A}$, the following holds:

$$Pr[\mathit{WeakHid}_{KZG}(\mathcal{A})] \leq Pr[\mathit{DL}(\mathit{redAdv})] + Pr[\mathit{t\text{-}DL}(\mathit{redAdv'})]$$

where redAdv *and* redAdv' *are reduction adversaries that use $\mathcal{A}$ to break the respective hardness assumption.*

We start the SoG proof from the weak evaluation hiding game (see Fig. 3): we restate the uniform polynomial as an interpolation of the adversary's arguments I (and some $i \notin I$) and uniform random evaluations via Lemma 2. We then expose the *KeyGen* secret α and rewrite *Commit* using only group-evaluation points (x, g^y) via Lemma 3. Next, we extract one uniform evaluation value and interpret it as a DL secret b. We then rewrite the evaluation proofs: $w = g^{q(\alpha)}$ for argument $j \in I$ can be computed as $(c/g^{p(j)})^{\frac{1}{\alpha - j}}$ (following [28]), which avoids the full polynomial but fails when $\alpha = j$. Via the fundamental lemma of SPMFs, this failure-event hop yields $Pr[\mathcal{G}_1 = 1] \leq Pr[\mathcal{G}_2 = 1] + Pr[\text{failure}]$, where the failure event—the adversary choosing $\alpha \in I$—reduces to t-DL by interpreting the *srs* as a t-DL instance and searching I for x with $g^x = g^\alpha$. At this point, all values for the adversary are independent of b. A successful adversary ($p = p'$) yields b by evaluating the guessed polynomial at i ($p'(i) = b$), reducing to DL. We encapsulate the game into a DL reduction adversary. See *KZG_hiding.thy* for the complete proof.

5 Formal Verification of the Batched KZG

We formalize the batched KZG from [27], fixing the same parameters as in Section 4.1. The batched KZG extends the standard KZG to evaluate and verify batches of points, reusing *KeyGen, Commit, VerifyPoly*, and *ValidPoly*, and redefining:

- **Eval**$(ck, p(x), I)$**:** let $Z_I(x) = \prod_{i \in I}(x - i)$, compute $r(x) = p(x) \bmod Z_I(x)$ and $q_I(x) = \frac{p(x) - r(x)}{Z_I(x)}$, return $(r, w_I := g^{q_I(a)})$. Note that $\forall\, i \in I : r(i) = p(i)$

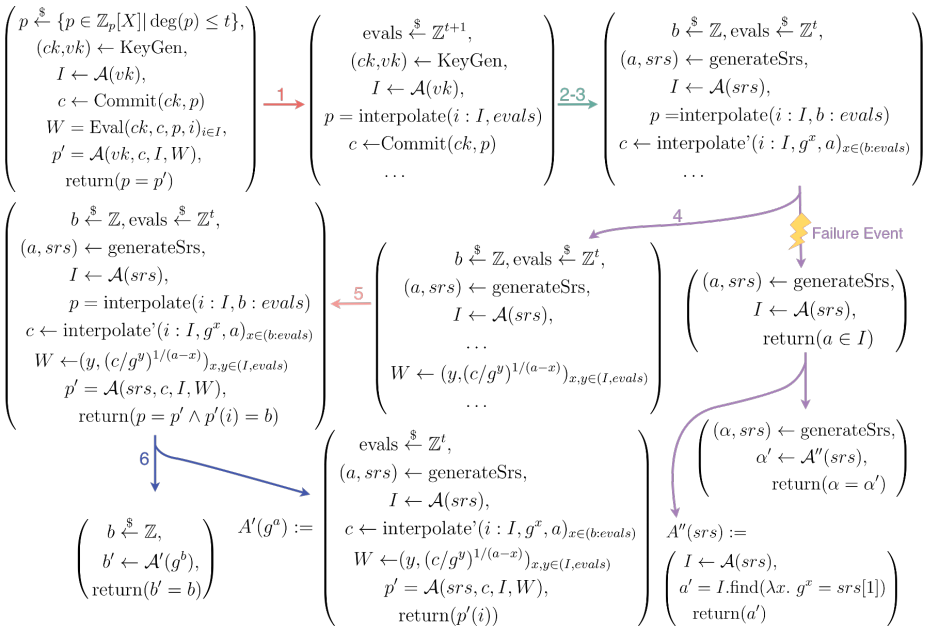

Fig. 3. Game-hops for theorem 3

- **VerifyEval**$(vk, c, I, (r, w_I))$**:** accept iff. the following pairing equation holds: $e(C, g) = e(g^{\prod_{i \in I}(a-i)}, w_I)e(g, g^{r(a)})$ Note, this expression comes from the equality: $p(x) = q_I(x) \cdot (\prod_{i \in I}(x - i)) + r(x)$.
- **ValidArgument**(I)**:** asserts the size of I is at most t.
- **ValidEval**(r, w_I)**:** asserts $w_I \in G$

These definitions are formalized in the theory *BatchKZG_def.thy*. We formally verify *perfect correctness* (3.2) in *BatchKZG_correct.thy* and reuse the standard KZG *polynomial binding* proof since the relevant functions are shared.

5.1 Hardness Assumptions

We need an additional hardness assumption for the batched KZG, a bilinear version of t-SDH. We define the t-BSDH game following [27] just like the t-SDH game, only the check in the return statement is replaced by $e(g, g)^{1/(a+c)} = h$.

5.2 Evaluation Binding

Following [27], we slightly adapt the notion of *evaluation binding* for the batched version. Let KZG denote the standard KZG and bKZG the batched KZG in this section. We show that the adversary cannot find a commitment c, argument i, standard KZG opening v, and a batched opening r for a set I (with respective opening proofs w_i and w_I) such that $i \in I$, $r(i) \neq v$, and both KZG.VerifyEval(srs, c, i, v, w_i) and bKZG.VerifyEval(srs, c, I, r, w_I) hold. We formalize this property in the following adapted *evaluation binding* game:

$$\underline{bEvalBind(\mathcal{A}):} \left(\begin{array}{l} (ck, vk) \leftarrow KeyGen, \\ (c, i, v, w_i, I, r, w_I) \leftarrow \mathcal{A}(ck), \\ \text{assert}(v \neq r(i) \wedge \text{KZG.ValidArgument}(i) \wedge \ \text{bKZG.ValidArgument}(I) \\ \qquad \wedge \text{KZG.ValidEval}(v, w_i) \wedge \ \text{bKZG.ValidEval}(r, w_I)), \\ \text{return}(\text{KZG.VerifyEval}(vk, c, i, (v, w_i)) \wedge \text{bKZG.VerifyEval}(vk, c, I, (r, w_I))) \end{array}\right)$$

Theorem 4 *(bKZG Evaluation Binding)*
For the batched KZG and any adversary $\mathcal{A}$, the following holds:

$$Pr[bEvalBind_{bKZG}(\mathcal{A})] \leq Pr[t\text{-}bSDH(redAdv)]$$

where redAdv *is a reduction adversary that uses $\mathcal{A}$ to break the t-BSDH hardness assumption.*

Following [28], the reduction interprets the *srs* as a t-BSDH instance. Given values that break *bEvalBind*, the definitions of *VerifyEval* yield

$$e(w_i, g^{a-i})\, e(g, g)^v = e(c, g) = e(g^{\prod_{j \in I}(a-j)}, w_I)\, e(g, g^{r(a)})$$

Setting $q_i(a) := \log_g w_i$, $q_I(a) := \log_g w_I$, $r'(x) := \frac{r(x)-r(i)}{x-i}$, and $s'(x) := \frac{\prod_{j \in I}(x-j)}{x-i}$, this implies

$$\begin{aligned} & q_i(a) \cdot (a-i) + v = \textstyle\prod_{j \in I}(a-j) \cdot q_I(a) + r(a) \\ \Leftrightarrow \quad & \frac{1}{a-i} = \frac{s'(a) \cdot q_I(a) - q_i(a) + r'(a)}{p(i) - r(i)} \end{aligned}$$

Hence, the following reduction value breaks t-BSDH for $-i$:

$$\left(e\left(g^{s'(a)}, w_I\right) \cdot e\left(\frac{g^{r'(a)}}{w_i}, g\right)\right)^{\frac{1}{p(i)-r(i)}} = e(g, g)^{\frac{1}{a-i}}$$

The SoG proof is analogous to Section 4.5; see *BatchKZG_eval_bind.thy*.

6 Algebraic Group Model

We formalize a generally applicable version of the AGM by Fuchsbauer, Kiltz, and Loss [22] in Isabelle and use it to prove *knowledge soundness* for both KZG versions.

6.1 Intuition

Computational models in proof assistants have traditionally been formalized via oracles [9, 3, 6, 7, 26]. However, AGM proofs in the literature [22, 10, 34, 23] often require algebraic adversaries to accompany each output group element with a *creation* vector instead (see Def. 2). We adopt this style via a novel constraint-programming-inspired approach: we guard every adversary call with an *assertion* that encodes the algebraic-algorithm rules. This slightly weakens the AGM (only *successful* adversaries must be algebraic), but since proofs typically assume adversary success, this has no impact.

6.2 Formalization

Our formalization semi-automatically produces the AGM assertions for a given adversary. The core idea is to decompose the generation of an assertion into three cleanly separated concerns:

1. **Select** — collect the group elements from the adversary's *input* (the *seen* values).
2. **Constrain** — formulate the AGM rules for the adversary's *output*: every output group element must equal the multi-exponentiation of its *creation* vector with the seen values.
3. **Restrict** — combine *select* and *constrain* into an assert that filters the adversary's output distribution, enforcing the AGM rules.

Formally, these correspond to three functions:

$$\begin{aligned} &\mathit{select} :: \alpha \Rightarrow \beta\ \mathit{list} \\ &\mathit{constrain} :: \beta\ \mathit{list} \Rightarrow \gamma \Rightarrow \mathit{bool} \\ &\mathit{restrict} :: (\alpha \Rightarrow \gamma\ \mathit{spmf}) \Rightarrow \alpha \Rightarrow \gamma\ \mathit{spmf} \end{aligned}$$

Given an adversary of type $\alpha \Rightarrow \gamma$, *restrict* yields a *restricted* adversary of the same type that additionally must pass the AGM assert. The separation of *select* (input-dependent) from *constrain* (output-dependent) is the key to composability: a generic *select* implementation for a given input can be paired with any *constrain* implementation for a given output. Meanwhile, *restrict* separates constraint *enforcement* from constraint *generation*.

Records as Abstractions. Since both *select* and *constrain* are parametric in two type variables with one of them (the group type β) fixed, a Haskell-like multiparameter type class would be the natural representation. Isabelle, however, only supports single-parameter type classes, so we represent them as Isabelle records [5]—extensible tuple types indexed by field names:

```
record (α, β) Select =
    select :: α ⇒ β list (⌑)

record (β, γ) Constrain =
    constrain :: β list ⇒ γ ⇒ bool (⊳)
```

Each record exposes a single field holding a function (⌑ and ⊳ as succinct notation for *select* and *constrain*), providing a uniform name for potentially many different implementations.

Primitive Instances. We provide two base instances for each record. The trivial instances *noSelect* and *noConstrain* return the empty list and *True*, respectively, imposing no constraints. The group-element instances *groupS* and *groupC* handle individual group elements: *groupS* maps a group element g to the singleton list $[g]$, while *groupC* generates the core algebraic-algorithm constraint for a group element g, given the seen elements s and a creation vector c: $\dim(s) = \dim(c) \wedge g = \prod_{i=0}^{dim(s)} {s_i}^{c_i}$

Select and Constrain as Combinators. We implement *Select* and *Constrain* for compound types as **combinators**—higher-order functions that lift a base instance over a data structure. Exploiting Isabelle's extensible records, we provide combinators for lists, products, trees, multisets, finite sets, and finite maps. Since combinators are closed under composition, they can be nested to handle complex types, e.g., (α *multiset list* $\times$ ($\beta \times \gamma$) *tree fset*).

The AlgebraicAlgorithm Locale. Using these record abstractions, we define an Isabelle locale for algebraic algorithms with a generic *restrict* definition:

```
locale AlgebraicAlgorithm = G : cyclic_group G +
  fixes sel :: (α, β) Select
  and con :: (β, γ) Constrain

definition restrict :: (α ⇒ γ spmf) ⇒ α ⇒ γ spmf
  where restrict 𝒜 arg ≡ do {
        res ← 𝒜 arg;
        _ ← assert_spmf
            ((⌑sel arg) ⊳con res);
        return_spmf res}
```

Operationally, *restrict* (1) runs the adversary to obtain its result, (2) applies *select* to the adversary's input to collect all seen group elements, (3) feeds those elements together with the result into *constrain* to check the AGM rules, and (4) embeds the check into an assert—the result is returned only if the assert passes. To obtain an adversary-specific *restrict*, one composes the appropriate *Select* and *Constrain* (combinator) instances for the adversary's input and output types and uses them to instantiate the locale. This yields a semi-automatically generated AGM guard that can be plugged directly into a game to enforce algebraic behavior.

Expressivity: Multi-Round Adversaries. Our AGM formalization is stateless, fitting into the pure functional language of Isabelle. Cryptographic proofs, however, sometimes require stateful multi-round adversaries (most often 2-round). In CryptHOL, such adversaries are modeled as separate pure functions that pass state between rounds; analogously, in the AGM, every adversary must have access to *all* seen group values at every round. This can be achieved by manually extending the adversary's input for round $n+1$ with the group values from round n. While embedding the AGM in an oracle could avoid this manual effort, our approach avoids reasoning about oracle state, trading statefulness for reduced proof complexity.

Compatibility with Oracles and other Computational Models. Since *restrict* operates within CryptHOL's standard SPMF monad, it lifts seamlessly to the formalized random oracle model [9] and oracles in general, enabling proofs in the combined AGM+ROM model [22]. Furthermore, our record structure and combinators readily generalize to other algebraic models—such as the Algebraic Isogeny Model (AIM) or the Algebraic Group Action Model (AGAM) [2]—by

simply swapping the primitive group instance for the appropriate primitive instance.

7 Knowledge Soundness for the KZGs

We use our AGM formalization to prove *knowledge soundness* for both KZG versions ("standard" KZG and batched KZG).

7.1 Knowledge Soundness for the "Standard" KZG

We lift the *knowledge soundness* game from section 3.2 into the AGM, extending the adversary to return a *creation* vector for each group element in its output, guarding every adversary call with *restrict*, and giving the extractor access to the *creation* vector; we mark the differences in yellow:

$$\underline{KSoundAGM(\mathcal{A},\mathcal{E}):} \left(\begin{array}{c} (ck, vk) \leftarrow \mathrm{KeyGen}, \\ (\boxed{\text{(c,cvec)}}, \sigma) \leftarrow \boxed{\text{restrict}}(\mathcal{A}1, ck), \\ (p, td) \leftarrow \mathcal{E}(c, \boxed{\text{cvec}}), \\ (i, p_i, \boxed{\text{(w, wvec)}}) \leftarrow \boxed{\text{restrict}}(\mathcal{A}2, (ck, \sigma)), \\ \mathrm{assert}(\ldots), \\ (p_i', w') = \mathrm{Eval}(ck, td, p, i), \\ \mathrm{return}(\mathrm{VerifyEval}(vk, c, i, (p_i, w)) \wedge p_i \neq p_i') \end{array}\right)$$

We semi-automatically generate *restrict* instances via our combinators. For example, $\mathcal{A}1$ has type $\alpha\ list \Rightarrow \alpha \times \sigma$ in the standard model, extended to $\alpha\ list \Rightarrow (\alpha \times int\ list) \times \sigma$ in the AGM (adding a creation vector). We compose *Select* from the input as "`listS Gp.groupS`" and *Constrain* from the output as "`prodC Gp.groupC noConstrain`". Concretely interpreting *AlgebraicAlgorithm* to obtain *restrict* for $\mathcal{A}1$ as so in Isabelle:

```
interpretation AGM1: AlgebraicAlgorithm Gp
  ''listS Gp.groupS'' ''prodC Gp.groupC noConstrain''
  by (unfold_locales)
```

Unlike typical literature proofs that reduce directly to t-DL [23, 34, 14], we give a simpler reduction from *knowledge soundness* to *evaluation binding*, basing our proof on t-SDH. In the AGM, t-DL and t-SDH are equivalent [10]. We interpret the t-SDH instance as the *srs* and use the extractor $\mathcal{E}$ that reads the *creation* vector *cvec* of c as coefficients: $p(x) := \sum_{i=0}^{t} cvec_i \cdot x$. Unfolding *restrict* gives:

$$c = \prod_{i=0}^{t} g^{(a^t)^{cvec_i}} = g^{\sum_{i=0}^{t} cvec_i \cdot a^t} = g^{p(a)} = \mathrm{Commit}(srs, p)$$

Hence, the extractor produces a polynomial consistent with c. By *perfect correctness*, every evaluation of p via *Eval* passes *VerifyEval*, so the *knowledge soundness* winning condition implies *evaluation binding*:

$$\begin{aligned} &\mathrm{VerifyEval}(vk, c, i, p_i', w_i') \wedge p_i \neq p_i' \\ &\Longrightarrow \mathrm{VerifyEval}(vk, c, i, p_i', w_i') \wedge \mathrm{VerifyEval}(vk, c, i, p_i, w_i) \end{aligned}$$

with $(p_i, w_i) := \text{Eval}(ck, td, p, i)$

Theorem 5 *(KZG Knowledge Soundness (AGM))*
For the KZG and any adversary $\mathcal{A} = (\mathcal{A}1, \mathcal{A}2)$, the following holds in the AGM:

$$Pr[KnowledgeSoundness_{KZG}(\mathcal{A}, \mathcal{E})] \leq Pr[t\text{-}SDH(redAdv)]$$

where $\mathcal{E}$ is our extractor and redAdv *is a reduction adversary that uses $\mathcal{A}$ and $\mathcal{E}$ to break t-SDH.*

The SoG proof starts from the *knowledge soundness* game, unfolds *restrict* for $\mathcal{A}1$, collapses winning condition and AGM asserts, applies the Implication (Lemma), reorders the result into the *evaluation binding* game (3.2) with a reduction adversary, and composes with Theorem 2 to reach the t-SDH game. See *KZG_knowledge_soundness.thy* for the complete proof.

The same overall proof strategy extends to the batched KZG, differing mostly in the implication step: we adapt the *knowledge soundness* game to the batched setting (asking the adversary to provide a point $i \in I$ where the evaluation polynomial disagrees with the extracted polynomial), reduce to *evaluation binding* via the extractor, and apply Theorem 4 to obtain a reduction to t-BSDH. The game-hops follow the same pattern; see the *BatchKZG_knowledge_sound.thy* theory for the complete proof.

8 Difficulties and Insights

8.1 Imprecise Definition and Failing Proof

The proofs in [28] are reduction sketches, not detailed SoG proofs. The hiding proof was particularly difficult to adapt: no game is defined, and the reduction computes evaluation proofs via $(c/g^{p(i)})^{\frac{1}{a-i}}$, which silently requires $a \neq i$ for all arguments $i \in I$ (a being the toxic waste from the srs). The reduction in [28] (unexplicitly) remedies this by assuming that I is uniformly random, under which $a \in I$ is negligible. However, as we understand the hiding definition in [28], this is not necessarily the case. Hence, there is no distributional guarantee that $a \notin I$, making the reduction from [28] inapplicable.

This difficulty led us to an insight: treating $a \in I$ as an *adversary achievement* rather than an unaccounted assumption both fixes and *strengthens* the proof. We handle the case $a \in I$ explicitly as a failure-event game-hop, splitting the proof into two reductions: $a \in I$ reduces to t-DL (the adversary has effectively found the toxic waste among its chosen arguments), while $a \notin I$ reduces to DL.

8.2 t-SDH is not required for Polynomial Binding

We can slightly advance on an answer for an open question of the KZG paper [27, 28]: Is it possible to construct efficient polynomial commitment schemes under weaker assumptions? We observe that the methods used to prove polynomial binding for the "standard" KZG actually suffice to extract the full DL exponent α in plain. Thus, we are able to prove polynomial binding under t-DL, a supposedly weaker assumption than the previously known t-SDH.

8.3 When Knowledge Soundness reduces to Evaluation Binding

This may be known to experts in the field; however, we could not find a reference that stated or proved that (the stronger notion of) knowledge soundness reduces to evaluation binding if you have an extractor that produces values accepted by *VerifyEval*. We give a proof of this in the appendix A.3. Concretely, this insight allowed us to drastically simplify the knowledge soundness proofs for our two KZG schemes by reusing the evaluation binding proofs we already formalized (as opposed to doing a new reduction proof to another hardness assumption).

8.4 Polynomial Factorization in Isabelle

The core of the reduction to t-DL in polynomial binding is polynomial factorization. However, the AFP contained no entry formalizing polynomial factorization beyond the square-free case. To generalize to arbitrary polynomials, we decided to formalize the Elimination-of-Repeated-Factors (ERF) algorithm [29], which transforms polynomials into square-free form while preserving the roots. We collaborated with Katharina Heidler (fka. Kreuzer) and Manuel Eberl, who formalized the ERF [29] for this purpose.

8.5 Reasoning about Asserts in CryptHOL

CryptHOL games use asserts for validity checks and winning conditions, but the framework provides little automation for assert-based reasoning. We extended CryptHOL with the Implication Lemma and many relational/equational lemmas for reasoning about assertion interaction, partial or complete removal/addition, and reordering in games; see *CryptHOL_ext.thy*.

9 Conclusion

We formally verified all applicable standard security properties for the "standard" KZG and a batched version. We formalized an abstract polynomial commitment scheme with standard games for *correctness, binding, hiding*, and *knowledge soundness*, and developed rigorous sequence-of-games proofs following [28]. We improved on their results in two cases: reducing *polynomial binding* to t-DL (prior t-SDH) and proving a stronger *weak evaluation hiding* for the standard KZG. Moreover, we developed the first generally applicable formalization of the Algebraic Group Model in a proof assistant, enabling formally verified proofs of *knowledge soundness* for both versions.

This work lays the foundation for the formal analysis of pairing-based succinct arguments following the IOP+PCS paradigm, such as Plonk[23], Sonic[34], or Marlin[17]. A natural next step is to formally verify interactive oracle proofs [12], enabling a complete formal analysis of the aforementioned succinct arguments.

Acknowledgments. Many thanks to Katharina Heidler (born Kreuzer) and Tobias Nipkow for their guidance and support. Thanks to Manuel Eberl, for formalizing the ERF algorithm with Katharina, and to Ian Goldberg for his insightful responses.

References

1. Archive of formal proofs, https://www.isa-afp.org
2. Aardal, M.A., Basso, A., Riepel, D.: The algebraic isogeny model: A general model with applications to SQIsign and key exchanges. Cryptology ePrint Archive, Paper 2026/032 (2026), https://eprint.iacr.org/2026/032
3. Abate, C., Haselwarter, P.G., Rivas, E., Muylder, A.V., Winterhalter, T., Hriţcu, C., Maillard, K., Spitters, B.: Ssprove: A foundational framework for modular cryptographic proofs in coq. In: 2021 IEEE 34th Computer Security Foundations Symposium (CSF). pp. 1–15 (2021). https://doi.org/10.1109/CSF51468.2021.00048
4. Bailey, B., Miller, A.: Formalizing soundness proofs of linear PCP SNARKs. In: Proceedings of the 33rd USENIX Conference on Security Symposium. SEC '24, USENIX Association, USA (2024)
5. Ballarin, C.: Locales and locale expressions in isabelle/isar. In: Berardi, S., Coppo, M., Damiani, F. (eds.) Types for Proofs and Programs. pp. 34–50. Springer Berlin Heidelberg, Berlin, Heidelberg (2004)
6. Barthe, G., Cederquist, J., Tarento, S.: A machine-checked formalization of the generic model and the random oracle model. In: Basin, D., Rusinowitch, M. (eds.) Automated Reasoning. pp. 385–399. Springer Berlin Heidelberg, Berlin, Heidelberg (2004)
7. Barthe, G., Grégoire, B., Heraud, S., Béguelin, S.Z.: Computer-aided security proofs for the working cryptographer. In: Rogaway, P. (ed.) Advances in Cryptology – CRYPTO 2011. pp. 71–90. Springer Berlin Heidelberg, Berlin, Heidelberg (2011)
8. Barthe, G., Hedin, D., Béguelin, S.Z., Grégoire, B., Heraud, S.: A Machine-Checked Formalization of Sigma-Protocols . In: 2010 IEEE 23rd Computer Security Foundations Symposium (CSF 2010). pp. 246–260. IEEE Computer Society, Los Alamitos, CA, USA (Jul 2010). https://doi.org/10.1109/CSF.2010.24, https://doi.ieeecomputersociety.org/10.1109/CSF.2010.24
9. Basin, D., Lochbihler, A., Sefidgar, S.: CryptHOL: Game-based proofs in higher-order logic. Journal of Cryptology **33**, 1–73 (01 2020). https://doi.org/10.1007/s00145-019-09341-z
10. Bauer, B., Fuchsbauer, G., Loss, J.: A classification of computational assumptions in the algebraic group model. In: Advances in Cryptology – CRYPTO 2020: 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17–21, 2020, Proceedings, Part II. p. 121–151. Springer-Verlag, Berlin, Heidelberg (2020). https://doi.org/10.1007/978-3-030-56880-1_5, https://doi.org/10.1007/978-3-030-56880-1_5
11. Ben-Sasson, E., Chiesa, A., Spooner, N.: Interactive oracle proofs. In: Proceedings, Part II, of the 14th International Conference on Theory of Cryptography - Volume 9986. p. 31–60. Springer-Verlag, Berlin, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53644-5_2, https://doi.org/10.1007/978-3-662-53644-5_2
12. Ben-Sasson, E., Chiesa, A., Spooner, N.: Interactive oracle proofs. In: Proceedings, Part II, of the 14th International Conference on Theory of Cryptography - Volume 9986. p. 31–60. Springer-Verlag, Berlin, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53644-5_2, https://doi.org/10.1007/978-3-662-53644-5_2
13. Bitansky, N., Canetti, R., Chiesa, A., Tromer, E.: From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again.

In: Proceedings of the 3rd Innovations in Theoretical Computer Science Conference. p. 326–349. ITCS '12, Association for Computing Machinery, New York, NY, USA (2012). https://doi.org/10.1145/2090236.2090263, https://doi.org/10.1145/2090236.2090263
14. Boneh, D., Drake, J., Fisch, B., Gabizon, A.: Halo infinite: Proof-carrying data from additive polynomial commitments. In: Advances in Cryptology – CRYPTO 2021: 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16–20, 2021, Proceedings, Part I. p. 649–680. Springer-Verlag, Berlin, Heidelberg (2021). https://doi.org/10.1007/978-3-030-84242-0_23, https://doi.org/10.1007/978-3-030-84242-0_23
15. Boneh, D., Shoup, V.: A graduate course in applied cryptography (2023), http://toc.cryptobook.us/book.pdf, http://toc.cryptobook.us/book.pdf
16. Butler, D., Lochbihler, A., Aspinall, D., Gascón, A.: Formalising Σ-protocols and commitment schemes using CryptHOL. J. Autom. Reason. **65**(4), 521–567 (Apr 2021). https://doi.org/10.1007/s10817-020-09581-w, https://doi.org/10.1007/s10817-020-09581-w
17. Chiesa, A., Hu, Y., Maller, M., Mishra, P., Vesely, N., Ward, N.: Marlin: Preprocessing zksnarks with universal and updatable srs. In: Advances in Cryptology – EUROCRYPT 2020: 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10–14, 2020, Proceedings, Part I. p. 738–768. Springer-Verlag, Berlin, Heidelberg (2020). https://doi.org/10.1007/978-3-030-45721-1_26, https://doi.org/10.1007/978-3-030-45721-1_26
18. European Digital Identity Cooperation Group: G - zero knowledge proof, https://eudi.dev/2.4.0/discussion-topics/g-zero-knowledge-proof/, discussion Paper — Topic G: Zero Knowledge Proof
19. Fiat, A., Shamir, A.: How to prove yourself: practical solutions to identification and signature problems. In: Proceedings on Advances in Cryptology—CRYPTO '86. p. 186–194. Springer-Verlag, Berlin, Heidelberg (1987)
20. Firsov, D., Unruh, D.: Zero-knowledge in EasyCrypt. Cryptology ePrint Archive, Paper 2022/926 (2022), https://eprint.iacr.org/2022/926
21. Frigo, M., abhi shelat: Anonymous credentials from ECDSA. Cryptology ePrint Archive, Paper 2024/2010 (2024), https://eprint.iacr.org/2024/2010
22. Fuchsbauer, G., Kiltz, E., Loss, J.: The algebraic group model and its applications. In: Shacham, H., Boldyreva, A. (eds.) Advances in Cryptology – CRYPTO 2018. pp. 33–62. Springer International Publishing, Cham (2018)
23. Gabizon, A., Williamson, Z.J., Ciobotaru, O.: PLONK: Permutations over Lagrange-bases for oecumenical noninteractive arguments of knowledge. Cryptology ePrint Archive, Paper 2019/953 (2019), https://eprint.iacr.org/2019/953, https://eprint.iacr.org/2019/953
24. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof-systems. In: Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing. p. 291–304. STOC '85, Association for Computing Machinery, New York, NY, USA (1985). https://doi.org/10.1145/22145.22178, https://doi.org/10.1145/22145.22178
25. Google: longfellow-zk: Implementation of the Google Zero-Knowledge library for Identity Protocols. https://github.com/google/longfellow-zk (2025), gitHub repository. Latest release v0.8.5 (2025-11-10). Accessed: 2025-12-31.
26. Jaeger, J., Mohan, D.I.: Generic and algebraic computation models: When agm proofs transfer to the ggm. In: Reyzin, L., Stebila, D. (eds.) Advances in Cryptology – CRYPTO 2024. pp. 14–45. Springer Nature Switzerland, Cham (2024)

27. Kate, A., Zaverucha, G.M., Goldberg, I.: Constant-size commitments to polynomials and their applications. In: Abe, M. (ed.) Advances in Cryptology - ASIACRYPT 2010. pp. 177–194. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)
28. Kate, A., Zaverucha, G.M., Goldberg, I.: Polynomial Commitments (2010), https://cacr.uwaterloo.ca/techreports/2010/cacr2010-10.pdf, this is an extended version of the Asiacrypt'10 paper
29. Kreuzer, K., Eberl, M.: Elimination of repeated factors algorithm. Archive of Formal Proofs (November 2023), https://isa-afp.org/entries/Elimination_Of_Repeated_Factors.html, Formal proof development
30. L2BEAT: Total value secured (rollups). https://l2beat.com/scaling/tvs (2025), accessed: 2025-12-31
31. Lavin, R., Liu, X., Mohanty, H., Norman, L., Zaarour, G., Krishnamachari, B.: A survey on the applications of zero-knowledge proofs (2024), https://arxiv.org/abs/2408.00243
32. Lee, J.: Dory: Efficient, transparent arguments for generalised inner products and polynomial commitments. In: Theory of Cryptography: 19th International Conference, TCC 2021, Raleigh, NC, USA, November 8–11, 2021, Proceedings, Part II. p. 1–34. Springer-Verlag, Berlin, Heidelberg (2021). https://doi.org/10.1007/978-3-030-90453-1_1, https://doi.org/10.1007/978-3-030-90453-1_1
33. Lipmaa, H.: Polymath: Groth16 is not the limit. In: Advances in Cryptology – CRYPTO 2024: 44th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18–22, 2024, Proceedings, Part X. p. 170–206. Springer-Verlag, Berlin, Heidelberg (2024). https://doi.org/10.1007/978-3-031-68403-6_6, https://doi.org/10.1007/978-3-031-68403-6_6
34. Maller, M., Bowe, S., Kohlweiss, M., Meiklejohn, S.: Sonic: Zero-knowledge snarks from linear-size universal and updatable structured reference strings. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. p. 2111–2128. CCS '19, Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3319535.3339817, https://doi.org/10.1145/3319535.3339817
35. Metere, R., Dong, C.: Automated cryptographic analysis of the pedersen commitment scheme. In: Rak, J., Bay, J., Kotenko, I., Popyack, L., Skormin, V., Szczypiorski, K. (eds.) Computer Network Security. pp. 275–287. Springer International Publishing, Cham (2017)
36. Nipkow, T., Paulson, L.C., Wenzel, M.: Isabelle/HOL: A Proof Assistant for Higher-Order Logic, vol. 1. Springer Berlin, Heidelberg (2002)
37. Palak, Haines, T.: Formalisation of the KZG polynomial commitment schemes in EasyCrypt. In: Computer Security – ESORICS 2025: 30th European Symposium on Research in Computer Security, Toulouse, France, September 22–24, 2025, Proceedings, Part II. p. 303–320. Springer-Verlag, Berlin, Heidelberg (2025). https://doi.org/10.1007/978-3-032-07891-9_16, https://doi.org/10.1007/978-3-032-07891-9_16
38. Petcher, A., Morrisett, G.: The foundational cryptography framework. In: Focardi, R., Myers, A. (eds.) Principles of Security and Trust. pp. 53–72. Springer Berlin Heidelberg, Berlin, Heidelberg (2015)
39. Rothmann, T.: Polynomial commitment schemes. Archive of Formal Proofs (July 2026), https://isa-afp.org/entries/Polynomial_Commitment_Schemes.html, Formal proof development
40. Rothmann, T., Kreuzer, K.: Formal verification of the kate-zaverucha-goldberg polynomial commitment scheme. 2024 Workshop on Foundations of Com-

puter Security (2024), https://fcs-workshop.github.io/fcs2024/papers/FCS_Rothmann_Kreuzer.pdf
41. Shoup, V.: Sequences of games: A tool for taming complexity in security proofs. IACR Cryptology ePrint Archive **2004**, 332 (01 2004)
42. Thaler, J.: Proofs, Arguments, and Zero-Knowledge. Now Publishers Inc (2022)
43. Xie, T., Zhang, Y., Song, D.: Orion: Zero knowledge proof with linear prover time. In: Advances in Cryptology – CRYPTO 2022: 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15–18, 2022, Proceedings, Part IV. p. 299–328. Springer-Verlag, Berlin, Heidelberg (2022). https://doi.org/10.1007/978-3-031-15985-5_11, https://doi.org/10.1007/978-3-031-15985-5_11

A Appendix

A.1 Eval Correctness Game

$$\mathit{EvalCorrect}(p) := \begin{pmatrix} (ck, vk) \leftarrow \mathrm{KeyGen}, \\ (c, d) \leftarrow \mathrm{Commit}(ck, p), \\ (p_i, w) = \mathrm{Eval}(ck, d, p, i), \\ \mathrm{return}(\mathrm{VerifyEval}(vk, c, i, (p_i, w))) \end{pmatrix}$$

A.2 Commitment Scheme Security Games

We point to [16] for the abstract commitment scheme security games. In the following, we give the concrete instantiation from the abstract PCS, i.e., the commitment scheme games instantiated with polynomials.

$$\underline{\mathit{PolyCorrect}(p):} \quad \begin{pmatrix} (ck, vk) \leftarrow \mathrm{KeyGen}, \\ (c, d) \leftarrow \mathrm{Commit}(ck, p), \\ \mathrm{return}(\mathrm{VerifyPoly}(vk, p, c, d)) \end{pmatrix}$$

$$\underline{\mathit{PolyBind}(\mathcal{A}):} \quad \begin{pmatrix} (ck, vk) \leftarrow \mathrm{KeyGen}, \\ (c, p, d, p', d') \leftarrow \mathcal{A}(ck), \\ _ \leftarrow \mathrm{assert}(p \neq p' \\ \quad \wedge \mathrm{ValidPoly}(p) \\ \quad \wedge \mathrm{ValidPoly}(p')), \\ b = \mathrm{VerifyPoly}(vk, p, c, d), \\ b' = \mathrm{VerifyPoly}(vk, p', c, d'), \\ \mathrm{return}(b \wedge b') \end{pmatrix}$$

$$\underline{\mathit{PolyHid}(p):} \quad \begin{pmatrix} (ck, vk) \leftarrow \mathrm{KeyGen}, \\ (p_0, p_1, \sigma) \leftarrow \mathcal{A}1(ck), \\ _ \leftarrow \mathrm{assert}(\mathrm{ValidPoly}(p_0) \\ \quad \wedge \mathrm{ValidPoly}(p_1)), \\ b \xleftarrow{\$} \{0, 1\}, \\ (c, d) \leftarrow \mathrm{Commit}(ck, p_b), \\ b' \leftarrow \mathcal{A}2(ck, \sigma), \\ \mathrm{return}(b = b') \end{pmatrix}$$

Fig. 4. The polynomial correctness, polynomial binding, and polynomial hiding (IND-CPA) games.

A.3 When Knowledge Soundness reduces to Evaluation Binding

Theorem 6 *For any PCS, adversary* $\mathcal{A} = (\mathcal{A}1, \mathcal{A}2)$*, and extractor* $\mathcal{E}$*, it holds:*

$$\forall\, c, i : \mathit{VerifyEval}(vk, c, i, \mathit{Eval}(ck, \mathcal{E}(c), i)) \\ \implies Pr[\mathit{KSound}(\mathcal{A}, \mathcal{E})] \leq Pr[\mathit{EvalBind}(\mathit{redAdv}(\mathcal{A}, \mathcal{E}))]$$

where (ck, vk) *are outputs of KeyGen and* redAdv *is a reduction adversary that simulates the KSound game using* $\mathcal{A}$ *and* $\mathcal{E}$*, returning* (c, i, p_i, w, p_i', w') *to the EvalBind game.*

Proof sketch. The KSound values satisfy most EvalBind requirements directly. The ValidEval check follows from PCS correctness; the VerifyEval check is exactly the implication condition. □