

On the Fiat–Shamir Security of Succinct Arguments from Functional Commitments

Alessandro Chiesa

alessandro.chiesa@epfl.ch
EPFL

Ziyi Guan

ziyi.guan@epfl.ch
EPFL

Christian Knabenhans

christian.knabenhans@epfl.ch
EPFL

Zihan Yu

z.yu@yale.edu
Yale University

June 2, 2026

Abstract

We study the security of a popular paradigm for constructing SNARGs, closing a key security gap left open by prior work. The paradigm consists of two steps: first, construct a public-coin succinct interactive argument by combining a functional interactive oracle proof (FIOP) and a functional commitment scheme (FC scheme); second, apply the Fiat–Shamir transformation in the random oracle model. Prior work did not consider this generalized setting nor prove the security of this second step (even in special cases).

We prove that the succinct argument obtained in the first step satisfies state-restoration security, thereby ensuring that the second step does in fact yield a succinct non-interactive argument. This holds provided the FIOP satisfies state-restoration security and the FC scheme satisfies a natural state-restoration variant of *function binding* (a generalization of position binding for vector commitment schemes).

Moreover, we prove that notable FC schemes satisfy state-restoration function binding, allowing us to establish, via our main result, the security of several SNARGs of interest (in the random oracle model). This includes a modular security proof of Plonk, in the ROM based on falsifiable Diffie–Hellman assumptions.

Keywords: Fiat–Shamir security; succinct arguments; functional commitment schemes

Contents

1	Introduction	3
1.1	Our results	4
2	Techniques	8
2.1	Limitations of prior analyses	9
2.2	Warm-up: interactive arguments based on functional PCPs	9
2.3	Fiat–Shamir security of interactive arguments based on functional PCPs	13
2.4	Succinct arguments based on public-coin functional IOPs	15
2.5	Function binding in action: On the security of Plonk	18
3	Preliminaries	21
3.1	Interactive arguments	21
3.2	Functional interactive oracle proofs	23
3.3	Functional commitment schemes	25
4	The Funky protocol	34
5	Solving time and tail errors	36
5.1	Inefficient baseline for the general case	36
5.2	Linear queries	37
5.3	Point queries	38
5.4	Univariate polynomial evaluation queries	39
5.5	Multivariate polynomial evaluation queries	40
5.6	Structured polynomial evaluation queries	41
5.7	Bounded-degree functions	42
6	State-restoration security reduction	44
6.1	Construction of the security reducers	45
6.2	Proof of Lemma 6.1	47
7	State-restoration security of the Funky protocol	52
7.1	Construction of the FIOP state-restoration adversary	53
7.2	State-restoration soundness	54
7.3	State-restoration knowledge soundness	56
8	Batching and linearization for homomorphic functional commitment schemes	59
8.1	Proof of Lemma 8.2 (batched-messages FC)	60
8.2	Proof of Lemma 8.3 (linearization trick)	62
9	Application: variants of the KZG polynomial commitment scheme	65
9.1	Proof of Lemma 9.1 (KZG)	66
9.2	Proof of Corollary 9.2 (batch KZG)	68
9.3	Proof of Lemma 9.3 (KZG with linear combinations)	69
9.4	Proof of Corollary 9.4 (linearized KZG)	71
A	Special function binding	72
B	Comparing function binding to other properties for KZG	75
B.1	Function binding implies strong correctness	75
B.2	Interpolation binding implies function binding	76
C	Function binding for polynomial commitment schemes based on DLog	80
C.1	Square-root-sized polynomial commitment scheme	80
C.2	Bulletproofs-style polynomial commitment scheme	83
	Acknowledgments	88
	References	88

1 Introduction

A succinct non-interactive argument (SNARG) for a relation R is a computationally-sound non-interactive proof system where, for a given instance $\mathbf{x}$, an argument prover aims to convince the argument verifier that there exists a witness $\mathbf{w}$ such that $(\mathbf{x}, \mathbf{w}) \in R$, while sending an argument string much shorter than $|\mathbf{w}|$ bits. SNARGs have numerous theoretical and practical applications, and are studied in a large body of work. Due to barriers [GW11], SNARGs are typically achieved in oracle models or from non-falsifiable (e.g., knowledge) assumptions (or both). In this paper we focus on the random oracle model (ROM).

SNARGs in the “pure” ROM. SNARGs exist unconditionally in the ROM [Mic00; Val08; BCS16; CY21b; CY21a; CY24]. They are obtained by combining a PCP/IOP and a ROM-based vector commitment scheme (specifically, a ROM-based Merkle commitment scheme). Security is proved via *straightline* (non-rewinding) extraction. Alas, a weak point of known SNARG constructions in this “pure” ROM setting is argument size, with state-of-the-art constructions achieving argument sizes around several tens of kilobytes. This state of affairs has motivated studying SNARGs in the ROM that *additionally rely on cryptographic assumptions*, to achieve smaller argument size (than what is currently possible unconditionally in the ROM).

Smaller: SNARGs in the ROM plus cryptography. Smaller argument sizes are achieved by applying the Fiat–Shamir transformation in the ROM [FS86] to succinct interactive arguments in the standard model (no oracles) that, in turn, rely on *cryptographic ingredients* for small communication complexity. Various works achieve argument sizes of several kilobytes this way [LM19; CHMMVW20; BFS20; CFFQR21; LPS24].

On Fiat–Shamir security. The Fiat–Shamir transformation in the ROM, which turns an interactive argument into a non-interactive argument in the ROM, requires the interactive argument to satisfy a soundness property *stronger* than standard soundness (and similarly for knowledge soundness). For every k , there exist k -round interactive arguments with soundness ϵ_{ARG} for which the soundness error of the resulting non-interactive argument in the ROM (after the transformation) is $m^{\Omega(k)} \cdot \epsilon_{\text{ARG}}$ against m -query adversaries. This does not suffice for asymptotic security when k is superconstant and, even when k is constant, yields poor concrete security. In sum, for Fiat–Shamir security, standard soundness is *not* the target.

The security target is, instead, *state-restoration soundness* [BCS16; CY24], which implies the security of the Fiat–Shamir transformation in its various flavors [CY24; CO25]. Establishing this stronger security notion for a given interactive argument requires, in general, a significantly more complex analysis. Moreover, “simpler” strong security notions such as round-by-round soundness [CCHLRR18], which suffices for Fiat–Shamir security, apply only to interactive *proofs* (rather than *arguments*).

Funky protocol. The succinct interactive arguments mentioned above all follow a familiar paradigm: they combine a type of probabilistic proof and a corresponding notion of commitment scheme. In this paper we describe and study this paradigm in *full generality*. We describe and study the **Funky protocol** (reminiscent of “function”), which constructs a succinct interactive argument by combining an *FC scheme* (functional commitment scheme) and an *FIOP* (functional interactive oracle proof); we elaborate on both notions later.

Specifically, to understand the security of all non-interactive arguments obtained by applying the Fiat–Shamir transformation in the ROM to the Funky protocol, we ask:

When does the Funky protocol satisfy state-restoration soundness?

Special cases of the Funky protocol are studied in prior work, typically *without consideration* of Fiat–Shamir security. In more detail: (i) [Kil92; BG08; CDGS23; CDGSY24] studies standard soundness of interactive arguments constructed from an IOP and vector commitment scheme; (ii) [LM19] studies standard soundness of interactive arguments constructed from a linear PCP and linear commitment scheme; (iii) [CHMMVW20;

BFS20; CFFQR21; LPS24] studies standard soundness of interactive arguments constructed from a polynomial IOP and polynomial commitment scheme. Some of these works describe SNARGs in the ROM obtained by applying the Fiat–Shamir transformation yet, since they only establish standard soundness of their interactive argument, they leave a *security gap*. One exception is [LPS25], which shows the special soundness of an interactive argument obtained from a *specific* polynomial IOP and a *specific* polynomial commitment scheme, which implies the Fiat–Shamir security of that construction. In contrast, we aim to establish the minimal conditions for state-restoration soundness of the Funky protocol in full generality, covering all the these cases.

What security for the FC scheme? Prior work studies the special case where the FC scheme is a PC scheme (polynomial commitment scheme) [CHMMVW20; BFS20; CFFQR21; LPS24], assuming that the PC scheme is extractable, a strong property stating that there exists a (non-black-box or rewinding) extractor that outputs a polynomial for the commitment output by the adversary. On the one hand, this does not suffice to ensure Fiat–Shamir security (as explained above); on the other hand, extractability is *unnecessary* to prove standard soundness (or knowledge soundness) of the interactive argument. Consider, by analogy, Kilian’s protocol [Kil92], a succinct interactive argument that combines a PCP and a VC scheme. The security of Kilian’s protocol relies on the *position binding* property of the VC scheme [CDGSY24], which ensures that different openings are consistent with one another; roughly, one rewinds the argument adversary sufficiently many times to recover enough consistent fragments of the PCP, without any need to actually recover a PCP that matches the commitment. A similar consideration applies to the IBCS protocol [CDGS23], which extends Kilian’s protocol to interactive arguments obtained from an IOP (and a position-binding VC scheme). Hence we ask: *what is the “right” security property for the FC scheme underlying the Funky protocol?* We seek the natural analogue of position binding for FC schemes, as well as its state-restoration variant that suffices for Fiat–Shamir security of the Funky protocol. As a result, constructions of FC schemes could be proved secure via the “right” amount of effort (without overshooting by, say, proving extractability).

1.1 Our results

We describe the Funky protocol, which achieves a succinct interactive argument in the standard model by combining an FC scheme (functional commitment scheme) and an FIOP (functional IOP). We establish the Fiat–Shamir security of the Funky protocol, closing a key security gap left open by prior work. Along the way, we identify (the state-restoration variant of) *function binding* as the “right” security property for this setting. We show that several FC schemes of interest satisfy this property and, in particular, establish that the influential Plonk argument (with optimizations) [GWC19] is secure in the ROM under falsifiable assumptions.

1.1.1 Fiat–Shamir security of Funky

State-restoration (knowledge) soundness [BCS16; CY24] is an idealized game for a public-coin interactive argument that implies the (knowledge) soundness of the non-interactive argument in the ROM that results from applying the Fiat–Shamir transformation in the ROM. The game considers an adversary against the interactive argument that aims to find an accepting transcript by making a bounded number of moves, with each move receiving corresponding randomness in response.

We denote by $\text{Funky}[\text{FIOP}, \text{FC}]$ the Funky protocol based on: (i) FIOP, a functional IOP with round complexity k_{FIOP} , proof length ℓ , and query complexity q ; (ii) FC, an interactive FC scheme. We upper bound the state-restoration soundness error $\epsilon_{\text{ARG}}^{\text{SR}}$ of $\text{Funky}[\text{FIOP}, \text{FC}]$ in terms of the state-restoration soundness

error $\epsilon_{\text{FIOP}}^{\text{SR}}$ of FIOP and the state-restoration function binding error $\epsilon_{\text{FC}}^{\text{SR}}$ of FC.¹ A similar statement (omitted below) holds for state-restoration knowledge soundness. We elaborate on function binding in Section 1.1.2.

Theorem 1 (informal). *The state-restoration soundness error $\epsilon_{\text{ARG}}^{\text{SR}}$ of Funky[FIOp, FC] against m -move t_{ARG} -size adversaries satisfies the following for every error tolerance $\epsilon > 0$:*

$$\epsilon_{\text{ARG}}^{\text{SR}}(m, t_{\text{ARG}}) \leq \epsilon_{\text{FIOP}}^{\text{SR}}(m + k_{\text{FIOP}}) + k_{\text{FIOP}} \cdot \epsilon_{\text{FC}}^{\text{SR}}(m_{\text{FC}}, t_{\text{FC}}) + k_{\text{FIOP}} \cdot \epsilon,$$

where

$$\begin{cases} m_{\text{FC}} = O\left(\frac{k_{\text{FIOP}} \cdot \ell}{\epsilon} \cdot m\right) \\ t_{\text{FC}} = O\left(\frac{\ell}{\epsilon} \cdot t_{\text{ARG}}\right) \end{cases}.$$

This implies negligible bounds in the negligible case.

The formal version of Theorem 1 is stated in Theorem 7.1. The error tolerance ϵ in the theorem reflects the fact that, in the security analysis, rewinding the adversary more times yields a smaller additive error. This is analogous to the error tolerance ϵ that arises in rewinding security analyses of succinct interactive arguments based on VC schemes [CDGSY24; CDGS23]. Despite the similarity, new technical challenges arise in the more general setting that we consider (see Section 2).

Theorem 1 implies, for special cases, bounds for standard (not state-restoration) security that are at least as good as, and sometimes better than, in prior work.

- *Vector commitment setting.* Theorem 1 implies the same bound on standard (not state-restoration) security as in [CDGS23; CDGSY24] for non-interactive VC schemes.
- *Linear commitment setting.* Theorem 1 implies a better bound compared to [LM19], which studies standard (not state-restoration) security assuming the underlying linear PCP has negligible error.
- *Polynomial commitment setting.* [CHMMVW20; BFS20] proves standard (not state-restoration) security based on extractability in the negligible regime. [LPS24] proves standard (not state-restoration) security for a *less efficient variant* of the protocol (it includes random openings that double the round complexity); moreover, [LPS24] relies on special soundness of the PC scheme, which cannot be achieved for notable PC schemes (see Section 1.1.2) and, also, incurs unnecessary multiplicative overheads in the upper bound.

Applicability of Theorem 1. Many known FIOps are (believed or) proven to have round-by-round (knowledge) soundness, which implies state-restoration (knowledge) soundness [CY24]; also, many FC schemes satisfy special soundness, which implies state-restoration function binding (Lemma 3.31). Hence Theorem 1 applies to many known FIOps and FC schemes. Moreover, [LPS25] shows that a widely used PC scheme is not special sound and, nevertheless, we show that it satisfies state-restoration function binding (see Section 1.1.2), deducing the security of Plonk [GWC19] in the ROM based on ARSDH and SplitRSDH (see Section 1.1.3).

Are probabilistic proofs inherent to succinct arguments? Prior work studies “reverse compilers” for succinct arguments: transformations that map any succinct argument in a certain class into a corresponding (unconditionally secure) probabilistic proof, proving the latter’s necessity. This paper puts on a formal footing intuitions about such compilers, as we now explain.

[RV09] studies succinct interactive arguments that are proved secure via a black-box reduction to a falsifiable assumption, and shows that any such construction implies a corresponding PCP. The parameters of PCPs achieved in [RV09] are rather poor. Intuitively, this is because succinct interactive arguments can be built not only from PCPs (plus a VC scheme) but also from: (i) IOPs (plus a VC scheme), or (ii) PIOPs (plus a PC scheme), or (iii) LIOPs (plus a LC scheme), or (iv) many other options enabled by the Funky protocol.

¹Both ingredients must be individually secure against state-restoration attacks, or else one can attack the Funky protocol.

Our proof of Theorem 1 is a black-box reduction to a falsifiable assumption, hence all these options are valid inputs to the compiler in [RV09]. We deduce that the poor PCP parameters in [RV09] are likely inherent: the succinct interactive argument input to the transformation in [RV09] may have been obtained from a PIOP or LIOP, both of which are much “weaker” notions compared to a PCP.

This stands in contrast to [CY20], which studies succinct (interactive and non-interactive) arguments with unconditional security in the ROM, and shows that any such construction implies a corresponding IOP with good parameters. This is consistent with our understanding of the “pure ROM” where we only know of the approach that combines an IOP and the Merkle commitment scheme (a VC scheme). So in this setting the probabilistic proofs input to a “forward compiler” and output by a “reverse compiler” essentially match.

Instantiating random oracles. In practice, random oracles are replaced by efficient hash functions. These heuristic instantiations do not always work in the sense that there are interactive protocols for which the Fiat–Shamir transformation is not secure for any efficient hash function [GK03; CGH04; BBHMR19; KRS25], including some examples of succinct interactive arguments. Understanding precisely for which interactive protocols the Fiat–Shamir transformation works in the standard model remains an open problem; moreover, the Fiat–Shamir transformation can be modified to circumvent some of the attacks [AY25].

1.1.2 Function binding

The security analysis of Kilian’s protocol [CDGSY24] and the IBCS protocol [CDGS23] relies on the position binding property of the underlying VC scheme; this property states that no efficient adversary can produce two inconsistent valid openings for the same commitment. We consider *function binding*, which is the analogous property for FC schemes that (we show) suffices for the standard security of the Funky protocol. This definition is inspired by the function binding property for linear commitment schemes in [LM19].

Definition 1 (informal). *FC has function binding error $\epsilon_{\text{FC}} = \epsilon_{\text{FC}}(t_{\text{FC}})$ if for every t_{FC} -size adversary that outputs a commitment cm and a list of openings $((\alpha_i, \beta_i, \text{pf}_i))_i$ valid for cm , with probability at most ϵ_{FC} , there does not exist Π such that $\alpha_i(\Pi) = \beta_i$ for every i .*

When the FC scheme has an interactive opening phase (this is common), we consider *state-restoration function binding*, which is the idealized state-restoration game for function binding. We defer the definition of state-restoration function binding to Section 3.3.

Targeting function binding (rather than special soundness or extractability) for FC schemes facilitates basing the security of FC schemes on falsifiable assumptions. (E.g., it avoids knowledge assumptions that are often used to prove extractability instead [CHMMVW20; BMMTV21].) We provide a notable example.

We study a widely used FC scheme that is obtained from batching the PC scheme in [KZG10] and applying the linearization trick in [GWC19], which we call *linKZG*. This FC scheme supports a rich query class $\mathbf{Q}_{\text{Struct}}[\mathbf{Q}, m, (h_k)_{k \in [n]}]$ that captures certain sums of products of low-degree polynomials (see Definition 5.9), and underlies concretely efficient SNARKs (it is used in Plonk [GWC19]).

We prove that *linKZG* satisfies state-restoration function binding based on ARSDH and SplitRSDH (falsifiable assumptions introduced by [LPS24; LPS25]).

Lemma 1 (informal). *Assume that the adaptive rational strong Diffie–Hellman (ARSDH) assumption holds with error $\epsilon_{\text{ARSDH}} = \epsilon_{\text{ARSDH}}(\lambda, t_{\text{ARSDH}})$ and the split rational strong Diffie–Hellman (SplitRSDH) assumption holds with error $\epsilon_{\text{SplitRSDH}} = \epsilon_{\text{SplitRSDH}}(\lambda, t_{\text{SplitRSDH}})$. Then, the FC scheme *linKZG* (Construction 8.9) has state-restoration function binding error $\epsilon_{\text{FC}}^{\text{SR}}$ against any adversary with move budget m_{FC} and size t_{FC} , where*

$$\epsilon_{\text{FC}}^{\text{SR}}(m_{\text{FC}}, t_{\text{FC}}) \leq \frac{m_{\text{FC}} + 1}{2^\lambda} + 2\epsilon_{\text{ARSDH}}(\lambda, t_{\text{ARSDH}}) + \epsilon_{\text{SplitRSDH}}(\lambda, t_{\text{SplitRSDH}}) ,$$

and $t_{\text{ARSDH}}, t_{\text{SplitRSDH}} = O((m_{\text{FC}} + 1) \cdot t_{\text{FC}})$.

Lemma 1 provides new insights over prior work. [LPS25] proves that `linkZG` does not satisfy special soundness (assuming the hardness of the DL problem); in fact, one can extend their analysis to show that `linkZG` does not even satisfy extractability (special soundness implies extractability, and extractability and state-restoration function binding are incomparable). No prior work establishes any security property of `linkZG` (based on falsifiable assumptions) that is useful towards succinct non-interactive arguments.

1.1.3 Application: Plonk is secure based on ARSDH and SplitRSDH

Our results (Theorem 1 and Lemma 1) imply that Plonk [GWC19] is secure, in the ROM, based on ARSDH and SplitRSDH.

Corollary 1. *Assuming ARSDH and SplitRSDH, Plonk is a succinct non-interactive argument of knowledge in the ROM.*

Prior work establishes the security of Plonk in the ROM based on ARSDH and SplitRSDH [LPS25] via a direct proof that makes non-black-box use of the FIOP and FC underlying Plonk, without distilling a security property for the FC, i.e., `linkZG`. In contrast, we obtain the same conclusion simply by establishing function binding for the relevant `linkZG`, which suffices for the security of the resulting argument.

2 Techniques

This section contains an extended discussion of the techniques underlying our results that highlights the main ideas. The full technical details, contained in the subsequent sections, require many more pages to spell out; this is due to the general setting that we consider and the complexity of establishing state-restoration security.

In Section 2.1 we describe challenges that arise when adapting ideas from prior work to the general setting of FIOPs and FC schemes. In Sections 2.2 and 2.3, as a warmup, we sketch how we address these challenges for the special case of FPCPs. In Section 2.4 we discuss the general case of FIOPs, including showing state-restoration security. In Section 2.5 we discuss Lemma 1 and Corollary 1.

Below is notation that we use throughout this section.

Query class. A query class $\mathbf{Q}$ is a set of functions of the form $\alpha: \Sigma^\ell \rightarrow \mathbb{D}$ for a given proof alphabet Σ , proof length ℓ , and answer domain $\mathbb{D}$. We define notable query classes that we use as examples in this section: *point queries* in Equation 1, *linear queries* in Equation 2, *univariate polynomial queries* in Equation 3, and *multivariate polynomial queries* in Equation 4. We also discuss evaluation queries on structured polynomials in Section 2.5, which arise in Plonk [GWC19].

$$\mathbf{Q}_{\text{Point}} := \left\{ \alpha: \mathbb{F}^\ell \rightarrow \mathbb{F} \mid \exists i \in [\ell] \text{ s.t. } \alpha(\Pi) = \Pi[i] \right\}, \quad (1)$$

$$\mathbf{Q}_{\text{Lin}} := \left\{ \alpha: \mathbb{F}^\ell \rightarrow \mathbb{F} \mid \exists \gamma \in \mathbb{F}^\ell \text{ s.t. } \alpha(\Pi) = \sum_{i=1}^{\ell} \gamma[i] \cdot \Pi[i] \right\}, \quad (2)$$

$$\mathbf{Q}_{\text{UniPoly}} := \left\{ \alpha: \mathbb{F}^\ell \rightarrow \mathbb{F} \mid \exists \gamma \in \mathbb{F} \text{ s.t. } \alpha(\Pi) = \sum_{i=1}^{\ell} \gamma^i \cdot \Pi[i] \right\}, \quad (3)$$

$$\mathbf{Q}_{\text{MultiPoly}}^{(m,D)} := \left\{ \ell := \binom{m+D}{D}, \alpha: \mathbb{F}^\ell \rightarrow \mathbb{F} \mid \begin{array}{l} \exists \gamma \in \mathbb{F}^m \text{ s.t.} \\ \alpha(\Pi) = \sum_{\substack{\omega \in \{0, \dots, D\}^m \\ \sum_{i \in [m]} \omega[i] \leq D}} \gamma^\omega \cdot \Pi[\omega] \\ \text{where } \gamma^\omega := \gamma[1]^{\omega[1]} \dots \gamma[m]^{\omega[m]} \end{array} \right\}. \quad (4)$$

These query classes implicitly depend on parameters such as the alphabets $(\Sigma, \mathbb{D}, \mathbb{F})$, proof length (ℓ) , degree bounds, and so on.

Public-coin FIOPs. We fix a public-coin functional interactive oracle proof $\text{FIOP} = (\mathbf{P}, \mathbf{V})$ with query class $\mathbf{Q}$. The prover $\mathbf{P}$ receives as input an instance-witness tuple (x, w) and the verifier $\mathbf{V}$ receives as input the instance x . For each round $i \in [k_{\text{FIOP}}]$, $\mathbf{P}$ outputs a proof string $\Pi \in \Sigma^{\ell_i}$ and $\mathbf{V}$ sends uniformly sampled randomness $\rho \in \{0, 1\}^{r_{\text{FIOP}, i}}$; afterwards, $\mathbf{V}$ can make queries $\alpha \in \mathbf{Q}$ to any Π_i to obtain answers $\alpha(\Pi_i) \in \mathbb{D}$; finally, $\mathbf{V}$ accepts or rejects. A functional probabilistically checkable proof FPCP is a special case of an FIOP with round complexity $k_{\text{FIOP}} = 1$. (See Section 3.2 for the formal definition of an FIOP.)

FC schemes. We fix a functional commitment scheme $\text{FC} = (\text{FC.Commit}, \text{FC.Open}, \text{FC.Check})$ with query class $\mathbf{Q}$ with the following syntax.

- **FC.Commit:** On input a message $m \in \Sigma^\ell$, FC.Commit outputs a commitment cm and auxiliary state aux .
- **FC.Open:** On input the auxiliary state aux and a query $\alpha \in \mathbf{Q}$, FC.Open outputs an opening proof pf .
- **FC.Check:** On input a commitment cm , query $\alpha \in \mathbf{Q}$, answer $\beta \in \mathbb{D}$, and opening proof pf , FC.Check determines if pf is valid for β being the evaluation of the query α on the message committed in cm .

We focus on non-interactive FC schemes in (most of) this section for simplicity. Theorem 1 works for interactive FC schemes as well. (See Section 3.3 for the formal definition of an FC scheme.)

2.1 Limitations of prior analyses

Attempting to straightforwardly adapt prior work to prove Theorem 1 runs into challenges.

Challenge 1: from point queries to function queries. An FIOP (resp., FPCP) for $\mathbf{Q}_{\text{Point}}$ is simply an IOP (resp., PCP) and an FC scheme for $\mathbf{Q}_{\text{Point}}$ is a VC scheme. The Funky protocol in the case $\mathbf{Q} = \mathbf{Q}_{\text{Point}}$ yields familiar protocols as special cases: Kilian’s protocol (when the FIOP is an FPCP) and the IBCS protocol. Prior work analyzes the security of these special cases in detail [CDGS23; CDGSY24]. For example, [CDGSY24] shows that the soundness error ϵ_{ARG} of Kilian’s protocol, when based on a PCP with proof length ℓ and soundness error ϵ_{PCP} and a VC scheme with position binding error ϵ_{VC} , satisfies the following bound against t_{ARG} -size adversaries:

$$\forall \epsilon > 0, \quad \epsilon_{\text{ARG}}(\mathbb{X}, t_{\text{ARG}}) \leq \epsilon_{\text{PCP}}(\mathbb{X}) + \epsilon_{\text{VC}}(t_{\text{VC}}) + \epsilon \quad \text{where} \quad t_{\text{VC}} = O\left(\frac{\ell}{\epsilon} \cdot t_{\text{ARG}}\right). \quad (5)$$

Reducing the general case of an FPCP and FC scheme for a function class $\mathbf{Q} = \{\alpha: \Sigma^\ell \rightarrow \mathbb{D}\}$ to the aforementioned special case is tempting though problematic. One could map an FPCP string $\Pi \in \Sigma^\ell$ to a corresponding PCP string $\Pi' := (\alpha(\Pi))_{\alpha \in \mathbf{Q}} \in \mathbb{D}^{\ell'}$ with proof length $\ell' := |\mathbf{Q}|$, viewing each query in $\mathbf{Q}$ as a point query into a string of all query evaluations. However, *this approach is insecure* because a malicious PCP string $\Pi' \in \mathbb{D}^{\ell'}$ need not be consistent with the all-query evaluation of any FPCP string $\Pi \in \Sigma^\ell$. Moreover, even if we were to ignore this problem, we would (at best) obtain a bound where

$$t_{\text{VC}} = O\left(\frac{\ell'}{\epsilon} \cdot t_{\text{ARG}}\right) = O\left(\frac{|\mathbf{Q}|}{\epsilon} \cdot t_{\text{ARG}}\right).$$

We *cannot afford such a bound*: while for point queries $|\mathbf{Q}_{\text{Point}}| = \ell$, in general $|\mathbf{Q}|$ may be *much larger*, even exponentially larger, than ℓ . For example, for univariate polynomial queries $|\mathbf{Q}_{\text{UniPoly}}| = |\mathbb{F}| \gg \ell$ and $\mathbb{F}$ typically has exponential size for PIOPs and PC schemes of interest. This would result in the size t_{VC} of the FC scheme adversary being unacceptably large (the FC scheme adversary would have exponential size).

In sum, we seek a “direct” reduction approach for a given query class $\mathbf{Q}$ that is both secure and efficient. In particular, security should depend on the FPCP/FIOP proof length ℓ , rather than on the size of $\mathbf{Q}$.

Challenge 2: Fiat–Shamir security. Analyses of SNARGs in the pure ROM constructed directly from PCPs/IOPs and Merkle commitment schemes prove Fiat–Shamir security, but rely on the *straightline extraction* property of the ROM-based Merkle commitment scheme [BCS16; CY24], which is not achievable by standard-model FC schemes. (Separately, as noted in Section 1, security analyses of succinct interactive arguments in the standard model obtained from VC schemes or PC schemes neglect Fiat–Shamir security.)

2.2 Warm-up: interactive arguments based on functional PCPs

We temporarily focus, as a warmup, on establishing the standard (not state-restoration) security of a special case of the Funky protocol, namely, the case of a **FPCP** and a **non-interactive FC scheme** for a generic query class $\mathbf{Q}$. We discuss state-restoration security of this special case in Section 2.3.

2.2.1 The Funky protocol (special case for FPCPs)

Funky[FPCP, FC] = $(\mathcal{P}, \mathcal{V})$ works as follows:

1. $\mathcal{P}$ computes the FPCP proof $\Pi \leftarrow \mathbf{P}(\mathbb{X}, \mathbb{W})$, the commitment $(\text{cm}, \text{aux}) \leftarrow \text{FC.Commit}(\Pi)$, and sends cm to $\mathcal{V}$.

2. $\mathcal{V}$ samples FPCP verifier randomness $\rho \leftarrow \{0, 1\}^r$ and sends it to $\mathcal{P}$.
3. $\mathcal{P}$ deduces the query set $\mathcal{Q} \subseteq \mathbf{Q}$ that $\mathbf{V}(\mathbf{x}; \rho)$ makes to Π . For each $\alpha \in \mathcal{Q}$, $\mathcal{P}$ sets the query answer $\beta_\alpha := \alpha(\Pi)$ and computes an opening proof $\text{pf}_\alpha := \text{FC.Open}(\text{aux}, \alpha)$. $\mathcal{P}$ sends $((\alpha, \beta_\alpha, \text{pf}_\alpha))_{\alpha \in \mathcal{Q}}$ to $\mathcal{V}$.
4. $\mathcal{V}$ performs the following checks:
 - (a) for every $\alpha \in \mathcal{Q}$, $\text{FC.Check}(\text{cm}, \alpha, \beta_\alpha, \text{pf}_\alpha) = 1$;
 - (b) $\mathbf{V}^{[(\alpha, \beta_\alpha)]_{\alpha \in \mathcal{Q}}}(\mathbf{x}; \rho) = 1$.

The notation $\mathbf{V}^{[(\alpha, \beta_\alpha)]_{\alpha \in \mathcal{Q}}}(\mathbf{x}; \rho)$ is the decision of the FPCP verifier $\mathbf{V}$, given instance $\mathbf{x}$ and FPCP randomness ρ , where each query $\alpha \in \mathcal{Q}$ is answered with β_α . (If $\mathbf{V}$ queries outside $\mathcal{Q}$ then $\mathbf{V}^{[(\alpha, \beta_\alpha)]_{\alpha \in \mathcal{Q}}}(\mathbf{x}; \rho) = 0$.)

2.2.2 Soundness analysis

We informally argue that the Funky protocol for FPCPs is sound. (Showing knowledge soundness is similar.) We review the proof of Kilian’s protocol in [CDGSY24] and explain how we build on it for our setting.

Review (special case): Kilian’s protocol. [CDGSY24] analyzes the security of Kilian’s protocol, i.e., Funky[PCP, VC] where PCP is a PCP and VC is a VC scheme. They describe a *reductor* $\mathfrak{R}$ that, given oracle access to an argument adversary $\tilde{\mathcal{P}}$, rewinds $\tilde{\mathcal{P}}$ for sufficiently many times to recover an underlying PCP string $\tilde{\Pi}$. Then they show that the probability of $\tilde{\mathcal{P}}$ convincing the argument verifier $\mathcal{V}$ is close to the probability of $\tilde{\Pi}$ convincing the PCP verifier $\mathbf{V}$. In particular, when $\tilde{\mathcal{P}}$ succeeds, one of the following happens: (i) $\mathbf{V}^{\tilde{\Pi}}$ accepts; (ii) $\tilde{\mathcal{P}}$ gives different answers to the same query during rewinding; or (iii) the verifier $\mathbf{V}$ queries a location that is “missing” from the extracted PCP string $\tilde{\Pi}$. The first event can be bounded by the PCP soundness error. The second event can be bounded by the position binding error of VC. The third event is related to determining the “right” number of rewinds N , per the following question.

Question 1. Fix an error $\epsilon \in (0, 1)$. Run $\tilde{\mathcal{P}}$ to obtain its first message cm (the commitment to an alleged PCP string). Independently run the rest of the interaction with the argument verifier $\mathcal{V}$ for $N + 1$ times (each time with fresh randomness). If $\tilde{\mathcal{P}}$ gives valid opening in the i -th run, record the query set $\mathcal{Q}_i$ and $\tilde{\mathcal{P}}$ ’s answer ans_i . For what N does the following hold:

$$\Pr \left[\begin{array}{l} (\forall i, j \in [N + 1] \ \forall q \in \mathcal{Q}_i \cap \mathcal{Q}_j, \ \text{ans}_i[q] = \text{ans}_j[q]) \\ \wedge \mathcal{Q}_{N+1} \not\subseteq \bigcup_{i \in [N]} \mathcal{Q}_i \end{array} \right] \leq \epsilon \ ?$$

(The first condition implies that $\tilde{\mathcal{P}}$ does not violate the position binding of VC.)

[CDGSY24] shows that for sufficiently large N , the probability above is small. In particular, they consider the distribution of a specific query location. For every $j \in [\ell]$, let δ_j be the probability that location j is queried by a uniformly sampled verifier randomness and the adversary $\tilde{\mathcal{P}}$ gives a valid opening. The probability that j does not appear in $\mathcal{Q}_i$ for any $i \in [N]$ but is included in $\mathcal{Q}_{N+1}$ is exactly

$$(1 - \delta_j)^N \cdot \delta_j \ ,$$

which is bounded by $\frac{1}{N}$ for large enough N . By a union bound, the probability that $\mathfrak{R}$ fails to obtain a valid opening for a query is upper-bounded by $\frac{\ell}{N}$, where ℓ is the proof length of the PCP. Setting $N := \frac{\ell}{\epsilon}$ yields (along with the rest of the security reduction) the upper bound stated in Equation 5.

Our reductor. We seek a reductor $\mathfrak{R}$ so that the soundness error of Funky[FPCP, FC] reduces to the soundness error of FPCP and the function binding error of FC. In the VC setting, each time it rewinds the adversary $\tilde{\mathcal{P}}$, the reductor $\mathfrak{R}$ records in a PCP string $\tilde{\Pi} \in \Sigma^\ell$ the answers given by $\tilde{\mathcal{P}}$ (if valid). However, for

a general query class $\mathbf{Q}$, recorded query-answer pairs $(\alpha_1, \beta_1), \dots, (\alpha_N, \beta_N) \in \mathbf{Q} \times \mathbb{D}$ are *constraints* on the set of possible proof strings $\tilde{\Pi} \in \Sigma^\ell$:

$$S := \left\{ \tilde{\Pi} \in \Sigma^\ell : \begin{array}{c} \alpha_1(\tilde{\Pi}) = \beta_1 \\ \vdots \\ \alpha_N(\tilde{\Pi}) = \beta_N \end{array} \right\}. \quad (6)$$

When $\mathbf{Q} = \mathbf{Q}_{\text{Point}}$ finding an element in this set is trivial: each query-answer directly specifies the location and value of an entry of $\tilde{\Pi}$, so the reducer sets the entries of $\tilde{\Pi}$ accordingly and sets arbitrarily the rest. For other query classes $\mathbf{Q}$, finding an element in this set may involve non-obvious computations. For example, when $\mathbf{Q} = \mathbf{Q}_{\text{UniPoly}}$, finding an element of this set is tantamount to interpolation, i.e., finding a bounded-degree polynomial that agrees with a list of evaluations (if one exists). In other cases still, there may not even be an efficient algorithm to find an element of this set. For example, suppose that the query set $\mathbf{Q}$ contains a single function $\alpha: \Sigma^\ell \rightarrow \mathbb{D}$ that is *one-way*; if the prover uses a random $\tilde{\Pi} \in \Sigma^\ell$ then no efficient algorithm can, given α and $\alpha(\tilde{\Pi})$, recover any Π' such that $\alpha(\tilde{\Pi}) = \alpha(\Pi')$.

In light of this, we require the query class $\mathbf{Q}$ to come with a *solver* $\text{Solver}_{\mathbf{Q}}$: an algorithm that receives as input a list of query-answer pairs $(\alpha_1, \beta_1), \dots, (\alpha_N, \beta_N) \in \mathbf{Q} \times \mathbb{D}$ and outputs an FPCP string $\tilde{\Pi}$ in the set S defined Equation 6, i.e., an FPCP string $\tilde{\Pi}$ that is consistent with all the given constraints (if one exists). We denote by $t_{\mathbf{Q}}$ the running time of $\text{Solver}_{\mathbf{Q}}$.

Informally, our reducer $\mathfrak{R}$ rewinds $\tilde{\mathcal{P}}$ for sufficiently many times, records all the valid query-answer pairs, and runs the solver to find a suitable FPCP string $\tilde{\Pi}$ for the recorded constraints, as follows.

$\mathfrak{R}^{\tilde{\mathcal{P}}(\text{aux}, \cdot)}(\text{cm})$: # aux is $\tilde{\mathcal{P}}$'s internal state after outputting the first message

1. Initialize an empty list K .
2. Repeat the following N times:
 - (a) Sample FPCP verifier randomness $\rho \leftarrow \{0, 1\}^r$.
 - (b) Run $\tilde{\mathcal{P}}(\text{aux}, \rho)$ to get the openings $((\alpha, \beta_\alpha, \text{pf}_\alpha))_{\alpha \in \mathcal{Q}}$.
 - (c) If $\text{FC.Check}(\text{cm}, \alpha, \beta_\alpha, \text{pf}_\alpha) = 1$ for all $\alpha \in \mathcal{Q}$, append $((\alpha, \beta_\alpha, \text{pf}_\alpha))_{\alpha \in \mathcal{Q}}$ to K .
3. Output $\tilde{\Pi} := \text{Solver}_{\mathbf{Q}}(K) \in \Sigma^\ell$.

Next, we want to show that, if the number of rewinds N is sufficiently large, the probability that $\tilde{\Pi}$ convinces the FPCP verifier $\mathbf{V}$ is close to the probability that $\tilde{\mathcal{P}}$ convinces the argument verifier $\mathcal{V}$.

How to set the number of rewinds N ? In the special case $\mathbf{Q} = \mathbf{Q}_{\text{Point}}$, the number of rewinds N comes from upper bounding the probability in Question 1. Moreover, each new filled-in location represents an additional constraint on the space of possible FPCP strings consistent with all the query-answer pairs recorded so far.

We consider an analogue of the above probability for a general query class $\mathbf{Q}$. Specifically, we introduce and analyze a probability $\epsilon_{\mathbf{Q}}(N)$ that we call the *tail error* of the query class $\mathbf{Q}$. Informally, $\epsilon_{\mathbf{Q}}(N)$ upper bounds the probability that after $N + 1$ rewinds: (i) the solution space is non-empty (i.e., function binding is not violated); and (ii) the solution space shrinks from the N -th rewinding to the $(N + 1)$ -th rewinding.

Definition 2 (informal). *A query class $\mathbf{Q}$ has **tail error** $\epsilon_{\mathbf{Q}} = \epsilon_{\mathbf{Q}}(N)$ if, for every distribution $\mathcal{D}$ over query-answer tuples $(\mathcal{Q} \subseteq \mathbf{Q}, \beta: \mathcal{Q} \rightarrow \mathbb{D})$,*

$$\Pr \left[\begin{array}{l} S_{N+1} \neq \emptyset \\ \wedge S_{N+1} \neq S_N \end{array} \middle| \begin{array}{l} S_0 := \Sigma^\ell \\ \text{For } i \in [N + 1] : \\ \quad (\mathcal{Q}_i, \beta_i) \leftarrow \mathcal{D} \\ \quad S_i := S_{i-1} \cap \{\Pi \in \Sigma^\ell : \forall \alpha \in \mathcal{Q}_i, \alpha(\Pi) = \beta_i(\alpha)\} \end{array} \right] \leq \epsilon_{\mathbf{Q}}(N).$$

The above event is a direct generalization of the event in Question 1 for $\mathbf{Q} = \mathbf{Q}_{\text{Point}}$. The condition $\forall i, j \in [N+1] \forall q \in \mathcal{Q}_i \cap \mathcal{Q}_j, \text{ans}_i[q] = \text{ans}_j[q]$ (position binding is not violated) corresponds to $S_{N+1} \neq \emptyset$ (function binding is not violated). The condition $\mathcal{Q}_{N+1} \not\subseteq \bigcup_{i \in [N]} \mathcal{Q}_i$ corresponds to $S_{N+1} \neq S_N$.

This definition enables us to obtain an upper bound. We let $\mathcal{D}$ be the output distribution of $\tilde{\mathcal{P}}$: $\mathcal{D}$ gives the query set and answers $(\mathcal{Q}, \beta)$ if $\tilde{\mathcal{P}}$ provides an accepting opening, otherwise $\mathcal{D}$ outputs $\emptyset$. Hence, fresh samples from $\mathcal{D}$ correspond to (fresh) rewinds of $\tilde{\mathcal{P}}$; and S_i is the set of all FPCP strings that are consistent with $\tilde{\mathcal{P}}$'s output after the i -th rewinding. This lets us prove that, for every N , the following holds:

$$\epsilon_{\text{ARG}}(\mathbb{X}, t_{\text{ARG}}) \leq \epsilon_{\text{FPCP}}(\mathbb{X}) + \epsilon_{\text{FC}}(t_{\text{VC}}) + \epsilon_{\mathbf{Q}}(N) \quad \text{where} \quad t_{\text{VC}} = O(N \cdot t_{\text{ARG}} + t_{\mathbf{Q}}) . \quad (7)$$

This leaves us with the task of setting N so that $\epsilon_{\mathbf{Q}}(N)$ is small, which leads to the next question.

How to compute tail errors? The analysis in [CDGSY24] (as explained in Section 2.2.2) shows that $\epsilon_{\mathbf{Q}_{\text{Point}}}(N) \leq \frac{\ell}{N}$. However their analysis does not generalize (see Section 2.1). We take a different approach.

Consider the following alternative argument for $\epsilon_{\mathbf{Q}_{\text{Point}}}(N) \leq \frac{\ell}{N}$. There are at most ℓ different $i \in [N+1]$ such that $\mathcal{Q}_i \not\subseteq \bigcup_{j \in [i-1]} \mathcal{Q}_j$ since $\mathcal{Q}_j \subseteq [\ell]$ for every $j \in [N+1]$; therefore

$$\sum_{i=1}^N \Pr [\mathcal{Q}_{i+1} \not\subseteq \bigcup_{j \in [i]} \mathcal{Q}_j] \leq \ell . \quad (8)$$

Moreover, each query set $\mathcal{Q}_i$ is a fresh sample, so $\mathcal{Q}_{i+1} \not\subseteq \bigcup_{j \in [i]} \mathcal{Q}_j$ is more likely to happen than $\mathcal{Q}_{i'+1} \not\subseteq \bigcup_{j \in [i']} \mathcal{Q}_j$ for $i' > i$:

$$\forall i, i' \in [N] \text{ s.t. } i' > i, \Pr [\mathcal{Q}_{i+1} \not\subseteq \bigcup_{j \in [i]} \mathcal{Q}_j] \geq \Pr [\mathcal{Q}_{i'+1} \not\subseteq \bigcup_{j \in [i']} \mathcal{Q}_j] . \quad (9)$$

Therefore:

$$\begin{aligned} & \Pr \left[\left(\forall i, j \in [N+1] \forall q \in \mathcal{Q}_i \cap \mathcal{Q}_j, \text{ans}_i[q] = \text{ans}_j[q] \right) \wedge \mathcal{Q}_{N+1} \not\subseteq \bigcup_{i \in [N]} \mathcal{Q}_i \right] \\ & \leq \Pr \left[\mathcal{Q}_{N+1} \not\subseteq \bigcup_{i \in [N]} \mathcal{Q}_i \right] \\ & \leq \frac{1}{N} \cdot \sum_{i=1}^N \Pr [\mathcal{Q}_{i+1} \not\subseteq \bigcup_{j \in [i]} \mathcal{Q}_j] \end{aligned} \quad \text{(Equation 9)}$$

$$\leq \frac{\ell}{N} . \quad \text{(Equation 8)}$$

The proof outline above can be adapted to other query classes. Specifically, we show that, for every $\mathbf{Q} \in \{\mathbf{Q}_{\text{Point}}, \mathbf{Q}_{\text{Lin}}, \mathbf{Q}_{\text{UniPoly}}, \mathbf{Q}_{\text{MultiPoly}}^{(m,D)}\}$,

$$\epsilon_{\mathbf{Q}}(N) \leq \frac{\ell}{N} .$$

In fact, in the technical sections, we consider a relaxed setting that suffices for us. We borrow the random-termination idea in [CDDGS25], leading to a minor modification of our reductor; then we suitably adapt the definition of a tail error, and establish the above bound with particularly simple analyses. See Section 5.

2.3 Fiat–Shamir security of interactive arguments based on functional PCPs

Let $\text{ARG} := \text{Funky}[\text{FPCP}, \text{FC}]$ be the warmup example discussed in Section 2.2, and let $\text{NARG} := \text{FS}[\text{ARG}]$ be the non-interactive argument in the ROM after applying the Fiat–Shamir transformation to ARG. Fiat–Shamir security in this warmup example is “for free”. Indeed, since the verifier sends only one message in ARG, the soundness error ϵ_{NARG} for NARG is closely related to the soundness error ϵ_{ARG} for ARG. Specifically, for an instance $\mathbb{x}$ not in the language, the soundness of NARG against t_{NARG} -size adversaries satisfies the following (see, e.g., [CY24, Lemma 12.2.1 and Lemma 12.3.1]):

$$\epsilon_{\text{NARG}}(\mathbb{x}, t_{\text{NARG}}) \leq O(t_{\text{NARG}}) \cdot \epsilon_{\text{ARG}}(\mathbb{x}, t_{\text{NARG}}) .$$

However, in general, it is *not* the case that the soundness error ϵ_{NARG} for $\text{NARG} := \text{FS}[\text{ARG}]$, where ARG is a (public-coin) k_{NARG} -round interactive argument, is closely related to the soundness ϵ_{ARG} for ARG. Specifically, it holds that $\epsilon_{\text{NARG}}(\mathbb{x}, t_{\text{NARG}}) \leq O(t_{\text{NARG}}^{k_{\text{NARG}}}) \cdot \epsilon_{\text{ARG}}(\mathbb{x}, t_{\text{NARG}})$, and this upper bound is essentially tight. This is why, when we leave the warmup and study the general case, the Fiat–Shamir security of the Funky protocol is not “for free” — we directly establish it based on suitable properties of the underlying ingredients.

As a stepping stone towards the general case (Section 2.4), below we give a proof sketch of Theorem 1 for the special case of FPCPs and non-interactive FC schemes.

2.3.1 State-restoration security

We review the notion of *state-restoration* soundness, which captures Fiat–Shamir security in the ROM.

Definition 2.1. *The argument state-restoration game for $\text{ARG} = (\mathcal{G}, \mathcal{P}, \mathcal{V})$ with functions $\text{rnd} = (\text{rnd}_i)_{i \in [k]}$ where $\text{rnd}_i \leftarrow \mathcal{U}(r_i)$ for every $i \in [k]$, and for the argument state-restoration prover $\tilde{\mathcal{P}}^{\text{SR}}$, is defined below.*

$\text{SRGame}(\text{rnd}, \tilde{\mathcal{P}}^{\text{SR}})$:

1. Repeat the following until $\tilde{\mathcal{P}}^{\text{SR}}$ decides to exit the loop:
 - (a) $\tilde{\mathcal{P}}^{\text{SR}}$ outputs $(\mathbb{x}, (m_j)_{j \in [i]})$, where $\mathbb{x}$ is an instance and $(m_j)_{j \in [i]}$ are prover messages.
 - (b) Set $\rho_i := \text{rnd}_i(\mathbb{x}, (m_j)_{j \in [i]})$.
 - (c) Send ρ_i to $\tilde{\mathcal{P}}^{\text{SR}}$.
2. $\tilde{\mathcal{P}}^{\text{SR}}$ outputs $(\mathbb{x}, (m_i)_{i \in [k]})$.
3. For every $i \in [k]$, set $\rho_i := \text{rnd}_i(\mathbb{x}, (m_j)_{j \in [i]})$.
4. Output $(\mathbb{x}^*, (m_i^*)_{i \in [k]}, (\rho_i^*)_{i \in [k]})$.

We denote by tr^{SR} the list of move-response pairs of the form $((\mathbb{x}, (m_j)_{j \in [i]}), \rho_i)$ performed in the loop. We show tr^{SR} in an execution of the argument state-restoration game SRGame using the following notation:

$$(\mathbb{x}^*, (m_i^*)_{i \in [k]}, (\rho_i^*)_{i \in [k]}) \xleftarrow{\text{tr}^{\text{SR}}} \text{SRGame}(\text{rnd}, \tilde{\mathcal{P}}^{\text{SR}}) .$$

We say that $\tilde{\mathcal{P}}^{\text{SR}}$ is *m-move* if $\tilde{\mathcal{P}}^{\text{SR}}$ exits the loop after at most m iterations.

For the sake of exposition, we do not include public parameters, auxiliary inputs, and salts in the definition above; we give a full definition of the argument state-restoration game in Definition 3.7. The adversary wins the state-restoration game if and only if the verifier accepts when given the instance $\mathbb{x}^* \notin L(R)$, the randomness $(\rho_i^*)_{i \in [k]}$, and prover messages $(m_i^*)_{i \in [k]}$.

The state-restoration soundness error $\epsilon_{\text{ARG}}^{\text{SR}}(m, t_{\text{ARG}})$ for an interactive argument is an upper-bound on the probability of any t_{ARG} -size m -move adversary winning the state-restoration game. This notion captures

Fiat–Shamir security: the soundness error of the non-interactive argument obtained from the Fiat–Shamir transformation (in the ROM) is upper-bounded by the state-restoration soundness error of the given interactive argument [CY24]. One can similarly define the state-restoration security for FIOPs; since FIOPs are information-theoretically secure, the state-restoration soundness error $\epsilon_{\text{FIOP}}^{\text{SR}}(m_{\text{FIOP}})$ depends only on the move budget m_{FIOP} of the FIOP state-restoration adversary (rather than also its size). The FPCP state-restoration FPCPSRGame and its soundness error $\epsilon_{\text{FPCP}}^{\text{SR}}(m_{\text{FPCP}})$ are defined analogously.

2.3.2 Proof sketch of Theorem 1 for FPCPs and non-interactive FC schemes

Analogously to standard (not state-restoration) security, we want to reduce the state-restoration security of Funky[FPCP, FC] to the state-restoration security of FPCP. Given an argument state-restoration adversary $\tilde{\mathcal{P}}^{\text{SR}}$, we want to construct a FPCP state-restoration adversary $\tilde{\mathbf{P}}^{\text{SR}} := \tilde{\mathbf{P}}^{\text{SR}}(\tilde{\mathcal{P}}^{\text{SR}})$ for FPCP where:

- whenever $\tilde{\mathcal{P}}^{\text{SR}}$ makes a move $(\mathbb{x}, \text{cm})$, $\tilde{\mathbf{P}}^{\text{SR}}$ rewinds $\tilde{\mathcal{P}}^{\text{SR}}$ (some number of times) to reconstruct a corresponding FPCP string $\tilde{\Pi}$, makes the move $(\mathbb{x}, \tilde{\Pi})$ in the FPCP state-restoration game FPCPSRGame, and forwards the game’s response (i.e., verifier randomness) to $\tilde{\mathcal{P}}^{\text{SR}}$;
- when $\tilde{\mathcal{P}}^{\text{SR}}$ halts and outputs $(\mathbb{x}^*, \text{cm}^*, ((\alpha^*, \beta_{\alpha^*}^*, \text{pf}_{\alpha^*}^*))_{\alpha^* \in Q})$, $\tilde{\mathbf{P}}^{\text{SR}}$ rewinds $\tilde{\mathcal{P}}^{\text{SR}}$ (some number of times) to reconstruct a corresponding FPCP string $\tilde{\Pi}^*$ and then halts and outputs $(\mathbb{x}^*, \tilde{\Pi}^*)$ in the FPCP state-restoration game FPCPSRGame.

Below we outline the main ideas to make this work.

Final output. When $\tilde{\mathcal{P}}^{\text{SR}}$ halts with its final output

$$(\mathbb{x}^*, \text{cm}^*, ((\alpha^*, \beta_{\alpha^*}^*, \text{pf}_{\alpha^*}^*))_{\alpha^* \in Q}) ,$$

the FPCP state-restoration adversary needs to reconstruct a corresponding FPCP string $\tilde{\Pi}^*$ that is consistent with the query-answer pairs in the final output. However, $\tilde{\mathcal{P}}^{\text{SR}}$ has already halted, and there is no more interaction with $\tilde{\mathcal{P}}^{\text{SR}}$ to be had, so $\tilde{\mathbf{P}}^{\text{SR}}$ cannot rewind $\tilde{\mathcal{P}}^{\text{SR}}$ to collect additional valid openings for the commitment.

We augment $\tilde{\mathcal{P}}^{\text{SR}}$ to output the partial move induced by the final output. We consider an extended argument state-restoration adversary $\hat{\mathcal{P}}^{\text{SR}}$ that works the same as $\tilde{\mathcal{P}}^{\text{SR}}$ except that, before halting, additionally makes the move $(\mathbb{x}, \text{cm})$. This increases the number of moves from m (for $\tilde{\mathcal{P}}^{\text{SR}}$) to $m + 1$ (for $\hat{\mathcal{P}}^{\text{SR}}$). This additional partial move gives $\tilde{\mathbf{P}}^{\text{SR}}$ the chance to rewind (and extract), as it is consistent with $\tilde{\mathcal{P}}^{\text{SR}}$ ’s final output.

Intermediate moves. Consider a move $(\mathbb{x}, \text{cm})$ by $\tilde{\mathcal{P}}^{\text{SR}}$. To reconstruct a corresponding FPCP string, $\tilde{\mathbf{P}}^{\text{SR}}$ should rewind $\tilde{\mathcal{P}}^{\text{SR}}$ multiple times in search of (valid) openings for the commitment in the move. Each fresh run of $\tilde{\mathcal{P}}^{\text{SR}}$ yields more moves (which do not contain any openings) and a final output

$$(\mathbb{x}^*, \text{cm}^*, ((\alpha^*, \beta_{\alpha^*}^*, \text{pf}_{\alpha^*}^*))_{\alpha^* \in Q}) .$$

If $(\mathbb{x}, \text{cm}) \neq (\mathbb{x}^*, \text{cm}^*)$ then this final output does not contribute any valid openings for the move. Worse, it can be that *no run of $\tilde{\mathcal{P}}^{\text{SR}}$ yields any progress for $(\mathbb{x}, \text{cm})$* ($\tilde{\mathcal{P}}^{\text{SR}}$ never opens the commitments for this move).

We resolve this issue as follows. If $\tilde{\mathbf{P}}^{\text{SR}}$ cannot reconstruct an FPCP string for $(\mathbb{x}, \text{cm})$, then it suffices for $\tilde{\mathbf{P}}^{\text{SR}}$ to use an *empty FPCP string* for this move because $\tilde{\mathcal{P}}^{\text{SR}}$ does not use this move to win SRGame. Hence, it suffices for $\tilde{\mathbf{P}}^{\text{SR}}$ to provide to $\tilde{\mathcal{P}}^{\text{SR}}$ random and consistent answers. (To ensure that the mapping of moves from one game to the other is injective, the commitment cm is included in the salt string.)

Reductor. Taking the above into account, we construct $\tilde{\mathbf{P}}^{\text{SR}}$ as follows. Here, N is the number of rewinds, to be set later, that $\tilde{\mathbf{P}}^{\text{SR}}$ performs for every move of $\tilde{\mathcal{P}}^{\text{SR}}$.

$\tilde{\mathbf{P}}^{\text{SR}}$:

1. Simulate SRGame using $\hat{\mathcal{P}}^{\text{SR}}$.
2. For every move $\text{mv} = (\mathbb{x}, \text{cm})$ made by $\hat{\mathcal{P}}^{\text{SR}}$:
 - (a) Let aux be the internal state of $\hat{\mathcal{P}}^{\text{SR}}$ after outputting the move.
 - (b) Repeat N times:
 - i. Continue the simulation of $\hat{\mathcal{P}}^{\text{SR}}(\text{aux})$ using fresh randomness to answer $\tilde{\mathcal{P}}^{\text{SR}}$'s moves.
 - ii. If $\hat{\mathcal{P}}^{\text{SR}}$ provides answers and valid openings consistent with mv , record the answers.
 - (c) Construct $\tilde{\Pi}$ using $\text{Solver}_{\mathbf{Q}}$ from the recorded answers.
 - (d) Forward $(\mathbb{x}, \tilde{\Pi})$ to FPCPSRGame to get the next verifier randomness and answer the move mv output by $\hat{\mathcal{P}}^{\text{SR}}$.
3. Let $\tilde{\Pi}^*$ be the FPCP string corresponding to the commitment cm^* in the final output of $\tilde{\mathcal{P}}^{\text{SR}}$.
4. Output $(\mathbb{x}^*, \tilde{\Pi}^*)$, where $\mathbb{x}^*$ is the instance in the final output of $\tilde{\mathcal{P}}^{\text{SR}}$.

We are left to argue that $\tilde{\mathbf{P}}^{\text{SR}}$ is “as good as” $\tilde{\mathcal{P}}^{\text{SR}}$. For this, we extend to state-restoration games the ideas that we introduced in Section 2.2.2. If $\tilde{\mathcal{P}}^{\text{SR}}$ succeeds then one of the following happens: (i) $\tilde{\mathbf{P}}^{\text{SR}}$ succeeds; (ii) $\tilde{\mathcal{P}}^{\text{SR}}$ violates function binding of FC; or (iii) $\tilde{\mathbf{P}}^{\text{SR}}$ fails to extract enough information about the FPCP string underlying the final output of $\tilde{\mathcal{P}}^{\text{SR}}$. Similar to Section 2.2.2, the probability of the second event can be bounded by the function binding error of FC and the probability of the third event can be bounded via tail errors as defined in Section 2.2.2.

Additionally, the probability of the first event can be bounded by the state-restoration soundness error of FPCP (this is not immediately true for the general case of FIOPs, see Section 2.4). Overall, we bound the state-restoration soundness error $\epsilon_{\text{ARG}}^{\text{SR}}$ of ARG as follows:

$$\epsilon_{\text{ARG}}^{\text{SR}}(m, t_{\text{ARG}}) \leq \epsilon_{\text{FPCP}}^{\text{SR}}(m_{\text{FPCP}}) + \epsilon_{\text{FC}}(t_{\text{FC}}) + \epsilon_{\mathbf{Q}}(N) \quad \text{where} \quad \begin{cases} m_{\text{FIOP}} = m + 1 \\ t_{\text{FC}} = O(N \cdot t_{\text{ARG}} + t_{\mathbf{Q}}) \end{cases}.$$

An analogous bound holds for state-restoration knowledge soundness.

2.4 Succinct arguments based on public-coin functional IOPs

We discuss the general case: the Fiat–Shamir security of $\text{ARG} := \text{Funky}[\text{FIOP}, \text{FC}]$, i.e., the Funky protocol based on a functional interactive oracle proof FIOP and a functional commitment scheme FC for a generic query class $\mathbf{Q}$. As already explained in Section 2.3, merely establishing the (standard) soundness of ARG is *insufficient*, since superconstant-round protocols may be insecure after the Fiat–Shamir transformation; rather, we directly establish the state-restoration soundness of ARG based on security properties of FIOP and of FC.

2.4.1 The Funky protocol (general case for every FIOP)

$\text{Funky}[\text{FIOP}, \text{FC}]$ is the interactive argument $\text{ARG} = (\mathcal{P}, \mathcal{V})$ defined as follows:

1. For every FIOP round $i \in [k_{\text{FIOP}}]$:
 - (a) $\mathcal{P}$ computes the i -th FIOP string $\Pi_i \leftarrow \mathbf{P}(\mathbb{x}, w, \rho_{i-1})$, the commitment $(\text{cm}_i, \text{aux}_i) \leftarrow \text{FC.Commit}(\Pi_i)$, and sends cm_i to $\mathcal{V}$.
 - (b) $\mathcal{V}$ samples the i -th FIOP randomness $\rho_i \leftarrow \{0, 1\}^{r_{\text{FIOP}, i}}$ and sends it to $\mathcal{P}$.
2. $\mathcal{P}$ simulates $\mathbf{V}^{(\Pi_i)_{i \in [k_{\text{FIOP}]}}(\mathbb{x}, (\rho_i)_{i \in [k_{\text{FIOP}]})}$ to deduce the query sets $(\mathcal{Q}_i)_{i \in [k_{\text{FIOP}]}$ ($\mathcal{Q}_i$ is $\mathbf{V}$'s queries to Π_i). For every $i \in [k_{\text{FIOP}}]$ and $\alpha \in \mathcal{Q}_i$:
 - (a) $\mathcal{P}$ sets $\beta_{i, \alpha} := \alpha(\Pi_i)$;

- (b) $\mathcal{P}$ computes the opening proof $\text{pf}_{i,\alpha} := \text{FC.Open}(\text{aux}_i, \alpha)$;
- (c) $\mathcal{P}$ sets $\beta_i := (\beta_{i,\alpha})_{\alpha \in \mathcal{Q}_i}$ and $\text{pf}_i := (\text{pf}_{i,\alpha})_{\alpha \in \mathcal{Q}_i}$.
- 3. $\mathcal{P}$ sends $((\alpha, \beta_{i,\alpha}, \text{pf}_{i,\alpha}))_{\alpha \in \mathcal{Q}_i, i \in [k_{\text{FIOP}}]}$ to $\mathcal{V}$.
- 4. $\mathcal{V}$ performs the following checks:
 - (a) for every $i \in [k_{\text{FIOP}}]$ and $\alpha \in \mathcal{Q}_i$, $\text{FC.Check}(\text{cm}_i, \alpha, \beta_{i,\alpha}, \text{pf}_{i,\alpha}) = 1$;
 - (b) $\mathbf{V}[(\alpha, \beta_{i,\alpha})_{i \in [k_{\text{FIOP}}], \alpha \in \mathcal{Q}_i}(\mathbb{x}, (\rho_i)_{i \in [k_{\text{FIOP}}]}) = 1$.

See Section 4 for a detailed description of the Funky protocol. (In that description we no longer assume that FC has a non-interactive opening phase, and for convenience we use a batch interface bFC for FC.)

2.4.2 State-restoration security of the Funky protocol

We describe how we extend the basic ideas from the special case in Section 2.3 to prove Theorem 1.

Proof sketch of Theorem 1 for non-interactive FC schemes. Analogously to the special case of FPCPs (Section 2.3), we want to reduce the state-restoration security of Funky[FIOP, FC] to the state-restoration security of FIOP. Ideally, given an argument state-restoration adversary $\tilde{\mathcal{P}}^{\text{SR}}$, we want to construct a FIOP state-restoration adversary $\tilde{\mathbf{P}}^{\text{SR}} := \tilde{\mathbf{P}}^{\text{SR}}(\tilde{\mathcal{P}}^{\text{SR}})$ for FIOP where:

- whenever $\tilde{\mathcal{P}}^{\text{SR}}$ makes a move $(\mathbb{x}, (\text{cm}_j)_{j \in [i]})$, $\tilde{\mathbf{P}}^{\text{SR}}$ rewinds $\tilde{\mathcal{P}}^{\text{SR}}$ (some number of times) to reconstruct corresponding FIOP strings $(\tilde{\Pi}_j)_{j \in [i]}$, makes the move $(\mathbb{x}, (\tilde{\Pi}_j)_{j \in [i]})$ in the FIOP state-restoration game FIOPSRGame, and forwards the game's response (i.e., verifier randomness) to $\tilde{\mathcal{P}}^{\text{SR}}$;
- when $\tilde{\mathcal{P}}^{\text{SR}}$ halts and outputs $(\mathbb{x}^*, (\text{cm}_i^*)_{i \in [k_{\text{FIOP}}]}, (((\alpha^*, \beta_{\alpha^*}^*, \text{pf}_{\alpha^*}^*))_{\alpha^* \in \mathcal{Q}_i})_{i \in [k_{\text{FIOP}}]})$, $\tilde{\mathbf{P}}^{\text{SR}}$ rewinds $\tilde{\mathcal{P}}^{\text{SR}}$ (some number of times) to reconstruct corresponding FIOP strings $(\tilde{\Pi}_i^*)_{i \in [k_{\text{FIOP}}]}$ and then halts and outputs $(\mathbb{x}^*, \tilde{\Pi}_i^*)_{i \in [k_{\text{FIOP}}]}$ in the FIOP state-restoration game FIOPSRGame.

Similar challenges arise as in the case of FPCPs as well as an additional one specific to the FIOP case. First we describe how we adapt the FIOP state-restoration adversary for the familiar challenges.

- *Final output.* We augment $\tilde{\mathcal{P}}^{\text{SR}}$ to output all partial moves induced by the final output

$$(\mathbb{x}^*, (\text{cm}_i^*)_{i \in [k_{\text{FIOP}}]}, (((\alpha^*, \beta_{\alpha^*}^*, \text{pf}_{\alpha^*}^*))_{\alpha^* \in \mathcal{Q}_i})_{i \in [k_{\text{FIOP}}]}) .$$

We consider an extended argument state-restoration adversary $\hat{\mathcal{P}}^{\text{SR}}$ that works the same as $\tilde{\mathcal{P}}^{\text{SR}}$ except that, before halting, additionally makes these moves (ordered by increasing i):

$$\{(\mathbb{x}, (\text{cm}_j)_{j \in [i]})\}_{i \in [k_{\text{FIOP}}]} .$$

This increases the number of moves from m (for $\tilde{\mathcal{P}}^{\text{SR}}$) to $m + k_{\text{FIOP}}$ (for $\hat{\mathcal{P}}^{\text{SR}}$). These additional partial moves give $\tilde{\mathbf{P}}^{\text{SR}}$ the chance to rewind (and extract) as they are all consistent with $\tilde{\mathcal{P}}^{\text{SR}}$'s final output.

- *Intermediate moves.* If $\tilde{\mathbf{P}}^{\text{SR}}$ cannot reconstruct FIOP strings for $\tilde{\mathcal{P}}^{\text{SR}}$'s move $(\mathbb{x}, (\text{cm}_j)_{j \in [i]})$, it suffices for $\tilde{\mathbf{P}}^{\text{SR}}$ to provide to $\tilde{\mathcal{P}}^{\text{SR}}$ random and consistent answers. (To ensure that the mapping of moves from one game to the other is injective, the commitments $(\text{cm}_j)_{j \in [i]}$ are included in the salt string.)

Hence we would construct the FIOP state-restoration adversary $\tilde{\mathbf{P}}^{\text{SR}}$ as follows.

$\tilde{\mathbf{P}}^{\text{SR}}$:

1. Simulate SRGame using $\hat{\mathcal{P}}^{\text{SR}}$.
2. For every move $mv = (\mathbb{x}, (cm_j)_{j \in [i]})$ made by $\hat{\mathcal{P}}^{\text{SR}}$:
 - (a) Let aux be the internal state of $\hat{\mathcal{P}}^{\text{SR}}$ after outputting the move.
 - (b) Repeat N times:
 - i. Continue the simulation of $\hat{\mathcal{P}}^{\text{SR}}(\text{aux})$ using fresh randomness to answer $\tilde{\mathcal{P}}^{\text{SR}}$'s moves.
 - ii. If $\hat{\mathcal{P}}^{\text{SR}}$ provides answers and valid openings consistent with mv, record the answers.
 - (c) For every $j \in [i]$, construct $(\tilde{\Pi}_j)_{j \in [i]}$ using Solver_Q from the recorded answers.
 - (d) Forward $(\mathbb{x}, (\tilde{\Pi}_j)_{j \in [i]})$ to FIOPSRGame to get the next verifier randomness and answer the move mv output by $\hat{\mathcal{P}}^{\text{SR}}$.
3. Let $(\tilde{\Pi}_i^*)_{i \in [k_{\text{FIOP}}]}$ be the FIOP strings corresponding to the commitments $(cm_i^*)_{i \in [k_{\text{FIOP}}]}$ in the final output of $\hat{\mathcal{P}}^{\text{SR}}$.
4. Output $(\mathbb{x}^*, (\tilde{\Pi}_i^*)_{i \in [k_{\text{FIOP}}]})$, where $\mathbb{x}^*$ is the instance in the final output of $\tilde{\mathcal{P}}^{\text{SR}}$.

As in Section 2.3, if $\tilde{\mathcal{P}}^{\text{SR}}$ succeeds then one of the following happens: (i) $\tilde{\mathcal{P}}^{\text{SR}}$ succeeds; (ii) $\tilde{\mathcal{P}}^{\text{SR}}$ violates function binding of FC; or (iii) $\tilde{\mathcal{P}}^{\text{SR}}$ fails to extract enough information about the FIOP strings underlying the final output of $\hat{\mathcal{P}}^{\text{SR}}$. The second event can be bounded by the function binding error of FC and the third event can be bounded via tail errors.

However, *contrary to the case of FPCPs, bounding the first event via the FIOP state-restoration soundness error is delicate*, which will require a fix in $\tilde{\mathcal{P}}^{\text{SR}}$ above. Suppose that $\tilde{\mathcal{P}}^{\text{SR}}$ makes moves $mv = (\mathbb{x}, (cm_1^*))$ and $mv' = (\mathbb{x}, (cm_1^*, cm_2^*))$. It can be that rewinding for mv leads to $\tilde{\Pi}_1$ and rewinding for mv' leads to $(\tilde{\Pi}'_1, \tilde{\Pi}'_2)$ with $\tilde{\Pi}'_1 \neq \tilde{\Pi}_1$. In the construction of $\tilde{\mathcal{P}}^{\text{SR}}$, we simulate SRGame by passing the randomness given by FIOPSRGame for the FIOP moves; thus, mv gets randomness ρ_1 corresponding to $\tilde{\Pi}_1$, and mv' gets randomness ρ'_2 corresponding to $(\tilde{\Pi}'_1, \tilde{\Pi}'_2)$. What should $\tilde{\Pi}_1^*$ be in Step 3 in $\tilde{\mathcal{P}}^{\text{SR}}$ in this case?

- Setting $\tilde{\Pi}_1^* := \tilde{\Pi}_1$: the FIOPSRGame must use a new randomness corresponding to $(\tilde{\Pi}_1, \tilde{\Pi}'_2)$ at the end, which is inconsistent with the randomness ρ'_2 given to the corresponding move mv' made by $\tilde{\mathcal{P}}^{\text{SR}}$.
- Setting $\tilde{\Pi}_1^* := \tilde{\Pi}'_1$: the FIOPSRGame must use a new randomness corresponding to $\tilde{\Pi}'_1$ at the end, which is inconsistent with the randomness ρ_1 given to the corresponding move mv made by $\tilde{\mathcal{P}}^{\text{SR}}$.

These inconsistencies prevent us from claiming that if $\tilde{\mathcal{P}}^{\text{SR}}$ succeeds, it must be the case that $\tilde{\mathcal{P}}^{\text{SR}}$ succeeds (when the second and the last event do not occur) as the randomness used by their games are not the same. We address this via memoization: for Step 2 in the above construction of $\tilde{\mathcal{P}}^{\text{SR}}$, instead of directly starting to rewind the current move, we first check if the commitments in the move are already extracted, if yes, we simply use the extracted FIOP strings directly. Returning to our example, for mv', since cm_1^* is already extracted, we just use $\tilde{\Pi}_1$ from the memory and only rewind to get $\tilde{\Pi}'_2$.

Overall, we bound the state-restoration soundness error $\epsilon_{\text{ARG}}^{\text{SR}}$ of ARG as follows:

$$\epsilon_{\text{ARG}}^{\text{SR}}(m, t_{\text{ARG}}) \leq \epsilon_{\text{FIOP}}^{\text{SR}}(m_{\text{FIOP}}) + k_{\text{FIOP}} \cdot \epsilon_{\text{FC}}(t_{\text{FC}}) + k_{\text{FIOP}} \cdot \epsilon_{\text{Q}}(N) ,$$

where

$$\begin{cases} m_{\text{FIOP}} = m + k_{\text{FIOP}} \\ t_{\text{FC}} = O(N \cdot t_{\text{ARG}} + t_{\text{Q}}) \end{cases} .$$

The round complexity k_{FIOP} of the FIOP multiplicatively affects two terms: (a) the term $k_{\text{FIOP}} \cdot \epsilon_{\text{FC}}(t_{\text{FC}})$, as the argument state-restoration adversary $\tilde{\mathcal{P}}^{\text{SR}}$ could violate function binding at any round, and for each round we construct a corresponding FC adversary A_{FC} that rewinds N times and runs Solver_Q; (b) the term $k_{\text{FIOP}} \cdot \epsilon_{\text{Q}}(N)$, as each extracted FIOP string in the final output incurs the tail error $\epsilon_{\text{Q}}(N)$.

Interactive FC schemes. FC schemes with an interactive opening phase are widely used (e.g., [BBBPWM18; Lee21; AGLMS23]). An interactive FC scheme FC is a tuple $(\text{FC.Commit}, \mathcal{P}_{\text{FC}}, \mathcal{V}_{\text{FC}})$ where $\mathcal{P}_{\text{FC}}$ and $\mathcal{V}_{\text{FC}}$ interact over k_{FC} rounds. Establishing the state-restoration security of the Funky protocol in this case demands that the FC scheme satisfies a *state-restoration strengthening* of the function binding property, which we introduce via a suitable game FCSRGame (Definition 3.19) and binding property (Definition 3.20).

The challenge in extending the security reduction to this case is that moves output by the state-restoration argument prover $\tilde{\mathcal{P}}^{\text{SR}}$ may attack the FIOP or the FC scheme. Hence we construct *two* state-restoration adversaries: (i) $\tilde{\mathcal{P}}^{\text{SR}} := \tilde{\mathcal{P}}^{\text{SR}}(\tilde{\mathcal{P}}^{\text{SR}})$ for FIOP's state-restoration game FIOPSRGame , analogously to the non-interactive FC scheme case; and (ii) $\tilde{\mathcal{P}}_{\text{FC}}^{\text{SR}} := \tilde{\mathcal{P}}_{\text{FC}}^{\text{SR}}(\tilde{\mathcal{P}}^{\text{SR}})$ for FC's state-restoration game FCSRGame .

In fact, $\tilde{\mathcal{P}}_{\text{FC}}^{\text{SR}}$ consists of k_{FIOP} adversaries $(\tilde{\mathcal{P}}_{\text{FC},i}^{\text{SR}})_{i \in [k_{\text{FIOP}}]}$ where $\tilde{\mathcal{P}}_{\text{FC},i}^{\text{SR}}$ seeks to break the function binding property in FCSRGame for the i -th commitment, by rewinding $\tilde{\mathcal{P}}^{\text{SR}}$'s moves corresponding to the FC opening phase for the i -th commitment (as opposed to the FIOP interaction phase). When $\tilde{\mathcal{P}}^{\text{SR}}$ outputs a move of the form $(\mathbb{x}, (\text{cm}_u)_{u \in [c]})$ for some $c \in [k_{\text{FIOP}}]$ (i.e., corresponding to an FIOP interaction), $\tilde{\mathcal{P}}_{\text{FC},i}^{\text{SR}}$ answers this move via fresh (and consistent) randomness. Otherwise, $\tilde{\mathcal{P}}^{\text{SR}}$'s move is of the form $(\mathbb{x}, (\text{cm}_u)_{u \in [k_{\text{FIOP}}]}, ((\mathcal{Q}_u, \beta_u))_{u \in [k_{\text{FIOP}}]}, (\text{pm}_{u,j})_{u \in [k_{\text{FIOP}}], j \in [c]})$ for some $c \in [k_{\text{FC}}]$ where:

- $((\mathcal{Q}_u, \beta_u))_{u \in [k_{\text{FIOP}}]}$ are the queries and answers for the corresponding commitments $(\text{cm}_u)_{u \in [k_{\text{FIOP}}]}$;
- for every $u \in [k_{\text{FIOP}}]$, $(\text{pm}_{u,j})_{j \in [c]}$ are c prover messages sent in the interactive opening phase for cm_u .

In this case, $\tilde{\mathcal{P}}_{\text{FC},i}^{\text{SR}}$ makes the move $(\text{cm}_u, (\mathcal{Q}_u, \beta_u), (\text{pm}_{u,j})_{j \in [c]})$ in the FC state-restoration game FCSRGame for every $u \in [k_{\text{FIOP}}]$. After this, $\tilde{\mathcal{P}}_{\text{FC},i}^{\text{SR}}$ runs $\text{Solver}_{\mathbf{Q}}$ on the (valid) queries and answers collected during rewinding, to determine if there is an FIOP string $\Pi_i \in \Sigma^{\ell_i}$ consistent with them. The adversary $\tilde{\mathcal{P}}_{\text{FC},i}^{\text{SR}}$ wins if $\text{Solver}_{\mathbf{Q}}$ does not find any such Π_i , since this means that $\tilde{\mathcal{P}}_{\text{FC},i}^{\text{SR}}$ broke function binding of FC. We prove that if Case ii happens when $\tilde{\mathcal{P}}^{\text{SR}}$ successfully wins the argument state-restoration game SRGame then there exists $i \in [k_{\text{FIOP}}]$ such that $\tilde{\mathcal{P}}_{\text{FC},i}^{\text{SR}}$ wins in the FC state-restoration game FCSRGame .

Note that since $\tilde{\mathcal{P}}_{\text{FC},i}^{\text{SR}}$ rewinds $\tilde{\mathcal{P}}^{\text{SR}}$ for N times, and in the worst case each move by $\tilde{\mathcal{P}}^{\text{SR}}$ results in k_{FIOP} moves by $\tilde{\mathcal{P}}_{\text{FC},i}^{\text{SR}}$, the total number of moves by $\tilde{\mathcal{P}}_{\text{FC},i}^{\text{SR}}$ is $O(N \cdot k_{\text{FIOP}} \cdot (m + k_{\text{FC}}))$. (Similarly to how we address Challenge 2 in Section 2.4.2 in $\tilde{\mathcal{P}}^{\text{SR}}$, we augment $\tilde{\mathcal{P}}^{\text{SR}}$ to output all partial moves corresponding to the FC opening phase in its final output, which increases the number of moves by $\tilde{\mathcal{P}}^{\text{SR}}$ from m to $m + k_{\text{FC}}$.)

Overall, for the Funky protocol, we upper-bound the state-restoration soundness error $\epsilon_{\text{ARG}}^{\text{SR}}$ as

$$\epsilon_{\text{ARG}}^{\text{SR}}(m, t_{\text{ARG}}) \leq \epsilon_{\text{FIOP}}^{\text{SR}}(m_{\text{FIOP}}) + k_{\text{FIOP}} \cdot \epsilon_{\text{FC}}^{\text{SR}}(m_{\text{FC}}, t_{\text{FC}}) + k_{\text{FIOP}} \cdot \epsilon_{\mathbf{Q}}(N)$$

$$\text{where } \begin{cases} m_{\text{FIOP}} = m + k_{\text{FIOP}} \\ m_{\text{FC}} = O(N \cdot k_{\text{FIOP}} \cdot (m + k_{\text{FC}})) \\ t_{\text{FC}} = O(N \cdot t_{\text{ARG}} + t_{\mathbf{Q}}) \end{cases},$$

and $\epsilon_{\text{FC}}^{\text{SR}}$ is the state-restoration function binding error of the FC scheme.

In the technical sections. We prove a security reduction lemma that enables us to “couple” the state-restoration security of the Funky protocol and the state-restoration security of the underlying FIOP in Section 6. From there, we derive the state-restoration soundness and state-restoration knowledge soundness of the Funky protocol in Section 7.

2.5 Function binding in action: On the security of Plonk

We outline how we prove Lemma 1 and Corollary 1.

Lemma 1: linkZG is state-restoration function binding. We prove the lemma via a sequence of simple black-box reductions that apply to any suitable homomorphic FC scheme FC, of which the KZG polynomial commitment scheme [KZG10] is an example. These reductions may be useful towards designing other optimized FC schemes.

- Every homomorphic functional commitment scheme FC for a query class $\mathbf{Q}$ gives rise to an optimized batched functional commitment scheme $\text{bFC} = \text{BatchMsg}[\text{FC}, m]$ for the query class $\mathbf{Q}_{\text{BatchMsg}}[\mathbf{Q}, m]$ (Definition 8.1). We show that if FC is state-restoration function binding, then so is bFC (Section 8.1).
- Every homomorphic functional commitment scheme FC for a query class $\mathbf{Q}$ gives rise to a functional commitment scheme $\text{combFC} = \text{combFC}[\text{FC}, n]$ for the query class $\mathbf{Q}_{\text{LinComb}}[\mathbf{Q}, n]$ (Definition 8.7).
- Every (non-interactive) homomorphic functional commitment scheme FC for the query class $\mathbf{Q}$ gives rise to an optimized linearized functional commitment scheme $\text{linFC} = \text{LinFC}[\text{FC}, m, (h_k)_{k \in [n]}]$ for the query class $\mathbf{Q}_{\text{Struct}}[\mathbf{Q}, m, (h_k)_{k \in [n]}]$ of structured queries (Definition 5.9), via a linearization technique [GWC19; CHMMVW20; LPS25; FFR24]. We show that if FC and $\text{combFC}[\text{FC}, n]$ are function binding, then so is linFC (Section 8.2).

In Lemma 9.1 we prove that the PC scheme KZG is state-restoration function binding under the (expected-time) ARSDH assumption. In Lemma 9.3 we prove that $\text{combFC}[\text{KZG}, n]$ is state-restoration function binding under the (expected-time) SDH and SplitRSDH assumptions. Since KZG is homomorphic we can apply the black-box reductions outlined above, establishing that

$$\text{linkZG} = \text{LinFC}[\text{KZG}, m, (h_k)_{k \in [n]}]$$

is state-restoration function binding (see Construction 8.9 and Corollaries 9.2 and 9.4 for details). In contrast, [LPS25] showed that under the discrete logarithm assumption, linkZG cannot be special sound in the standard model (in fact, their attack can be extended to show that linkZG cannot be knowledge sound). Finally, we compare function binding to other properties of KZG in Appendix B.

Corollary 1: Plonk is secure assuming ARSDH and SplitRSDH. Recall how Plonk is constructed.

- Let PlonkIOP be the FIOP for the query class $\mathbf{Q}_{\text{Struct}}[\mathbf{Q}_{\text{UniPoly}}, m, (h_k)_{k \in [n]}]$ described in [GWC19].
- Let linkZG be the FC scheme for the query class $\mathbf{Q}_{\text{Struct}}[\mathbf{Q}_{\text{UniPoly}}, m, (h_k)_{k \in [n]}]$ discussed above.
- Let $\text{iPlonk} := \text{Funky}[\text{PlonkIOP}, \text{linkZG}]$ be the interactive argument obtained via the Funky protocol.

Plonk (in the ROM) is the result of applying the Fiat–Shamir transformation in the ROM to iPlonk. Therefore, to establish the soundness of Plonk, it suffices to show state-restoration soundness of iPlonk (and similarly for knowledge soundness).

Our Theorem 1 reduces the above goal to upper bounding: (i) the state-restoration soundness error $\epsilon_{\text{FIOP}}^{\text{SR}}$ of PlonkIOP , (ii) the state-restoration function binding error $\epsilon_{\text{FC}}^{\text{SR}}$ of linkZG, and (iii) the tail error of $\mathbf{Q}_{\text{Struct}}[\mathbf{Q}_{\text{UniPoly}}, m, (h_k)_{k \in [n]}]$. We discuss each in turn.

- PlonkIOP is a constant-round FIOP, so its state-restoration soundness is polynomially related to its standard soundness.
- We bound the state-restoration function binding error $\epsilon_{\text{FC}}^{\text{SR}}$ for linkZG (our Lemma 1).
- We prove that $\epsilon_{\mathbf{Q}_{\text{Struct}}[\mathbf{Q}, m, (h_k)_{k \in [n]}}](N) \leq \frac{(m+n)(D+1)}{N}$ (see Lemma 5.10).

[LPS25] shows that iPlonk is special-sound (which implies Fiat–Shamir security in the ROM) assuming the hardness of ARSDH and SplitRSDH (a new falsifiable assumption they propose), using an ad-hoc security analysis. In contrast, our analysis is generic, and we believe is applicable to other FC schemes of interest.

Remark 2.2. To showcase how to prove function binding for other functional commitment schemes of interest, we prove function binding for two schemes in Appendix C: a non-interactive polynomial commitment scheme with square-root-sized proofs in Appendix C.1, and an interactive polynomial commitment scheme with logarithmically many rounds inspired by Bulletproofs in Appendix C.2.

3 Preliminaries

Definition 3.1. A **relation** R is a set of pairs $(\mathbb{x}, \mathbb{w})$ where $\mathbb{x}$ is an instance and $\mathbb{w}$ a witness. The corresponding **language** $L(R)$ is the set of instances $\mathbb{x}$ for which there exists a witness $\mathbb{w}$ such that $(\mathbb{x}, \mathbb{w}) \in R$.

Definition 3.2. A **query class** $\mathbf{Q}$ is a set of functions of the form $\alpha: \Sigma^\ell \rightarrow \mathbb{D}$, where Σ is the input alphabet, $\ell \in \mathbb{N}$ is the input length, and $\mathbb{D}$ is the answer domain.

3.1 Interactive arguments

An *interactive argument* for a relation R (in the common reference string model) is a tuple of algorithms $\text{ARG} = (\mathcal{G}, \mathcal{P}, \mathcal{V})$ satisfying the following properties. We define the round complexity k as the number of rounds (back-and-forth interactions) between the argument prover and argument verifier. Additionally, for each $i \in [k]$, we define r_i as the number of random bits the argument verifier sends in round i .

Definition 3.3. $\text{ARG} = (\mathcal{G}, \mathcal{P}, \mathcal{V})$ has **perfect completeness** if for every security parameter $\lambda \in \mathbb{N}$, instance size bound $n \in \mathbb{N}$, public parameter $\text{pp} \in \mathcal{G}(1^\lambda, n)$, and instance-witness pair $(\mathbb{x}, \mathbb{w}) \in R$ with $|\mathbb{x}| \leq n$,

$$\Pr [\langle \mathcal{P}(\text{pp}, \mathbb{x}, \mathbb{w}), \mathcal{V}(\text{pp}, \mathbb{x}) \rangle = 1] = 1 .$$

Definition 3.4. $\text{ARG} = (\mathcal{G}, \mathcal{P}, \mathcal{V})$ has **(adaptive) soundness error** ϵ_{ARG} if for every security parameter $\lambda \in \mathbb{N}$, instance size bound $n \in \mathbb{N}$, auxiliary input distribution $\mathcal{D}$, circuit size bound $t_{\text{ARG}} \in \mathbb{N}$, and t_{ARG} -size circuit $\tilde{\mathcal{P}}$,

$$\Pr \left[\begin{array}{l} |\mathbb{x}| \leq n \\ \wedge \mathbb{x} \notin L(R) \\ \wedge b = 1 \end{array} \middle| \begin{array}{l} \text{pp} \leftarrow \mathcal{G}(1^\lambda, n) \\ \text{ai} \leftarrow \mathcal{D} \\ (\mathbb{x}, \text{aux}) \leftarrow \tilde{\mathcal{P}}(\text{pp}, \text{ai}) \\ b \leftarrow \langle \tilde{\mathcal{P}}(\text{aux}), \mathcal{V}(\text{pp}, \mathbb{x}) \rangle \end{array} \right] \leq \epsilon_{\text{ARG}}(\lambda, n, t_{\text{ARG}}) .$$

Definition 3.5. $\text{ARG} = (\mathcal{G}, \mathcal{P}, \mathcal{V})$ has **(adaptive) knowledge soundness error** κ_{ARG} **with extraction time** $t_{\mathcal{E}}$ if there exists a probabilistic algorithm $\mathcal{E}$ such that for every security parameter $\lambda \in \mathbb{N}$, instance size bound $n \in \mathbb{N}$, auxiliary input distribution $\mathcal{D}$, circuit size bound $t_{\text{ARG}} \in \mathbb{N}$, and t_{ARG} -size circuit $\tilde{\mathcal{P}}$,

$$\Pr \left[\begin{array}{l} |\mathbb{x}| \leq n \\ \wedge (\mathbb{x}, \mathbb{w}) \notin R \\ \wedge b = 1 \end{array} \middle| \begin{array}{l} \text{pp} \leftarrow \mathcal{G}(1^\lambda, n) \\ \text{ai} \leftarrow \mathcal{D} \\ (\mathbb{x}, \text{aux}) \leftarrow \tilde{\mathcal{P}}(\text{pp}, \text{ai}) \\ b \stackrel{\text{tr}}{\leftarrow} \langle \tilde{\mathcal{P}}(\text{aux}), \mathcal{V}(\text{pp}, \mathbb{x}) \rangle \\ \mathbb{w} \leftarrow \mathcal{E}^{\tilde{\mathcal{P}}(\text{aux})}(\text{pp}, \mathbb{x}, \text{tr}) \end{array} \right] \leq \kappa_{\text{ARG}}(\lambda, n, t_{\text{ARG}}) ;$$

moreover, $\mathcal{E}$ runs in time $t_{\mathcal{E}}(\lambda, n, t_{\text{ARG}})$.

Above, $b \stackrel{\text{tr}}{\leftarrow} \langle \tilde{\mathcal{P}}(\text{aux}), \mathcal{V}(\text{pp}, \mathbb{x}) \rangle$ denotes the fact that tr is the transcript of the interaction (i.e., the messages exchanged between $\tilde{\mathcal{P}}$ and $\mathcal{V}$). Moreover, $\mathcal{E}^{\tilde{\mathcal{P}}}$ means that $\mathcal{E}$ has black-box access to (each next-message function of) $\tilde{\mathcal{P}}$; in particular $\mathcal{E}$ can send verifier messages to $\tilde{\mathcal{P}}$ to obtain the next message of $\tilde{\mathcal{P}}$ (for a partial interaction where $\mathcal{V}$ sent those messages).

Definition 3.6. For $n \in \mathbb{N}$, $\mathcal{U}(n)$ is the uniform distribution over all functions $f: \{0, 1\}^* \rightarrow \{0, 1\}^n$. Moreover, for $m \in \mathbb{N}$ and $(n_1, \dots, n_m) \in \mathbb{N}^m$, $\mathcal{U}((n_1, \dots, n_m)) := \mathcal{U}(n_1) \times \dots \times \mathcal{U}(n_m)$.

Definition 3.7. The argument state-restoration game for $\text{ARG} = (\mathcal{G}, \mathcal{P}, \mathcal{V})$ with salt size $s \in \mathbb{N}$, functions $\text{rnd} = (\text{rnd}_i)_{i \in [k]}$ where $\text{rnd}_i \leftarrow \mathcal{U}(r_i)$ for every $i \in [k]$, argument state-restoration prover $\tilde{\mathcal{P}}^{\text{SR}}$, public parameter pp , and auxiliary input ai is defined below.

$\text{SRGame}(s, \text{rnd}, \tilde{\mathcal{P}}^{\text{SR}}, \text{pp}, \text{ai})$:

1. Repeat the following until $\tilde{\mathcal{P}}^{\text{SR}}(\text{pp}, \text{ai})$ decides to exit the loop:
 - (a) $\tilde{\mathcal{P}}^{\text{SR}}$ outputs $(\mathbb{x}, (m_j)_{j \in [i]}, (\sigma_j)_{j \in [i]})$, where $\mathbb{x}$ is an instance, $(m_j)_{j \in [i]}$ are prover messages, and $(\sigma_j)_{j \in [i]}$ are salt strings in $\{0, 1\}^s$.
 - (b) Set $\rho_i := \text{rnd}_i(\mathbb{x}, (m_j)_{j \in [i]}, (\sigma_j)_{j \in [i]})$.
 - (c) Send ρ_i to $\tilde{\mathcal{P}}^{\text{SR}}$.
2. $\tilde{\mathcal{P}}^{\text{SR}}$ outputs $(\mathbb{x}, (m_i)_{i \in [k]}, (\sigma_i)_{i \in [k]})$.
3. For every $i \in [k]$, set $\rho_i := \text{rnd}_i(\mathbb{x}, (m_j)_{j \in [i]}, (\sigma_j)_{j \in [i]})$.
4. Output $(\mathbb{x}, (m_i)_{i \in [k]}, (\sigma_i)_{i \in [k]}, (\rho_i)_{i \in [k]})$.

We denote by tr^{SR} the list of move-response pairs of the form $((\mathbb{x}, (m_j)_{j \in [i]}, (\sigma_j)_{j \in [i]}), \rho_i)$ performed in the loop. We show tr^{SR} in an execution of the argument state-restoration game SRGame using the following notation:

$$(\mathbb{x}, (m_i)_{i \in [k]}, (\sigma_i)_{i \in [k]}, (\rho_i)_{i \in [k]}) \xleftarrow{\text{tr}^{\text{SR}}} \text{SRGame}(s, \text{rnd}, \tilde{\mathcal{P}}^{\text{SR}}).$$

We say that $\tilde{\mathcal{P}}^{\text{SR}}$ is m-move if $\tilde{\mathcal{P}}^{\text{SR}}$ exits the loop after at most m iterations.

Definition 3.8. $\text{ARG} = (\mathcal{G}, \mathcal{P}, \mathcal{V})$ has **state-restoration soundness error** $\epsilon_{\text{ARG}}^{\text{SR}}$ if for every security parameter $\lambda \in \mathbb{N}$, instance size bound $n \in \mathbb{N}$, auxiliary input distribution $\mathcal{D}$, salt size $s \in \mathbb{N}$, move budget $\text{m} \in \mathbb{N}$, circuit size bound $t_{\text{ARG}} \in \mathbb{N}$, and m-move t_{ARG} -size circuit $\tilde{\mathcal{P}}^{\text{SR}}$,

$$\Pr \left[\begin{array}{l} |\mathbb{x}| \leq n \\ \wedge \mathbb{x} \notin L(R) \\ \wedge b = 1 \end{array} \middle| \begin{array}{l} \text{pp} \leftarrow \mathcal{G}(1^\lambda, n) \\ \text{ai} \leftarrow \mathcal{D} \\ \text{rnd} := (\text{rnd}_i)_{i \in [k]} \leftarrow \mathcal{U}((r_i)_{i \in [k]}) \\ (\mathbb{x}, (m_i)_{i \in [k]}, (\sigma_i)_{i \in [k]}, (\rho_i)_{i \in [k]}) \leftarrow \text{SRGame}(s, \text{rnd}, \tilde{\mathcal{P}}^{\text{SR}}, \text{pp}, \text{ai}) \\ b \leftarrow \mathcal{V}(\mathbb{x}, (m_i)_{i \in [k]}, (\rho_i)_{i \in [k]}) \end{array} \right] \leq \epsilon_{\text{ARG}}^{\text{SR}}(\lambda, n, s, \text{m}, t_{\text{ARG}}).$$

Definition 3.9. $\text{ARG} = (\mathcal{G}, \mathcal{P}, \mathcal{V})$ has **state-restoration knowledge soundness error** $\kappa_{\text{ARG}}^{\text{SR}}$ **with extraction time** t_{ESR} if there exists a probabilistic algorithm $\mathcal{E}_{\text{SR}}$ such that for every security parameter $\lambda \in \mathbb{N}$, instance size bound $n \in \mathbb{N}$, auxiliary input distribution $\mathcal{D}$, salt size $s \in \mathbb{N}$, move budget $\text{m} \in \mathbb{N}$, circuit size bound $t_{\text{ARG}} \in \mathbb{N}$, and m-move t_{ARG} -size deterministic state-restoration prover $\tilde{\mathcal{P}}^{\text{SR}}$,

$$\Pr \left[\begin{array}{l} |\mathbb{x}| \leq n \\ \wedge (\mathbb{x}, \mathbb{w}) \notin R \\ \wedge b = 1 \end{array} \middle| \begin{array}{l} \text{pp} \leftarrow \mathcal{G}(1^\lambda, n) \\ \text{ai} \leftarrow \mathcal{D} \\ \text{rnd} := (\text{rnd}_i)_{i \in [k]} \leftarrow \mathcal{U}((r_i)_{i \in [k]}) \\ (\mathbb{x}, (m_i)_{i \in [k]}, (\sigma_i)_{i \in [k]}, (\rho_i)_{i \in [k]}) \xleftarrow{\text{tr}^{\text{SR}}} \text{SRGame}(s, \text{rnd}, \tilde{\mathcal{P}}^{\text{SR}}, \text{pp}, \text{ai}) \\ \mathbb{w} \leftarrow \mathcal{E}_{\text{SR}}^{\tilde{\mathcal{P}}^{\text{SR}}(\text{pp}, \text{ai})}(\mathbb{x}, (m_i)_{i \in [k]}, (\sigma_i)_{i \in [k]}, (\rho_i)_{i \in [k]}, \text{tr}^{\text{SR}}) \\ b \leftarrow \mathcal{V}(\mathbb{x}, (m_i)_{i \in [k]}, (\rho_i)_{i \in [k]}) \end{array} \right] \leq \kappa_{\text{ARG}}^{\text{SR}}(\lambda, n, s, \text{m}, t_{\text{ARG}});$$

moreover, $\mathcal{E}_{\text{SR}}$ runs in time $t_{\text{ESR}}(\lambda, n, s, \text{m}, t_{\text{ARG}})$.

We list the efficiency measures for an argument:

- the round complexity k is the number of rounds (back-and-forth interactions) between the argument prover and argument verifier;
- the randomness complexity r is the total number of random bits the argument verifier $\mathcal{V}$ sends, r_i is the number of random bits the argument verifier sends in round i , and $r_{\max} := \max_{i \in [k]} \{r_i\}$;
- the prover-to-verifier communication complexity pc is the total number of bits sent by the argument prover, pc_i is the number of bits sent by the argument prover in round i , and $pc_{\max} := \max_{i \in [k]} \{pc_i\}$;
- the argument generator running time is t_G ;
- the argument prover running time is t_P ;
- the argument verifier running time is t_V ;
- the public parameter complexity p_{ARG} is the number of bits of the public parameter pp .

3.2 Functional interactive oracle proofs

The definition of (public-coin) functional interactive oracle proofs (FIOPs) is adapted from the definition of IOPs with special queries in [BCG20].

Definition 3.10. A public-coin FIOP with query classes $\mathbf{Q}_1, \dots, \mathbf{Q}_{k_{\text{FIOP}}}$ is a tuple $\text{FIOP} = (\mathbf{P}, \mathbf{V})$ where $\mathbf{P}$ is the prover and $\mathbf{V}$ the verifier. The prover receives as input an instance-witness tuple $(\mathbb{x}, \mathbb{w})$ and the verifier receives as input the instance $\mathbb{x}$. The protocol has k_{FIOP} rounds, and in each round $i \in [k_{\text{FIOP}}]$ the prover sends a message $\Pi_i \in \Sigma_i^{\ell_i}$ where Σ_i is the alphabet and ℓ_i is the proof length of round i , and the verifier replies with a random string $\rho_i \in \{0, 1\}^{r_{\text{FIOP}, i}}$ where $r_{\text{FIOP}, i}$ is the number of random bits that the FIOP verifier uses in round i . The verifier has access to $(\Pi_1, \Pi_2, \dots, \Pi_{k_{\text{FIOP}}})$ through queries in $\mathbf{Q}_1, \dots, \mathbf{Q}_{k_{\text{FIOP}}}$. In more detail, the answer of a query $\alpha \in \mathbf{Q}_i$ to Π_i for some $i \in [k_{\text{FIOP}}]$ is $\alpha(\Pi_i) \in \mathbb{D}$ (the answer may be an error value $\perp$ if Π_i is not according to the expected format). After the interaction and queries, the verifier $\mathbf{V}$ accepts or rejects.

A FIOP system for a relation R satisfies the following.

Definition 3.11. $\text{FIOP} = (\mathbf{P}, \mathbf{V})$ has **perfect completeness** if, for every instance-witness pair $(\mathbb{x}, \mathbb{w}) \in R$,

$$\Pr [\langle \mathbf{P}(\mathbb{x}, \mathbb{w}), \mathbf{V}(\mathbb{x}) \rangle = 1] = 1 .$$

Definition 3.12. $\text{FIOP} = (\mathbf{P}, \mathbf{V})$ has **soundness error** ϵ_{FIOP} if, for every (unbounded) circuit $\tilde{\mathbf{P}}$ and auxiliary input distribution $\mathbf{D}$,

$$\Pr \left[\begin{array}{l} |\mathbb{x}| \leq n \\ \wedge \mathbb{x} \notin L(R) \\ \wedge b = 1 \end{array} \middle| \begin{array}{l} \mathbf{ai} \leftarrow \mathbf{D} \\ (\mathbb{x}, \mathbf{aux}) \leftarrow \tilde{\mathbf{P}}(\mathbf{ai}) \\ b \leftarrow \langle \tilde{\mathbf{P}}(\mathbf{aux}), \mathbf{V}(\mathbb{x}) \rangle \end{array} \right] \leq \epsilon_{\text{FIOP}}(n) .$$

Definition 3.13. $\text{FIOP} = (\mathbf{P}, \mathbf{V})$ has **knowledge soundness error** κ_{FIOP} if there exists a probabilistic algorithm $\mathbf{E}$ such that, for every (unbounded) circuit $\tilde{\mathbf{P}}$ and auxiliary input distribution $\mathbf{D}$,

$$\Pr \left[\begin{array}{l} |\mathbb{x}| \leq n \\ \wedge (\mathbb{x}, \mathbb{w}) \notin R \\ \wedge b = 1 \end{array} \middle| \begin{array}{l} \mathbf{ai} \leftarrow \mathbf{D} \\ (\mathbb{x}, \mathbf{aux}) \leftarrow \tilde{\mathbf{P}}(\mathbf{ai}) \\ b \xleftarrow{\text{tr}} \langle \tilde{\mathbf{P}}(\mathbf{aux}), \mathbf{V}(\mathbb{x}) \rangle \\ \mathbb{w} \leftarrow \mathbf{E}^{\tilde{\mathbf{P}}(\mathbf{aux})}(\mathbb{x}, \text{tr}) \end{array} \right] \leq \kappa_{\text{FIOP}}(n) ;$$

moreover, $\mathbf{E}$ runs in time $t_{\mathbf{E}}(n)$.

Definition 3.14. The **FIOP state-restoration game** for $\text{FIOP} = (\mathbf{P}, \mathbf{V})$ with salt size $s_{\text{FIOP}} \in \mathbb{N}$, functions $\text{rnd}_{\text{FIOP}} = (\text{rnd}_{\text{FIOP},i})_{i \in [k_{\text{FIOP}}]}$ where $\text{rnd}_{\text{FIOP},i} : \{0, 1\}^* \rightarrow \{0, 1\}^{r_{\text{FIOP},i}}$ for every $i \in [k_{\text{FIOP}}]$, **FIOP state-restoration prover** $\tilde{\mathbf{P}}^{\text{SR}}$, and auxiliary input $\mathbf{ai}$ is defined below.

$\text{FIOPSRGame}(s_{\text{FIOP}}, \text{rnd}_{\text{FIOP}}, \tilde{\mathbf{P}}^{\text{SR}}, \mathbf{ai})$:

1. Repeat the following until $\tilde{\mathbf{P}}^{\text{SR}}(\mathbf{ai})$ decides to exit the loop:
 - (a) $\tilde{\mathbf{P}}^{\text{SR}}$ outputs $(\mathbb{x}, (m_j)_{j \in [i]}, (\gamma_j)_{j \in [i]})$, where $\mathbb{x}$ is an instance, $(m_j)_{j \in [i]}$ are prover messages, and $(\gamma_j)_{j \in [i]}$ are salt strings in $\{0, 1\}^{s_{\text{FIOP}}}$.
 - (b) Set $\rho_i := \text{rnd}_{\text{FIOP},i}(\mathbb{x}, (m_j)_{j \in [i]}, (\gamma_j)_{j \in [i]})$.
 - (c) Send ρ_i to $\tilde{\mathbf{P}}^{\text{SR}}$.
2. $\tilde{\mathbf{P}}^{\text{SR}}$ outputs $(\mathbb{x}, (m_i)_{i \in [k_{\text{FIOP}}]}, (\gamma_i)_{i \in [k_{\text{FIOP}}]})$.
3. For every $i \in [k_{\text{FIOP}}]$, set $\rho_i := \text{rnd}_{\text{FIOP},i}(\mathbb{x}, (m_j)_{j \in [i]}, (\gamma_j)_{j \in [i]})$.
4. Output $(\mathbb{x}, (m_i)_{i \in [k_{\text{FIOP}}]}, (\gamma_i)_{i \in [k_{\text{FIOP}}]}, (\rho_i)_{i \in [k_{\text{FIOP}}]})$.

We denote by $\text{tr}_{\text{FIOP}}^{\text{SR}}$ the list of move-response pairs of the form $((\mathbb{x}, (m_j)_{j \in [i]}, (\gamma_j)_{j \in [i]}), \rho_i)$ performed in the loop. We show $\text{tr}_{\text{FIOP}}^{\text{SR}}$ in an execution of the **FIOP state-restoration game** FIOPSRGame using the following notation:

$$(\mathbb{x}, (m_i)_{i \in [k_{\text{FIOP}}]}, (\gamma_i)_{i \in [k_{\text{FIOP}}]}, (\rho_i)_{i \in [k_{\text{FIOP}}]}) \xleftarrow{\text{tr}_{\text{FIOP}}^{\text{SR}}} \text{FIOPSRGame}(s_{\text{FIOP}}, \text{rnd}_{\text{FIOP}}, \tilde{\mathbf{P}}^{\text{SR}}).$$

We say that $\tilde{\mathbf{P}}^{\text{SR}}$ is $\mathbf{m}_{\text{FIOP}}$ -move if $\tilde{\mathbf{P}}^{\text{SR}}$ exits the loop after at most $\mathbf{m}_{\text{FIOP}}$ iterations.

Definition 3.15. $\text{FIOP} = (\mathbf{P}, \mathbf{V})$ has **state-restoration soundness error** $\epsilon_{\text{FIOP}}^{\text{SR}}$ if for every instance size bound $n \in \mathbb{N}$, auxiliary input distribution $\mathbf{D}$, salt size $s_{\text{FIOP}} \in \mathbb{N}$, move budget $\mathbf{m}_{\text{FIOP}} \in \mathbb{N}$, and $\mathbf{m}_{\text{FIOP}}$ -move circuit $\tilde{\mathbf{P}}^{\text{SR}}$,

$$\Pr \left[\begin{array}{l} |\mathbb{x}| \leq n \\ \wedge \mathbb{x} \notin L(R) \\ \wedge b = 1 \end{array} \middle| \begin{array}{l} \mathbf{ai} \leftarrow \mathbf{D} \\ \text{rnd}_{\text{FIOP}} := (\text{rnd}_{\text{FIOP},i})_{i \in [k_{\text{FIOP}}]} \leftarrow \mathcal{U}((r_{\text{FIOP},i})_{i \in [k_{\text{FIOP}}]}) \\ \left(\begin{array}{l} \mathbb{x}, (m_i)_{i \in [k_{\text{FIOP}}]}, \\ (\gamma_i)_{i \in [k_{\text{FIOP}}]}, (\rho_i)_{i \in [k_{\text{FIOP}}]} \end{array} \right) \leftarrow \text{FIOPSRGame}(s_{\text{FIOP}}, \text{rnd}_{\text{FIOP}}, \tilde{\mathbf{P}}^{\text{SR}}, \mathbf{ai}) \\ b \leftarrow \mathbf{V}(\mathbb{x}, (m_i)_{i \in [k_{\text{FIOP}}]}, (\rho_i)_{i \in [k_{\text{FIOP}}]}) \end{array} \right] \\ \leq \epsilon_{\text{FIOP}}^{\text{SR}}(n, s_{\text{FIOP}}, \mathbf{m}_{\text{FIOP}}).$$

Definition 3.16. $\text{FIOP} = (\mathbf{P}, \mathbf{V})$ has **state-restoration knowledge soundness error** $\kappa_{\text{FIOP}}^{\text{SR}}$ with extraction time $t_{\text{E}_{\text{SR}}}$ if there exists a probabilistic algorithm $\mathbf{E}_{\text{SR}}$ such that for every instance size bound $n \in \mathbb{N}$, auxiliary input distribution $\mathbf{D}$, salt size $s_{\text{FIOP}} \in \mathbb{N}$, move budget $\mathbf{m}_{\text{FIOP}} \in \mathbb{N}$, circuit size bound $t_{\text{FIOP}} \in \mathbb{N}$, and $\mathbf{m}_{\text{FIOP}}$ -move t_{FIOP} -size deterministic state-restoration prover $\tilde{\mathbf{P}}^{\text{SR}}$,

$$\Pr \left[\begin{array}{l} |\mathbb{x}| \leq n \\ \wedge (\mathbb{x}, \mathbb{w}) \notin R \\ \wedge b = 1 \end{array} \middle| \begin{array}{l} \mathbf{ai} \leftarrow \mathbf{D} \\ \text{rnd}_{\text{FIOP}} := (\text{rnd}_{\text{FIOP},i})_{i \in [k_{\text{FIOP}}]} \leftarrow \mathcal{U}((r_{\text{FIOP},i})_{i \in [k_{\text{FIOP}}]}) \\ \left(\begin{array}{l} \mathbb{x}, (m_i)_{i \in [k_{\text{FIOP}}]}, \\ (\gamma_i)_{i \in [k_{\text{FIOP}}]}, (\rho_i)_{i \in [k_{\text{FIOP}}]} \end{array} \right) \xleftarrow{\text{tr}_{\text{FIOP}}^{\text{SR}}} \text{FIOPSRGame}(s_{\text{FIOP}}, \text{rnd}_{\text{FIOP}}, \tilde{\mathbf{P}}^{\text{SR}}, \mathbf{ai}) \\ \mathbb{w} \leftarrow \mathbf{E}_{\text{SR}}^{\tilde{\mathbf{P}}^{\text{SR}}(\mathbf{ai})}(\mathbb{x}, (m_i)_{i \in [k_{\text{FIOP}}]}, (\gamma_i)_{i \in [k_{\text{FIOP}}]}, (\rho_i)_{i \in [k_{\text{FIOP}}]}, \text{tr}_{\text{FIOP}}^{\text{SR}}) \\ b \leftarrow \mathbf{V}(\mathbb{x}, (m_i)_{i \in [k_{\text{FIOP}}]}, (\rho_i)_{i \in [k_{\text{FIOP}}]}) \end{array} \right] \\ \leq \kappa_{\text{FIOP}}^{\text{SR}}(n, s_{\text{FIOP}}, \mathbf{m}_{\text{FIOP}}).$$

Moreover, $\mathbf{E}_{\text{SR}}$ runs in time $t_{\text{E}_{\text{SR}}}(n, s_{\text{FIOP}}, \mathbf{m}_{\text{FIOP}}, t_{\text{FIOP}})$.

We list the efficiency measures for an FIOP:

- the round complexity k_{FIOP} is the number of rounds (back-and-forth interactions) between the FIOP prover and FIOP verifier;
- the alphabet Σ is the set over which symbols of prover messages are defined (we assume the alphabet across all rounds);
- the proof length ℓ is the total number of alphabet symbols sent by the FIOP prover, ℓ_i is the length of the proof sent by the FIOP prover in round i , and $\ell_{\max} := \max_{i \in [k_{\text{FIOP}}]} \{\ell_i\}$;
- the query complexity q is the total number of queries that the FIOP verifier makes (each query is a element in $\mathbf{Q}_i$ and is answered by the corresponding evaluation in $\mathbb{D}$), q_i is the number of queries that the FIOP verifier makes to the i -th FIOP string, and $q_{\max} := \max_{i \in [k_{\text{FIOP}}]} \{q_i\}$;
- the randomness complexity r_{FIOP} is the total number of random bits used by the FIOP verifier, $r_{\text{FIOP},i}$ is the number of random bits that the FIOP verifier uses in round i , and $r_{\text{FIOP},\max} := \max_{i \in [k_{\text{FIOP}}]} \{r_{\text{FIOP},i}\}$.

Any efficiency measure may be a function of the instance $\mathbb{x}$ (e.g., of the instance size $|\mathbb{x}|$).

3.3 Functional commitment schemes

3.3.1 Non-interactive functional commitment schemes

A non-interactive functional commitment (FC) scheme for a query class $\mathbf{Q} = \{\alpha : \Sigma^\ell \rightarrow \mathbb{D}\}$ is a tuple of algorithms

$$\text{FC} = (\text{FC.Gen}, \text{FC.Commit}, \text{FC.Open}, \text{FC.Check})$$

that satisfies the following syntax.

- $\text{FC.Gen}(1^\lambda, \ell) \rightarrow \text{pp}_{\text{FC}}$: On input a security parameter $\lambda \in \mathbb{N}$ and message length $\ell \in \mathbb{N}$, FC.Gen samples public parameter pp_{FC} .
- $\text{FC.Commit}(\text{pp}_{\text{FC}}, \Pi) \rightarrow (\text{cm}, \text{aux})$: On input a public parameter pp_{FC} and a message vector $\Pi \in \Sigma^\ell$, FC.Commit produces a commitment cm and the corresponding auxiliary state aux .
- $\text{FC.Open}(\text{pp}_{\text{FC}}, \text{aux}, \alpha, \beta) \rightarrow \text{pf}$: On input a public parameter pp_{FC} , an auxiliary state aux , a query $\alpha \in \mathbf{Q}$ and a purported evaluation $\beta \in \mathbb{D}$, FC.Open outputs an opening proof pf for the claim “ $\beta = \alpha(\Pi)$ ”.
- $\text{FC.Check}(\text{pp}_{\text{FC}}, \text{cm}, \alpha, \beta, \text{pf}) \rightarrow \{0, 1\}$: On input a public parameter pp_{FC} , a commitment cm , a query $\alpha \in \mathbf{Q}$, an answer $\beta \in \mathbb{D}$, and an opening proof pf , FC.Check determines if pf is a valid proof for β being the answer of the query α on the vector committed in cm .

Definition 3.17 (Function binding). $\text{FC} = (\text{FC.Gen}, \text{FC.Commit}, \text{FC.Open}, \text{FC.Check})$ with query class $\mathbf{Q}$ has **function binding error** ϵ_{FC} if for every security parameter $\lambda \in \mathbb{N}$, message length $\ell \in \mathbb{N}$, sample set size $L \in \mathbb{N}$, auxiliary input distribution $\mathcal{D}_{\text{FC}}$, adversary size bound t_{FC} , and t_{FC} -size circuit A_{FC} ,

$$\Pr \left[\begin{array}{l} \forall i \in [L] : b_i = 1 \\ \wedge \nexists \Pi \text{ s.t. } \forall i \in [L], \alpha_i(\Pi) = \beta_i \end{array} \middle| \begin{array}{l} \text{pp}_{\text{FC}} \leftarrow \text{FC.Gen}(1^\lambda, \ell) \\ \text{ai}_{\text{FC}} \leftarrow \mathcal{D}_{\text{FC}} \\ (\text{cm}, ((\alpha_i, \beta_i, \text{pf}_i))_{i \in [L]}) \leftarrow A_{\text{FC}}(\text{pp}_{\text{FC}}, \text{ai}_{\text{FC}}) \\ \text{For } i \in [L]: \\ \quad b_i \leftarrow \text{FC.Check}(\text{pp}_{\text{FC}}, \text{cm}, \alpha_i, \beta_i, \text{pf}_i) \end{array} \right] \\ \leq \epsilon_{\text{FC}}(\lambda, \ell, L, t_{\text{FC}}) .$$

Remark 3.18 (Monotonicity of ϵ_{FC}). We assume hereafter that the function binding error ϵ_{FC} is monotone in each coordinate in the natural direction:

- $\epsilon_{\text{FC}}(\cdot, \ell, L, t_{\text{FC}})$ is non-increasing (larger security parameters decrease an adversary's success);
- $\epsilon_{\text{FC}}(\lambda, \cdot, L, t_{\text{FC}})$ is non-decreasing (longer message vector decrease an adversary's success);
- $\epsilon_{\text{FC}}(\lambda, \ell, \cdot, t_{\text{FC}})$ is non-decreasing (The solution space with a set of constraints is smaller than that of its subset); and
- $\epsilon_{\text{FC}}(\lambda, \ell, L, \cdot)$ is non-decreasing (the success of an adversary increases with its computational power).

The last condition is trivially satisfied, while the first should also hold in any reasonable commitment scheme. The remaining two are natural; in any case, otherwise one may replace, in our computations, expressions of the type $\epsilon_{\text{FC}}(\lambda, \ell_{\max}, L_{\max}, t_{\text{FC}})$, when $\ell_{\max} = \max_i \{ \ell_i \}$ and $L_{\max} = \max_j \{ L_j \}$, with

$$\max_{i,j} \{ \epsilon_{\text{FC}}(\lambda, \ell_i, L_j, t_{\text{FC}}) \} .$$

3.3.2 Interactive functional commitment schemes

An interactive functional commitment scheme $\text{FC} = (\text{FC.Gen}, \text{FC.Commit}, \mathcal{P}_{\text{FC}}, \mathcal{V}_{\text{FC}})$ is defined similarly. In particular, FC.Gen and FC.Commit are defined as in the non-interactive case, while the opening and verification phases follow the interactive protocol below: given an public parameter pp_{FC} , an instance $(\text{cm}, \alpha, \beta)$ and an auxiliary state aux , the FC prover $\mathcal{P}_{\text{FC}}$ and the FC verifier $\mathcal{V}_{\text{FC}}$ engage in a public-coin k_{FC} -round interactive protocol:

$$\langle \mathcal{P}_{\text{FC}}(\text{pp}_{\text{FC}}, \text{aux}, \alpha, \beta), \mathcal{V}_{\text{FC}}(\text{pp}_{\text{FC}}, \text{cm}, \alpha, \beta) \rangle \xrightarrow{((\text{pm}_i, \text{vm}_i))_{i \in [k_{\text{FC}}]}} b .$$

Above, $((\text{pm}_i, \text{vm}_i))_{i \in [k_{\text{FC}}]}$ is the transcript of the interaction (that is, the k_{FC} -round of prover messages and the verifier randomness exchanged between $\mathcal{P}_{\text{FC}}$ and $\mathcal{V}_{\text{FC}}$). For every $j \in [k_{\text{FC}}]$:

- $\mathcal{P}_{\text{FC}}$'s j -th message.
 1. Compute the j -th FC prover message pm_j and auxiliary state:

$$(\text{pm}_j, \text{aux}_j) \leftarrow \begin{cases} \mathcal{P}_{\text{FC}}(\text{pp}_{\text{FC}}, \text{aux}, \alpha, \beta) & \text{if } j = 1 \\ \mathcal{P}_{\text{FC}}(\text{aux}_{j-1}, \text{vm}_{j-1}) & \text{if } j > 1 \end{cases} .$$

2. Send pm_j to $\mathcal{V}_{\text{FC}}$.

- $\mathcal{V}_{\text{FC}}$'s j -th message.
 1. Sample the j -th FC verifier randomness $\text{vm}_j \leftarrow \{0, 1\}^{\text{r}_{\text{FC},j}}$.
 2. Send vm_j to $\mathcal{P}_{\text{FC}}$.

Definition 3.19 (FC state-restoration game). *The FC state-restoration game for FC with salt size $s_{\text{FC}} \in \mathbb{N}$, functions $\text{rnd}_{\text{FC}} = (\text{rnd}_{\text{FC},i})_{i \in [k_{\text{FC}}]}$ where $\text{rnd}_{\text{FC},i} \leftarrow \mathcal{U}(\text{r}_{\text{FC},i})$ for every $i \in [k_{\text{FC}}]$, public parameter pp_{FC} , auxiliary input ai_{FC} , sample size $L \in \mathbb{N}$, and FC state-restoration prover $\tilde{\mathcal{P}}_{\text{FC}}^{\text{SR}}$ is defined below.*

$\text{FCSRGame}(s_{\text{FC}}, \text{rnd}_{\text{FC}}, \tilde{\mathcal{P}}_{\text{FC}}^{\text{SR}}, \text{pp}_{\text{FC}}, \text{ai}_{\text{FC}})$:

1. Repeat the following until $\tilde{\mathcal{P}}_{\text{FC}}^{\text{SR}}(\text{pp}_{\text{FC}}, \text{ai}_{\text{FC}})$ decides to exit the loop:
 - (a) $\tilde{\mathcal{P}}_{\text{FC}}^{\text{SR}}$ outputs $(\text{cm}, \alpha, \beta, (\text{pm}_j)_{j \in [i]}, (\eta_j)_{j \in [i]})$, where cm is a commitment, α is a query in $\mathbf{Q}$, β is a corresponding answer, $(\text{pm}_j)_{j \in [i]}$ are prover messages, and $(\eta_j)_{j \in [i]}$ are salt strings in $\{0, 1\}^{s_{\text{FC}}}$.

- (b) Set $\text{vm}_i := \text{rnd}_{\text{FC},i}(\text{cm}, \alpha, \beta, (\text{pm}_j)_{j \in [i]}, (\eta_j)_{j \in [i]})$.
- (c) Send vm_i to $\tilde{\mathcal{P}}_{\text{FC}}^{\text{SR}}$.
2. $\tilde{\mathcal{P}}_{\text{FC}}^{\text{SR}}$ outputs $(\text{cm}, ((\alpha_i, \beta_i, (\text{pm}_{i,j})_{j \in [\text{k}_{\text{FC}}]}, (\eta_{i,j})_{j \in [\text{k}_{\text{FC}}]}))_{i \in [L]})$.
3. For every $i \in [L], j \in [\text{k}_{\text{FC}}]$, set $\text{vm}_{i,j} := \text{rnd}_{\text{FC},j}(\text{cm}, \alpha_i, \beta_i, (\text{pm}_{i,k})_{k \in [j]}, (\eta_{i,k})_{k \in [j]})$.
4. Output $(\text{cm}, ((\alpha_i, \beta_i, ((\text{pm}_{i,j}, \eta_{i,j}, \text{vm}_{i,j}))_{j \in [\text{k}_{\text{FC}}]}))_{i \in [L]})$.

We denote by $\text{tr}_{\text{FC}}^{\text{SR}}$ the list of move-response pairs of the form $(\text{cm}, \alpha, \beta, (\text{pm}_j)_{j \in [i]}, (\eta_j)_{j \in [i]})$ performed in the loop. We show $\text{tr}_{\text{FC}}^{\text{SR}}$ in an execution of the FC state-restoration game FCStateRestGame using the following notation:

$$(\text{cm}, ((\alpha_i, \beta_i, ((\text{pm}_{i,j}, \eta_{i,j}, \text{vm}_{i,j}))_{j \in [\text{k}_{\text{FC}}]}))_{i \in [L]}) \xleftarrow{\text{tr}_{\text{FC}}^{\text{SR}}} \text{FCStateRestGame}(s_{\text{FC}}, \text{rnd}_{\text{FC}}, \tilde{\mathcal{P}}_{\text{FC}}^{\text{SR}}, \text{pp}_{\text{FC}}, \text{ai}_{\text{FC}}) .$$

We say that $\tilde{\mathcal{P}}_{\text{FC}}^{\text{SR}}$ is m_{FC} -move if $\tilde{\mathcal{P}}_{\text{FC}}^{\text{SR}}$ exits the loop after at most m_{FC} iterations.

Definition 3.20 (State-restoration function binding). An k_{FC} -round interactive FC with query class $\mathbf{Q}$ has **state-restoration function binding error** $\epsilon_{\text{FC}}^{\text{SR}}$ if for every security parameter $\lambda \in \mathbb{N}$, message length $\ell \in \mathbb{N}$, sample set size $L \in \mathbb{N}$, auxiliary input distribution $\mathcal{D}_{\text{FC}}$, salt size $s_{\text{FC}} \in \mathbb{N}$, move budget $\text{m}_{\text{FC}} \in \mathbb{N}$, circuit size bound $t_{\text{FC}} \in \mathbb{N}$, and m_{FC} -move t_{FC} -size circuit $\tilde{\mathcal{P}}_{\text{FC}}^{\text{SR}}$,

$$\Pr \left[\begin{array}{l} \forall i \in [L], b_i = 1 \\ \wedge \nexists \Pi \text{ s.t. } \forall i \in [L], \alpha_i(\Pi) = \beta_i \end{array} \middle| \begin{array}{l} \text{pp}_{\text{FC}} \leftarrow \text{FC.Gen}(1^\lambda, \ell) \\ \text{ai}_{\text{FC}} \leftarrow \mathcal{D}_{\text{FC}} \\ \text{rnd}_{\text{FC}} := (\text{rnd}_{\text{FC},i})_{i \in [\text{k}_{\text{FC}}]} \leftarrow \mathcal{U}((r_{\text{FC},i})_{i \in [\text{k}_{\text{FC}}]}) \\ \left(\text{cm}, \left(\left(\begin{array}{l} \alpha_i, \beta_i, \\ ((\text{pm}_{i,j}, \eta_{i,j}, \text{vm}_{i,j}))_{j \in [\text{k}_{\text{FC}}]} \end{array} \right) \right)_{i \in [L]} \right) \\ \leftarrow \text{FCStateRestGame}(s_{\text{FC}}, \text{rnd}_{\text{FC}}, \tilde{\mathcal{P}}_{\text{FC}}^{\text{SR}}, \text{pp}_{\text{FC}}, \text{ai}_{\text{FC}}) \\ \text{For } i \in [L]: \\ b_i \leftarrow \mathcal{V}_{\text{FC}}(\text{pp}_{\text{FC}}, \text{cm}, \alpha_i, \beta_i, ((\text{pm}_{i,j}, \text{vm}_{i,j}))_{j \in [\text{k}_{\text{FC}}]}) \end{array} \right] \\ \leq \epsilon_{\text{FC}}^{\text{SR}}(\lambda, \ell, L, s_{\text{FC}}, \text{m}_{\text{FC}}, t_{\text{FC}}) .$$

3.3.3 Batched functional commitment schemes

Definition 3.21 (Batched query class). Let $\mathbf{Q} = \{\alpha : \Sigma^\ell \rightarrow \mathbb{D}\}$ be a query class, and let $m \in \mathbb{N}$. The batched query class $\text{Batch}[\mathbf{Q}, m]$ is defined as follows:

$$\text{Batch}[\mathbf{Q}, m] := \left\{ \alpha' : \Sigma^\ell \rightarrow \mathbb{D}^m \mid \exists \alpha_1, \dots, \alpha_m \in \mathbf{Q} \text{ s.t. } \alpha'(\Pi) = (\alpha_i(\Pi))_{i \in [m]} \right\} .$$

Non-interactive batched functional commitment schemes. A non-interactive batched FC scheme $\text{bFC} = (\text{bFC.Gen}, \text{bFC.Commit}, \text{bFC.Open}, \text{bFC.Check})$ for a query class $\mathbf{Q} = \{\alpha : \Sigma^\ell \rightarrow \mathbb{D}\}$ is a non-interactive FC scheme for $\text{Batch}[\mathbf{Q}, m]$ (for some $m \in \mathbb{N}$) that satisfies the following syntax.

- $\text{bFC.Gen}(1^\lambda, \ell) \rightarrow \text{pp}_{\text{bFC}}$: On input a security parameter $\lambda \in \mathbb{N}$ and message length $\ell \in \mathbb{N}$, bFC.Gen samples public parameter pp_{bFC} .
- $\text{bFC.Commit}(\text{pp}_{\text{bFC}}, \Pi) \rightarrow (\text{cm}, \text{aux})$: On input a public parameter pp_{bFC} and a message vector $\Pi \in \Sigma^\ell$, bFC.Commit produces a commitment cm and the corresponding auxiliary state aux .

- $\text{bFC.Open}(\text{pp}_{\text{bFC}}, \text{aux}, \mathcal{Q}, \beta) \rightarrow \text{pf}$: On input a public parameter pp_{bFC} , an auxiliary state aux , a query set $\mathcal{Q} \subseteq \mathbf{Q}$ and a corresponding answer set $\beta \in \mathbb{D}^{|\mathcal{Q}|}$, bFC.Open outputs an opening proof pf for the claim “ $\forall \alpha \in \mathcal{Q}, \beta^{(\alpha)} = \alpha(\Pi)$ ”.
- $\text{bFC.Check}(\text{pp}_{\text{bFC}}, \text{cm}, \mathcal{Q}, \beta, \text{pf}) \rightarrow \{0, 1\}$: On input a public parameter pp_{bFC} , a commitment cm , a query set $\mathcal{Q} \subseteq \mathbf{Q}$, an answer set $\beta \in \mathbb{D}^{|\mathcal{Q}|}$, and an opening proof pf , bFC.Check determines if pf is a valid proof for β being the answer of the query set $\mathcal{Q}$ on the vector committed in cm .

Definition 3.22 ((Batched) function binding). $\text{bFC} = (\text{bFC.Gen}, \text{bFC.Commit}, \text{bFC.Open}, \text{bFC.Check})$ with query class $\mathbf{Q}$ has **function binding error** ϵ_{bFC} if for every security parameter $\lambda \in \mathbb{N}$, message length $\ell \in \mathbb{N}$, query set size $m \in \mathbb{N}$, sample set size $L \in \mathbb{N}$, auxiliary input distribution $\mathcal{D}_{\text{bFC}}$, adversary size bound t_{bFC} , and t_{bFC} -size circuit A_{bFC} ,

$$\Pr \left[\begin{array}{l} \forall i \in [L], |\mathcal{Q}_i| = m \wedge b_i = 1 \\ \wedge \nexists \Pi \text{ s.t. } \forall i \in [L], \alpha \in \mathcal{Q}_i, \alpha(\Pi) = \beta_i^{(\alpha)} \end{array} \mid \begin{array}{l} \text{pp}_{\text{bFC}} \leftarrow \text{bFC.Gen}(1^\lambda, \ell) \\ \text{ai}_{\text{bFC}} \leftarrow \mathcal{D}_{\text{bFC}} \\ (\text{cm}, ((\mathcal{Q}_i, \beta_i, \text{pf}_i))_{i \in [L]}) \leftarrow A_{\text{bFC}}(\text{pp}_{\text{bFC}}, \text{ai}_{\text{bFC}}) \\ \text{For } i \in [L]: \\ \quad b_i \leftarrow \text{bFC.Check}(\text{pp}_{\text{bFC}}, \text{cm}, \mathcal{Q}_i, \beta_i, \text{pf}_i) \end{array} \right] \leq \epsilon_{\text{bFC}}(\lambda, \ell, m, L, t_{\text{bFC}}) .$$

Remark 3.23 (Monotonicity of ϵ_{bFC}). We assume hereafter that the (batched) function binding error ϵ_{bFC} is monotone in each coordinate in the natural direction:

- $\epsilon_{\text{bFC}}(\cdot, \ell, m, L, t_{\text{bFC}})$ is non-increasing (larger security parameters decrease an adversary's success);
- $\epsilon_{\text{bFC}}(\lambda, \cdot, m, L, t_{\text{bFC}})$ is non-decreasing (longer message vector decrease an adversary's success);
- $\epsilon_{\text{bFC}}(\lambda, \ell, \cdot, L, t_{\text{bFC}})$ is non-decreasing (The solution space with a set of constraints is smaller than that of its subset);
- $\epsilon_{\text{bFC}}(\lambda, \ell, m, \cdot, t_{\text{bFC}})$ is non-decreasing (The solution space with a set of constraints is smaller than that of its subset); and
- $\epsilon_{\text{bFC}}(\lambda, \ell, m, L, \cdot)$ is non-decreasing (the success of an adversary increases with its computational power).

The last condition is trivially satisfied, while the first should also hold in any reasonable commitment scheme. The remaining three are natural; in any case, otherwise one may replace, in our computations, expressions of the type $\epsilon_{\text{bFC}}(\lambda, \ell_{\max}, m_{\max}, L_{\max}, t_{\text{bFC}})$, when $\ell_{\max} = \max_i \{ \ell_i \}$, $m_{\max} = \max_j \{ m_j \}$ and $L_{\max} = \max_k \{ L_k \}$, with

$$\max_{i,j,k} \{ \epsilon_{\text{bFC}}(\lambda, \ell_i, m_j, L_k, t_{\text{bFC}}) \} .$$

Interactive batched functional commitment schemes. An interactive batched FC scheme $\text{bFC} = (\text{bFC.Gen}, \text{bFC.Commit}, \mathcal{P}_{\text{bFC}}, \mathcal{V}_{\text{bFC}})$ for a query class $\mathbf{Q} = \{ \alpha: \Sigma^\ell \rightarrow \mathbb{D} \}$ is an interactive FC scheme for $\text{Batch}[\mathbf{Q}, m]$. In particular, bFC.Gen and bFC.Commit are defined to be the same as the ones for non-interactive batched FC scheme, while the opening and verification phases follow the interactive protocol below: given an public parameter pp_{bFC} , an instance $(\text{cm}, \mathcal{Q}, \beta)$ and an auxiliary state aux , the prover and verifier engage in a public-coin k_{bFC} -round interactive protocol:

$$\langle \mathcal{P}_{\text{bFC}}(\text{pp}_{\text{bFC}}, \text{aux}, \mathcal{Q}, \beta), \mathcal{V}_{\text{bFC}}(\text{pp}_{\text{bFC}}, \text{cm}, \mathcal{Q}, \beta) \rangle \xrightarrow{((\text{pm}_i, \text{vm}_i))_{i \in [k_{\text{bFC}}]}} b .$$

Above, $((\text{pm}_i, \text{vm}_i))_{i \in [k_{\text{bFC}}]}$ is the transcript of the interaction (that is, the k_{bFC} -round of prover messages and the verifier randomness exchanged between $\mathcal{P}_{\text{bFC}}$ and $\mathcal{V}_{\text{bFC}}$). For every $j \in [k_{\text{bFC}}]$:

- $\mathcal{P}_{\text{bFC}}$'s j -th message.

1. Compute the j -th FC prover message pm_j and auxiliary state:

$$(\text{pm}_j, \text{aux}_j) \leftarrow \begin{cases} \mathcal{P}_{\text{bFC}}(\text{pp}_{\text{bFC}}, \text{aux}, \mathcal{Q}, \beta) & \text{if } j = 1 \\ \mathcal{P}_{\text{bFC}}(\text{aux}_{j-1}, \text{vm}_{j-1}) & \text{if } j > 1 \end{cases}.$$

2. Send pm_j to $\mathcal{V}_{\text{bFC}}$.

- $\mathcal{V}_{\text{bFC}}$'s j -th message.

1. Sample the j -th FC verifier randomness $\text{vm}_j \leftarrow \{0, 1\}^{r_{\text{bFC},j}}$.

2. Send vm_j to $\mathcal{P}_{\text{bFC}}$.

Definition 3.24 ((Batched) FC state-restoration game). *The FC state-restoration game for bFC with salt size $s_{\text{bFC}} \in \mathbb{N}$, functions $\text{rnd}_{\text{bFC}} = (\text{rnd}_{\text{bFC},i})_{i \in [k_{\text{bFC}}]}$ where $\text{rnd}_{\text{bFC},i} \leftarrow \mathcal{U}(r_{\text{bFC},i})$ for every $i \in [k_{\text{bFC}}]$, public parameter pp_{bFC} , auxiliary input ai_{bFC} , sample size $L \in \mathbb{N}$, and FC state-restoration prover $\tilde{\mathcal{P}}_{\text{bFC}}^{\text{SR}}$ is defined below.*

$\text{bFCsRGame}(s_{\text{bFC}}, \text{rnd}_{\text{bFC}}, \tilde{\mathcal{P}}_{\text{bFC}}^{\text{SR}}, \text{pp}_{\text{bFC}}, \text{ai}_{\text{bFC}})$:

1. Repeat the following until $\tilde{\mathcal{P}}_{\text{bFC}}^{\text{SR}}(\text{pp}_{\text{bFC}}, \text{ai}_{\text{bFC}})$ decides to exit the loop:

(a) $\tilde{\mathcal{P}}_{\text{bFC}}^{\text{SR}}$ outputs $(\text{cm}, \mathcal{Q}, \beta, (\text{pm}_j)_{j \in [i]}, (\eta_j)_{j \in [i]})$, where cm is a commitment, $\mathcal{Q}$ is a query set from $\mathbf{Q}$, β is the claimed answer, $(\text{pm}_j)_{j \in [i]}$ are prover messages, and $(\eta_j)_{j \in [i]}$ are salt strings in $\{0, 1\}^{s_{\text{bFC}}}$.

(b) Set $\text{vm}_i := \text{rnd}_{\text{bFC},i}(\text{cm}, \mathcal{Q}, \beta, (\text{pm}_j)_{j \in [i]}, (\eta_j)_{j \in [i]})$.

(c) Send vm_i to $\tilde{\mathcal{P}}_{\text{bFC}}^{\text{SR}}$.

2. $\tilde{\mathcal{P}}_{\text{bFC}}^{\text{SR}}$ outputs $(\text{cm}, ((\mathcal{Q}_i, \beta_i, (\text{pm}_{i,j})_{j \in [k_{\text{bFC}}]}, (\eta_{i,j})_{j \in [k_{\text{bFC}}]}))_{i \in [L]})$.

3. For every $i \in [L]$, $j \in [k_{\text{bFC}}]$, set $\text{vm}_{i,j} := \text{rnd}_{\text{bFC},j}(\text{cm}, \mathcal{Q}_i, \beta_i, (\text{pm}_{i,k})_{k \in [j]}, (\eta_{i,k})_{k \in [j]})$.

4. Output $(\text{cm}, ((\mathcal{Q}_i, \beta_i, ((\text{pm}_{i,j}, \eta_{i,j}, \text{vm}_{i,j}))_{j \in [k_{\text{bFC}}]}))_{i \in [L]})$.

We denote by $\text{tr}_{\text{bFC}}^{\text{SR}}$ the list of move-response pairs of the form $(\text{cm}, \mathcal{Q}, \beta, (\text{pm}_j)_{j \in [i]}, (\eta_j)_{j \in [i]})$ performed in the loop. We show $\text{tr}_{\text{bFC}}^{\text{SR}}$ in an execution of the FC state-restoration game FCSRGame using the following notation:

$$(\text{cm}, ((\mathcal{Q}_i, \beta_i, ((\text{pm}_{i,j}, \eta_{i,j}, \text{vm}_{i,j}))_{j \in [k_{\text{bFC}}]}))_{i \in [L]}) \leftarrow \text{tr}_{\text{bFC}}^{\text{SR}} \text{bFCsRGame}(s_{\text{bFC}}, \text{rnd}_{\text{bFC}}, \tilde{\mathcal{P}}_{\text{bFC}}^{\text{SR}}, \text{pp}_{\text{bFC}}, \text{ai}_{\text{bFC}}).$$

We say that $\tilde{\mathcal{P}}_{\text{bFC}}^{\text{SR}}$ is m_{bFC} -move if $\tilde{\mathcal{P}}_{\text{bFC}}^{\text{SR}}$ exits the loop after at most m_{bFC} iterations.

Definition 3.25 ((Batched) state-restoration function binding). *An k_{bFC} -round interactive batched FCS bFC with query class $\mathbf{Q}$ has state-restoration function binding error $\epsilon_{\text{bFC}}^{\text{SR}}$ if for every security parameter $\lambda \in \mathbb{N}$, message length $\ell \in \mathbb{N}$, query set size $m \in \mathbb{N}$, sample set size $L \in \mathbb{N}$, auxiliary input distribution $\mathcal{D}_{\text{bFC}}$, salt size $s_{\text{bFC}} \in \mathbb{N}$, move budget $m_{\text{bFC}} \in \mathbb{N}$, circuit size bound $t_{\text{bFC}} \in \mathbb{N}$, and m_{bFC} -move t_{bFC} -size circuit $\tilde{\mathcal{P}}_{\text{bFC}}^{\text{SR}}$,*

$$\Pr \left[\begin{array}{l} (\forall i \in [L], |\mathcal{Q}_i| = m \wedge b_i = 1) \\ \wedge \nexists \Pi \text{ s.t. } \forall i \in [L], \alpha \in \mathcal{Q}_i, \alpha(\Pi) = \beta_i^{(\alpha)} \end{array} \middle| \begin{array}{l} \text{pp}_{\text{bFC}} \leftarrow \text{bFC.Gen}(1^\lambda, \ell) \\ \text{ai}_{\text{bFC}} \leftarrow \mathcal{D}_{\text{bFC}} \\ \text{rnd}_{\text{bFC}} := (\text{rnd}_{\text{bFC},i})_{i \in [k_{\text{bFC}}]} \leftarrow \mathcal{U}((r_{\text{bFC},i})_{i \in [k_{\text{bFC}}]}) \\ \left(\text{cm}, \left(\left(\mathcal{Q}_i, \beta_i, \right. \right. \right. \\ \left. \left. \left. ((\text{pm}_{i,j}, \eta_{i,j}, \text{vm}_{i,j}))_{j \in [k_{\text{bFC}}]} \right) \right)_{i \in [L]} \right) \\ \leftarrow \text{bFCsRGame}(s_{\text{bFC}}, \text{rnd}_{\text{bFC}}, \tilde{\mathcal{P}}_{\text{bFC}}^{\text{SR}}, \text{pp}_{\text{bFC}}, \text{ai}_{\text{bFC}}) \\ \text{For } i \in [L]: \\ b_i \leftarrow \mathcal{V}_{\text{bFC}}(\text{pp}_{\text{bFC}}, \text{cm}, \mathcal{Q}_i, \beta_i, ((\text{pm}_{i,j}, \text{vm}_{i,j}))_{j \in [k_{\text{bFC}}]}) \end{array} \right] \\ \leq \epsilon_{\text{bFC}}^{\text{SR}}(\lambda, \ell, m, L, s_{\text{bFC}}, m_{\text{bFC}}, t_{\text{bFC}}).$$

3.3.4 From FC to batched FC

Given a FC scheme for a query class $\mathbf{Q}$, one can construct a batched FC scheme for $\mathbf{Q}$ by naively applying bFC to each query in the query set.

Construction 3.26 (Non-interactive batched FC). Let FC be a non-interactive functional commitment for the query class $\mathbf{Q}$. We construct a functional commitment bFC for $\mathbf{Q}$ as follows:

- $\text{bFC.Gen}(1^\lambda, \ell)$: Output $\text{pp}_{\text{bFC}} \leftarrow \text{FC.Gen}(1^\lambda, \ell)$.
- $\text{bFC.Commit}(\text{pp}_{\text{bFC}}, \Pi)$: Output $(\text{cm}, \text{aux}) \leftarrow \text{FC.Commit}(\text{pp}_{\text{bFC}}, \Pi)$.
- $\text{bFC.Open}(\text{pp}_{\text{bFC}}, \text{aux}, \mathcal{Q}, \beta)$:
 1. Parse $\mathcal{Q}$ as $(\alpha_i)_{i \in [|\mathcal{Q}|]}$ and β as $(\beta_i)_{i \in [|\mathcal{Q}|]}$
 2. For $i \in [|\mathcal{Q}|]$: compute $\text{pf}_i \leftarrow \text{FC.Open}(\text{pp}_{\text{bFC}}, \text{aux}, \alpha_i, \beta_i)$.
 3. Output $(\text{pf}_i)_{i \in [|\mathcal{Q}|]}$.
- $\text{bFC.Check}(\text{pp}_{\text{bFC}}, \text{cm}, \mathcal{Q}, \beta, \text{pf})$:
 1. Parse $\mathcal{Q}$ as $(\alpha_i)_{i \in [|\mathcal{Q}|]}$, β as $(\beta_i)_{i \in [|\mathcal{Q}|]}$ and pf as $(\text{pf}_i)_{i \in [|\mathcal{Q}|]}$.
 2. For $i \in [|\mathcal{Q}|]$, compute $b_i \leftarrow \text{FC.Check}(\text{pp}_{\text{bFC}}, \text{cm}, \alpha_i, \beta_i, \text{pf}_i)$.
 3. If $\forall i \in [|\mathcal{Q}|], b_i = 1$, output 1; otherwise, output 0.

Lemma 3.27. Let FC be a non-interactive FC scheme for the query class $\mathbf{Q}$ with function binding error $\epsilon_{\text{FC}} = \epsilon_{\text{FC}}(\lambda, \ell, L, t_{\text{FC}})$. Let bFC be defined as in Construction 3.26. Then, bFC has batched function binding error $\epsilon_{\text{bFC}} = \epsilon_{\text{bFC}}(\lambda, \ell, m, L, t_{\text{bFC}})$ such that

$$\epsilon_{\text{bFC}}(\lambda, \ell, m, L, t_{\text{bFC}}) \leq \epsilon_{\text{FC}}(\lambda, \ell, m \cdot L, t_{\text{FC}}) ,$$

where $t_{\text{FC}} \leq t_{\text{bFC}} + m \cdot L$.

Proof. Let A_{bFC} be an adversary for bFC. We construct an adversary A_{FC} for FC as follows:

$A_{\text{FC}}(\text{pp}_{\text{FC}}, \text{ai}_{\text{FC}})$:

1. Run $(\text{cm}, ((\mathcal{Q}_i, \beta_i, \text{pf}_i))_{i \in [L]}) \leftarrow A_{\text{bFC}}(\text{pp}_{\text{FC}}, \text{ai}_{\text{FC}})$.
2. For every $i \in [L]$, parse $\mathcal{Q}_i$ as $(\alpha_{i,j})_{j \in [m]}$, β_i as $(\beta_{i,j})_{j \in [m]}$ and pf_i as $(\text{pf}_{i,j})_{j \in [m]}$.
3. Output $(\text{cm}, ((\alpha_{i,j}, \beta_{i,j}, \text{pf}_{i,j}))_{i \in [L], j \in [m]})$.

Note that A_{FC} has size $t_{\text{FC}} \leq t_{\text{bFC}} + m \cdot L$.

Moreover,

$$\begin{aligned} & \Pr \left[\begin{array}{l} \forall i \in [L], |\mathcal{Q}_i| = m \wedge b_i = 1 \\ \wedge \nexists \Pi \text{ s.t. } \forall i \in [L], \alpha \in \mathcal{Q}_i, \alpha(\Pi) = \beta_i^{(\alpha)} \end{array} \right] \left[\begin{array}{l} \text{pp}_{\text{bFC}} \leftarrow \text{bFC.Gen}(1^\lambda, \ell) \\ \text{ai}_{\text{bFC}} \leftarrow \mathcal{D}_{\text{bFC}} \\ (\text{cm}, ((\mathcal{Q}_i, \beta_i, \text{pf}_i))_{i \in [L]}) \leftarrow A_{\text{bFC}}(\text{pp}_{\text{bFC}}, \text{ai}_{\text{bFC}}) \\ \text{For } i \in [L]: \\ \quad b_i \leftarrow \text{bFC.Check}(\text{pp}_{\text{bFC}}, \text{cm}, \mathcal{Q}_i, \beta_i, \text{pf}_i) \end{array} \right] \\ & \leq \Pr \left[\begin{array}{l} (\forall i \in [L], j \in [m], b_{i,j} = 1) \\ \wedge \nexists \Pi \text{ s.t. } \forall i \in [L], j \in [m], \alpha_{i,j}(\Pi) = \beta_{i,j} \end{array} \right] \left[\begin{array}{l} \text{pp}_{\text{FC}} \leftarrow \text{FC.Gen}(1^\lambda, \ell) \\ \text{ai}_{\text{FC}} \leftarrow \mathcal{D}_{\text{FC}} \\ (\text{cm}, ((\alpha_{i,j}, \beta_{i,j}, \text{pf}_{i,j}))_{i \in [L], j \in [m]}) \leftarrow A_{\text{FC}}(\text{pp}_{\text{FC}}, \text{ai}_{\text{FC}}) \\ \text{For } i \in [L], j \in [m]: \\ \quad b_{i,j} \leftarrow \text{FC.Check}(\text{pp}_{\text{FC}}, \text{cm}, \alpha_{i,j}, \beta_{i,j}, \text{pf}_{i,j}) \end{array} \right] \end{aligned}$$

$$\leq \epsilon_{\text{FC}}(\lambda, \ell, m \cdot L, t_{\text{FC}}) .$$

□

Construction 3.28 (Interactive batched FC). Let FC be an interactive functional commitment for the query class $\mathbf{Q}$. We construct a functional commitment bFC for $\mathbf{Q}$ as follows:

- $\text{bFC.Gen}(1^\lambda, \ell)$: Output $\text{pp}_{\text{bFC}} \leftarrow \text{FC.Gen}(1^\lambda, \ell)$.
- $\text{bFC.Commit}(\text{pp}_{\text{bFC}}, \Pi)$: Output $(\text{cm}, \text{aux}) \leftarrow \text{FC.Commit}(\text{pp}_{\text{bFC}}, \Pi)$.
- $\langle \mathcal{P}_{\text{bFC}}(\text{pp}_{\text{bFC}}, \text{aux}, \mathcal{Q}, \beta), \mathcal{V}_{\text{bFC}}(\text{pp}_{\text{bFC}}, \text{cm}, \mathcal{Q}, \beta) \rangle$:
 1. Both $\mathcal{P}_{\text{bFC}}$ and $\mathcal{V}_{\text{bFC}}$ parse $\mathcal{Q}$ as $(\alpha_i)_{i \in [|\mathcal{Q}|]}$ and β as $(\beta_i)_{i \in [|\mathcal{Q}|]}$.
 2. For $j \in [k_{\text{bFC}}]$:
 - (a) $\mathcal{P}_{\text{bFC}}$'s j -th message.
 - i. For $i \in [|\mathcal{Q}|]$, compute

$$(\text{pm}_{i,j}, \text{aux}_{i,j}) \leftarrow \begin{cases} \mathcal{P}_{\text{FC}}(\text{pp}_{\text{bFC}}, \text{aux}, \alpha_i, \beta_i) & \text{if } j = 1 \\ \mathcal{P}_{\text{FC}}(\text{aux}_{i,j-1}, \text{vm}_{i,j-1}) & \text{if } j > 1 \end{cases} .$$

- ii. Send $(\text{pm}_{i,j})_{i \in [|\mathcal{Q}|]}$ to $\mathcal{V}_{\text{bFC}}$.
- (b) $\mathcal{V}_{\text{bFC}}$'s j -th message.
 - i. For $i \in [|\mathcal{Q}|]$, sample $\text{vm}_{i,j} \leftarrow \{0, 1\}^{r_{\text{bFC},j}}$.
 - ii. Send $(\text{vm}_{i,j})_{i \in [|\mathcal{Q}|]}$ to $\mathcal{P}_{\text{bFC}}$.
3. For $i \in [|\mathcal{Q}|]$, compute $b_i = \mathcal{V}_{\text{FC}}(\text{pp}_{\text{bFC}}, \text{cm}, \alpha_i, \beta_i, ((\text{pm}_{i,j}, \text{vm}_{i,j}))_{j \in [k_{\text{bFC}}]})$.
4. If $\forall i \in [|\mathcal{Q}|], b_i = 1$, output 1; otherwise, output 0.

Lemma 3.29. Let FC be an interactive FC scheme for query class $\mathbf{Q}$ with state-restoration function binding error $\epsilon_{\text{FC}}^{\text{SR}} = \epsilon_{\text{FC}}^{\text{SR}}(\lambda, \ell, L, s_{\text{FC}}, m_{\text{FC}}, t_{\text{FC}})$. Let bFC be defined as in Construction 3.28. Then, bFC has batched state-restoration function binding error $\epsilon_{\text{bFC}}^{\text{SR}} = \epsilon_{\text{bFC}}^{\text{SR}}(\lambda, \ell, m, L, s_{\text{bFC}}, m_{\text{bFC}}, t_{\text{bFC}})$ such that

$$\epsilon_{\text{bFC}}^{\text{SR}}(\lambda, \ell, m, L, s_{\text{bFC}}, m_{\text{bFC}}, t_{\text{bFC}}) \leq \epsilon_{\text{FC}}^{\text{SR}}(\lambda, \ell, m \cdot L, \frac{s_{\text{bFC}}}{m}, m \cdot m_{\text{bFC}}, t_{\text{FC}}) ,$$

where $t_{\text{FC}} \leq t_{\text{bFC}} + m \cdot L$.

Proof. Let $\tilde{\mathcal{P}}_{\text{bFC}}^{\text{SR}}$ be a state-restoration adversary for bFC. We construct a state-restoration adversary $\tilde{\mathcal{P}}_{\text{FC}}^{\text{SR}}$ for FC as follows:

- $\tilde{\mathcal{P}}_{\text{FC}}^{\text{SR}}(\text{pp}_{\text{FC}}, \text{ai}_{\text{FC}})$:
1. Simulate the batched FC state-restoration game bFC SRGame with $\tilde{\mathcal{P}}_{\text{bFC}}^{\text{SR}}(\text{pp}_{\text{FC}}, \text{ai}_{\text{FC}})$ as follows:
 - (a) Get $(\text{cm}, \mathcal{Q}, \beta, (\text{pm}_j)_{j \in [i]}, (\eta_j)_{j \in [i]})$ from $\tilde{\mathcal{P}}_{\text{bFC}}^{\text{SR}}$.
 - (b) Parse $\mathcal{Q}$ as $(\alpha_u)_{u \in [m]}$ and β as $(\beta_u)_{u \in [m]}$.
 - (c) For $j \in [i]$, parse pm_j as $(\text{pm}_{u,j})_{u \in [m]}$ and η_j as $(\eta_{u,j})_{u \in [m]}$.
 - (d) For $u \in [m]$, output $(\text{cm}, \alpha_u, \beta_u, (\text{pm}_{u,j})_{j \in [i]}, (\eta_{u,j})_{j \in [i]})$ and receive $\text{vm}_{u,i}$.
 - (e) Send $(\text{vm}_{u,i})_{u \in [m]}$ to $\tilde{\mathcal{P}}_{\text{bFC}}^{\text{SR}}$.
 2. Get $(\text{cm}, ((\mathcal{Q}_i, \beta_i, (\text{pm}_{i,j})_{j \in [k_{\text{bFC}}]}, (\eta_{i,j})_{j \in [k_{\text{bFC}}]}))_{i \in [L]})$ from $\tilde{\mathcal{P}}_{\text{bFC}}^{\text{SR}}$.
 3. Parse $\mathcal{Q}_i$ as $(\alpha_{u,i})_{u \in [m]}$, β_i as $(\beta_{u,i})_{u \in [m]}$, $\text{pm}_{i,j}$ as $(\text{pm}_{u,i,j})_{u \in [m]}$, and $\eta_{i,j}$ as $(\eta_{u,i,j})_{u \in [m]}$.

4. Output $(\text{cm}, ((\alpha_{u,i}, \beta_{u,i}, (\text{pm}_{u,i,j})_{j \in [k_{\text{bFC}}]}, (\eta_{u,i,j})_{j \in [k_{\text{bFC}}]}))_{i \in [L], u \in [m]})$.

Note that $\tilde{\mathcal{P}}_{\text{FC}}^{\text{SR}}$ has size $t_{\text{FC}} \leq t_{\text{bFC}} + m \cdot L$, salt size $s_{\text{FC}} \leq \frac{s_{\text{bFC}}}{m}$, and move budget $m_{\text{FC}} \leq m \cdot m_{\text{bFC}}$.

Moreover,

$$\begin{aligned}
& \Pr \left[\begin{array}{l} (\forall i \in [L] : |\mathcal{Q}_i| = m \wedge b_i = 1) \\ \wedge \nexists \Pi \text{ s.t. } \forall i \in [L], \alpha \in \mathcal{Q}_i, \alpha(\Pi) = \beta_i^{(\alpha)} \end{array} \middle| \begin{array}{l} \text{pp}_{\text{bFC}} \leftarrow \text{bFC.Gen}(1^\lambda, \ell) \\ \text{ai}_{\text{bFC}} \leftarrow \mathcal{D}_{\text{bFC}} \\ \text{rnd}_{\text{bFC}} := (\text{rnd}_{\text{bFC},i})_{i \in [k_{\text{bFC}}]} \leftarrow \mathcal{U}((r_{\text{bFC},i})_{i \in [k_{\text{bFC}}]}) \\ \left(\text{cm}, \left(\left(\begin{array}{l} \mathcal{Q}_i, \beta_i, \\ ((\text{pm}_{i,j}, \eta_{i,j}, \text{vm}_{i,j}))_{j \in [k_{\text{bFC}}]} \end{array} \right) \right)_{i \in [L]} \right) \\ \leftarrow \text{bFCSRGame}(s_{\text{bFC}}, \text{rnd}_{\text{bFC}}, \tilde{\mathcal{P}}_{\text{bFC}}^{\text{SR}}, \text{pp}_{\text{bFC}}, \text{ai}_{\text{bFC}}) \\ \text{For } i \in [L]: \\ b_i \leftarrow \mathcal{V}_{\text{bFC}} \left(\begin{array}{l} \text{pp}_{\text{bFC}}, \text{cm}, \\ \mathcal{Q}_i, \beta_i, \\ ((\text{pm}_{i,j}, \text{vm}_{i,j}))_{j \in [k_{\text{bFC}}]} \end{array} \right) \end{array} \right] \\
& \leq \Pr \left[\begin{array}{l} (\forall i \in [L], u \in [m] : b_{u,i} = 1) \\ \wedge \nexists \Pi \text{ s.t. } \forall i \in [L], u \in [m], \alpha_{u,i}(\Pi) = \beta_{u,i} \end{array} \middle| \begin{array}{l} \text{pp}_{\text{FC}} \leftarrow \text{FC.Gen}(1^\lambda, \ell) \\ \text{ai}_{\text{FC}} \leftarrow \mathcal{D}_{\text{FC}} \\ \text{rnd}_{\text{FC}} := (\text{rnd}_{\text{FC},i})_{i \in [k_{\text{FC}}]} \leftarrow \mathcal{U}((r_{\text{FC},i})_{i \in [k_{\text{FC}}]}) \\ \left(\text{cm}, \left(\left(\begin{array}{l} \alpha_{u,i}, \beta_{u,i}, \\ ((\text{pm}_{u,i,j}, \eta_{u,i,j}, \text{vm}_{u,i,j}))_{j \in [k_{\text{FC}}]} \end{array} \right) \right)_{i \in [L], u \in [m]} \right) \\ \leftarrow \text{FCSRGame}(s_{\text{FC}}, \text{rnd}_{\text{FC}}, \tilde{\mathcal{P}}_{\text{FC}}^{\text{SR}}, \text{pp}_{\text{FC}}, \text{ai}_{\text{FC}}) \\ \text{For } i \in [L], u \in [m]: \\ b_{u,i} \leftarrow \mathcal{V}_{\text{FC}} \left(\begin{array}{l} \text{pp}_{\text{FC}}, \text{cm}, \\ \alpha_{u,i}, \beta_{u,i}, \\ ((\text{pm}_{u,i,j}, \text{vm}_{u,i,j}))_{j \in [k_{\text{bFC}}]} \end{array} \right) \end{array} \right] \\
& \leq \epsilon_{\text{FC}}^{\text{SR}}(\lambda, \ell, m \cdot L, \frac{s_{\text{bFC}}}{m}, m \cdot m_{\text{bFC}}, t_{\text{FC}}) .
\end{aligned}$$

□

3.3.5 State-restoration function binding from state-restoration knowledge soundness and binding

We show that if a functional commitment scheme FC is state-restoration knowledge sound and binding, then it is also state-restoration function binding.

Definition 3.30 (Binding). *An k_{FC} -round interactive FC with query class $\mathbf{Q}$ has **binding error** ϵ_{B} if for every security parameter $\lambda \in \mathbb{N}$, message length $\ell \in \mathbb{N}$, auxiliary input distribution $\mathcal{D}_{\text{FC}}$, circuit size bound $t_{\text{FC}} \in \mathbb{N}$, and t_{FC} -size circuit $\tilde{\mathcal{P}}_{\text{FC}}^{\text{SR}}$,*

$$\Pr \left[\begin{array}{l} \Pi \neq \Pi' \wedge \\ \text{FC.Commit}(\text{pp}_{\text{FC}}, \Pi) = \text{FC.Commit}(\text{pp}_{\text{FC}}, \Pi') \end{array} \middle| \begin{array}{l} \text{pp}_{\text{FC}} \leftarrow \text{FC.Gen}(1^\lambda, \ell) \\ \text{ai} \leftarrow \mathcal{D} \\ (\Pi, \Pi') \leftarrow A_{\text{B}}(\text{pp}_{\text{FC}}, \text{ai}) \end{array} \right] \leq \epsilon_{\text{B}}(\lambda, \ell, t_{\text{B}}) .$$

Lemma 3.31. *Let FC be a functional commitment scheme with state-restoration knowledge soundness error $\kappa_{\text{KS}}^{\text{SR}} = \kappa_{\text{KS}}^{\text{SR}}(\lambda, \ell, s_{\text{FC}}, m_{\text{FC}}, t_{\text{KS}}^{\text{SR}})$ for the ternary relation*

$$\{(\text{pp}_{\text{FC}}, (\text{cm}, \alpha, \beta), \Pi) \mid \text{cm} = \text{FC.Commit}(\text{pp}_{\text{FC}}, \Pi) \wedge \alpha(\Pi) = \beta\} ,$$

with extractor running time $t_{\mathcal{E}_{\text{SR}}}$. Further, let FC have binding error $\epsilon_B = \epsilon_B(\lambda, \ell, t_B)$ (Definition 3.30). Then FC has state-restoration function binding error $\epsilon_{\text{FC}}^{\text{SR}}$ such that for every security parameter $\lambda \in \mathbb{N}$, message length $\ell \in \mathbb{N}$, sample set size $L \in \mathbb{N}$, and adversary size bound $t_{\text{FC}}^{\text{SR}} \in \mathbb{N}$,

$$\epsilon_{\text{FC}}^{\text{SR}}(\lambda, \ell, L, s_{\text{FC}}, m_{\text{FC}}, t_{\text{FC}}^{\text{SR}}) \leq L \cdot \kappa_{\text{KS}}^{\text{SR}}(\lambda, \ell, s_{\text{FC}}, m_{\text{FC}}, t_{\text{KS}}^{\text{SR}}) + \epsilon_B(\lambda, \ell, t_B) ,$$

where $t_{\text{KS}}^{\text{SR}} \leq t_{\text{FC}}^{\text{SR}} + L\ell$ and $t_B \leq t_{\text{FC}}^{\text{SR}} + Lt_{\mathcal{E}_{\text{SR}}} + L(\ell + s_{\text{FC}}m_{\text{FC}})$.

Lemma 3.31 generically gives a concrete bound on the state-restoration function binding error for constructions which have already been shown to satisfy state-restoration knowledge soundness and binding.

Proof sketch. Let A_{SRFB} be a state-restoration function binding adversary against FC. Further, let $\mathcal{E}_{\text{SR}}$ be a state-restoration knowledge soundness extractor for FC, running in time $t_{\mathcal{E}_{\text{SR}}}$. For each $i \in [L]$, we define the following state-restoration knowledge soundness adversary $(\tilde{\mathcal{P}}_{\text{FC}}^{\text{SR}})^{(i)}$, which runs A_{SRFB} and simulates FCSRGame using its own state-restoration knowledge soundness game. When A_{SRFB} outputs

$$(\text{cm}, ((\alpha_i, \beta_i), ((\text{pm}_{i,j}, \eta_{i,j}, \text{vm}_{i,j}))_{j \in [k_{\text{FC}}]}))_{i \in [L]} ,$$

$(\tilde{\mathcal{P}}_{\text{FC}}^{\text{SR}})^{(i)}$ outputs $(\text{cm}, \alpha_i, \beta_i, (\text{pm}_{i,j})_{j \in [k_{\text{FC}}]}, (\eta_{i,j})_{j \in [k_{\text{FC}}]})$.

Consider the following adversary A_B against the binding property of FC. For each $i \in [L]$, A_B simulates FCSRGame towards $(\tilde{\mathcal{P}}_{\text{FC}}^{\text{SR}})^{(i)}$, and then runs the knowledge extractor $\mathcal{E}_{\text{SR}}$ with rewinding access to $(\tilde{\mathcal{P}}_{\text{FC}}^{\text{SR}})^{(i)}$, taking as input the instance $(\text{cm}, \alpha_i, \beta_i)$, prover messages $(\text{pm}_{i,j})_{j \in [k_{\text{FC}}]}$ and salts $(\eta_{i,j})_{j \in [k_{\text{FC}}]}$. The i -th extraction will yield a valid witness Π_i , except with probability $\kappa_{\text{KS}}^{\text{SR}}(\lambda, \ell, s_{\text{FC}}, m_{\text{FC}}, t_{\text{KS}}^{\text{SR}})$. If all extractions succeeds, A_B searches for indices $i, j \in [L]$ such that $\Pi_i \neq \Pi_j$ and outputs (Π_i, Π_j) , and aborts if no such indices exist.

If A_{SRFB} is successful, then there is no Π such that $\alpha_i(\Pi) = \beta_i$ for all $i \in [L]$. If all L extractions succeed, this implies that there exist $i, j \in [L]$ such that $\Pi_i \neq \Pi_j$, and A_B is successful. $\square$

4 The Funky protocol

Consider the following two ingredients:

- $\text{FIOP} = (\mathbf{P}, \mathbf{V})$, a public-coin functional IOP system for a relation R with alphabet Σ , domain $\mathbb{D}$, query complexity q , round complexity k_{FIOP} , query classes $\mathbf{Q}_1, \dots, \mathbf{Q}_{k_{\text{FIOP}}}$, and proof lengths $\ell_1, \dots, \ell_{k_{\text{FIOP}}}$; and
- $\text{bFC} = (\text{bFC.Gen}, \text{bFC.Commit}, \mathcal{P}_{\text{bFC}}, \mathcal{V}_{\text{bFC}})$, an k_{bFC} -round interactive batched functional commitment scheme over the alphabet Σ , and domain $\mathbb{D}$. We assume for simplicity that bFC supports a query class that includes $\mathbf{Q}_1, \dots, \mathbf{Q}_{k_{\text{FIOP}}}$. In its full generality, the Funky protocol is built from k_{FIOP} batched functional commitment schemes $\text{bFC}_1, \dots, \text{bFC}_{k_{\text{FIOP}}}$, where bFC_i supports $\mathbf{Q}_i$.

The construction of $\text{ARG} := \text{Funky}[\text{FIOP}, \text{bFC}]$ is specified below.

Construction 4.1. The argument generator $\mathcal{G}$ receives as input a security parameter $\lambda \in \mathbb{N}$ and an instance size bound $n \in \mathbb{N}$, and works as follows.

$\mathcal{G}(\lambda, n)$:

1. Sample FC scheme public parameter: $\text{pp}_{\text{bFC}} \leftarrow \text{bFC.Gen}(1^\lambda, \ell_{\max}(n))$.
2. Set public parameter for the interactive argument: $\text{pp} := \text{pp}_{\text{bFC}}$.
3. Output pp .

The argument prover $\mathcal{P}$ receives as input the public parameter pp , an instance $\mathbf{x}$, and a witness $\mathbf{w}$, and the argument verifier $\mathcal{V}$ receives as input the public parameter pp and the instance $\mathbf{x}$. Then $\mathcal{P}$ and $\mathcal{V}$ interact as follows.

1. $\mathcal{P}$'s commitments and $\mathcal{V}$'s query challenges.

For $i \in [k_{\text{FIOP}}]$:

(a) $\mathcal{P}$'s i -th commitment.

- i. Compute the i -th FIOP string $\Pi_i \in \Sigma^{\ell_i}$ and auxiliary state:

$$(\Pi_i, \mathbf{aux}_i) \leftarrow \begin{cases} \mathbf{P}(\mathbf{x}, \mathbf{w}) & \text{if } i = 1 \\ \mathbf{P}(\mathbf{aux}_{i-1}, \rho_{i-1}) & \text{if } i > 1 \end{cases}.$$

- ii. Compute a functional commitment to the FIOP string: $(\text{cm}_i, \mathbf{aux}_i) \leftarrow \text{bFC.Commit}(\text{pp}, \Pi_i)$.

iii. Send cm_i to $\mathcal{V}$.

(b) $\mathcal{V}$'s i -th query challenge.

- i. Sample the i -th FIOP verifier randomness $\rho_i \leftarrow \{0, 1\}^{r_{\text{FIOP}, i}}$.
- ii. Send ρ_i to $\mathcal{P}$.

2. $\mathcal{P}$'s response.

- (a) Run the FIOP verifier $\mathbf{V}^{\Pi_1, \dots, \Pi_{k_{\text{FIOP}}}}(\mathbf{x}; \rho_1, \dots, \rho_{k_{\text{FIOP}}})$ to deduce $\mathcal{Q}_1, \dots, \mathcal{Q}_{k_{\text{FIOP}}}$, where $\mathcal{Q}_i \subseteq \mathbf{Q}_i$ is the query set of $\mathbf{V}$ to Π_i .

- (b) For every $i \in [k_{\text{FIOP}}]$, $\alpha \in \mathcal{Q}_i$, compute the evaluation $\beta_i^{(\alpha)} := \alpha(\Pi_i)$.

- (c) Set $\beta_i := (\beta_i^{(\alpha)})_{\alpha \in \mathcal{Q}_i}$.

- (d) Send $((\mathcal{Q}_i, \beta_i))_{i \in [k_{\text{FIOP}}]}$ to $\mathcal{V}$.

3. Interaction for the opening proofs.

For every $j \in [k_{\text{bFC}}]$:

- (a) $\mathcal{P}$'s j -th message.
- i. For every $i \in [k_{\text{FIOP}}]$, compute the j -th FC prover message $\text{pm}_{i,j}$ and auxiliary state:

$$(\text{pm}_{i,j}, \text{aux}_{i,j}) \leftarrow \begin{cases} \mathcal{P}_{\text{bFC}}(\text{pp}, \text{aux}_i, \mathcal{Q}_i, \beta_i) & \text{if } j = 1 \\ \mathcal{P}_{\text{bFC}}(\text{aux}_{i,j-1}, \text{vm}_{i,j-1}) & \text{if } j > 1 \end{cases}.$$

- ii. Send $(\text{pm}_{i,j})_{i \in [k_{\text{FIOP}}]}$ to $\mathcal{V}$.
- (b) $\mathcal{V}$'s j -th message.
- i. For every $i \in [k_{\text{FIOP}}]$, sample the j -th FC verifier randomness $\text{vm}_{i,j} \leftarrow \{0, 1\}^{r_{\text{bFC},j}}$.
 - ii. Send $(\text{vm}_{i,j})_{i \in [k_{\text{FIOP}}]}$ to $\mathcal{P}$.

Set $\text{pf}_i := ((\text{pm}_{i,j}, \text{vm}_{i,j}))_{j \in [k_{\text{bFC}}]}$.

4. $\mathcal{V}$'s decision.

Check that the following holds:

- (a) $\mathbf{V}([Q_i, \beta_i])_{i \in [k_{\text{FIOP}}]}(\mathbf{x}; (\rho_i)_{i \in [k_{\text{FIOP}}]}) = 1$;
- (b) For every $i \in [k_{\text{FIOP}}]$, $\mathcal{V}_{\text{bFC}}(\text{pp}, \text{cm}_i, \mathcal{Q}_i, \beta_i, \text{pf}_i) = 1$.

The protocol has $2k_{\text{FIOP}} + 1 + 2k_{\text{bFC}}$ messages: the first $2k_{\text{FIOP}} + 1$ simulate the FIOP, and the rest simulates the bFC. For notational simplicity, we view the $(2k_{\text{FIOP}} + 1)$ -st message (Step 2d) of the prover and the $(2k_{\text{FIOP}} + 2)$ -nd message (Step 3a for $j = 1$) as one single message. Hence, the protocol has $2k_{\text{FIOP}} + 2k_{\text{bFC}}$ in total and the prover $\mathcal{P}$ and the verifier $\mathcal{V}$ interact for $k_{\text{FIOP}} + k_{\text{bFC}}$ rounds.

Moreover, the protocol is public-coin because the verifier's messages are uniformly sampled random strings. We comment on the protocol's efficiency measures:

- the generator outputs a public parameter of size $p_{\text{ARG}} := |\text{pp}_{\text{bFC}}|$ bits;
- the round complexity $k := k_{\text{FIOP}} + k_{\text{bFC}}$;
- the prover-to-verifier communication complexity, in bits, is

$$\text{pc} := \sum_{i \in [k_{\text{FIOP}}]} |\text{cm}_i| + q \cdot (\log |\mathcal{Q}_i| + \log |\mathbb{D}|) + \sum_{i \in [k_{\text{FIOP}}]} \sum_{j \in [k_{\text{bFC}}]} |\text{pm}_{i,j}|,$$

moreover,

$$\begin{aligned} \text{pc}_i &:= |\text{cm}_i| \text{ for } i \in [k_{\text{FIOP}}], \\ \text{pc}_{k_{\text{FIOP}}+1} &:= q \cdot (\log |\mathcal{Q}_i| + \log |\mathbb{D}|) + \sum_{i \in [k_{\text{FIOP}}]} |\text{pm}_{i,1}|, \\ \text{pc}_{k_{\text{FIOP}}+j} &:= \sum_{i \in [k_{\text{FIOP}}]} |\text{pm}_{i,j}| \text{ for } 2 \leq j \leq k_{\text{bFC}}; \end{aligned}$$

- the randomness complexity, in bits, is

$$r := \sum_{i \in [k_{\text{FIOP}}]} r_{\text{FIOP},i} + k_{\text{FIOP}} \cdot \sum_{j \in [k_{\text{bFC}}]} r_{\text{bFC},j},$$

moreover,

$$\begin{aligned} r_i &:= r_{\text{FIOP},i} \text{ for } i \in [k_{\text{FIOP}}], \\ r_{k_{\text{FIOP}}+j} &:= k_{\text{FIOP}} \cdot r_{\text{bFC},j} \text{ for } j \in [k_{\text{bFC}}]; \end{aligned}$$

- the argument generator running time is $t_{\mathcal{G}} := t_{\text{bFC.Gen}}$.
- the argument prover running time is $t_{\mathcal{P}} := t_{\mathcal{P}} + k_{\text{FIOP}} \cdot t_{\text{bFC.Commit}} + k_{\text{FIOP}} \cdot t_{\mathcal{P}_{\text{bFC}}} + t_{\mathcal{V}}$;
- the argument verifier running time is $t_{\mathcal{V}} := t_{\mathcal{V}} + k_{\text{FIOP}} \cdot t_{\mathcal{V}_{\text{bFC}}}$.

5 Solving time and tail errors

We introduce notions for a query class $\mathbf{Q}$ that we use in later sections: *solver* and *tail error*. Then we study these notions for several query classes of interest.

Definition 5.1 (Solver). Fix $n \in \mathbb{N}$. A **solver** for a query class $\mathbf{Q}$ is an algorithm $\text{Solver}_{\mathbf{Q}}$ that receives as input a list of query-answer pairs $((\alpha_i, \beta_i))_{i \in [n]}$ and outputs an arbitrary element in the set

$$\cap_{i=1}^n \{\Pi \in \Sigma^\ell : \forall i \in [n], \alpha_i(\Pi) = \beta_i\} .$$

We denote by $t_{\mathbf{Q}}(\ell, n)$ the running time of $\text{Solver}_{\mathbf{Q}}$ on n query-answer pairs.

Definition 5.2. Fix $k \in \mathbb{N}$. A distribution $\mathcal{D}$ is **k -admissible** for the query class $\mathbf{Q}$ if

$$\text{supp}(\mathcal{D}) \subseteq \{(Q, \beta) : Q \subseteq \mathbf{Q}, \beta \subseteq \mathbb{D}, |Q| = |\beta| \leq k\} \cup \emptyset .$$

Definition 5.3 (Tail error). A query class $\mathbf{Q}$ has **tail error** $\epsilon_{\mathbf{Q}}$ if for every $N, k, \ell \in \mathbb{N}$ and k -admissible distribution $\mathcal{D}$,

$$\Pr \left[\begin{array}{c|c} S_{n+1} \neq \emptyset & n \leftarrow [N] \\ \wedge S_{n+1} \neq S_n & (S_1, \dots, S_{n+1}) \leftarrow \text{Sampler}(n, \mathcal{D}) \end{array} \right] \leq \epsilon_{\mathbf{Q}}(\ell, k, N) ,$$

where Sampler works as follows:

$\text{Sampler}(n, \mathcal{D})$:

1. Set $S_0 := \Sigma^\ell$.
2. For $i \in [n+1]$:
 - (a) Sample $(Q_i, \beta_i) \leftarrow \mathcal{D}$.
 - (b) Set $S_i := S_{i-1} \cap \{\Pi \in \Sigma^\ell : \forall \alpha \in Q_i, \alpha(\Pi) = \beta_i^{(\alpha)}\}$.
3. Output $(S_1, \dots, S_{n+1})$.

5.1 Inefficient baseline for the general case

We prove (inefficient) baselines that hold for every query class $\mathbf{Q}$. In subsequent sections we provide (efficient) improvements for specific query classes of interest.

Lemma 5.4. Every query class $\mathbf{Q}$ has the following:

- tail error $\epsilon_{\mathbf{Q}}(\ell, k, N) \leq \frac{|\mathbf{Q}| \cdot (\ln N + 1)}{N}$,
- solving time $t_{\mathbf{Q}}(\ell, n) \leq n \cdot |\Sigma|^\ell$.

Proof. Fix a query class $\mathbf{Q}$.

Tail error. We adapt the proof from [CDDGS25]. Let $\mathcal{D}$ be a k -admissible distribution for some $k \in \mathbb{N}$. Fix $\ell, N \in \mathbb{N}$. For every $m \in [N]$, define the indicator variable χ_m to be 1 if and only if the following holds:

$$[S_{m+1} \neq \emptyset \wedge S_{m+1} \neq S_m \mid (S_1 = (Q_1, \beta_1), \dots, S_{N+1} = (Q_{N+1}, \beta_{N+1})) \leftarrow \text{Sampler}(N, \mathcal{D})] .$$

Note that $\chi_m = 1$ implies that $Q_{m+1} \setminus \cup_{i=1}^m Q_i \neq \emptyset$. Hence,

$$\mathbb{E}[\chi_m]$$

$$\begin{aligned}
&\leq \Pr [\mathcal{Q}_{m+1} \setminus \cup_{i=1}^m \mathcal{Q}_i \neq \emptyset] \\
&= \Pr [\exists \alpha \in \mathbf{Q}: \alpha \in \mathcal{Q}_{m+1} \wedge \alpha \notin \cup_{i=1}^m \mathcal{Q}_i] \\
&\leq \sum_{\alpha \in \mathbf{Q}} \Pr [\alpha \in \mathcal{Q}_{m+1} \wedge \alpha \notin \cup_{i=1}^m \mathcal{Q}_i] \\
&\leq \frac{|\mathbf{Q}|}{m} .
\end{aligned}$$

We conclude that

$$\Pr \left[\begin{array}{c} S_{m+1} \neq \emptyset \\ \wedge S_{m+1} \neq S_m \end{array} \middle| \begin{array}{c} m \leftarrow [N] \\ (S_1, \dots, S_{m+1}) \leftarrow \text{Sampler}(m, \mathcal{D}) \end{array} \right] = \frac{1}{N} \sum_{m=1}^N \mathbb{E} [\chi_m] \leq \frac{|\mathbf{Q}| \cdot (\ln N + 1)}{N} .$$

Solving time. Fix $n \in \mathbb{N}$. We construct $\text{Solver}_{\mathbf{Q}}$ for $\mathbf{Q}$ below:

$\text{Solver}_{\mathbf{Q}}(((\alpha_i, \beta_i))_{i \in [n]}):$

1. For $\Pi \in \Sigma^\ell$:
 - (a) Check if $\forall i \in [n], \alpha_i(\Pi) = \beta_i$. Continue the loop if the check fails.
 - (b) Output Π .
2. Output $\perp$.

The running time of $\text{Solver}_{\mathbf{Q}}$ can be upper-bounded by $n \cdot |\Sigma|^\ell$. Hence,

$$t_{\mathbf{Q}}(\ell, n) \leq n \cdot |\Sigma|^\ell .$$

□

5.2 Linear queries

We prove bounds for the query class $\mathbf{Q}_{\text{Lin}}$ of linear queries.

Lemma 5.5. *The query class $\mathbf{Q}_{\text{Lin}}$ (Eq. 2) has the following:*

- tail error $\epsilon_{\mathbf{Q}_{\text{Lin}}}(\ell, k, N) \leq \frac{\ell}{N}$;
- solving time $t_{\mathbf{Q}_{\text{Lin}}}(\ell, n) \leq n \cdot \ell^2$;

Proof. We compute the tail error and solving time for $\mathbf{Q}_{\text{Lin}}$ separately.

Tail error. Let $\mathcal{D}$ be a k -admissible distribution for some $k \in \mathbb{N}$. Fix $\ell, N \in \mathbb{N}$. For every $m \in [N]$, define the indicator variable χ_m to be 1 if and only if the following holds:

$$[S_{m+1} \neq \emptyset \wedge S_{m+1} \neq S_m \mid (S_1 = (\mathcal{Q}_1, \beta_1), \dots, S_{N+1} = (\mathcal{Q}_{N+1}, \beta_{N+1})) \leftarrow \text{Sampler}(N, \mathcal{D})] .$$

For each $i \in [N+1]$, $(\mathcal{Q}_i, \beta_i)$ defines a linear equation system. Note that $\chi_m = 1$ implies that $\mathcal{Q}_{m+1}$ lies outside the linear span of $\cup_{i=1}^m \mathcal{Q}_i$. Since the linear span of $\cup_{i=1}^{N+1} \mathcal{Q}_i$ has dimension at most ℓ ,

$$\Pr \left[\sum_{m=1}^N \chi_m \leq \ell \right] = 1 .$$

Hence,

$$\sum_{m=1}^N \mathbb{E} [\chi_m] \leq \ell .$$

We conclude that

$$\Pr \left[\begin{array}{c} S_{m+1} \neq \emptyset \\ \wedge S_{m+1} \neq S_m \end{array} \mid \begin{array}{c} m \leftarrow [N] \\ (S_1, \dots, S_{m+1}) \leftarrow \text{Sampler}(m, \mathcal{D}) \end{array} \right] = \frac{1}{N} \sum_{m=1}^N \mathbb{E}[\chi_m] \leq \frac{\ell}{N} .$$

Solving time. Fix $n \in \mathbb{N}$. We construct a solver for $\mathbf{Q}_{\text{Lin}}$ below:

$\text{Solver}_{\mathbf{Q}_{\text{Lin}}}(((\alpha_i, \beta_i))_{i \in [n]}):$

1. Construct the linear equation system:

$$\left\{ \begin{array}{c} \alpha_1(X) = \beta_1 \\ X \in \mathbb{F}^\ell : \quad \vdots \\ \alpha_n(X) = \beta_n \end{array} \right\} .$$

2. Use Gaussian elimination to solve the linear equation system.

3. Output an arbitrary solution Π . If there is no solution, output $\perp$.

The running time of $\text{Solver}_{\mathbf{Q}_{\text{Lin}}}$ can be upper-bounded by $n \cdot \ell^2$. Hence,

$$t_{\mathbf{Q}_{\text{Lin}}}(\ell, n) \leq n \cdot \ell^2 .$$

□

5.3 Point queries

We prove bounds for the query class $\mathbf{Q}_{\text{Point}}$ of point queries.

Lemma 5.6. *The query class $\mathbf{Q}_{\text{Point}}$ (Eq. 1) has the following:*

- *tail error $\epsilon_{\mathbf{Q}_{\text{Point}}}(\ell, k, N) \leq \frac{\ell}{N}$;*
- *solving time $t_{\mathbf{Q}_{\text{Point}}}(\ell, n) \leq n + \ell$;*

Proof. We compute the tail error and solving time for $\mathbf{Q}_{\text{Point}}$ separately.

Tail error. Let $\mathcal{D}$ be a k -admissible distribution for some $k \in \mathbb{N}$. Fix $\ell, N \in \mathbb{N}$. For every $m \in [N]$, define the indicator variable χ_m to be 1 if and only if the following holds:

$$[S_{m+1} \neq \emptyset \wedge S_{m+1} \neq S_m \mid (S_1 = (\mathcal{Q}_1, \beta_1), \dots, S_{N+1} = (\mathcal{Q}_{N+1}, \beta_{N+1})) \leftarrow \text{Sampler}(N, \mathcal{D})] .$$

Note that $\chi_m = 1$ implies that $\mathcal{Q}_{m+1}$ contains a new position that is not in $\cup_{i=1}^m \mathcal{Q}_m$. Since there are ℓ positions in total,

$$\Pr \left[\sum_{m=1}^N \chi_m \leq \ell \right] = 1 .$$

Hence,

$$\sum_{m=1}^N \mathbb{E}[\chi_m] \leq \ell .$$

We conclude that

$$\Pr \left[\begin{array}{c} S_{m+1} \neq \emptyset \\ \wedge S_{m+1} \neq S_m \end{array} \mid \begin{array}{c} m \leftarrow [N] \\ (S_1, \dots, S_{m+1}) \leftarrow \text{Sampler}(m, \mathcal{D}) \end{array} \right] = \frac{1}{N} \sum_{m=1}^N \mathbb{E}[\chi_m] \leq \frac{\ell}{N} .$$

Solving time. Fix $n \in \mathbb{N}$. We construct a solver for $\mathbf{Q}_{\text{Point}}$ below:

$\text{Solver}_{\mathbf{Q}_{\text{Point}}}(((\alpha_i, \beta_i))_{i \in [n]}):$

1. Set $\Pi := (\perp)^\ell$.
2. For every $i \in [n]$:
 - (a) If $\alpha_i(\Pi)$ (the location in Π corresponding to α_i) is $\perp$, set it to be β_i .
 - (b) Otherwise, if $\beta_i \neq \alpha_i(\Pi)$, output $\perp$.
3. For every position in Π that is $\perp$, set it to be an arbitrary value in Σ .
4. Output Π .

The running time of $\text{Solver}_{\mathbf{Q}_{\text{Point}}}$ can be upper-bounded by $n + \ell$. Hence,

$$t_{\mathbf{Q}_{\text{Point}}}(\ell, n) \leq n + \ell .$$

□

5.4 Univariate polynomial evaluation queries

We prove bounds for the query class $\mathbf{Q}_{\text{UniPoly}}$ of univariate polynomial queries.

Lemma 5.7. *The query class $\mathbf{Q}_{\text{UniPoly}}$ (Eq. 3) has the following:*

- tail error $\epsilon_{\mathbf{Q}_{\text{UniPoly}}}(\ell, k, N) \leq \frac{\ell}{N}$;
- solving time $t_{\mathbf{Q}_{\text{UniPoly}}}(\ell, n) \leq n \cdot \ell$;

Proof. We compute the tail error and solving time for $\mathbf{Q}_{\text{UniPoly}}$ separately.

Tail error. Let $\mathcal{D}$ be a k -admissible distribution for some $k \in \mathbb{N}$. Fix $\ell, N \in \mathbb{N}$. For every $m \in [N]$, define the indicator variable χ_m to be 1 if and only if the following holds:

$$[S_{m+1} \neq \emptyset \wedge S_{m+1} \neq S_m \mid (S_1 = (\mathcal{Q}_1, \beta_1), \dots, S_{N+1} = (\mathcal{Q}_{N+1}, \beta_{N+1})) \leftarrow \text{Sampler}(N, \mathcal{D})] .$$

Note that $\chi_m = 1$ implies that the Lagrange interpolation $\text{Lagrange}(((\mathcal{Q}_i, \beta_i))_{i \in [m+1]})$ exceeds the degree bound $\ell - 1$ or has larger degree than $\text{Lagrange}(((\mathcal{Q}_i, \beta_i))_{i \in [m]})$. Therefore,

$$\Pr \left[\sum_{m=1}^N \chi_m \leq \ell \right] = 1 .$$

Hence,

$$\sum_{m=1}^N \mathbb{E}[\chi_m] \leq \ell .$$

We conclude that

$$\Pr \left[\begin{array}{c} S_{m+1} \neq \emptyset \\ \wedge S_{m+1} \neq S_m \end{array} \mid \begin{array}{c} m \leftarrow [N] \\ (S_1, \dots, S_{m+1}) \leftarrow \text{Sampler}(m, \mathcal{D}) \end{array} \right] = \frac{1}{N} \sum_{m=1}^N \mathbb{E}[\chi_m] \leq \frac{\ell}{N} .$$

Solving time. Fix $n \in \mathbb{N}$. We construct a solver for $\mathbf{Q}_{\text{UniPoly}}$ below:

$\text{Solver}_{\mathbf{Q}_{\text{UniPoly}}}(((\alpha_i, \beta_i))_{i \in [n]}):$

1. If $n \leq \ell$, compute $\text{Lagrange}(((\alpha_i, \beta_i))_{i \in [\ell]})$ and output its coefficient vector.
2. Compute $f := \text{Lagrange}(((\alpha_i, \beta_i))_{i \in [\ell]})$.

3. For $\ell + 1 \leq i \leq n$, if $f(\alpha_i) \neq \beta_i$, output $\perp$.
4. Output the coefficient vector of f .

The running time of $\text{Solver}_{\mathbf{Q}_{\text{UniPoly}}}$ can be upper-bounded by $n \cdot \ell$. Hence,

$$t_{\mathbf{Q}_{\text{UniPoly}}}(\ell, n) \leq n \cdot \ell .$$

□

5.5 Multivariate polynomial evaluation queries

We prove bounds for the query class $\mathbf{Q}_{\text{MultiPoly}}^{(m,D)}$ of multivariate polynomial queries.

Lemma 5.8. *The query class $\mathbf{Q}_{\text{MultiPoly}}^{(m,D)}$ (Equation 4) has the following:*

- *tail error $\epsilon_{\mathbf{Q}_{\text{MultiPoly}}^{(m,D)}}(\ell, k, N) \leq \frac{\ell}{N}$;*
- *solving time $t_{\mathbf{Q}_{\text{MultiPoly}}^{(m,D)}}(\ell, n) \leq n \cdot \ell^2$;*

Proof. We compute the tail error and solving time for $\mathbf{Q}_{\text{MultiPoly}}^{(m,D)}$ separately.

Tail error. Let $\mathcal{D}$ be a k -admissible distribution for some $k \in \mathbb{N}$. Fix $\ell, N \in \mathbb{N}$. For every $m \in [N]$, define the indicator variable χ_m to be 1 if and only if the following holds:

$$[S_{m+1} \neq \emptyset \wedge S_{m+1} \neq S_m \mid (S_1 = (\mathcal{Q}_1, \beta_1), \dots, S_{N+1} = (\mathcal{Q}_{N+1}, \beta_{N+1})) \leftarrow \text{Sampler}(N, \mathcal{D})] .$$

Let $\gamma' := (\gamma^\omega)_{\omega \in \{0, \dots, D\}^m, \sum_{i \in [m]} \omega[i] \leq D}$, each query in $\mathbf{Q}_{\text{MultiPoly}}^{(m,D)}$ can be expressed as an inner product $\langle \gamma', \Pi \rangle$ for some $\gamma \in \mathbb{F}^m$. Therefore, for each $i \in [N+1]$, $(\mathcal{Q}_i, \beta_i)$ defines a linear equation system. Note that $\chi_m = 1$ implies that $\mathcal{Q}_{m+1}$ lies outside the linear span of $\cup_{i=1}^m \mathcal{Q}_i$. Since the linear span of $\cup_{i=1}^{N+1} \mathcal{Q}_i$ has dimension at most ℓ ,

$$\Pr \left[\sum_{m=1}^N \chi_m \leq \ell \right] = 1 .$$

Hence,

$$\sum_{m=1}^N \mathbb{E}[\chi_m] \leq \ell .$$

We conclude that

$$\Pr \left[\begin{array}{c} S_{m+1} \neq \emptyset \\ \wedge S_{m+1} \neq S_m \end{array} \mid \begin{array}{c} m \leftarrow [N] \\ (S_1, \dots, S_{m+1}) \leftarrow \text{Sampler}(m, \mathcal{D}) \end{array} \right] = \frac{1}{N} \sum_{m=1}^N \mathbb{E}[\chi_m] \leq \frac{\ell}{N} .$$

Solving time. Fix $n \in \mathbb{N}$. We construct a solver for $\mathbf{Q}_{\text{MultiPoly}}^{(m,D)}$ below:

$\text{Solver}_{\mathbf{Q}_{\text{MultiPoly}}^{(m,D)}}(((\alpha_i, \beta_i))_{i \in [n]}):$

1. Construct the linear equation system:

$$\left\{ X \in \mathbb{F}^\ell : \begin{array}{c} \alpha_1(X) = \beta_1 \\ \vdots \\ \alpha_n(X) = \beta_n \end{array} \right\} .$$

2. Use Gaussian elimination to solve the system.
3. Output a random solution Π . If there is no solution, output $\perp$.

The running time of $\text{Solver}_{\mathbf{Q}_{\text{MultiPoly}}^{(m,D)}}$ can be upper-bounded by $n \cdot \ell^2$. Hence,

$$t_{\mathbf{Q}_{\text{MultiPoly}}^{(m,D)}}(\ell, n) \leq n \cdot \ell^2 .$$

□

5.6 Structured polynomial evaluation queries

We prove bounds for the query class $\mathbf{Q}_{\text{Struct}}[\mathbf{Q}, m, (\mathbf{h}_k)_{k \in [n]}]$ defined below.

Definition 5.9. For parameters $m, n \in \mathbb{N}$, a degree bound $D \in \mathbb{N}$, public multivariate polynomials $\mathbf{h}_i \in \mathbb{F}^{\leq D_h}[X_1, \dots, X_m]$ and $\ell = (m + n)(D + 1)$, the query class $\mathbf{Q}_{\text{Struct}}[\mathbf{Q}, m, (\mathbf{h}_k)_{k \in [n]}]$ is

$$\left\{ \alpha: \mathbb{F}^\ell \rightarrow \mathbb{F}^{m+1} \left| \begin{array}{l} \exists \gamma \in \mathbb{F} \text{ s.t.} \\ \alpha(\Pi) = \left(\mathbf{f}_1(\gamma), \dots, \mathbf{f}_m(\gamma), \sum_{k \in [n]} \mathbf{h}_k(\mathbf{f}_1(\gamma), \dots, \mathbf{f}_m(\gamma)) \cdot \mathbf{g}_k(\gamma) \right), \\ \mathbf{f}_j(\gamma) := \sum_{i \in [D+1]} \gamma^{i-1} \cdot \Pi[(j-1)(D+1) + i] \text{ for } j \in [m], \\ \mathbf{g}_k(\gamma) := \sum_{i \in [D+1]} \gamma^{i-1} \cdot \Pi[(m+k-1)(D+1) + i] \text{ for } k \in [n] \end{array} \right. \right\} .$$

Lemma 5.10. The query class $\mathbf{Q}_{\text{Struct}}[\mathbf{Q}, m, (\mathbf{h}_k)_{k \in [n]}]$ has the following:

- tail error $\epsilon_{\mathbf{Q}_{\text{Struct}}[\mathbf{Q}, m, (\mathbf{h}_k)_{k \in [n]}]}(\ell, k, N) \leq \frac{\ell}{N}$;
- solving time $t_{\mathbf{Q}_{\text{Struct}}[\mathbf{Q}, m, (\mathbf{h}_k)_{k \in [n]}]}(\ell, u) \leq u \cdot \ell^2$;

Proof. We compute the tail error and solving time for $\mathbf{Q}_{\text{Struct}}[\mathbf{Q}, m, (\mathbf{h}_k)_{k \in [n]}]$ separately.

Tail error. Let $\mathcal{D}$ be a k -admissible distribution for some $k \in \mathbb{N}$. Fix $\ell, N \in \mathbb{N}$. For every $m \in [N]$, define the indicator variable χ_m to be 1 if and only if the following holds:

$$[S_{m+1} \neq \emptyset \wedge S_{m+1} \neq S_m \mid (S_1 = (\mathcal{Q}_1, \beta_1), \dots, S_{N+1} = (\mathcal{Q}_{N+1}, \beta_{N+1})) \leftarrow \text{Sampler}(N, \mathcal{D})] .$$

For each $i \in [N+1]$, $(\mathcal{Q}_i, \beta_i)$ defines a constraint system with $\ell = (m + n)(D + 1)$ variables, corresponding to the coefficients of $(\mathbf{f}_1, \dots, \mathbf{f}_m, \mathbf{g}_1, \dots, \mathbf{g}_n)$. Since $\mathbf{h}_1, \dots, \mathbf{h}_m$ are public, the system indeed forms a linear equation system. Specifically, a query-answer pair (α, β) can be rewritten as

$$\left\{ X \in \mathbb{F}^\ell : \begin{array}{l} \sum_{i \in [D+1]} \gamma^{i-1} \cdot X[i] = \beta[1] \\ \sum_{i \in [D+1]} \gamma^{i-1} \cdot X[(D+1) + i] = \beta[2] \\ \vdots \\ \sum_{i \in [D+1]} \gamma^{i-1} \cdot X[(m-1)(D+1) + i] = \beta[m] \\ \sum_{k \in [n]} \mathbf{h}_k(\beta[1], \dots, \beta[m]) \sum_{i \in [D+1]} \gamma^{i-1} \cdot X[(m+k-1)(D+1) + i] = \beta[m+1] \end{array} \right\} .$$

Consequently, $\chi_j = 1$ implies that $\mathcal{Q}_{j+1}$ lies outside the linear span of $\cup_{i=1}^j \mathcal{Q}_i$. Since the linear span of $\cup_{i=1}^{N+1} \mathcal{Q}_i$ has dimension at most ℓ ,

$$\Pr \left[\sum_{j=1}^N \chi_j \leq \ell \right] = 1 ,$$

and

$$\sum_{j=1}^N \mathbb{E} [\chi_j] \leq \ell .$$

We conclude that

$$\Pr \left[\begin{array}{c} S_{j+1} \neq \emptyset \\ \wedge S_{j+1} \neq S_j \end{array} \mid \begin{array}{c} j \leftarrow [N] \\ (S_1, \dots, S_{j+1}) \leftarrow \text{Sampler}(j, \mathcal{D}) \end{array} \right] = \frac{1}{N} \sum_{j=1}^N \mathbb{E} [\chi_j] \leq \frac{\ell}{N} .$$

Solving time. Fix $u \in \mathbb{N}$. We construct a solver for $\mathbf{Q}_{\text{Struct}}[\mathbf{Q}, m, (\mathbf{h}_k)_{k \in [n]}]$ below:

$\text{Solver}_{\mathbf{Q}_{\text{Struct}}[\mathbf{Q}, m, (\mathbf{h}_k)_{k \in [n]}]}(((\alpha_i, \beta_i))_{i \in [u]}):$

1. Construct the linear equation system:

$$\left\{ \begin{array}{l} X \in \mathbb{F}^\ell : \\ \alpha_1(X) = \beta_1 \\ \vdots \\ \alpha_u(X) = \beta_u \end{array} \right\} .$$

2. Use Gaussian elimination to solve the system.

3. Output a random solution Π . If there is no solution, output $\perp$.

The running time of $\text{Solver}_{\mathbf{Q}_{\text{Struct}}[\mathbf{Q}, m, (\mathbf{h}_k)_{k \in [n]}]}$ can be upper-bounded by $u \cdot \ell^2$. Hence,

$$t_{\mathbf{Q}_{\text{Struct}}[\mathbf{Q}, m, (\mathbf{h}_k)_{k \in [n]}]}(\ell, u) \leq u \cdot \ell^2 .$$

□

5.7 Bounded-degree functions

We analyze the tail error for query classes consisting of multivariate polynomials of bounded total degree. Specifically, a query $\alpha(\Pi)$ in the query class $\mathbf{Q}$ is an ℓ -variate polynomial with a total degree bounded by D .

In particular, $\mathbf{Q}_{\text{Lin}}$, $\mathbf{Q}_{\text{Point}}$, $\mathbf{Q}_{\text{UniPoly}}$, $\mathbf{Q}_{\text{MultiPoly}}$ and $\mathbf{Q}_{\text{Struct}}[\mathbf{Q}, m, (\mathbf{h}_k)_{k \in [n]}]$ are special cases of $\mathbf{Q}_{\text{Func}}^{(D)}$.

Lemma 5.11. Fix a field $\mathbb{F}$ and $\ell, D, q \in \mathbb{N}$. Let

$$\mathbf{Q}_{\text{Func}}^{(D)} := \mathbb{F}^{\leq D}[X_1, \dots, X_\ell]$$

be the query class of ℓ -variate polynomial queries of total degree at most D . The query class $\mathbf{Q}_{\text{Func}}^{(D)}$ has the following:

- tail error $\epsilon_{\mathbf{Q}_{\text{Func}}^{(D)}}(\ell, k, N) \leq \frac{\binom{\ell+D}{D}-1}{N}$;
- solving time $t_{\mathbf{Q}_{\text{Func}}^{(D)}}(\ell, n) \leq n \cdot |\mathbb{F}|^\ell$;

Proof. We compute the tail error and solving time for $\mathbf{Q}_{\text{Func}}^{(\text{D})}$ separately.

Tail error. Let $\mathcal{D}$ be a k -admissible distribution for some $k \in \mathbb{N}$. Fix $\ell, N \in \mathbb{N}$. For every $m \in [N]$, define the indicator variable χ_m to be 1 if and only if the following holds:

$$[S_{m+1} \neq \emptyset \wedge S_{m+1} \neq S_m \mid (S_1 = (\mathcal{Q}_1, \beta_1), \dots, S_{N+1} = (\mathcal{Q}_{N+1}, \beta_{N+1})) \leftarrow \text{Sampler}(N, \mathcal{D})]$$

For each $i \in [N+1]$, $(\mathcal{Q}_i, \beta_i)$ defines a constraint system with ℓ variables, which can be relaxed into a linear equation system with $\binom{\ell+\text{D}}{\text{D}} - 1$ variables by treating each possible monomial as a new variable. We denote the relaxed linear equation system as $(\mathcal{Q}'_i, \beta_i)$, $i \in [N+1]$. Consequently, $\chi_m = 1$ implies that $\mathcal{Q}'_{m+1}$ lies outside the linear span of $\cup_{i=1}^m \mathcal{Q}'_i$. Since the linear span of $\cup_{i=1}^{N+1} \mathcal{Q}'_i$ has dimension at most $\binom{\ell+\text{D}}{\text{D}} - 1$,

$$\Pr \left[\sum_{m=1}^N \chi_m \leq \binom{\ell+\text{D}}{\text{D}} - 1 \right] = 1 .$$

Hence

$$\sum_{m=1}^N \mathbb{E}[\chi_m] \leq \binom{\ell+\text{D}}{\text{D}} - 1 .$$

We conclude that

$$\Pr \left[\begin{array}{c} S_{m+1} \neq \emptyset \\ \wedge S_{m+1} \neq S_m \end{array} \mid \begin{array}{c} m \leftarrow [N] \\ (S_1, \dots, S_{m+1}) \leftarrow \text{Sampler}(m, \mathcal{D}) \end{array} \right] = \frac{1}{N} \sum_{m=1}^N \mathbb{E}[\chi_m] \leq \frac{\binom{\ell+\text{D}}{\text{D}} - 1}{N} .$$

Solving time. Fix $n \in \mathbb{N}$. We construct a solver for $\mathbf{Q}_{\text{Func}}^{(\text{D})}$ below:

$\text{Solver}_{\mathbf{Q}_{\text{Func}}^{(\text{D})}}(((\alpha_i, \beta_i))_{i \in [n]}):$

1. For $\Pi \in \mathbb{F}^\ell$:
 - (a) Check if $\forall i \in [n], \alpha_i(\Pi) = \beta_i$. Continue the loop if the check fails.
 - (b) Output Π .
2. Output $\perp$.

The running time of $\text{Solver}_{\mathbf{Q}_{\text{Func}}^{(\text{D})}}$ can be upper-bounded by $n \cdot |\mathbb{F}|^\ell$. Hence,

$$t_{\mathbf{Q}_{\text{Func}}^{(\text{D})}}(\ell, n) \leq n \cdot |\mathbb{F}|^\ell .$$

□

6 State-restoration security reduction

We establish a security reduction lemma for the state-restoration security of $\text{ARG} := \text{Funky}[\text{FIOP}, \text{bFC}]$. Our primary goal is to bound the probability that the argument state-restoration adversary convinces the argument verifier, yet the extracted FIOP messages from rewinding fail to satisfy the FIOP verifier.

We start with some definitions. Let $\tilde{\mathcal{P}}^{\text{SR}}$ be a state-restoration adversary for ARG with size t_{ARG} and move budget m . A move mv output by $\tilde{\mathcal{P}}^{\text{SR}}$ can be one of the following two forms:

- $(\mathbb{X}', (\text{cm}'_j)_{j \in [i]}, (\sigma'_j)_{j \in [i]})$ for some $i \in [\text{k}_{\text{FIOP}}]$, which we rewrite as $(\mathbb{X}', ((\text{cm}'_j, \sigma'_j))_{j \in [i]})$ for simplicity;
- $(\mathbb{X}', ((\text{cm}'_u)_{u \in [\text{k}_{\text{FIOP}}]}, ((\mathcal{Q}'_u, \beta'_u))_{u \in [\text{k}_{\text{FIOP}}]}, (\text{pm}'_{i,u})_{i \in [\text{k}_{\text{FIOP}}], u \in [j]}, ((\sigma'_u)_{u \in [\text{k}_{\text{FIOP}}]}, (\eta'_{i,u})_{i \in [\text{k}_{\text{FIOP}}], u \in [j]}))$ for some $j \in [\text{k}_{\text{bFC}}]$, which we rewrite as

$$(\mathbb{X}', ((\text{cm}'_u, \sigma'_u))_{u \in [\text{k}_{\text{FIOP}}]}, ((\mathcal{Q}'_u, \beta'_u))_{u \in [\text{k}_{\text{FIOP}}]}, ((\text{pm}'_{i,u}, \eta'_{i,u}))_{i \in [\text{k}_{\text{FIOP}}], u \in [j]})$$

for simplicity.

Moreover, we assume that $\tilde{\mathcal{P}}^{\text{SR}}$ has final output

$$(\mathbb{X}, ((\text{cm}_i, \sigma_i))_{i \in [\text{k}_{\text{FIOP}}]}, ((\mathcal{Q}_i, \beta_i))_{i \in [\text{k}_{\text{FIOP}}]}, ((\text{pm}_{i,j}, \eta_{i,j}))_{i \in [\text{k}_{\text{FIOP}}], j \in [\text{k}_{\text{bFC}}]})$$

such that all its *partial moves* (prefixes of the final output) are included in tr^{SR} , which can be ensured by increasing the move budget from m to $\hat{m} := m + \text{k}_{\text{FIOP}} + \text{k}_{\text{bFC}}$. Specifically, the following moves are included in tr^{SR} :

- for every $i \in [\text{k}_{\text{FIOP}}]$, $(\mathbb{X}, ((\text{cm}_j, \sigma_j))_{j \in [i]})$;
- for every $i \in [\text{k}_{\text{bFC}}]$, $(\mathbb{X}, ((\text{cm}_j, \sigma_j))_{j \in [\text{k}_{\text{FIOP}}]}, ((\mathcal{Q}_j, \beta_j))_{j \in [\text{k}_{\text{FIOP}}]}, ((\text{pm}_{j,u}, \eta_{j,u}))_{j \in [\text{k}_{\text{FIOP}}], u \in [i]})$.

Finally, for any move mv , we set $\text{round}(mv) := i$ if mv is a move for round i (this is always unambiguous).

Lemma 6.1. *There exists algorithms $\mathfrak{R}_m$ and $\mathfrak{R}$ such that for every security parameter $\lambda \in \mathbb{N}$, instance size bound $n \in \mathbb{N}$, number of rewindings $N \in \mathbb{N}$, auxiliary input distribution $\mathcal{D}$, adversary size bound $t_{\text{ARG}} \in \mathbb{N}$, adversary move budget $m \in \mathbb{N}$, and t_{ARG} -size m -move circuit $\tilde{\mathcal{P}}^{\text{SR}}$, the following holds:*

$$\Pr \left[\begin{array}{l} \mathbf{V}^{(\tilde{\Pi}_i)_{i \in [\text{k}_{\text{FIOP}}]}}(\mathbb{X}; (\rho_i)_{i \in [\text{k}_{\text{FIOP}}]}) \neq 1 \\ \wedge \mathbf{V}^{([\mathcal{Q}_i, \beta_i])_{i \in [\text{k}_{\text{FIOP}}]}}(\mathbb{X}; (\rho_i)_{i \in [\text{k}_{\text{FIOP}}]}) = 1 \\ \wedge \left(\forall i \in [\text{k}_{\text{FIOP}}] : \right. \\ \left. \mathcal{V}_{\text{bFC}} \left(\begin{array}{l} \text{pp}, \text{cm}_i, \mathcal{Q}_i, \beta_i, \\ ((\text{pm}_{i,j}, \text{vm}_{i,j}))_{j \in [\text{k}_{\text{bFC}}]} \end{array} \right) = 1 \right) \end{array} \right] \left[\begin{array}{l} \text{pp} \leftarrow \mathcal{G}(1^\lambda, n) \\ \text{ai} \leftarrow \mathcal{D} \\ \text{rnd} := (\text{rnd}_i)_{i \in [\text{k}_{\text{FIOP}} + \text{k}_{\text{bFC}}]} \leftarrow \mathcal{U}((r_i)_{i \in [\text{k}_{\text{FIOP}} + \text{k}_{\text{bFC}}]}) \\ \left(\begin{array}{l} \mathbb{X}, ((\text{cm}_i, \sigma_i))_{i \in [\text{k}_{\text{FIOP}}]}, ((\mathcal{Q}_i, \beta_i))_{i \in [\text{k}_{\text{FIOP}}]}, \\ ((\text{pm}_{i,j}, \eta_{i,j}))_{i \in [\text{k}_{\text{FIOP}}], j \in [\text{k}_{\text{bFC}}]}, \\ (\rho_i)_{i \in [\text{k}_{\text{FIOP}}]}, (\text{vm}_{i,j})_{i \in [\text{k}_{\text{FIOP}}], j \in [\text{k}_{\text{bFC}}]} \end{array} \right) \\ \xleftarrow{\text{tr}^{\text{SR}}} \text{SRGame}(s, \text{rnd}, \tilde{\mathcal{P}}^{\text{SR}}, \text{pp}, \text{ai}) \\ \text{Parse } \text{tr}^{\text{SR}} \text{ as } ((\text{mv}_a, \text{res}_a))_{a \in [\hat{m}]} \\ \text{For every } a \in [\hat{m}] : \\ (\tilde{\Pi}_i^{(a)})_{i \in [\text{round}(\text{mv}_a)]} \leftarrow \mathfrak{R}_m^{\tilde{\mathcal{P}}^{\text{SR}}(\text{pp}, \text{ai})}(a, (\text{res}_i)_{i \in [a-1]}) \\ (\tilde{\Pi}_i)_{i \in [\text{k}_{\text{FIOP}}]} \leftarrow \mathfrak{R} \left(\begin{array}{l} \text{tr}^{\text{SR}}, (\mathbb{X}, ((\text{cm}_i, \sigma_i))_{i \in [\text{k}_{\text{FIOP}}]}), \\ (\tilde{\Pi}_i^{(a)})_{a \in [\hat{m}], i \in [\text{round}(\text{mv}_a)]} \end{array} \right) \end{array} \right] \\ \leq \sum_{i \in [\text{k}_{\text{FIOP}}]} (\epsilon_{\text{bFC}}^{\text{SR}}(\lambda, \ell_i, \mathbf{q}_i, N + 1, s_{\text{bFC}}, m_{\text{bFC}}, t_{\text{bFC}}) + \epsilon_{\mathbf{Q}_i}(\ell_i, \mathbf{q}_i, N)) ,$$

where

- $s_{\text{bFC}} \leq s + \log((k_{\text{FIOP}} \cdot N + 1) \cdot (m + k_{\text{bFC}}))$;
- $m_{\text{bFC}} \leq (k_{\text{FIOP}} \cdot N + 1) \cdot (m + k_{\text{bFC}})$; and
- $t_{\text{bFC}} \leq O(N \cdot (t_{\text{ARG}} + m + k_{\text{FIOP}} + k_{\text{bFC}} + t_{\text{VbFC}})) + t_{\text{Q}}(\ell_{\text{max}}, N \cdot q_{\text{max}})$.

Roadmap. The proof of Lemma 6.1 proceeds in three steps:

- **Reductor construction (Section 6.1):** We first construct the reducers. The core component is $\mathfrak{R}_m$ (Construction 6.2), which extracts the underlying FIOP message for a single move. Crucially, it operates by *rewinding* the adversary to the state immediately preceding the move and exploring alternative execution branches by simulating the random oracles. This allows collecting sufficient query-answer pairs to invoke the query class solver. A global algorithm $\mathfrak{R}$ (Construction 6.3) then aggregates these extractions into the FIOP messages. For each round, $\mathfrak{R}$ identifies the first move in the trace consistent with the final output's commitment prefix and adopts its extracted FIOP message.
- **Error analysis (Section 6.2):** We then analyze the probability that the reduction fails (i.e., the argument verifier accepts but the extraction fails to satisfy the FIOP verifier). We decompose this failure event into two disjoint cases:
 - *Binding violation (Section 6.2.1):* The reductor cannot reconstruct the FIOP messages, i.e., the constraint system has no solution. We bound this probability by constructing an adversary against the state-restoration function binding of the FC scheme (Construction 6.5).
 - *Tail failure (Section 6.2.2):* The reconstructed FIOP messages are inconsistent with the openings. By viewing the openings as a fresh query, this probability is bounded by the *tail error* of the query class.

6.1 Construction of the security reducers

We construct the algorithms $\mathfrak{R}_m$ and $\mathfrak{R}$ in Lemma 6.1: $\mathfrak{R}_m$ rewinds $\tilde{\mathcal{P}}^{\text{SR}}$ multiple times to obtain FIOP strings for a specific move by $\tilde{\mathcal{P}}^{\text{SR}}$; and $\mathfrak{R}$ outputs the FIOP strings for $\tilde{\mathcal{P}}^{\text{SR}}$'s final output (given all the obtained FIOP strings).

Construction 6.2. $\mathfrak{R}_m$ has oracle access to $\tilde{\mathcal{P}}^{\text{SR}}$, takes in input a move index $a \in [\hat{m}]$ and responses $(\text{res}_i)_{i \in [a-1]}$, and outputs FIOP strings $(\tilde{\Pi}_i^{(a)})_{i \in [\text{round}(\text{mv}_a)]}$.

$\mathfrak{R}_m^{\tilde{\mathcal{P}}^{\text{SR}}(\text{pp}, \text{ai})}(a, (\text{res}_i)_{i \in [a-1]}):$

1. Simulate SRGame until $\tilde{\mathcal{P}}^{\text{SR}}$ makes the a -th move mv_a , answering the first $a-1$ moves with $(\text{res}_i)_{i \in [a-1]}$.
2. If $\text{round}(\text{mv}_a) > k_{\text{FIOP}}$, output $\perp$; otherwise parse mv_a as $(\mathbb{x}^*, ((\text{cm}_i^*, \sigma_i^*))_{i \in [\text{round}(\text{mv}_a)]})$.
3. For every $i \in [\text{round}(\text{mv}_a)]$, initialize a set of query-answer pairs $K_i^{(a)} := \emptyset$.
4. *Rewind to obtain query-answer pairs.* Sample $N_a \leftarrow [N]$ and repeat the following N_a times.
 - (a) For every $i \in [k]$, initialize an empty partial function $\text{rnd}_i: \{0, 1\}^* \rightarrow \{0, 1\}^{r_i}$ where

$$\text{rnd}_i(\text{mv}) := \begin{cases} \text{res}_v & \text{if mv is the } v\text{-th move of the simulation} \\ & \text{with } v \in [a-1] \text{ and } \text{round}(\text{mv}) = i; \\ \perp & \text{otherwise.} \end{cases}$$

- (b) Continue simulating SRGame using $(\text{rnd}_i)_{i \in [k]}$ to obtain the final output

$$(\mathbb{x}, ((\text{cm}_i, \sigma_i))_{i \in [k_{\text{FIOP}}]}, ((\mathcal{Q}_i, \beta_i))_{i \in [k_{\text{FIOP}}]}, ((\text{pm}_{i,j}, \eta_{i,j}))_{i \in [k_{\text{FIOP}}], j \in [k_{\text{bFC}}]}) \cdot$$

When $\tilde{\mathcal{P}}^{\text{SR}}$ makes a move mv for round $i \in [k]$: if $\text{rnd}_i(mv) = \perp$ (meaning this is a fresh move in this execution branch), then sample fresh $\rho \in \{0, 1\}^{r_i}$ and set $\text{rnd}_i(mv) := \rho$; either way, answer mv with $\text{rnd}_i(mv)$.

(c) For every $j \in [k_{\text{bFC}}]$, let $(\text{vm}_{i,j})_{i \in [k_{\text{FIOP}}]}$ be the response of SRGame for the move

$$(\mathbb{x}, ((\text{cm}_i, \sigma_i))_{i \in [k_{\text{FIOP}}]}, ((\mathcal{Q}_i, \beta_i))_{i \in [k_{\text{FIOP}}]}, ((\text{pm}_{i,u}, \eta_{i,u}))_{i \in [k_{\text{FIOP}}], u \in [j]}) .$$

(d) For every $i \in [\text{round}(mv_a)]$: if $\mathcal{V}_{\text{bFC}}(\text{pp}, \text{cm}_i^*, \mathcal{Q}_i, \beta_i, ((\text{pm}_{i,j}, \text{vm}_{i,j}))_{j \in [k_{\text{bFC}}]}) = 1$, add $(\mathcal{Q}_i, \beta_i)$ to $K_i^{(a)}$.

5. *Solve for the FIOP strings.* For every $i \in [\text{round}(mv_a)]$:

(a) Parse $K_i^{(a)}$ as $((\alpha_j, \beta_j))_{j \in [u]}$, where u is the total number of query-answer pairs in $K_i^{(a)}$.

(b) For analysis, define $S_i^{(a)}$ to be the set of FIOP strings in Σ^{ℓ_i} consistent with $K_i^{(a)}$:

$$S_i^{(a)} := \{\Pi_i \in \Sigma^{\ell_i} : \forall j \in [u], \alpha_j(\Pi_i) = \beta_j\} .$$

(c) Run $\tilde{\Pi}_i^{(a)} \leftarrow \text{Solver}_{\mathbf{Q}_i}(((\alpha_j, \beta_j))_{j \in [u]})$.

6. Output $(\tilde{\Pi}_i^{(a)})_{i \in [\text{round}(mv_a)]}$.

The algorithm $\mathfrak{R}_m$ rewinds $\tilde{\mathcal{P}}^{\text{SR}}$ and performs additional checks for at most N times, with each loop costing at most time $t_{\text{ARG}} + k_{\text{FIOP}} \cdot t_{\mathcal{V}_{\text{bFC}}}$. Moreover, for each round $i \in [k_{\text{FIOP}}]$, $\mathfrak{R}_m$ uses the query class solver $\text{Solver}_{\mathbf{Q}}$ to reconstruct the i -th FIOP string. Hence, the running time of $\mathfrak{R}_m$ is

$$\begin{aligned} t_{\mathfrak{R}_m} &\leq N \cdot (t_{\text{ARG}} + k_{\text{FIOP}} \cdot t_{\mathcal{V}_{\text{bFC}}}) + \sum_{i \in [k_{\text{FIOP}}]} t_{\mathbf{Q}}(\ell_i, N \cdot q_i) \\ &\leq N \cdot (t_{\text{ARG}} + k_{\text{FIOP}} \cdot t_{\mathcal{V}_{\text{bFC}}}) + k_{\text{FIOP}} \cdot t_{\mathbf{Q}}(\ell_{\max}, N \cdot q_{\max}) . \end{aligned}$$

Construction 6.3. $\mathfrak{R}$ takes as input a move-response trace tr^{SR} , a move $(\mathbb{x}, ((\text{cm}_i, \sigma_i))_{i \in [k_{\text{FIOP}}]})$, and a list of FIOP strings $(\tilde{\Pi}_i^{(a)})_{a \in [\hat{m}], i \in [c_a]}$, and outputs k_{FIOP} FIOP strings.

$\mathfrak{R}(\text{tr}^{\text{SR}}, (\mathbb{x}, ((\text{cm}_i, \sigma_i))_{i \in [k_{\text{FIOP}}]}), (\tilde{\Pi}_i^{(a)})_{a \in [\hat{m}], i \in [c_a]})$:

1. For every $i \in [k_{\text{FIOP}}]$:

(a) Let a be the index of the first move mv in tr^{SR} such that $\text{round}(mv) \leq k_{\text{FIOP}}$ and mv is of the form

$$(\mathbb{x}, ((\text{cm}_j, \sigma_j))_{j \in [i]}, ((\text{cm}'_j, \sigma'_j))_{j \in [u] \setminus [i]}) \text{ for some } u \geq i. \text{ Set } \tilde{\Pi}_i := \tilde{\Pi}_i^{(a)}.$$

(b) If there is no such move, set $\tilde{\Pi}_i := \perp$.

2. Output $(\tilde{\Pi}_i)_{i \in [k_{\text{FIOP}}]}$.

For every $i \in [k_{\text{FIOP}}]$, the algorithm $\mathfrak{R}$ scans all moves in search of a match, which takes time at most $\hat{m}$. Hence, $\mathfrak{R}$ runs in time

$$t_{\mathfrak{R}} \leq k_{\text{FIOP}} \cdot \hat{m} \leq k_{\text{FIOP}} \cdot (m + k_{\text{FIOP}} + k_{\text{bFC}}) .$$

6.2 Proof of Lemma 6.1

Throughout this proof, probabilities are with respect to the following experiment unless stated otherwise:

$$\left[\begin{array}{l} \text{pp} \leftarrow \mathcal{G}(1^\lambda, n) \\ \text{ai} \leftarrow \mathcal{D} \\ \text{rnd} := (\text{rnd}_i)_{i \in [\text{k}_{\text{FIOP}} + \text{k}_{\text{bFC}}]} \leftarrow \mathcal{U}((r_i)_{i \in [\text{k}_{\text{FIOP}} + \text{k}_{\text{bFC}}]}) \\ \left(\begin{array}{l} \mathbb{X}, ((\text{cm}_i, \sigma_i))_{i \in [\text{k}_{\text{FIOP}}]}, ((\mathcal{Q}_i, \beta_i))_{i \in [\text{k}_{\text{FIOP}}]}, \\ ((\text{pm}_{i,j}, \eta_{i,j}))_{i \in [\text{k}_{\text{FIOP}}], j \in [\text{k}_{\text{bFC}}]}, \\ (\rho_i)_{i \in [\text{k}_{\text{FIOP}}]}, (\text{vm}_{i,j})_{i \in [\text{k}_{\text{FIOP}}], j \in [\text{k}_{\text{bFC}}]} \end{array} \right) \xleftarrow{\text{tr}^{\text{SR}}} \text{SRGame}(s, \text{rnd}, \tilde{\mathcal{P}}^{\text{SR}}, \text{pp}, \text{ai}) \\ \text{For } a \in [\hat{m}] : (\tilde{\Pi}_i^{(a)})_{i \in [\text{round}(\text{mv}_a)]} \leftarrow \mathfrak{R}_m^{\tilde{\mathcal{P}}^{\text{SR}}(\text{pp}, \text{ai})}(a, (\text{res}_i)_{i \in [a-1]}) \\ (\tilde{\Pi}_i)_{i \in [\text{k}_{\text{FIOP}}]} \leftarrow \mathfrak{R}(\text{tr}^{\text{SR}}, (\mathbb{X}, ((\text{cm}_i, \sigma_i))_{i \in [\text{k}_{\text{FIOP}}]}), (\tilde{\Pi}_i^{(a)})_{a \in [\hat{m}], i \in [\text{round}(\text{mv}_a)]}) \end{array} \right]. \quad (10)$$

Our goal is to upper-bound the probability of the following event:

$$E_{\text{reduction}} \iff \left[\begin{array}{l} \mathbf{V}^{(\tilde{\Pi}_i)_{i \in [\text{k}_{\text{FIOP}}]}}(\mathbb{X}; (\rho_i)_{i \in [\text{k}_{\text{FIOP}}]}) \neq 1 \\ \wedge \mathbf{V}^{((\mathcal{Q}_i, \beta_i)_{i \in [\text{k}_{\text{FIOP}}]}}(\mathbb{X}; (\rho_i)_{i \in [\text{k}_{\text{FIOP}}]}) = 1 \\ \wedge \left(\forall i \in [\text{k}_{\text{FIOP}}] : \right. \\ \left. \mathcal{V}_{\text{bFC}} \left(\begin{array}{l} \text{pp}, \text{cm}_i, \mathcal{Q}_i, \beta_i, \\ ((\text{pm}_{i,j}, \text{vm}_{i,j}))_{j \in [\text{k}_{\text{bFC}}]} \end{array} \right) = 1 \right) \end{array} \right]. \quad (11)$$

Let S_i be the solution space from which $\tilde{\Pi}_i$ is selected as defined in Construction 6.2. Set $S_i^* := S_i \cap \{\Pi \in \Sigma^\ell : \forall \alpha \in \mathcal{Q}_i, \alpha(\Pi) = \beta_i^{(\alpha)}\}$. The event above indicates that for some round $i \in [\text{k}_{\text{FIOP}}]$, either the reducer fails to reconstruct the FIOP proof string $\tilde{\Pi}_i$, or $\tilde{\Pi}_i$ is inconsistent with $(\mathcal{Q}_i, \beta_i)$. This further implies that one of the following two cases must hold:

1. There exists $i \in [\text{k}_{\text{FIOP}}]$ such that $S_i^* = \emptyset$.
2. There exists $i \in [\text{k}_{\text{FIOP}}]$ such that $S_i^* \neq \emptyset$ and $S_i^* \neq S_i$.

In other words, by union bound, we just need to upper-bound the following two probabilities to get a bound for the target probability:

$$\Pr \left[\exists i \in [\text{k}_{\text{FIOP}}], \left(\begin{array}{l} S_i^* = \emptyset \\ \wedge \mathcal{V}_{\text{bFC}} \left(\begin{array}{l} \text{pp}, \text{cm}_i, \mathcal{Q}_i, \beta_i, \\ ((\text{pm}_{i,j}, \text{vm}_{i,j}))_{j \in [\text{k}_{\text{bFC}}]} \end{array} \right) = 1 \end{array} \right) \right]; \quad (12)$$

and

$$\Pr \left[\exists i \in [\text{k}_{\text{FIOP}}], \left(\begin{array}{l} S_i^* \neq \emptyset \\ \wedge S_i^* \neq S_i \\ \wedge \mathcal{V}_{\text{bFC}} \left(\begin{array}{l} \text{pp}, \text{cm}_i, \mathcal{Q}_i, \beta_i, \\ ((\text{pm}_{i,j}, \text{vm}_{i,j}))_{j \in [\text{k}_{\text{bFC}}]} \end{array} \right) = 1 \end{array} \right) \right]. \quad (13)$$

6.2.1 Bound for Equation 12 from state-restoration function binding

We prove that the probability in Equation 12 is bounded by the state-restoration function binding error of bFC. To achieve this, we construct a family of FC state-restoration adversaries $(\tilde{\mathcal{P}}_{\text{bFC}, i}^{\text{SR}})_{i \in [\text{k}_{\text{FIOP}}]}$.

The strategy is to simulate the behavior of the reducer while forwarding the FC opening moves of $\tilde{\mathcal{P}}^{\text{SR}}$ to the external FC challenger. Crucially, to elicit fresh randomness, if a truncated FC opening move has been queried previously, the adversary modifies the associated salt. Consequently, if the reducer fails to reconstruct the FIOP messages, it implies that the underlying constraint system has no solution, thereby allowing some $\tilde{\mathcal{P}}_{\text{bFC},i}^{\text{SR}}$ to break the state-restoration function binding property.

Construction 6.4. Let $\mathcal{D}$ be the auxiliary input distribution for ARG. The auxiliary input distribution $\mathcal{D}_{\text{bFC}}$ for bFC is as follows.

$\mathcal{D}_{\text{bFC}}$: Output $\text{ai} \leftarrow \mathcal{D}$.

Construction 6.5. For every $i \in [k_{\text{FIOP}}]$, we define $\tilde{\mathcal{P}}_{\text{bFC},i}^{\text{SR}}$ as follows.

$\tilde{\mathcal{P}}_{\text{bFC},i}^{\text{SR}}(\text{pp}_{\text{bFC}}, \text{ai}_{\text{bFC}})$:

1. Parse pp_{bFC} as pp and ai_{bFC} as ai .
2. Set $I := \emptyset$.
3. Simulate the execution of SRGame with $\tilde{\mathcal{P}}^{\text{SR}}(\text{pp}, \text{ai})$ as follows.
 - (a) For every $u \in [k]$, initialize an empty partial function $\text{rnd}_u: \{0, 1\}^* \rightarrow \{0, 1\}^{r_u}$.
 - (b) When $\tilde{\mathcal{P}}^{\text{SR}}$ outputs a move mv of the form $(\mathbb{x}', ((\text{cm}'_j, \sigma'_j))_{j \in [u]})$ for some $u \in [k_{\text{FIOP}}]$: if $\text{rnd}_u(\text{mv}) = \perp$ then sample $\rho \in \{0, 1\}^{r_u}$ and set $\text{rnd}_u(\text{mv}) := \rho$; either way, answer mv with $\text{rnd}_u(\text{mv})$.
 - (c) When $\tilde{\mathcal{P}}^{\text{SR}}$ outputs a move mv of the form

$$(\mathbb{x}', ((\text{cm}'_u, \sigma'_u))_{u \in [k_{\text{FIOP}}]}, ((\mathcal{Q}'_u, \beta'_u))_{u \in [k_{\text{FIOP}}]}, ((\text{pm}'_{u,j}, \eta'_{u,j}))_{u \in [k_{\text{FIOP}}], j \in [c]})$$

for some $c \in [k_{\text{bFC}}]$:

- i. If $\text{rnd}_{k_{\text{FIOP}}+c}(\text{mv}) = \perp$ then set $\text{rnd}_{k_{\text{FIOP}}+c}(\text{mv}) := (\text{vm}'_{u,c})_{u \in [k_{\text{FIOP}}]}$ where, for every $u \in [k_{\text{FIOP}}]$, $\text{vm}'_{u,c}$ is obtained as follows:
 - A. Set $\text{mv}_{\text{bFC}} := (\text{cm}'_u, \mathcal{Q}'_u, \beta'_u, (\text{pm}'_{u,j})_{j \in [c]}, (\eta'_{u,j})_{j \in [c]})$.
 - B. If $\text{mv}_{\text{bFC}} \in I$, modify the salt $\eta'_{u,c}$ (e.g., by flipping a bit) to obtain a new move mv'_{bFC} not in I ; add mv'_{bFC} to I .
 - C. Make the move mv'_{bFC} in FCSRGame to get the answer $\text{vm}'_{u,c}$.
 - ii. Answer mv with $\text{rnd}_{k_{\text{FIOP}}+c}(\text{mv})$.
4. Let tr^{SR} be the query-answer trace of SRGame and $(\text{res}_a)_{a \in [\tilde{m}]}$ be the responses. Let the final output of $\tilde{\mathcal{P}}^{\text{SR}}$ be:

$$(\mathbb{x}, ((\text{cm}_u, \sigma_u))_{u \in [k_{\text{FIOP}}]}, ((\mathcal{Q}_u, \beta_u))_{u \in [k_{\text{FIOP}}]}, ((\text{pm}_{u,j}, \eta_{u,j}, \text{vm}_{u,j}))_{u \in [k_{\text{FIOP}}], j \in [k_{\text{bFC}}]}) \cdot$$

5. Let a be the index of the first move mv_a in tr^{SR} such that $\text{round}(\text{mv}_a) \leq k_{\text{FIOP}}$ and mv_a is of the form $(\mathbb{x}, ((\text{cm}_j, \sigma_j))_{j \in [i]}, ((\text{cm}'_j, \sigma'_j))_{j \in [u] \setminus [i]})$ for some $u \geq i$.
6. Simulate SRGame until $\tilde{\mathcal{P}}^{\text{SR}}$ outputs the a -th move by answering the first $a-1$ moves with $(\text{res}_u)_{u \in [a-1]}$ obtained from tr^{SR} .
7. Initialize $K_i := \emptyset$.
8. *Rewind to obtain query-answer pairs.* Sample $N_i \leftarrow [N]$ and repeat the following for N_i times.
 - (a) Continue the simulation of SRGame as follows and obtain the final output

$$(\mathbb{x}^*, ((\text{cm}^*_u, \sigma^*_u))_{u \in [k_{\text{FIOP}}]}, ((\mathcal{Q}^*_u, \beta^*_u))_{u \in [k_{\text{FIOP}}]}, ((\text{pm}^*_{u,j}, \eta^*_{u,j}))_{u \in [k_{\text{FIOP}}], j \in [k_{\text{bFC}}]}) \cdot$$

- i. For every $u \in [k]$, reinitialize the function $\text{rnd}_u : \{0, 1\}^* \rightarrow \{0, 1\}^{r_u}$ where

$$\text{rnd}_u(\text{mv}) := \begin{cases} \text{res}_v & \text{if mv is the } v\text{-th move of the simulation} \\ & \text{with } v \in [a-1] \text{ and } \text{round}(\text{mv}) = u; \\ \perp & \text{otherwise.} \end{cases}$$

- ii. When $\tilde{\mathcal{P}}^{\text{SR}}$ outputs a move of the form $\text{mv} := (\mathbb{x}', ((\text{cm}'_j, \sigma'_j))_{j \in [u]})$ for some $u \in [k_{\text{FIOP}}]$: if $\text{rnd}_u(\text{mv}) = \perp$ then sample $\rho \in \{0, 1\}^{r_u}$ and set $\text{rnd}_u(\text{mv}) := \rho$; either way, answer mv with $\text{rnd}_u(\text{mv})$.
- iii. When $\tilde{\mathcal{P}}^{\text{SR}}$ outputs a move mv of the form

$$(\mathbb{x}', ((\text{cm}'_u, \sigma'_u))_{u \in [k_{\text{FIOP}}]}, ((\mathcal{Q}'_u, \beta'_u))_{u \in [k_{\text{FIOP}}]}, ((\text{pm}'_{u,j}, \eta'_{u,j}))_{u \in [k_{\text{FIOP}}], j \in [c]})$$

for some $c \in [k_{\text{bFC}}]$:

- If $\text{rnd}_{k_{\text{FIOP}}+c}(\text{mv}) = \perp$ then set $\text{rnd}_{k_{\text{FIOP}}+c}(\text{mv}) := (\text{vm}'_{u,c})_{u \in [k_{\text{FIOP}}]}$ where, for every $u \in [k_{\text{FIOP}}]$, $\text{vm}'_{u,c}$ is obtained as follows:
 - * Set $\text{mv}_{\text{bFC}} := (\text{cm}'_u, \mathcal{Q}'_u, \beta'_u, (\text{pm}'_{u,j})_{j \in [c]}, (\eta'_{u,j})_{j \in [c]})$.
 - * If $\text{mv}_{\text{bFC}} \in I$, modify $\eta'_{u,c}$ to obtain a new move mv'_{bFC} not in I ; add mv_{bFC} to I .
 - * Make the move mv_{bFC} in FCSRGame to get the answer $\text{vm}'_{u,c}$.
 - Answer mv with $\text{rnd}_{k_{\text{FIOP}}+c}(\text{mv})$.
- (b) For every $c \in [k_{\text{bFC}}]$, let $(\text{vm}^*_{u,c})_{u \in [k_{\text{FIOP}}]}$ be the response of SRGame for move

$$(\mathbb{x}^*, ((\text{cm}^*_u, \sigma^*_u))_{u \in [k_{\text{FIOP}}]}, ((\mathcal{Q}^*_u, \beta^*_u))_{u \in [k_{\text{FIOP}}]}, ((\text{pm}^*_{u,j}, \eta^*_{u,j}))_{u \in [k_{\text{FIOP}}], j \in [c]}) .$$

- (c) If $\mathcal{V}_{\text{bFC}}(\text{pp}, \text{cm}_i, \mathcal{Q}^*_i, \beta^*_i, ((\text{pm}^*_{i,c}, \text{vm}^*_{i,c}))_{c \in [k_{\text{bFC}}]}) = 1$, add $(\mathcal{Q}^*_i, \beta^*_i, ((\text{pm}^*_{i,c}, \eta^*_{i,c}))_{c \in [k_{\text{bFC}}]})$ to K_i .

9. Solve for the FIOP strings. Run $\tilde{\Pi}_i \leftarrow \text{Solver}_{\mathbf{Q}_i}(((\alpha_j, \beta_j))_{j \in [u]})$ where $((\alpha_j, \beta_j))_{j \in [u]}$ are all query-answer pairs in K_i .
10. If $\tilde{\Pi}_i \neq \perp$, output the “dummy” tuple $(\text{cm}_1, \perp)$.
11. If $\tilde{\Pi}_i = \perp$, implying the gathered openings are inconsistent with any valid codeword (thus breaking function binding), add $(\mathcal{Q}_i, \beta_i, ((\text{pm}_{i,j}, \eta_{i,c}))_{c \in [k_{\text{bFC}}]})$ to K_i and output (cm_i, K_i) .

We discuss efficiency parameters of $\tilde{\mathcal{P}}^{\text{SR}}_{\text{bFC},i}$.

- *Salt size s_{bFC} of $\tilde{\mathcal{P}}^{\text{SR}}_{\text{bFC},i}$.* The algorithm $\tilde{\mathcal{P}}^{\text{SR}}_{\text{bFC},i}$ relies on having enough distinct salts, specifically at most $(k_{\text{FIOP}} \cdot N + 1) \cdot (m + k_{\text{bFC}})$ distinct salts to avoid duplicates in the set I (see Step 3c and Step 8(a)iii). This demands setting the salt size s_{bFC} larger than s , specifically at least $s + \log((k_{\text{FIOP}} \cdot N + 1) \cdot (m + k_{\text{bFC}}))$.
- *Move budget m_{bFC} .* In Step 3c and Step 8(a)iii, whenever $\tilde{\mathcal{P}}^{\text{SR}}$ makes a move of the form

$$(\mathbb{x}', ((\text{cm}'_u, \sigma'_u))_{u \in [k_{\text{FIOP}}]}, ((\mathcal{Q}'_u, \beta'_u))_{u \in [k_{\text{FIOP}}]}, ((\text{pm}'_{u,j}, \eta'_{u,j}))_{u \in [k_{\text{FIOP}}], j \in [c]})$$

for some $c \in [k_{\text{bFC}}]$, $\tilde{\mathcal{P}}^{\text{SR}}_{\text{bFC},i}$ may query bFCSRGame for the answer. Hence the move budget m_{bFC} is at most $(k_{\text{FIOP}} \cdot N + 1) \cdot (m + k_{\text{bFC}})$.

- *Running time t_{bFC} .*
 - Step 3 takes time at most $t_{\text{ARG}} + \hat{m}$.
 - Step 5 takes time $\hat{m}$.

- Step 8 takes time $N \cdot (t_{\text{ARG}} + \hat{m} + t_{\mathcal{V}_{\text{bFC}}})$.
- Step 9 takes time $t_{\mathbf{Q}}(\ell_i, N \cdot \mathbf{q}_i)$.

Hence the time of $\tilde{\mathcal{P}}_{\text{bFC},i}^{\text{SR}}$ is at most

$$\begin{aligned} t_{\text{bFC}} &\leq t_{\text{ARG}} + \hat{m} + (\hat{m} + N \cdot (t_{\text{ARG}} + \hat{m} + t_{\mathcal{V}_{\text{bFC}}}) + t_{\mathbf{Q}}(\ell_i, N \cdot \mathbf{q}_i)) \\ &= t_{\text{ARG}} + 2\hat{m} + N \cdot (t_{\text{ARG}} + \hat{m} + t_{\mathcal{V}_{\text{bFC}}}) + t_{\mathbf{Q}}(\ell_i, N \cdot \mathbf{q}_i) \\ &= O(N \cdot (t_{\text{ARG}} + m + k_{\text{FIOP}} + k_{\text{bFC}} + t_{\mathcal{V}_{\text{bFC}}})) + t_{\mathbf{Q}}(\ell_{\max}, N \cdot \mathbf{q}_{\max}) . \end{aligned}$$

We denote the experiment of the bFC state-restoration game as:

$$\text{Exp}_{\text{bFC},i} := \left[\begin{array}{l} \text{pp}_{\text{bFC}} \leftarrow \text{bFC.Gen}(1^\lambda, \ell_{\max}) \\ \text{ai}_{\text{bFC}} \leftarrow \mathcal{D}_{\text{bFC}} \\ \text{rnd}_{\text{bFC}} := (\text{rnd}_{\text{bFC},u})_{u \in [\mathbf{k}_{\text{bFC}}]} \leftarrow \mathcal{U}((r_{\text{bFC},u})_{u \in [\mathbf{k}_{\text{bFC}}]}) \\ (\text{cm}_i, K_i) \leftarrow \text{bFC SRGame}(s_{\text{bFC}}, \text{rnd}_{\text{bFC}}, \tilde{\mathcal{P}}_{\text{bFC},i}^{\text{SR}}, \text{pp}_{\text{bFC}}, \text{ai}_{\text{bFC}}) \\ \text{Parse } K_i \text{ as } ((\mathcal{Q}_j, \beta_j, ((\text{pm}_{j,u}, \eta_{j,u}, \text{vm}_{j,u}))_{u \in [\mathbf{k}_{\text{bFC}}]}))_{j \in [\mathbf{N}+1]} \end{array} \right] .$$

We bound

$$\begin{aligned} &\Pr \left[\exists i \in [\mathbf{k}_{\text{FIOP}}], \left(\begin{array}{l} S_i^* = \emptyset \\ \wedge \mathcal{V}_{\text{bFC}} \left(\begin{array}{l} \text{pp}, \text{cm}_i, \mathcal{Q}_i, \beta_i, \\ ((\text{pm}_{i,j}, \text{vm}_{i,j}))_{j \in [\mathbf{k}_{\text{bFC}}]} \end{array} \right) = 1 \end{array} \right) \right] \\ &\leq \sum_{i \in [\mathbf{k}_{\text{FIOP}}]} \Pr \left[\begin{array}{l} S_i^* = \emptyset \\ \wedge \mathcal{V}_{\text{bFC}} \left(\begin{array}{l} \text{pp}, \text{cm}_i, \mathcal{Q}_i, \beta_i, \\ ((\text{pm}_{i,j}, \text{vm}_{i,j}))_{j \in [\mathbf{k}_{\text{bFC}}]} \end{array} \right) = 1 \end{array} \right] \\ &\leq \sum_{i \in [\mathbf{k}_{\text{FIOP}}]} \Pr \left[\begin{array}{l} \left(\begin{array}{l} \forall j \in [\mathbf{N}+1] : \\ |\mathcal{Q}_j| \leq \mathbf{q}_{\max} \\ \wedge \mathcal{V}_{\text{bFC}} \left(\begin{array}{l} \text{pp}, \text{cm}_i, \mathcal{Q}_j, \beta_j, \\ ((\text{pm}_{j,u}, \text{vm}_{j,u}))_{u \in [\mathbf{k}_{\text{bFC}}]} \end{array} \right) = 1 \end{array} \right) \\ \wedge \nexists \Pi \text{ s.t. } \forall j \in [\mathbf{N}+1], \alpha \in \mathcal{Q}_j, \alpha(\Pi) = \beta_j^{(\alpha)} \end{array} \right] \Bigg| E_{\text{bFC},i} \\ &\leq \sum_{i \in [\mathbf{k}_{\text{FIOP}}]} \epsilon_{\text{bFC}}^{\text{SR}}(\lambda, \ell_i, \mathbf{q}_i, \mathbf{N}+1, s_{\text{bFC}}, \mathbf{m}_{\text{bFC}}, t_{\text{bFC}}) . \end{aligned}$$

where $s_{\text{bFC}}, \mathbf{m}_{\text{bFC}}, t_{\text{bFC}}$ satisfy the same conditions as in Lemma 6.1. The first inequality results from the union bound. The second inequality follows from the definition of the experiment $\text{Exp}_{\text{bFC},i}$ (implicitly invoking Construction 3.26). The final inequality is derived from the definition of state-restoration function binding (Definition 3.25).

6.2.2 Bound for Equation 13 from tail errors

We bound the probability in Equation 13 via the tail error of the query classes.

Recall from Construction 6.3 that for each round i , the final FIOP string $\tilde{\Pi}_i$ is selected from the extraction of a specific move a_i , i.e., $\tilde{\Pi}_i := \tilde{\Pi}_i^{(a_i)}$. We define a $\mathbf{q}_i$ -admissible distribution $\mathcal{D}_{(i, a_i, (\text{res}_j)_{j \in [a_i-1]})}$ as follows.

$$\mathcal{D}_{(i, a_i, (\text{res}_j)_{j \in [a_i-1]})} :$$

1. Simulate SRGame until $\tilde{\mathcal{P}}^{\text{SR}}$ outputs the a_i -th move mv by answering the first $a_i - 1$ moves with $(\text{res}_j)_{j \in [a-1]}$.
2. Let $(\mathbf{x}^*, ((\text{cm}_i^*, \sigma_i^*))_{i \in [\text{round}(\text{mv})]})$ be the a_i -th move mv of $\tilde{\mathcal{P}}^{\text{SR}}$.
3. Let $\mathbf{r} := \max(\max_{i \in [\mathbf{k}_{\text{FIOP}}]} r_{\text{FIOP}, i}, \max_{j \in [\mathbf{k}_{\text{bFC}}]} r_{\text{bFC}, j} \cdot \mathbf{k}_{\text{FIOP}})$, sample the reductor randomness

$$\boldsymbol{\rho} := (\rho_a)_{a \in [\hat{\mathbf{m}}]} \leftarrow \{0, 1\}^{\mathbf{r} \cdot \hat{\mathbf{m}}}.$$

4. Continue the simulation of SRGame with $\boldsymbol{\rho}$ and obtain the final output

$$(\mathbf{x}, ((\text{cm}_j, \sigma_j))_{j \in [\mathbf{k}_{\text{FIOP}}]}, ((\mathcal{Q}_j, \beta_j))_{j \in [\mathbf{k}_{\text{FIOP}}]}, ((\text{pm}_{j,u}, \eta_{j,u}))_{j \in [\mathbf{k}_{\text{FIOP}}], u \in [\mathbf{k}_{\text{bFC}}]}) .$$

5. For every $u \in [\mathbf{k}_{\text{bFC}}]$, let $(\text{vm}_{j,u})_{j \in [\mathbf{k}_{\text{FIOP}}]}$ be the response of SRGame for move

$$(\mathbf{x}, ((\text{cm}_j, \sigma_j))_{j \in [\mathbf{k}_{\text{FIOP}}]}, ((\mathcal{Q}_j, \beta_j))_{j \in [\mathbf{k}_{\text{FIOP}}]}, ((\text{pm}_{j,c}, \eta_{j,c}))_{i \in [\mathbf{k}_{\text{FIOP}}], c \in [u]}) .$$

6. If $\mathcal{V}_{\text{bFC}}(\text{pp}, \text{cm}_i^*, \mathcal{Q}_i, \beta_i, ((\text{pm}_{i,j}, \text{vm}_{i,j}))_{j \in [\mathbf{k}_{\text{bFC}}]}) = 1$, output $(\mathcal{Q}_i, \beta_i)$; otherwise output $\emptyset$.

Let Sampler be the sampling procedure in Definition 5.3. For every $i \in [\mathbf{k}_{\text{FIOP}}]$,

$$\begin{aligned} & \Pr \left[\begin{array}{l} S_i^* \neq \emptyset \\ \wedge S_i^* \neq S_i \\ \wedge \mathcal{V}_{\text{bFC}} \left(\text{pp}, \text{cm}_i, \mathcal{Q}_i, \beta_i, \right. \\ \left. ((\text{pm}_{i,j}, \text{vm}_{i,j}))_{j \in [\mathbf{k}_{\text{bFC}}]} \right) = 1 \end{array} \right] \\ & \leq \Pr \left[\begin{array}{l} S_{n+1} \neq \emptyset \\ \wedge S_{n+1} \neq S_n \end{array} \mid \begin{array}{l} n \leftarrow [\mathbf{N}] \\ (S_1, \dots, S_{n+1}) \leftarrow \text{Sampler}(n, \mathcal{D}_{(i, a_i, (\text{res}_j)_{j \in [a_i-1]})}) \end{array} \right] \\ & \leq \epsilon_{\mathbf{Q}_i}(\ell_i, \mathbf{q}_i, \mathbf{N}) . \end{aligned}$$

We conclude that

$$\begin{aligned} & \Pr \left[\begin{array}{l} S_i^* \neq \emptyset \\ \wedge S_i^* \neq S_i \\ \wedge \mathcal{V}_{\text{bFC}} \left(\text{pp}, \text{cm}_i, \mathcal{Q}_i, \beta_i, \right. \\ \left. ((\text{pm}_{i,j}, \text{vm}_{i,j}))_{j \in [\mathbf{k}_{\text{bFC}}]} \right) = 1 \end{array} \right] \\ & \leq \sum_{i \in [\mathbf{k}_{\text{FIOP}}]} \Pr \left[\begin{array}{l} S_i^* \neq \emptyset \\ \wedge S_i^* \neq S_i \\ \wedge \mathcal{V}_{\text{bFC}} \left(\text{pp}, \text{cm}_i, \mathcal{Q}_i, \beta_i, \right. \\ \left. ((\text{pm}_{i,j}, \text{vm}_{i,j}))_{j \in [\mathbf{k}_{\text{bFC}}]} \right) = 1 \end{array} \right] \\ & \leq \sum_{i \in [\mathbf{k}_{\text{FIOP}}]} \epsilon_{\mathbf{Q}_i}(\ell_i, \mathbf{q}_i, \mathbf{N}) . \end{aligned}$$

7 State-restoration security of the Funky protocol

We analyze the security of the compiled interactive argument $\text{ARG} := \text{Funky}[\text{FIOP}, \text{bFC}]$. Our main result is Theorem 7.1 (the formal version of Theorem 1), which establishes the state-restoration soundness (and knowledge soundness) of ARG.

Theorem 7.1. *Let $\mathbf{Q}_1, \dots, \mathbf{Q}_{k_{\text{FIOP}}}$ be query classes, where $\mathbf{Q}_i = \{\alpha : \Sigma^{\ell_i} \rightarrow \mathbb{D}\}$ has tail error $\epsilon_{\mathbf{Q}_i}$ and solving time $t_{\mathbf{Q}_i}$. Consider the following two ingredients.*

- $\text{FIOP} = (\mathbf{P}, \mathbf{V})$ is a public-coin FIOP with query classes $\mathbf{Q}_1, \dots, \mathbf{Q}_{k_{\text{FIOP}}}$ for a relation R with round complexity k_{FIOP} , alphabet Σ , proof length ℓ , and query complexity q . We denote by $\epsilon_{\text{FIOP}}^{\text{SR}}$ and $\kappa_{\text{FIOP}}^{\text{SR}}$ the state-restoration soundness and state-restoration knowledge soundness errors of FIOP, respectively.
- $\text{bFC} = (\text{bFC.Gen}, \text{bFC.Commit}, \mathcal{P}_{\text{bFC}}, \mathcal{V}_{\text{bFC}})$ is a k_{bFC} -round FC scheme with query classes $\mathbf{Q}_1, \dots, \mathbf{Q}_{k_{\text{FIOP}}}$. We denote by $\epsilon_{\text{bFC}}^{\text{SR}}$ the state-restoration function binding error of bFC.

Then $\text{ARG} = (\mathcal{G}, \mathcal{P}, \mathcal{V}) := \text{Funky}[\text{FIOP}, \text{bFC}]$ (Construction 4.1) is a $(2k_{\text{FIOP}} + 1 + 2k_{\text{bFC}})$ -message public-coin interactive argument system for R with state-restoration soundness error $\epsilon_{\text{ARG}}^{\text{SR}}$ and state-restoration knowledge soundness error $\kappa_{\text{ARG}}^{\text{SR}}$ such that for every $N \in \mathbb{N}$,

$$\begin{aligned} \epsilon_{\text{ARG}}^{\text{SR}}(\lambda, n, s, m, t_{\text{ARG}}) &\leq \epsilon_{\text{FIOP}}^{\text{SR}}(n, s + \lambda, m_{\text{FIOP}}) + \sum_{i \in [k_{\text{FIOP}}]} (\epsilon_{\text{bFC}}^{\text{SR}}(\lambda, \ell_i, q_i, N + 1, s_{\text{bFC}}, m_{\text{bFC}}, t_{\text{bFC}}) + \epsilon_{\mathbf{Q}_i}(\ell_i, q_i, N)) \text{ and} \\ \kappa_{\text{ARG}}^{\text{SR}}(\lambda, n, s, m, t_{\text{ARG}}) &\leq \kappa_{\text{FIOP}}^{\text{SR}}(n, s + \lambda, m_{\text{FIOP}}) + \sum_{i \in [k_{\text{FIOP}}]} (\epsilon_{\text{bFC}}^{\text{SR}}(\lambda, \ell_i, q_i, N + 1, s_{\text{bFC}}, m_{\text{bFC}}, t_{\text{bFC}}) + \epsilon_{\mathbf{Q}_i}(\ell_i, q_i, N)) . \end{aligned}$$

where

- $m_{\text{FIOP}} \leq m + k_{\text{FIOP}}$;
- $t_{\text{FIOP}} \leq (m + k_{\text{FIOP}}) \cdot (N \cdot (t_{\text{ARG}} + k_{\text{FIOP}} \cdot t_{\mathcal{V}_{\text{bFC}}}) + k_{\text{FIOP}} \cdot t_{\mathbf{Q}}(\ell_{\max}, N \cdot q_{\max}))$;
- $s_{\text{bFC}} \leq s + \log((k_{\text{FIOP}} \cdot N + 1) \cdot (m + k_{\text{bFC}}))$;
- $m_{\text{bFC}} \leq (k_{\text{FIOP}} \cdot N + 1) \cdot (m + k_{\text{bFC}})$; and
- $t_{\text{bFC}} \leq O(N \cdot (t_{\text{ARG}} + m + k_{\text{FIOP}} + k_{\text{bFC}} + t_{\mathcal{V}_{\text{bFC}}})) + t_{\mathbf{Q}}(\ell_{\max}, N \cdot q_{\max})$.

Moreover, ARG's extractor runs in time $t_{\mathcal{E}_{\text{SR}}}(\lambda, n, s, m, t_{\text{ARG}}) \leq t_{\mathcal{E}_{\text{SR}}}(n, s_{\text{FIOP}}, m_{\text{FIOP}}, t_{\text{FIOP}}) + t_{\text{FIOP}}$.

Corollary 7.2 (negligible regime). *Let ARG be defined as in Theorem 7.1. Assume that:*

- $\epsilon_{\text{bFC}}^{\text{SR}}(\lambda, \ell, L, s_{\text{bFC}}, m_{\text{bFC}}, t_{\text{bFC}}) = \text{negl}(\lambda)$ when $\ell, L, s_{\text{bFC}}, m_{\text{bFC}}, t_{\text{bFC}}$ are polynomials in λ ;
- for every polynomial p there is a polynomial p' such that $\epsilon_{\mathbf{Q}_i}(\ell, q, N) = 1/p(\lambda)$ for all $i \in [k_{\text{FIOP}}]$ and $N = p'(\lambda)$.

Then, if $m, t_{\text{ARG}}, \ell, q, k_{\text{FIOP}}$ are polynomials in λ , we have:

$$\begin{aligned} \epsilon_{\text{ARG}}^{\text{SR}}(\lambda, n, s, m, t_{\text{ARG}}) &\leq \epsilon_{\text{FIOP}}^{\text{SR}}(n, s + \lambda, m_{\text{FIOP}}) + \text{negl}(\lambda) \text{ and} \\ \kappa_{\text{ARG}}^{\text{SR}}(\lambda, n, s, m, t_{\text{ARG}}) &\leq \kappa_{\text{FIOP}}^{\text{SR}}(n, s + \lambda, m_{\text{FIOP}}) + \text{negl}(\lambda) . \end{aligned}$$

Proof. Suppose $k_{\text{FIOP}} = p_1(\lambda)$ for some polynomial p_1 . Let $p_2(\lambda)$ be an arbitrary polynomial. Choose $N = \text{poly}(\lambda)$ such that $\epsilon_{\mathbf{Q}_i}(\ell, q, N) \leq \frac{1}{2p_1(\lambda)p_2(\lambda)}$ for all $i \in [k_{\text{FIOP}}]$. Hence, $m_{\text{bFC}} = \text{poly}(\lambda)$ and $t_{\text{bFC}} = \text{poly}(\lambda)$. Therefore,

$$\epsilon_{\text{ARG}}^{\text{SR}}(\lambda, n, s, m, t_{\text{ARG}}) \leq \epsilon_{\text{FIOP}}^{\text{SR}}(n, s + \lambda, m_{\text{FIOP}}) + \text{negl}(\lambda) + \frac{1}{2p_2(\lambda)}$$

$$< \epsilon_{\text{FIOP}}^{\text{SR}}(n, s + \lambda, \mathbf{m}_{\text{FIOP}}) + \text{negl}(\lambda) + \frac{1}{p_2(\lambda)} .$$

Since p_2 is an arbitrary polynomial, we conclude that

$$\epsilon_{\text{ARG}}^{\text{SR}}(\lambda, n, s, \mathbf{m}, t_{\text{ARG}}) \leq \epsilon_{\text{FIOP}}^{\text{SR}}(n, s + \lambda, \mathbf{m}_{\text{FIOP}}) + \text{negl}(\lambda) .$$

An analogous argument holds for $\kappa_{\text{ARG}}^{\text{SR}}$. □

Roadmap. The proof of Theorem 7.1 proceeds in three steps:

- **Adversary Simulation (Section 7.1):** We construct an FIOP state-restoration adversary $\tilde{\mathbf{P}}^{\text{SR}}$ that internally simulates the argument adversary $\tilde{\mathcal{P}}^{\text{SR}}$ by invoking the reducers. This construction bridges the gap between the compiled argument and the underlying FIOP.
- **Soundness Analysis (Section 7.2):** We define a state-restoration experiment and decompose the malicious prover's success probability into two disjoint events: (1) the underlying FIOP state-restoration adversary does not succeed, whose probability is bounded by the reduction error via Lemma 6.1, and (2) a soundness failure of the underlying FIOP (bounded by $\epsilon_{\text{FIOP}}^{\text{SR}}$).
- **Knowledge Soundness Analysis (Section 7.3):** Similarly, we show that any malicious prover breaking argument knowledge soundness either triggers a reducer failure or can be used to extract a witness via the underlying FIOP extractor.

7.1 Construction of the FIOP state-restoration adversary

To prove soundness, we first need to bridge the gap between the compiled argument and the underlying FIOP. We achieve this by constructing an FIOP state-restoration adversary $\tilde{\mathbf{P}}^{\text{SR}}$.

The strategy of $\tilde{\mathbf{P}}^{\text{SR}}$ is to simulate the execution of $\tilde{\mathcal{P}}^{\text{SR}}$ and translate its commitment-based moves into FIOP messages by invoking the reducers $\mathfrak{R}_m$ (Construction 6.2) and $\mathfrak{R}$ (Construction 6.3). Crucially, to ensure consistency between the extracted messages and the random oracle queries across different execution branches, $\tilde{\mathbf{P}}^{\text{SR}}$ employs *memoization* to reuse previously extracted FIOP strings for identical commitment prefixes.

Construction 7.3. Let $\mathcal{D}$ be the auxiliary input distribution for ARG. The auxiliary input distribution $\mathbf{D}$ for FIOP is defined as follows.

D:

1. Compute $\text{pp} \leftarrow \mathcal{G}(1^\lambda, n)$.
2. Sample $\text{ai} \leftarrow \mathcal{D}$.
3. Output $\text{ai} := (\text{pp}, \text{ai})$.

Construction 7.4. The FIOP state-restoration prover $\tilde{\mathbf{P}}^{\text{SR}}$ is defined as follows.

$\tilde{\mathbf{P}}^{\text{SR}}(\text{ai}):$

1. Parse ai as (pp, ai) .
2. Simulate SRGame with $\tilde{\mathcal{P}}^{\text{SR}}(\text{pp}, \text{ai})$ as follows.
 - (a) For every $i \in [k]$, initialize an empty partial function $\text{rnd}_i: \{0, 1\}^* \rightarrow \{0, 1\}^{r_i}$.
 - (b) Initialize an empty partial function $M: \{0, 1\}^* \rightarrow \{0, 1\}^*$.

(c) For $a \in [\hat{m}]$, when $\tilde{\mathcal{P}}^{\text{SR}}$ makes its a -th move mv_a :

i. If mv_a has the form $(\mathbb{x}', ((\text{cm}'_j, \sigma'_j))_{j \in [i]})$ for some $i \in [\text{k}_{\text{FIOP}}]$:

A. If $\text{rnd}_i(\text{mv}_a) = \perp$:

- Run $(\tilde{\Pi}_j^{(a)})_{j \in [i]} := \mathfrak{R}_m^{\tilde{\mathcal{P}}^{\text{SR}}(\text{pp}, \text{ai})}(a, (\text{res}_i)_{i \in [a-1]})$ (Construction 6.2).
- For every $j \in [i]$, if $M((\mathbb{x}', ((\text{cm}'_u, \sigma'_u))_{u \in [j]})) = \perp$, set $M((\mathbb{x}', ((\text{cm}'_u, \sigma'_u))_{u \in [j]})) = \tilde{\Pi}_j^{(a)}$; otherwise, set $\tilde{\Pi}_j^{(a)} := M((\mathbb{x}', ((\text{cm}'_u, \sigma'_u))_{u \in [j]}))$. (*Memoization step*)
- For every $j \in [i]$, set the FIOP salt string $\gamma_j := (\text{cm}'_j, \sigma'_j)$.
- Make the FIOP move $(\mathbb{x}', (\tilde{\Pi}_j^{(a)})_{j \in [i]}, (\gamma_j)_{j \in [i]})$ in FIOPSRGame to get an answer ρ_a .
- Set $\text{rnd}_i(\text{mv}_a) := \rho_a$.

B. Answer mv with $\text{rnd}_i(\text{mv})$.

ii. Otherwise, mv_a has the form

$$(\mathbb{x}', ((\text{cm}'_u, \sigma'_u))_{u \in [\text{k}_{\text{FIOP}}]}, ((\mathcal{Q}'_u, \beta'_u))_{u \in [\text{k}_{\text{FIOP}}]}, ((\text{pm}'_{i,u}, \eta'_{i,u}))_{i \in [\text{k}_{\text{FIOP}}], u \in [j]})$$

for some $j \in [\text{k}_{\text{bFC}}]$:

A. If $\text{rnd}_{\text{k}_{\text{FIOP}}+j}(\text{mv}_a) = \perp$, sample $\rho \in \{0, 1\}^{\text{r}_{\text{k}_{\text{FIOP}}+j}}$ and set $\text{rnd}_{\text{k}_{\text{FIOP}}+j}(\text{mv}_a) := \rho$.

B. Set $(\tilde{\Pi}_i^{(a)})_{i \in [\text{round}(\text{mv}_a)]} := \perp$ and answer mv_a with $\text{rnd}_{\text{k}_{\text{FIOP}}+j}(\text{mv}_a)$.

3. Let tr^{SR} be the move-response trace of SRGame and let the final output of $\tilde{\mathcal{P}}^{\text{SR}}$ be:

$$(\mathbb{x}, ((\text{cm}_i, \sigma_i))_{i \in [\text{k}_{\text{FIOP}}]}, ((\mathcal{Q}_i, \beta_i))_{i \in [\text{k}_{\text{FIOP}}]}, ((\text{pm}_{i,j}, \eta_{i,j}))_{i \in [\text{k}_{\text{FIOP}}], j \in [\text{k}_{\text{bFC}}]}) .$$

4. Run $(\tilde{\Pi}_i)_{i \in [\text{k}_{\text{FIOP}}]} \leftarrow \mathfrak{R}(\text{tr}^{\text{SR}}, (\mathbb{x}, ((\text{cm}_i, \sigma_i))_{i \in [\text{k}_{\text{FIOP}}]}), (\tilde{\Pi}_i^{(a)})_{a \in [\hat{m}], i \in [\text{round}(\text{mv}_a)]})$.

5. For every $i \in [\text{k}_{\text{FIOP}}]$, set the FIOP salt string $\gamma_i := (\text{cm}_i, \sigma_i)$.

6. Output $(\mathbb{x}, (\tilde{\Pi}_i)_{i \in [\text{k}_{\text{FIOP}}]}, (\gamma_i)_{i \in [\text{k}_{\text{FIOP}}]})$.

We compute the efficiency parameters for $\tilde{\mathcal{P}}^{\text{SR}}$.

- *Salt size s_{FIOP} of $\tilde{\mathcal{P}}^{\text{SR}}$.* Salt strings are used in Step 2(c)iA and the salt size s_{FIOP} is at most $s + \lambda$.
- *Move budget m_{FIOP} .* When $\tilde{\mathcal{P}}^{\text{SR}}$ makes a move of the form $(\mathbb{x}', ((\text{cm}'_j, \sigma'_j))_{j \in [i]})$ for some $i \in [\text{k}]$, $\tilde{\mathcal{P}}^{\text{SR}}$ may make a move in FIOPSRGame (see Step 2(c)iA). Hence the move budget m_{FIOP} is at most $\text{m} + \text{k}_{\text{FIOP}}$.
- *Running time t_{FIOP} .* When $\tilde{\mathcal{P}}^{\text{SR}}$ makes a move of the form $(\mathbb{x}', ((\text{cm}'_j, \sigma'_j))_{j \in [i]})$ for some $i \in [\text{k}]$, $\tilde{\mathcal{P}}^{\text{SR}}$ may run the reductor $\mathfrak{R}_m$ (Step 2(c)iA). From Construction 6.2, the running time of $\tilde{\mathcal{P}}^{\text{SR}}$ is at most

$$t_{\text{FIOP}} \leq (\text{m} + \text{k}_{\text{FIOP}}) \cdot (\text{N} \cdot (t_{\text{ARG}} + \text{k}_{\text{FIOP}} \cdot t_{\mathcal{V}_{\text{bFC}}}) + \text{k}_{\text{FIOP}} \cdot t_{\mathbf{Q}}(\ell_{\text{max}}, \text{N} \cdot \mathbf{q}_{\text{max}})) .$$

7.2 State-restoration soundness

By Definition 3.8, we wish to upper-bound the following expression:

$$\Pr \left[\begin{array}{l} |\mathbb{x}| \leq n \\ \wedge \mathbb{x} \notin L(R) \\ \wedge \mathcal{V} \left(\begin{array}{l} \text{pp}, \mathbb{x}, (\text{cm}_i)_{i \in [\text{k}_{\text{FIOP}}]}, \\ ((\mathcal{Q}_i, \beta_i))_{i \in [\text{k}_{\text{FIOP}}]}, (\rho_i)_{i \in [\text{k}_{\text{FIOP}}]}, \\ ((\text{pm}_{i,j}, \text{vm}_{i,j}))_{i \in [\text{k}_{\text{FIOP}}], j \in [\text{k}_{\text{bFC}}]} \end{array} \right) \end{array} \right] = 1 \quad \left| \begin{array}{l} \text{pp} \leftarrow \mathcal{G}(1^\lambda, n) \\ \text{ai} \leftarrow \mathcal{D} \\ \text{rnd} := (\text{rnd}_i)_{i \in [\text{k}_{\text{FIOP}} + \text{k}_{\text{bFC}}]} \leftarrow \mathcal{U}((r_i)_{i \in [\text{k}_{\text{FIOP}} + \text{k}_{\text{bFC}}]}) \\ \left(\begin{array}{l} \mathbb{x}, ((\text{cm}_i, \sigma_i))_{i \in [\text{k}_{\text{FIOP}}]}, ((\mathcal{Q}_i, \beta_i))_{i \in [\text{k}_{\text{FIOP}}]}, \\ ((\text{pm}_{i,j}, \eta_{i,j}))_{i \in [\text{k}_{\text{FIOP}}], j \in [\text{k}_{\text{bFC}}]}, \\ (\rho_i)_{i \in [\text{k}_{\text{FIOP}}]}, (\text{vm}_{i,j})_{i \in [\text{k}_{\text{FIOP}}], j \in [\text{k}_{\text{bFC}}]} \end{array} \right) \\ \xleftarrow{\text{tr}^{\text{SR}}} \text{SRGame}(s, \text{rnd}, \tilde{\mathcal{P}}^{\text{SR}}, \text{pp}, \text{ai}) \end{array} \right] . \quad (14)$$

We augment the experiment above by running $\mathfrak{R}_m$ (Construction 6.2) for every $a \in [\hat{m}]$ and $\mathfrak{R}$ as follows, where the augmentations are shown in blue.

$$\left[\begin{array}{l} \text{pp} \leftarrow \mathcal{G}(1^\lambda, n) \\ \text{ai} \leftarrow \mathcal{D} \\ \text{rnd} := (\text{rnd}_i)_{i \in [\text{k}_{\text{FIOP}} + \text{k}_{\text{bFC}}]} \leftarrow \mathcal{U}((r_i)_{i \in [\text{k}_{\text{FIOP}} + \text{k}_{\text{bFC}}]}) \\ \left(\begin{array}{l} \mathbb{X}, ((\text{cm}_i, \sigma_i))_{i \in [\text{k}_{\text{FIOP}}]}, ((\mathcal{Q}_i, \beta_i))_{i \in [\text{k}_{\text{FIOP}}]}, \\ ((\text{pm}_{i,j}, \eta_{i,j}))_{i \in [\text{k}_{\text{FIOP}}], j \in [\text{k}_{\text{bFC}}]}, \\ (\rho_i)_{i \in [\text{k}_{\text{FIOP}}]}, (\text{vm}_{i,j})_{i \in [\text{k}_{\text{FIOP}}], j \in [\text{k}_{\text{bFC}}]} \end{array} \right) \xleftarrow{\text{tr}^{\text{SR}}} \text{SRGame}(s, \text{rnd}, \tilde{\mathcal{P}}^{\text{SR}}, \text{pp}, \text{ai}) \\ \text{For } a \in [\hat{m}] : (\tilde{\Pi}_i^{(a)})_{i \in [\text{round}(\text{mv}_a)]} \leftarrow \mathfrak{R}_m^{\tilde{\mathcal{P}}^{\text{SR}}(\text{pp}, \text{ai})}(a, (\text{res}_i)_{i \in [a-1]}) \\ (\tilde{\Pi}_i)_{i \in [\text{k}_{\text{FIOP}}]} \leftarrow \mathfrak{R}(\text{tr}^{\text{SR}}, (\mathbb{X}, ((\text{cm}_i, \sigma_i))_{i \in [\text{k}_{\text{FIOP}}]}), (\tilde{\Pi}_i^{(a)})_{a \in [\hat{m}], i \in [\text{round}(\text{mv}_a)]}) \end{array} \right] .$$

Throughout the proof, probabilities are with respect to the above experiment unless stated otherwise. Note that it is the same experiment as Experiment 10.

To simplify the notation, we define the event E_{Π} as

$$E_{\Pi} \iff \mathbf{V}^{(\tilde{\Pi}_i)_{i \in [\text{k}_{\text{FIOP}}]}}(\mathbb{X}; (\rho_i)_{i \in [\text{k}_{\text{FIOP}}]}) = 1 ,$$

and the event E_{sound} as:

$$\begin{aligned} E_{\text{sound}} &\iff \left[\begin{array}{l} |\mathbb{X}| \leq n \\ \wedge \mathbb{X} \notin L(R) \\ \wedge \mathcal{V} \left(\begin{array}{l} \text{pp}, \mathbb{X}, (\text{cm}_i)_{i \in [\text{k}_{\text{FIOP}}]}, \\ ((\mathcal{Q}_i, \beta_i))_{i \in [\text{k}_{\text{FIOP}}]}, (\rho_i)_{i \in [\text{k}_{\text{FIOP}}]}, \\ ((\text{pm}_{i,j}, \text{vm}_{i,j}))_{i \in [\text{k}_{\text{FIOP}}], j \in [\text{k}_{\text{bFC}}]} \end{array} \right) = 1 \end{array} \right] \\ &\iff \left[\begin{array}{l} |\mathbb{X}| \leq n \\ \wedge \mathbb{X} \notin L(R) \\ \wedge \mathbf{V}^{((\mathcal{Q}_i, \beta_i))_{i \in [\text{k}_{\text{FIOP}}]}}(\mathbb{X}; (\rho_i)_{i \in [\text{k}_{\text{FIOP}}]}) = 1 \\ \wedge \left(\forall i \in [\text{k}_{\text{FIOP}}] : \mathcal{V}_{\text{bFC}} \left(\begin{array}{l} \text{pp}, \text{cm}_i, \mathcal{Q}_i, \beta_i, \\ ((\text{pm}_{i,j}, \text{vm}_{i,j}))_{j \in [\text{k}_{\text{bFC}}]} \end{array} \right) = 1 \right) \end{array} \right] . \end{aligned}$$

By the law of total probability,

$$\Pr[E_{\text{sound}}] \leq \Pr[E_{\text{sound}} \wedge \neg E_{\Pi}] + \Pr[E_{\text{sound}} \wedge E_{\Pi}] .$$

We bound the former term and then the latter term.

Bound from security reduction. Note that $E_{\text{sound}} \wedge \neg E_{\Pi}$ directly implies $E_{\text{reduction}}$ in Event 11. Therefore, according to Lemma 6.1,

$$\begin{aligned} &\Pr[E_{\text{sound}} \wedge \neg E_{\Pi}] \\ &\leq \Pr[E_{\text{reduction}}] \\ &\leq \sum_{i \in [\text{k}_{\text{FIOP}}]} (\epsilon_{\text{bFC}}^{\text{SR}}(\lambda, \ell_i, \mathbf{q}_i, \mathbf{N} + 1, s_{\text{bFC}}, \mathbf{m}_{\text{bFC}}, t_{\text{bFC}}) + \epsilon_{\mathbf{Q}_i}(\ell_i, \mathbf{q}_i, \mathbf{N})) . \end{aligned}$$

Bound from FIOP state-restoration soundness. Let $\mathbf{D}$ be as in Construction 7.3 and let $\tilde{\mathbf{P}}^{\text{SR}}$ be the FIOP state-restoration adversary in Construction 7.4. We have

$$\begin{aligned} & \Pr [E_{\text{sound}} \wedge E_{\Pi}] \\ & \leq \Pr \left[\begin{array}{l} |\mathbb{X}| \leq n \\ \wedge \mathbb{X} \notin L(R) \\ \wedge \mathbf{V}(\mathbb{X}, (\tilde{\Pi}_i)_{i \in [\text{k}_{\text{FIOP}}]}, (\rho_i)_{i \in [\text{k}_{\text{FIOP}}]}) = 1 \end{array} \right] \left[\begin{array}{l} \mathbf{ai} \leftarrow \mathbf{D} \\ \text{rnd}_{\text{FIOP}} := (\text{rnd}_{\text{FIOP},i})_{i \in [\text{k}_{\text{FIOP}}]} \leftarrow \mathcal{U}((r_{\text{FIOP},i})_{i \in [\text{k}_{\text{FIOP}}]}) \\ (\mathbb{X}, (\tilde{\Pi}_i)_{i \in [\text{k}_{\text{FIOP}}]}, (\gamma_i)_{i \in [\text{k}_{\text{FIOP}}]}, (\rho_i)_{i \in [\text{k}_{\text{FIOP}}]}) \\ \leftarrow \text{FIOPSRGame}(s_{\text{FIOP}}, \text{rnd}_{\text{FIOP}}, \tilde{\mathbf{P}}^{\text{SR}}, \mathbf{ai}) \end{array} \right] \\ & \leq \epsilon_{\text{FIOP}}^{\text{SR}}(n, s_{\text{FIOP}}, m_{\text{FIOP}}) . \end{aligned}$$

The first inequality holds because the randomness computed by the FIOP random oracles correctly simulates that of the argument random oracles due to memoization. The second inequality follows directly from Definition 3.15.

7.3 State-restoration knowledge soundness

By Definition 3.9, we wish to upper-bound the following expression:

$$\Pr \left[\begin{array}{l} |\mathbb{X}| \leq n \\ \wedge (\mathbb{X}, \mathbb{W}) \notin R \\ \wedge \mathcal{V} \left(\begin{array}{l} \text{pp}, \mathbb{X}, (\text{cm}_i)_{i \in [\text{k}_{\text{FIOP}}]}, \\ ((\mathcal{Q}_i, \beta_i))_{i \in [\text{k}_{\text{FIOP}}]}, (\rho_i)_{i \in [\text{k}_{\text{FIOP}}]}, \\ ((\text{pm}_{i,j}, \text{vm}_{i,j}))_{i \in [\text{k}_{\text{FIOP}}], j \in [\text{k}_{\text{bFC}}]} \end{array} \right) = 1 \end{array} \right] \left[\begin{array}{l} \text{pp} \leftarrow \mathcal{G}(1^\lambda, n) \\ \mathbf{ai} \leftarrow \mathcal{D} \\ \text{rnd} := (\text{rnd}_i)_{i \in [\text{k}_{\text{FIOP}} + \text{k}_{\text{bFC}}]} \leftarrow \mathcal{U}((r_i)_{i \in [\text{k}_{\text{FIOP}} + \text{k}_{\text{bFC}}]}) \\ \left(\begin{array}{l} \mathbb{X}, ((\text{cm}_i, \sigma_i))_{i \in [\text{k}_{\text{FIOP}}]}, ((\mathcal{Q}_i, \beta_i))_{i \in [\text{k}_{\text{FIOP}}]}, \\ ((\text{pm}_{i,j}, \eta_{i,j}))_{i \in [\text{k}_{\text{FIOP}}], j \in [\text{k}_{\text{bFC}}]}, \\ (\rho_i)_{i \in [\text{k}_{\text{FIOP}}]}, (\text{vm}_{i,j})_{i \in [\text{k}_{\text{FIOP}}], j \in [\text{k}_{\text{bFC}}]} \end{array} \right) \\ \xleftarrow{\text{tr}^{\text{SR}}} \text{SRGame}(s, \text{rnd}, \tilde{\mathcal{P}}^{\text{SR}}, \text{pp}, \mathbf{ai}) \\ \mathbb{W} \leftarrow \mathcal{E}_{\text{SR}}^{\tilde{\mathcal{P}}^{\text{SR}}(\text{pp}, \mathbf{ai})} \left(\begin{array}{l} \mathbb{X}, ((\text{cm}_i, \sigma_i))_{i \in [\text{k}_{\text{FIOP}}]}, \\ ((\mathcal{Q}_i, \beta_i))_{i \in [\text{k}_{\text{FIOP}}]}, \\ ((\text{pm}_{i,j}, \eta_{i,j}))_{i \in [\text{k}_{\text{FIOP}}], j \in [\text{k}_{\text{bFC}}]}, \\ (\rho_i)_{i \in [\text{k}_{\text{FIOP}}]}, (\text{vm}_{i,j})_{i \in [\text{k}_{\text{FIOP}}], j \in [\text{k}_{\text{bFC}}]}, \\ \text{tr}^{\text{SR}} \end{array} \right) \end{array} \right] . \quad (15)$$

Similarly to Section 7.2, we augment the experiment above by running $\mathfrak{R}_m$ (Construction 6.2) for every $a \in [\hat{m}]$ and $\mathfrak{R}$ as follows, where the augmentations are shown in blue.

$$\left[\begin{array}{l} \text{pp} \leftarrow \mathcal{G}(1^\lambda, n) \\ \mathbf{ai} \leftarrow \mathcal{D} \\ \text{rnd} := (\text{rnd}_i)_{i \in [\text{k}_{\text{FIOP}} + \text{k}_{\text{bFC}}]} \leftarrow \mathcal{U}((r_i)_{i \in [\text{k}_{\text{FIOP}} + \text{k}_{\text{bFC}}]}) \\ \left(\begin{array}{l} \mathbb{X}, ((\text{cm}_i, \sigma_i))_{i \in [\text{k}_{\text{FIOP}}]}, ((\mathcal{Q}_i, \beta_i))_{i \in [\text{k}_{\text{FIOP}}]}, \\ ((\text{pm}_{i,j}, \eta_{i,j}))_{i \in [\text{k}_{\text{FIOP}}], j \in [\text{k}_{\text{bFC}}]}, \\ (\rho_i)_{i \in [\text{k}_{\text{FIOP}}]}, (\text{vm}_{i,j})_{i \in [\text{k}_{\text{FIOP}}], j \in [\text{k}_{\text{bFC}}]} \end{array} \right) \xleftarrow{\text{tr}^{\text{SR}}} \text{SRGame}(s, \text{rnd}, \tilde{\mathcal{P}}^{\text{SR}}, \text{pp}, \mathbf{ai}) \\ \mathbb{W} \leftarrow \mathcal{E}_{\text{SR}}^{\tilde{\mathcal{P}}^{\text{SR}}(\text{pp}, \mathbf{ai})} \left(\begin{array}{l} \mathbb{X}, ((\text{cm}_i, \sigma_i))_{i \in [\text{k}_{\text{FIOP}}]}, ((\mathcal{Q}_i, \beta_i))_{i \in [\text{k}_{\text{FIOP}}]}, \\ ((\text{pm}_{i,j}, \eta_{i,j}))_{i \in [\text{k}_{\text{FIOP}}], j \in [\text{k}_{\text{bFC}}]}, \\ (\rho_i)_{i \in [\text{k}_{\text{FIOP}}]}, (\text{vm}_{i,j})_{i \in [\text{k}_{\text{FIOP}}], j \in [\text{k}_{\text{bFC}}]}, \text{tr}^{\text{SR}} \end{array} \right) \\ \text{For } a \in [\hat{m}] : (\tilde{\Pi}_i^{(a)})_{i \in [\text{round}(\text{mv}_a)]} \leftarrow \mathfrak{R}_m^{\tilde{\mathcal{P}}^{\text{SR}}(\text{pp}, \mathbf{ai})}(a, (\text{res}_i)_{i \in [a-1]}) \\ (\tilde{\Pi}_i)_{i \in [\text{k}_{\text{FIOP}}]} \leftarrow \mathfrak{R}(\text{tr}^{\text{SR}}, (\mathbb{X}, ((\text{cm}_i, \sigma_i))_{i \in [\text{k}_{\text{FIOP}}]}), (\tilde{\Pi}_i^{(a)})_{a \in [\hat{m}], i \in [\text{round}(\text{mv}_a)]}) \end{array} \right] .$$

Throughout the proof, probabilities are with respect to the above experiment unless stated otherwise.

Similarly, we define the event E_{ks} as:

$$E_{\text{ks}} \iff \left[\begin{array}{l} |\mathbb{X}| \leq n \\ \wedge (\mathbb{X}, \mathbb{W}) \notin R \\ \wedge \mathcal{V} \left(\begin{array}{l} \text{pp}, \mathbb{X}, (\text{cm}_i)_{i \in [\text{k}_{\text{FIOP}}]}, \\ ((\mathcal{Q}_i, \beta_i))_{i \in [\text{k}_{\text{FIOP}}]}, (\rho_i)_{i \in [\text{k}_{\text{FIOP}}]}, \\ ((\text{pm}_{i,j}, \text{vm}_{i,j}))_{i \in [\text{k}_{\text{FIOP}}], j \in [\text{k}_{\text{bFC}}]} \end{array} \right) = 1 \end{array} \right] \\ \iff \left[\begin{array}{l} |\mathbb{X}| \leq n \\ \wedge (\mathbb{X}, \mathbb{W}) \notin R \\ \wedge \mathbf{V}^{((\mathcal{Q}_i, \beta_i))_{i \in [\text{k}_{\text{FIOP}}]} (\mathbb{X}; (\rho_i)_{i \in [\text{k}_{\text{FIOP}}]}) = 1 \\ \wedge \left(\begin{array}{l} \forall i \in [\text{k}_{\text{FIOP}}] : \\ \mathcal{V}_{\text{bFC}} \left(\begin{array}{l} \text{pp}, \text{cm}_i, \mathcal{Q}_i, \beta_i, \\ ((\text{pm}_{i,j}, \text{vm}_{i,j}))_{j \in [\text{k}_{\text{bFC}}]} \end{array} \right) = 1 \end{array} \right) \end{array} \right] .$$

By the law of total probability,

$$\Pr[E_{\text{ks}}] \leq \Pr[E_{\text{ks}} \wedge \neg E_{\text{II}}] + \Pr[E_{\text{ks}} \wedge E_{\text{II}}] .$$

We bound the former term and then the latter term.

Bound from security reduction. Note that $E_{\text{ks}} \wedge \neg E_{\text{II}}$ directly implies $E_{\text{reduction}}$ in Event 11. Therefore, according to Lemma 6.1,

$$\begin{aligned} & \Pr[E_{\text{ks}} \wedge \neg E_{\text{II}}] \\ & \leq \Pr[E_{\text{reduction}}] \\ & \leq \sum_{i \in [\text{k}_{\text{FIOP}}]} (\epsilon_{\text{bFC}}^{\text{SR}}(\lambda, \ell_i, \mathbf{q}_i, \mathbf{N} + 1, s_{\text{bFC}}, \mathbf{m}_{\text{bFC}}, t_{\text{bFC}}) + \epsilon_{\mathbf{Q}_i}(\ell_i, \mathbf{q}_i, \mathbf{N})) . \end{aligned}$$

Bound from FIOP state-restoration knowledge soundness. Let $\mathbf{D}$ be defined as in Construction 7.3, $\tilde{\mathbf{P}}^{\text{SR}}$ be the FIOP state-restoration adversary constructed in Construction 7.4. We construct the state-restoration knowledge extractor $\mathcal{E}_{\text{SR}}$ for ARG, which invokes the underlying FIOP extractor $\mathbf{E}_{\text{SR}}$ by giving it oracle access to $\tilde{\mathbf{P}}^{\text{SR}}$ (from Section 7.1) and the simulation of the FIOP state-restoration game.

Construction 7.5. Let $\mathbf{E}_{\text{SR}}$ be the state-restoration extractor for FIOP. The state-restoration knowledge extractor $\mathcal{E}_{\text{SR}}$ for ARG is as follows.

$\mathcal{E}_{\text{SR}}^{\tilde{\mathcal{P}}^{\text{SR}}(\text{pp}, \mathbf{ai})}(\mathbb{X}, ((\text{cm}_i, \sigma_i))_{i \in [\text{k}_{\text{FIOP}}]}, ((\mathcal{Q}_i, \beta_i))_{i \in [\text{k}_{\text{FIOP}}]}, (\rho_i)_{i \in [\text{k}_{\text{FIOP}}]}, (\text{pm}_{i,j}, \eta_{i,j}, \text{vm}_{i,j})_{i \in [\text{k}_{\text{FIOP}}], j \in [\text{k}_{\text{bFC}}]}, \text{tr}^{\text{SR}}):$

1. Set $\mathbf{ai} := (\text{pp}, \mathbf{ai})$.
2. Construct $\tilde{\mathbf{P}}^{\text{SR}}$ using $\tilde{\mathcal{P}}^{\text{SR}}$ as in Construction 7.4.
3. Simulate FIOPSRGame with $\tilde{\mathbf{P}}^{\text{SR}}(\mathbf{ai})$ as follows.
4. Whenever $\tilde{\mathbf{P}}^{\text{SR}}$ makes a move of the form $(\mathbb{X}', (\tilde{\Pi}_j^{(a)})_{j \in [i]}, (\gamma_j)_{j \in [i]})$ for some $i \in [\text{k}]$:
 - (a) Parse $(\gamma_j)_{j \in [i]}$ as $((\text{cm}'_j, \sigma'_j))_{j \in [i]}$.
 - (b) Set $\text{mv}' := (\mathbb{X}', ((\text{cm}'_j, \sigma'_j))_{j \in [i]})$.
 - (c) Answer mv' with the corresponding response in tr^{SR} .
5. Let $\text{tr}_{\text{FIOP}}^{\text{SR}}$ be the move-response trace of FIOPSRGame and let the final output of $\tilde{\mathbf{P}}^{\text{SR}}$ be:

$$(\mathbb{X}, (\tilde{\Pi}_i)_{i \in [\text{k}_{\text{FIOP}}]}, (\gamma_i)_{i \in [\text{k}_{\text{FIOP}}]}, (\rho_i)_{i \in [\text{k}_{\text{FIOP}}]}) .$$

6. Compute $\mathbb{w} \leftarrow \mathbf{E}_{\text{SR}}^{\tilde{\mathbf{P}}^{\text{SR}}(\mathbf{ai})}(\mathbb{x}, (\tilde{\Pi}_i)_{i \in [k_{\text{FIOP}}]}, (\gamma_i)_{i \in [k_{\text{FIOP}}]}, (\rho_i)_{i \in [k_{\text{FIOP}}]}, \text{tr}_{\text{FIOP}}^{\text{SR}})$.
7. Output $\mathbb{w}$.

$\mathcal{E}_{\text{SR}}$ simulates the FIOP state-restoration game FIOPSRGame with $\tilde{\mathbf{P}}^{\text{SR}}$ and runs the FIOP state-restoration extractor $\mathbf{E}_{\text{SR}}$. The running time of $\mathcal{E}_{\text{SR}}$ is at most

$$t_{\mathcal{E}_{\text{SR}}}(\lambda, n, s, \mathbf{m}, t_{\text{ARG}}) \leq t_{\mathbf{E}_{\text{SR}}}(n, s_{\text{FIOP}}, \mathbf{m}_{\text{FIOP}}, t_{\text{FIOP}}) + t_{\text{FIOP}} .$$

Since the randomness provided by the FIOP random oracles correctly simulates that of the argument random oracles, according to Definition 3.16, the following holds:

$$\begin{aligned} & \Pr[E_{\text{KS}} \wedge E_{\Pi}] \\ & \leq \Pr \left[\begin{array}{l} |\mathbb{x}| \leq n \\ \wedge (\mathbb{x}, \mathbb{w}) \notin R \\ \wedge \mathbf{V}(\mathbb{x}, (\tilde{\Pi}_i)_{i \in [k_{\text{FIOP}}]}, (\rho_i)_{i \in [k_{\text{FIOP}}]}) = 1 \end{array} \right] \\ & \leq \kappa_{\text{FIOP}}^{\text{SR}}(n, s_{\text{FIOP}}, \mathbf{m}_{\text{FIOP}}) \cdot \left[\begin{array}{l} \mathbf{ai} \leftarrow \mathbf{D} \\ \text{rnd}_{\text{FIOP}} := (\text{rnd}_{\text{FIOP}, i})_{i \in [k_{\text{FIOP}}]} \leftarrow \mathcal{U}((r_{\text{FIOP}, i})_{i \in [k_{\text{FIOP}}]}) \\ (\mathbb{x}, (\tilde{\Pi}_i)_{i \in [k_{\text{FIOP}}]}, (\gamma_i)_{i \in [k_{\text{FIOP}}]}, (\rho_i)_{i \in [k_{\text{FIOP}}]}) \\ \quad \xleftarrow{\text{tr}_{\text{FIOP}}^{\text{SR}}} \text{FIOPSRGame}(s_{\text{FIOP}}, \text{rnd}_{\text{FIOP}}, \tilde{\mathbf{P}}^{\text{SR}}, \mathbf{ai}) \\ \mathbb{w} \leftarrow \mathbf{E}_{\text{SR}}^{\tilde{\mathbf{P}}^{\text{SR}}(\mathbf{ai})}(\mathbb{x}, (\tilde{\Pi}_i)_{i \in [k_{\text{FIOP}}]}, (\gamma_i)_{i \in [k_{\text{FIOP}}]}, (\rho_i)_{i \in [k_{\text{FIOP}}]}, \text{tr}_{\text{FIOP}}^{\text{SR}}) \end{array} \right] \end{aligned}$$

8 Batching and linearization for homomorphic functional commitment schemes

We describe generic FC-to-FC compilers for query classes of interest, and we show that these compilers preserve the state-restoration function binding property.

Batching messages. We consider the batch-messages query class $\mathbf{Q}_{\text{BatchMsg}}[\mathbf{Q}, m]$, where a query $\alpha \in \mathbf{Q}$ is applied to a batch of messages.

Definition 8.1. Let $\mathbf{Q} \subseteq \{\alpha : \Sigma^\ell \rightarrow \mathbb{D}\}$ be a query class, and let $m \in \mathbb{N}$. The batched-messages query class $\mathbf{Q}_{\text{BatchMsg}}[\mathbf{Q}, m]$ with message length $m \cdot \ell$ is the class of all queries $\alpha \in \mathbf{Q}$ applied to m subvectors of size ℓ , i.e.,

$$\mathbf{Q}_{\text{BatchMsg}}[\mathbf{Q}, m] := \left\{ \alpha' : \Sigma^{m \cdot \ell} \rightarrow \mathbb{D}^m \mid \exists \alpha \in \mathbf{Q} : \text{s.t. } \alpha'(\Pi) = (\alpha(\Pi[i\ell + 1], \dots, \Pi[i\ell + \ell]))_{i \in \{0, \dots, m-1\}} \right\} .$$

Every suitably homomorphic functional commitment FC for some query class $\mathbf{Q}$ gives rise to a natural optimized batched functional commitment $\text{BatchMsg}[\text{FC}]$ for the batched query class $\mathbf{Q}_{\text{BatchMsg}}[\mathbf{Q}, m]$, we give a detailed construction in Construction 8.5. We show that if FC is state-restoration function binding, then so is $\text{BatchMsg}[\text{FC}]$:

Lemma 8.2 (state-restoration function binding of $\text{BatchMsg}[\text{FC}, m]$). *Let FC be a non-interactive, triply homomorphic FC scheme (Definition 8.4) with expected-time function binding error $\epsilon_{\text{FC}}^* = \epsilon_{\text{FC}}^*(\lambda, \ell, L, t_{\text{FC}}^*)$. Then for any batch size $m \in \mathbb{N}$, the batch FC scheme $\text{BatchMsg}[\text{FC}, m]$ for the query class $\mathbf{Q}_{\text{BatchMsg}}[\mathbf{Q}, m]$ has state-restoration function binding error $\epsilon_{\text{FC}}^{\text{SR}}$ such that for every security parameter $\lambda \in \mathbb{N}$, message length $\ell \in \mathbb{N}$, sample set size $L \in \mathbb{N}$, salt size $s_{\text{FC}} \in \mathbb{N}$, move budget $m_{\text{FC}} \in \mathbb{N}$, circuit size bound $t_{\text{bFC}} \in \mathbb{N}$,*

$$\epsilon_{\text{bFC}}^{\text{SR}}(\lambda, m \cdot \ell, L, s_{\text{FC}}, m_{\text{FC}}, t_{\text{bFC}}) \leq L \cdot (m_{\text{FC}} + 1) \cdot \frac{m - 1}{2^\lambda} + \epsilon_{\text{FC}}^*(\lambda, \ell, L, t_{\text{FC}}^*) ,$$

where $t_{\text{FC}}^* \leq L(m_{\text{FC}} + 1)(m - 1)(t_{\text{bFC}} + \text{poly}(m \cdot \ell)) + m \cdot t_{\mathbf{Q}}(\ell, L) + L \cdot (m^3 + m^2)$.

Linearization trick. We consider the query class $\mathbf{Q}_{\text{Struct}}[\mathbf{Q}, m, (h_k)_{k \in [n]}]$ (Definition 5.9), where queries are non-linear combinations (parametrized by $(h_k)_{k \in [n]}$) of the outputs of queries α in the base query class $\mathbf{Q}$. Such query classes arise in optimized argument constructions such as Plonk [GWC19].

As a stepping stone, we consider the query class $\mathbf{Q}_{\text{LinComb}}[\mathbf{Q}, n]$ of linear combinations of n queries in the base query class $\mathbf{Q}$ (Definition 8.7). Given any homomorphic functional commitment FC for $\mathbf{Q}$, we construct a functional commitment $\text{combFC}[\text{FC}, n]$ for $\mathbf{Q}_{\text{LinComb}}[\mathbf{Q}, n]$ in Construction 8.8.

Then, given a functional commitment FC for $\mathbf{Q}$, we construct an optimized functional commitment $\text{LinFC}[\text{FC}, m, (h_k)_{k \in [n]}]$ for $\mathbf{Q}_{\text{Struct}}[\mathbf{Q}, m, (h_k)_{k \in [n]}]$ in Construction 8.9 by applying the linearization trick, and we show that if FC and $\text{combFC}[\text{FC}, n]$ are function binding, then $\text{LinFC}[\text{FC}, m, (h_k)_{k \in [n]}]$ is state-restoration function binding:

Lemma 8.3 (state-restoration function binding for linearized FCs). *Let FC be a non-interactive functional commitment scheme with function binding error $\epsilon_{\text{FC}} = \epsilon_{\text{FC}}(\lambda, \ell, L, t_{\text{FC}})$, and let $\text{combFC}[\text{FC}, n]$ be the functional commitment scheme from Construction 8.8 with function binding error $\epsilon_{\text{combFC}} = \epsilon_{\text{combFC}}(\lambda, \ell, L, t_{\text{combFC}})$. For any $m \in \mathbb{N}$ and public polynomials $(h_k)_{k \in [n]}$, the linearized functional commitment scheme*

$$\text{linFC}[\text{FC}, m, (h_k)_{k \in [n]}]$$

has state-restoration function binding error $\epsilon_{\text{linFC}}^{\text{SR}}$ such that for every security parameter $\lambda \in \mathbb{N}$, polynomial degree bound $D \in \mathbb{N}$, sample set size $L \in \mathbb{N}$, and adversary size bound $t_{\text{linFC}} \in \mathbb{N}$,

$$\epsilon_{\text{linFC}}^{\text{SR}}(\lambda, (m + n)\ell, L, s_{\text{FC}}, m_{\text{FC}}, t_{\text{linFC}}) \leq L \cdot (m_{\text{FC}} + 1) \cdot \frac{m - 1}{2^\lambda} + \epsilon_{\text{FC}}(\lambda, \ell, L, t_{\text{FC}}^*) + \epsilon_{\text{combFC}}(\lambda, \ell, n, L, t_{\text{combFC}}^*) ,$$

$$\text{where } \begin{cases} t_{\text{FC}}^* \leq L(m_{\text{FC}} + 1)(m - 1)(t_{\text{linFC}} + \text{poly}(m \cdot \ell)) + L \cdot (m^3 + m^2) + m \cdot t_{\mathbf{Q}}(\ell, L) , \\ t_{\text{combFC}}^* \leq L(m_{\text{FC}} + 1)(m - 1)(t_{\text{linFC}} + \text{poly}(m \cdot \ell)) + L \cdot (m^3 + m^2) + \ell \cdot n . \end{cases}$$

8.1 Proof of Lemma 8.2 (batched-messages FC)

Definition 8.4. Let FC be a non-interactive functional commitment scheme for a query set of functions $\alpha \in \Sigma^\ell \rightarrow \mathbb{D}$. Let the query set $\mathcal{Q}$, the evaluation domain $\mathbb{D}$, and the set of proofs $\mathbb{P}$ be $\mathbb{F}$ -modules. FC is **triply homomorphic** if for every $\text{cm}, \text{aux}, \text{aux}', \alpha, \beta, \beta', \text{pf}, \text{pf}'$:

$$\left(\begin{array}{l} \text{FC.Check}(\text{pp}_{\text{FC}}, \text{cm}, \alpha, \beta, \text{pf}) = 1 \\ \wedge \text{FC.Check}(\text{pp}_{\text{FC}}, \text{cm}', \alpha, \beta', \text{pf}') = 1 \end{array} \right) \Rightarrow \text{FC.Check}(\text{pp}_{\text{FC}}, \text{cm} + \text{cm}', \alpha, \beta + \beta', \text{pf} + \text{pf}') .$$

Construction 8.5 (FC for batched messages). Let FC be a triply homomorphic functional commitment for the query class $\mathbf{Q}$. For any $m \in \mathbb{N}$, we construct a functional commitment $\text{bFC} := \text{BatchMsg}[\text{FC}, m]$ for the query class $\mathbf{Q}_{\text{BatchMsg}[\mathbf{Q}, m]}$ as follows:

- $\text{bFC.Gen}(1^\lambda, \ell, m)$: Output $\text{pp}_{\text{FC}} \leftarrow \text{FC.Gen}(1^\lambda, \ell)$.
- $\text{bFC.Commit}(\text{pp}_{\text{FC}}, f = (f_b)_{b \in [m]})$:
 1. For $b \in [m]$: Compute $\text{cm}_b := \text{FC.Commit}(\text{pp}_{\text{FC}}, f_b)$.
 2. Set $\text{cm} := (\text{cm}_b)_{b \in [m]}$.
 3. Set $\text{aux} := f$.
 4. Output (cm, aux) .
- $\langle \mathcal{P}_{\text{bFC}}(\text{pp}_{\text{FC}}, \text{cm}, \text{aux}, \alpha, \beta), \mathcal{V}_{\text{bFC}}(\text{pp}_{\text{FC}}, \text{cm}, \alpha, \beta) \rangle$
 1. Both $\mathcal{P}_{\text{bFC}}$ and $\mathcal{V}_{\text{bFC}}$ parse cm as $(\text{cm}_b)_{b \in [m]}$ and β as $(\beta_b)_{b \in [m]}$.
 2. $\mathcal{P}_{\text{bFC}}$ parses aux as $(f_b)_{b \in [m]}$.
 3. $\mathcal{V}_{\text{bFC}}$ sends a random challenge $\gamma \leftarrow \mathbb{F}$ to $\mathcal{P}_{\text{bFC}}$.
 4. $\mathcal{P}_{\text{bFC}}$ computes $\text{pf}_b \leftarrow \text{FC.Open}(\text{cm}_b, f_b, \alpha, \beta_b)$ for all $b \in [m]$ and sends $\text{pf} := \sum_{b \in [m]} \gamma^{b-1} \text{pf}_b$ to $\mathcal{V}_{\text{bFC}}$.
 5. $\mathcal{V}_{\text{bFC}}$ outputs $\text{FC.Check}(\text{pp}_{\text{FC}}, \sum_{b \in [m]} \gamma^{b-1} \text{cm}_b, \alpha, \sum_{b \in [m]} \gamma^{b-1} \beta_b, \text{pf})$

In order to prove that $\text{BatchMsg}[\text{FC}, m]$ satisfies state-restoration function binding (Definition 3.25), we prove that it is m -special function binding (Definition A.1). Lemma 8.2 follows from Lemmas 8.6 and A.2.

Lemma 8.6. Let FC be a functional commitment scheme with function binding error $\epsilon_{\text{FC}} = \epsilon_{\text{FC}}(\lambda, \ell, L, t_{\text{FC}})$. Then for any batch size $m \in \mathbb{N}$, $\text{BatchMsg}[\text{FC}, m]$ has m -special function binding error $\epsilon_{\text{bSFB}} = \epsilon_{\text{bSFB}}(\lambda, \ell, m \cdot L, t_{\text{bSFB}}^*)$ such that for every security parameter $\lambda \in \mathbb{N}$, message size $\ell \in \mathbb{N}$, sample set size $L \in \mathbb{N}$, and expected adversary size bound $t_{\text{bSFB}}^* \in \mathbb{N}$,

$$\epsilon_{\text{bSFB}}(\lambda, m \cdot \ell, L, t_{\text{bSFB}}^*) \leq \epsilon_{\text{FC}}(\lambda, \ell, L, t_{\text{FC}}) ,$$

where $t_{\text{FC}} \leq t_{\text{bSFB}}^* + m \cdot t_{\mathbf{Q}}(\ell, L) + L \cdot (m^3 + m^2)$.

Proof of Lemma 8.6. Given an m -special function binding adversary A_{bSFB} against bFC, we construct the following function binding adversary A_{FC} against FC:

- $A_{\text{FC}}(\text{pp}, \text{ai}_{\text{FC}})$:
1. Run $(\text{cm}, ((\alpha_i, \beta_i, T_i))_{i \in [L]}) \leftarrow A_{\text{bSFB}}(\text{pp}, \text{ai}_{\text{FC}})$

2. If there is no Π such that $\forall i \in [L] : \alpha_i(\Pi) = \beta_i$, output $\perp$.
3. Parse cm as $(\text{cm}_b)_{b \in [m]}$.
4. For $i \in [L]$:
 - (a) Parse β_i as $(\beta_{i,b})_{b \in [m]}$.
 - (b) Parse T_i as verifier challenges $(\gamma_i^{(b)})_{b \in [m]}$ and final prover messages $(\text{pf}_i^{(b)})_{b \in [m]}$.
 - (c) Compute $V_i := \begin{bmatrix} 1 & \gamma_i^{(1)} & \cdots & (\gamma_i^{(1)})^{m-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \gamma_i^{(m)} & \cdots & (\gamma_i^{(m)})^{m-1} \end{bmatrix}$.
 - (d) If V_i is invertible, compute $V_i^{-1} = \begin{bmatrix} \bar{\gamma}_{i,1,1} & \cdots & \bar{\gamma}_{i,1,m-1} \\ \vdots & \ddots & \vdots \\ \bar{\gamma}_{i,m,1} & \cdots & \bar{\gamma}_{i,m,m} \end{bmatrix}$, and output $\perp$ otherwise.
 - (e) Set $\begin{bmatrix} \text{pf}'_{i,1} \\ \vdots \\ \text{pf}'_{i,m} \end{bmatrix} := V_i^{-1} \begin{bmatrix} \text{pf}_i^{(1)} \\ \vdots \\ \text{pf}_i^{(m)} \end{bmatrix}$.
5. For $j \in [m]$:

If there is no Π_j such that $\forall i \in [L] : \alpha_i(\Pi_j) = \beta_{i,j}$, output $(\text{cm}_j, ((\alpha_i, \beta_{i,j}))_{i \in [L]}, (\text{pf}'_{i,j})_{i \in [L]})$.
6. Output $\perp$.

Running time. Each iteration of the loop in Step 4 requires $m^3 + m^2$ operations (to invert the Vandermonde matrix and compute a matrix-vector product), and each iteration of the loop in Step 5 requires $t_{\mathbf{Q}}(\ell, L)$ operations, so A_{FC} runs in expected time at most $t_{\text{bSFB}}^* + m \cdot t_{\mathbf{Q}}(\ell, L) + L \cdot (m^3 + m^2)$.

Success probability. If A_{bSFB} is successful, A_{FC} does not abort in Step 4d, and for every $i \in [L]$ and $k \in [m]$,

$$\text{FC.Check}\left(\sum_{b \in [m]} (\gamma_i^{(k)})^{b-1} \cdot \text{cm}_b, \alpha_i, \sum_{b \in [m]} (\gamma_i^{(k)})^{b-1} \cdot \beta_i^{(b)}, \text{pf}_i^{(k)}\right) = 1.$$

By construction, we have that for all $i \in [L]$ and $j \in [m]$,

$$\begin{aligned} \sum_{k \in [m]} \bar{\gamma}_{i,j,k} \sum_{b \in [m]} (\gamma_i^{(k)})^{b-1} \cdot \text{cm}_b &= \text{cm}_j, \\ \sum_{k \in [m]} \bar{\gamma}_{i,j,k} \sum_{b \in [m]} (\gamma_i^{(k)})^{b-1} \cdot \beta_{i,b} &= \beta_{i,j}. \end{aligned}$$

By triple homomorphism, this implies that for all $i \in [L]$ and $j \in [m]$:

$$\begin{aligned} &\text{FC.Check}(\text{cm}_j, \alpha_i, \beta_{i,j}, \text{pf}'_{i,j}) \\ &= \text{FC.Check}\left(\sum_{k \in [m]} \bar{\gamma}_{i,j,k} \sum_{b \in [m]} (\gamma_i^{(k)})^{b-1} \cdot \text{cm}_b, \alpha_i, \sum_{k \in [m]} \bar{\gamma}_{i,j,k} \sum_{b \in [m]} (\gamma_i^{(k)})^{b-1} \cdot \beta_{i,b}, \sum_{k \in [m]} \bar{\gamma}_{i,j,k} \cdot \text{pf}_i^{(k)}\right) \\ &= 1, \text{ i.e., } \text{pf}'_{i,j} \text{ is an accepting proof for } (\text{cm}_j, \alpha_i, \beta_{i,j}). \end{aligned}$$

Further, the function binding condition

$$\nexists (\Pi_b)_{b \in [m]} \in (\mathbb{F}^\ell)^m : \forall i \in [L] : \forall b \in [m] : \alpha_i(\Pi_b) = \beta_{i,b}$$

for bFC implies that the condition in Step 5 is satisfied for at least one j . To see why, assume towards a contradiction that

$$\begin{aligned} & \neg(\exists j \in [m] : \nexists \Pi_j : \forall i \in [L] : \alpha_i(\Pi_j) = \beta_{i,j}) \\ & \equiv \forall j \in [m] : \exists \Pi_j : \forall i \in [L] : \alpha_i(\Pi_j) = \beta_{i,j} \\ & \Rightarrow \exists (\Pi_b)_{b \in [m]} : \forall i \in [L] : \forall j \in [m] : \alpha_i(\Pi_j) = \beta_{i,j} , \end{aligned}$$

which contradicts our assumption on the success of A_{bSFB} . $\square$

8.2 Proof of Lemma 8.3 (linearization trick)

We generalize the linearization trick [GWC19; CHMMVW20; FFR24] [LPS25, Section 4.2] for any homomorphic functional commitment: given a functional commitment scheme FC, we construct a functional commitment scheme $\text{LinFC}[\text{FC}, m, (h_k)_{k \in [n]}]$ for $\mathbf{Q}_{\text{Struct}}[\mathbf{Q}, m, (h_k)_{k \in [n]}]$ (Definition 5.9).

Linear combination queries. As a building block, we consider the query class $\mathbf{Q}_{\text{LinComb}}[\mathbf{Q}, n]$ of linear combinations of n queries in the base query class $\mathbf{Q}$.

Definition 8.7. For a query class $\mathbf{Q}$ and a parameter $n \in \mathbb{N}$, the query class $\mathbf{Q}_{\text{LinComb}}[\mathbf{Q}, n]$ is

$$\left\{ \alpha' : \mathbb{F}^{\ell n} \rightarrow \mathbb{F} \mid \begin{array}{l} \Pi = \mathbf{g}_1 \parallel \dots \parallel \mathbf{g}_n \text{ and} \\ \exists \alpha \in \mathbf{Q}, \gamma_1, \dots, \gamma_n \in \mathbb{F} \text{ s.t.} \\ \alpha'(\Pi) = \sum_{\ell \in [n]} \gamma_\ell \cdot \alpha(\mathbf{g}_\ell) \end{array} \right\} .$$

Given any homomorphic functional commitment FC for $\mathbf{Q}$, we construct a functional commitment $\text{combFC}[\text{FC}, n]$ for $\mathbf{Q}_{\text{LinComb}}[\mathbf{Q}, n]$.

Construction 8.8 (FC for $\mathbf{Q}_{\text{LinComb}}$). Let FC be a homomorphic functional commitment scheme for the query class $\mathbf{Q}$. For any $n \in \mathbb{N}$, we construct a functional commitment $\text{combFC} = \text{combFC}[\text{FC}]$ for the query class $\mathbf{Q}_{\text{LinComb}}[\mathbf{Q}, n]$ as follows:

$\text{combFC.Gen}(1^\lambda, \ell \cdot n)$: Output $\text{pp}_{\text{FC}} \leftarrow \text{FC.Gen}(1^\lambda, \ell)$.

$\text{combFC.Commit}(\text{pp}_{\text{FC}}, \Pi)$:

1. Parse Π as $(\mathbf{g}_1, \dots, \mathbf{g}_n)$.
2. Output $\text{cm} := (\text{FC.Commit}(\text{pp}_{\text{FC}}, \mathbf{p}_\ell))_{\ell \in [n]}$.

$\langle \mathcal{P}_{\text{comb}}(\text{pp}_{\text{FC}}, \text{cm}, \text{aux}, \alpha', \beta), \mathcal{V}_{\text{comb}}(\text{pp}_{\text{FC}}, \text{cm}, \alpha', \beta) \rangle$:

1. $\mathcal{P}_{\text{comb}}$ and $\mathcal{V}_{\text{comb}}$ both parse α' as $(\alpha, h_1, \dots, h_n)$.
2. Run $\langle \mathcal{P}_{\text{FC}}, \mathcal{V}_{\text{FC}} \rangle$ for the commitment $\sum_{\ell \in [n]} h_\ell \cdot \text{cm}_\ell$, the query α , evaluation β , and for the witness $\sum_{\ell \in [n]} h_\ell \cdot \mathbf{g}_\ell$.

Generalized linearization trick.

Construction 8.9. Let FC be a non-interactive triply homomorphic functional commitment for the query class $\mathbf{Q}$, and let $\text{combFC} = \text{combFC}[\text{FC}, n]$ be the functional commitment scheme for the class $\mathbf{Q}' := \mathbf{Q}_{\text{LinComb}}[\mathbf{Q}, n]$ from Construction 8.8. For any $m \in \mathbb{N}$, we construct a functional commitment $\text{linFC} := \text{LinFC}[\text{FC}, m, (h_k)_{k \in [n]}]$ for the query class $\mathbf{Q}_{\text{Struct}}[\mathbf{Q}, m, (h_k)_{k \in [n]}]$ as follows:

- $\text{linFC.Gen}(1^\lambda, \ell, m)$: Output $\text{pp}_{\text{FC}} \leftarrow \text{FC.Gen}(1^\lambda, \ell)$.
- $\text{linFC.Commit}(\text{pp}_{\text{FC}}, \mathbf{f} = (\mathbf{f}_b)_{b \in [m+n]})$:

1. For $b \in [m+n]$: Compute $\text{cm}_b := \text{FC.Commit}(\text{pp}_{\text{FC}}, \mathbf{f}_b)$.
 2. Set $\text{cm} := (\text{cm}_b)_{b \in [m+n]}$.
 3. Set $\text{aux} := \mathbf{f}$.
 4. Output (cm, aux) .
- $\langle \mathcal{P}_{\text{linFC}}(\text{pp}_{\text{FC}}, \text{cm}, \text{aux}, \alpha, \beta), \mathcal{V}_{\text{linFC}}(\text{pp}_{\text{FC}}, \text{cm}, \alpha, \beta) \rangle$
 1. Both $\mathcal{P}_{\text{linFC}}$ and $\mathcal{V}_{\text{linFC}}$ parse cm as $(\text{cm}_b)_{b \in [m+n]}$ and β as $(\beta_b)_{b=0}^m$.
 2. $\mathcal{P}_{\text{linFC}}$ parses aux as $(\mathbf{f}_b)_{b \in [m+n]}$.
 3. $\mathcal{V}_{\text{linFC}}$ sends a random challenge $\gamma \leftarrow \mathbb{F}$ to $\mathcal{P}_{\text{linFC}}$.
 4. $\mathcal{P}_{\text{linFC}}$ and $\mathcal{V}_{\text{linFC}}$ compute $\text{pf}_b \leftarrow \text{FC.Open}(\text{cm}_b, \mathbf{f}_b, \alpha, \beta_b)$ for all $b \in [m]$.
 5. $\mathcal{P}_{\text{linFC}}$ computes $h_\ell := h_\ell(\beta_1, \dots, \beta_m)$ for all $\ell \in [n]$.
 6. $\mathcal{P}_{\text{linFC}}$ computes $\text{pf}_0 \leftarrow \text{combFC.Open}((\text{cm}_{s+1}, \dots, \text{cm}_{s+n}), (\mathbf{f}_{s+1}, \dots, \mathbf{f}_{s+n}), (\alpha, h_1, \dots, h_n), \beta_{m+1})$.
 7. $\mathcal{P}_{\text{linFC}}$ sends $\text{pf} := \sum_{b \in [m+1]} \gamma^{b-1} \text{pf}_b$ to $\mathcal{V}_{\text{linFC}}$.
 8. $\mathcal{V}_{\text{linFC}}$ outputs $\text{FC.Check}(\text{pp}_{\text{FC}}, \sum_{\ell \in [n]} h_\ell \cdot \text{cm}_{m+\ell} + \sum_{b \in [m]} \gamma^b \text{cm}_b, \alpha, \sum_{b \in [m+1]} \gamma^{b-1} \beta_b, \text{pf})$.

In order to prove that linFC satisfies state-restoration function binding (Definition 3.25), we prove that it is $(m+1)$ -special function binding (Definition A.1). Lemma 8.3 follows from Lemmas 8.10 and A.2.

Lemma 8.10. *Let FC be a functional commitment scheme with function binding error $\epsilon_{\text{FC}} = \epsilon_{\text{FC}}(\lambda, \ell, \mathbf{L}, t_{\text{FC}})$. Let $n \in \mathbb{N}$, and let $\text{combFC} = \text{combFC}[\text{FC}, n]$ be the functional commitment scheme for the class $\mathbf{Q}_{\text{LinComb}}[\mathbf{Q}, n]$ from Construction 8.8, with function binding error $\epsilon_{\text{combFC}} = \epsilon_{\text{combFC}}(\lambda, \ell, n, \mathbf{L}, t_{\text{combFC}})$. Then for any parameter $m \in \mathbb{N}$, $\text{LinFC}[\text{FC}, m, (\mathbf{h}_k)_{k \in [n]}]$ has $(m+1)$ -special function binding error $\epsilon_{\text{linSFB}} = \epsilon_{\text{linSFB}}(\lambda, \ell, (m+n) \cdot \mathbf{L}, t_{\text{linSFB}})$ such that for every security parameter $\lambda \in \mathbb{N}$, message size $\ell \in \mathbb{N}$, sample set size $\mathbf{L} \in \mathbb{N}$, and adversary size bound $t_{\text{linSFB}} \in \mathbb{N}$,*

$$\epsilon_{\text{linSFB}}(\lambda, (m+n) \cdot \ell, \mathbf{L}, t_{\text{linSFB}}) \leq \epsilon_{\text{FC}}(\lambda, \ell, \mathbf{L}, t_{\text{FC}}) + \epsilon_{\text{combFC}}(\lambda, \ell, n, \mathbf{L}, t_{\text{combFC}}),$$

where $t_{\text{FC}} \leq t_{\text{linSFB}} + \mathbf{L} \cdot (m^3 + m^2) + m \cdot t_{\mathbf{Q}}(\ell, \mathbf{L})$ and $t_{\text{combFC}} \leq t_{\text{linSFB}} + \mathbf{L} \cdot (m^3 + m^2) + \mathbf{L} \cdot n$.

Proof of Lemma 8.10. Given an $(m+1)$ -special function binding adversary A_{linSFB} against linFC , we construct function binding adversaries against FC and combFC .

Special function binding helper. Both adversaries run the following algorithm as a subroutine:

$A(\text{pp}, \text{ai}_{\text{FC}})$:

1. Run $(\text{cm}, ((\alpha_i, \beta_i, \mathbf{T}_i))_{i \in [\mathbf{L}]}) \leftarrow t_{\text{combFC}}(\text{pp}, \text{ai}_{\text{FC}})$
2. If there is no Π such that $\forall i \in [\mathbf{L}] : \alpha_i(\Pi) = \beta_i$, output $\perp$.
3. Parse cm as $(\text{cm}_b)_{b \in [m+n]}$.
4. For $i \in [\mathbf{L}]$:
 - (a) Parse β_i as $(\beta_{i,b})_{b=0}^m$.
 - (b) Parse $\mathbf{T}_i$ as verifier challenges $(\gamma_i^{(b)})_{b=0}^m$ and final prover messages $(\text{pf}_i^{(b)})_{b=0}^m$.
 - (c) Compute $V_i := \begin{bmatrix} 1 & \gamma_i^{(0)} & \cdots & (\gamma_i^{(0)})^m \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \gamma_i^{(m)} & \cdots & (\gamma_i^{(m)})^m \end{bmatrix}$.
 - (d) If V_i is invertible, compute $V_i^{-1} = \begin{bmatrix} \bar{\gamma}_{i,0,0} & \cdots & \bar{\gamma}_{i,0,m} \\ \vdots & \ddots & \vdots \\ \bar{\gamma}_{i,m,0} & \cdots & \bar{\gamma}_{i,m,m} \end{bmatrix}$, and output $\perp$ otherwise.

$$(e) \text{ Set } \begin{bmatrix} \text{pf}'_{i,0} \\ \vdots \\ \text{pf}'_{i,m} \end{bmatrix} := V_i^{-1} \begin{bmatrix} \text{pf}_i^{(0)} \\ \vdots \\ \text{pf}_i^{(m)} \end{bmatrix}.$$

5. Output $(\text{cm}, ((\alpha_i, \beta_i, (\text{pf}'_{i,j})_{j=0}^m)_{i \in [L]}).$

Running time: Each iteration of the loop in Step 4 requires $m^3 + m^2$ operations (to invert the Vandermonde matrix and compute a matrix-vector product), so A runs in time $t_A \leq t_{\text{linSFB}} + L \cdot (m^3 + m^2)$.

Success probability: If A_{linSFB} is successful, then A does not abort in Step 4d, and for every $i \in [L]$ and $k \in [m+1]$,

$$\text{FC.Check}(\text{pp}_{\text{FC}}, \sum_{\ell \in [n]} h_\ell \cdot \text{cm}_{m+\ell} + \sum_{b \in [m]} (\gamma_i^{(k)})^b \text{cm}_b, \alpha, \sum_{b=0}^m (\gamma_i^{(k)})^{b-1} \beta_b, \text{pf}^{(k)}) = 1.$$

By triple homomorphism, this implies that for all $i \in [L]$ and $j \in [m]$:

$$\begin{aligned} & \text{FC.Check}(\text{cm}_j, \alpha_i, \beta_{i,j}, \text{pf}'_{i,j}) \\ &= \text{FC.Check}(\sum_{k \in [m+1]} \bar{\gamma}_{i,j,k} \sum_{b \in [m+1]} (\gamma_i^{(k)})^{b-1} \cdot \text{cm}_b, \alpha_i, \sum_{k \in [m+1]} \bar{\gamma}_{i,j,k} \sum_{b \in [m+1]} (\gamma_i^{(k)})^{b-1} \cdot \beta_{i,b}, \sum_{k \in [m+1]} \bar{\gamma}_{i,j,k} \cdot \text{pf}_i^{(k)}) \\ &= 1, \text{ i.e., } \text{pf}'_{i,j} \text{ is an accepting proof for } (\text{cm}_j, \alpha_i, \beta_{i,j}). \end{aligned}$$

Similarly, $\text{FC.Check}(\sum_{\ell \in [n]} h_{i,\ell} \cdot \text{cm}_{m+\ell}, \alpha_i, \beta_{i,0}, \text{pf}'_{i,0}) = 1$.

Function binding adversary against FC.

$A_{\text{FC}}(\text{pp}, \text{ai}_{\text{FC}})$:

1. Run $(\text{cm}, ((\alpha_i, \beta_i), \text{pf}'_{i,j})_{j \in [m+1]})_{i \in [L]} \leftarrow A_{\text{bSFB}}(\text{pp}, \text{ai}_{\text{FC}})$.
2. For $j \in [m]$:

If there is no Π_j such that $\forall i \in [L] : \alpha_i(\Pi_j) = \beta_{i,j}$, output $(\text{cm}_j, ((\alpha_i, \beta_{i,j}))_{j \in [L]}, (\text{pf}'_{i,j})_{i \in [L]}).$

3. Output $\perp$.

Running time: Each iteration of the loop requires $t_{\mathbf{Q}}(\ell, L)$ operations, so A_{FC} runs in time at most $t_A + m \cdot t_{\mathbf{Q}}(\ell, L)$.

Success probability: If A_{linSFB} is successful and A_{FC} does not abort, then A_{FC} breaks function binding for FC.

Function binding adversary against combFC.

$A_{\text{combFC}}(\text{pp}, \text{ai}_{\text{FC}})$:

1. Run $((\text{cm}_i)_{i \in [m+n]}, ((\alpha_i, (\beta_{i,b})_{b \in [m+1]}), \text{pf}'_{i,j})_{j \in [m+1]})_{i \in [L]} \leftarrow A_{\text{bSFB}}(\text{pp}, \text{ai}_{\text{FC}})$.
2. For $i \in [L]$: For $\ell \in [m]$: Compute $h_{i,\ell} := h_\ell(\beta_{i,1}, \dots, \beta_{i,m})$.
3. Output $((\text{cm}_{m+i})_{i \in [n]}, ((\alpha_i, h_{i,1}, \dots, h_{i,m}), \beta_{i,0}, \text{pf}'_{i,0})_{i \in [L]}).$

Running time: A_{combFC} runs in time at most $t_A + L \cdot n$.

Success probability: If A_{linSFB} is successful and A_{FC} outputs $\perp$, then

$$\nexists (\Pi_\ell)_{\ell \in [n]} : \forall i \in [L] : \sum_{\ell \in [n]} h_\ell(\beta_{i,1}, \dots, \beta_{i,m}) \cdot \alpha_i(\Pi_\ell) = \beta_{i,0},$$

and A_{combFC} breaks function binding for combFC. □

9 Application: variants of the KZG polynomial commitment scheme

We apply the generic results from Section 8 to variants of the KZG polynomial commitment scheme [KZG10]. In Section 9.1, we show that the KZG polynomial commitment scheme KZG is function binding under the falsifiable ARSDH assumption.

Lemma 9.1 (KZG is function binding). *Suppose the ARSDH assumption (Definition 9.6) holds with error $\epsilon_{\text{ARSDH}} = \epsilon_{\text{ARSDH}}(\lambda, D, t_{\text{ARSDH}})$. The KZG polynomial commitment scheme (Construction 9.5) has function binding error ϵ_{FC} such that for every security parameter $\lambda \in \mathbb{N}$, polynomial degree bound $D \in \mathbb{N}$, sample set size $L \in \mathbb{N}$, and adversary size bound $t_{\text{FC}} \in \mathbb{N}$,*

$$\epsilon_{\text{FC}}(\lambda, D + 1, L, t_{\text{FC}}) \leq \epsilon_{\text{ARSDH}}(\lambda, D, t_{\text{ARSDH}}) ,$$

where $t_{\text{ARSDH}} \leq t_{\text{FC}} + 2L^2 + L + D(L + 8) + 2(D + 1)^2$.

In Section 9.2, we then show that the batched PC bKZG = BatchMsg[KZG, m] is state-restoration function binding under the same assumption.

Corollary 9.2 (Batched KZG is state-restoration function binding). *Suppose the expected-time ARSDH assumption (Definition 9.9) holds with error $\epsilon_{\text{ARSDH}} = \epsilon_{\text{ARSDH}}(\lambda, D, t_{\text{ARSDH}}^*)$. Then for any batch size $m \in \mathbb{N}$, the batch polynomial commitment scheme bKZG = BatchMsg[KZG, m] has state-restoration function binding error $\epsilon_{\text{FC}}^{\text{SR}}$ such that for every security parameter $\lambda \in \mathbb{N}$, polynomial degree bound $D \in \mathbb{N}$, sample set size $L \in \mathbb{N}$, and adversary size bound $t_{\text{bFC}} \in \mathbb{N}$,*

$$\epsilon_{\text{bFC}}^{\text{SR}}(\lambda, m(D + 1), L, s_{\text{FC}}, m_{\text{FC}}, t_{\text{bFC}}) \leq L \cdot (m_{\text{FC}} + 1) \cdot \frac{m - 1}{2\lambda} + \epsilon_{\text{ARSDH}}(\lambda, D, t_{\text{ARSDH}}^*) ,$$

where $t_{\text{ARSDH}}^* \leq L(m_{\text{FC}} + 1)(m - 1)(t_{\text{bFC}} + \text{poly}(m \cdot D, L)) + m \cdot ((D + 1)^2 + (D + 1) \cdot L) + L \cdot (m^3 + m^2)$.

In Section 9.4, we show that the linearized KZG functional commitment linKZG = LinFC[bKZG, m , $(h_k)_{k \in [n]}$] (as used in the Plonk construction [GWC19]) is state-restoration function binding.

Lemma 9.3. *Assume the SplitRSDH assumption holds with parameters $(\psi_\ell)_{\ell \in [n]}$ and d . Assume that KZG is evaluation binding with error $\epsilon_{\text{EB}}(\lambda, n_1, t_{\text{EB}})$. Then combFC[KZG] (Construction 8.8) is function binding with function binding error*

$$\epsilon_{\text{combFC}}(\lambda, n \cdot n_1, L, t_{\text{combFC}}) \leq \epsilon_{\text{EB}}(\lambda, n_1, t_{\text{EB}}) + \epsilon_{\text{SplitRSDH}}(\lambda, n_1, n, d, L, (\psi_k)_{k \in [n]}, t_{\text{SplitRSDH}}) ,$$

where, $t_{\text{EB}} \leq t_{\text{combFC}} + L^2$ and $t_{\text{SplitRSDH}} \leq t_{\text{combFC}} + 3L^2$.

Corollary 9.4 (Linearized KZG is state-restoration function binding). *Suppose the expected-time SDH assumption holds with error $\epsilon_{\text{SDH}} = \epsilon_{\text{SDH}}(\lambda, D, t_{\text{SDH}}^*)$, the expected-time ARSDH assumption (Definition 9.9) holds with error $\epsilon_{\text{ARSDH}} = \epsilon_{\text{ARSDH}}(\lambda, D, t_{\text{ARSDH}}^*)$, and the expected-time SplitRSDH assumption (Definition 9.11) holds with error $\epsilon_{\text{SplitRSDH}} = \epsilon_{\text{SplitRSDH}}(\lambda, D, n, d, L, (\psi_\ell)_{\ell \in [n]}, t_{\text{SplitRSDH}}^*)$ for the parameters $(\psi_\ell)_{\ell \in [n]}$ and d . Then for any $m \in \mathbb{N}$ and public polynomials $(h_k)_{k \in [n]}$, the linearized KZG functional commitment scheme linKZG = LinFC[bKZG, m , $(h_k)_{k \in [n]}$] (Construction 8.9) has state-restoration function binding error $\epsilon_{\text{FC}}^{\text{SR}}$ such that for every security parameter $\lambda \in \mathbb{N}$, polynomial degree bound $D \in \mathbb{N}$, sample set size $L \in \mathbb{N}$, and adversary size bound $t_{\text{linFC}} \in \mathbb{N}$,*

$$\begin{aligned} \epsilon_{\text{linFC}}^{\text{SR}}(\lambda, (m + n)(D + 1), L, s_{\text{FC}}, m_{\text{FC}}, t_{\text{linFC}}) &\leq L \cdot (m_{\text{FC}} + 1) \cdot \frac{m}{2\lambda} \\ &\quad + \epsilon_{\text{SDH}}(\lambda, n_1, t_{\text{SDH}}^*) \\ &\quad + \epsilon_{\text{ARSDH}}(\lambda, D, t_{\text{ARSDH}}^*) \\ &\quad + \epsilon_{\text{SplitRSDH}}(\lambda, n_1, n, d, L, (\psi_k)_{k \in [n]}, t_{\text{SplitRSDH}}) , \end{aligned}$$

$$\text{where } \begin{cases} t_{\text{SDH}}^* \leq L(m_{\text{FC}} + 1)(m - 1)(t_{\text{linFC}} + \text{poly}(m \cdot \ell)) + L \cdot (m^3 + m^2) + \ell \cdot n + L^2 \\ t_{\text{ARSDH}}^* \leq L(m_{\text{FC}} + 1)(m - 1)(t_{\text{linFC}} + \text{poly}(m \cdot \ell)) + L \cdot (m^3 + m^2) + m \cdot t_{\mathbf{Q}}(\ell, L) \\ t_{\text{SplitRSDH}}^* \leq L(m_{\text{FC}} + 1)(m - 1)(t_{\text{linFC}} + \text{poly}(m \cdot \ell)) + L \cdot (m^3 + m^2) + \ell \cdot n + 3L^2 \end{cases}$$

9.1 Proof of Lemma 9.1 (KZG)

Construction 9.5 (KZG). The FC scheme KZG for the query class $\mathbf{Q}_{\text{UniPoly}}$ is defined as follows.

- $\text{KZG.Gen}(1^\lambda, D)$:
 1. Choose $(p, \mathbb{G}_1, \mathbb{G}_2, [1]_1, [1]_2, \mathbb{G}_T, e)$ where $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are additive groups of prime order p , $[1]_1$ is a generator of $\mathbb{G}_1$, $[1]_2$ is a generator of $\mathbb{G}_2$, and $e: \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is a bilinear pairing.
 2. Sample $\tau \leftarrow \mathbb{Z}_p \setminus \{0\}$.
 3. Output $\text{pp}_{\text{PC}} := ([1]_1, [\tau]_1, [\tau^2]_1, \dots, [\tau^D]_1, [1]_2, [\tau]_2)$.
- $\text{KZG.Commit}(\text{pp}_{\text{PC}}, p)$:
 1. Parse pp_{PC} as $([1]_1, [\tau]_1, [\tau^2]_1, \dots, [\tau^D]_1, [1]_2, [\tau]_2)$.
 2. Compute $\text{cm} := [p(\tau)]_1$.
 3. Set $\text{aux} := p$.
 4. Output (cm, aux) .
- $\text{KZG.Open}(\text{pp}_{\text{PC}}, \text{aux}, \alpha, \beta)$:
 1. Parse pp_{PC} as $([1]_1, [\tau]_1, [\tau^2]_1, \dots, [\tau^D]_1, [1]_2, [\tau]_2)$ and aux as p .
 2. Compute $p_\alpha(X) := \frac{p(X) - \beta}{X - \alpha}$.
 3. Compute $\text{pf} := [p_\alpha(\tau)]_1$.
 4. Output pf .
- $\text{KZG.Check}(\text{pp}_{\text{PC}}, \text{cm}, \alpha, \beta, \text{pf})$:
 1. Parse pp_{PC} as $([1]_1, [\tau]_1, [\tau^2]_1, \dots, [\tau^D]_1, [1]_2, [\tau]_2)$.
 2. Check that $e(\text{cm} - [\beta]_1, [1]_2) = e(\text{pf}, [\tau - \alpha]_2)$.

We reduce the function binding property of the KZG PCS to the following Diffie-Hellman-type assumption over bilinear groups.

Definition 9.6 (ARSDH assumption [LPS24]). *Let $\mathbb{F}$ be a field and $Z_S(X) := \prod_{s \in S} (X - s)$ for $S \subseteq \mathbb{F}$, the **adaptive rational strong Diffie-Hellman (ARSDH)** holds with error ϵ_{ARSDH} if for every security parameter λ , adversary size bound t_{ARSDH} , t_{ARSDH} -size circuit A_{ARSDH} , and tuple $(p, \mathbb{G}_1, \mathbb{G}_2, [1]_1, [1]_2, \mathbb{G}_T, e)$ where $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are groups of prime order $p(\lambda)$, $[1]_1$ is a generator of $\mathbb{G}_1$, $[1]_2$ is a generator of $\mathbb{G}_2$, and $e: \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is a bilinear pairing,*

$$\Pr \left[\begin{array}{l} S \subseteq \mathbb{F}, |S| = D + 1 \\ \wedge h_1, h_2 \in \mathbb{G}_1, h_1 \neq [0]_1 \\ \wedge h_2 = \frac{1}{Z_S(\tau)} \cdot h_1 \end{array} \middle| \begin{array}{l} \tau \leftarrow \mathbb{Z}_p \setminus \{0\} \\ (S, h_1, h_2) \leftarrow A_{\text{ARSDH}}([1]_1, [\tau]_1, [\tau^2]_1, \dots, [\tau^D]_1, [1]_2, [\tau]_2) \end{array} \right] \leq \epsilon_{\text{ARSDH}}(\lambda, D, t_{\text{ARSDH}}) .$$

Proof of Lemma 9.1. Fix the security parameter $\lambda \in \mathbb{N}$, polynomial degree bound $D \in \mathbb{N}$, sample set size $L \in \mathbb{N}$, and adversary size bound $t_{\text{FC}} \in \mathbb{N}$. For every auxiliary input distribution $\mathcal{D}_{\text{FC}}$ and t_{FC} -size

adversary A_{FC} against function binding, consider the following adversary A_{ARSDH} against ARSDH. Below, $\text{Lagrange}(\{(x_i, y_i)\}_{i \in [n]})$ denotes the unique polynomial of degree at most $n - 1$ that interpolates all points $\{(x_i, y_i)\}_{i \in [n]}$.

- $A_{ARSDH}([1]_1, [\tau]_1, [\tau^2]_1, \dots, [\tau^D]_1, [1]_2, [\tau]_2)$:
1. Sample $\text{ai}_{FC} \leftarrow \mathcal{D}_{FC}$ and set $\text{pp}_{PC} := ([1]_1, [\tau]_1, [\tau^2]_1, \dots, [\tau^D]_1, [1]_2, [\tau]_2)$.
 2. Run $(\text{cm}, \{(\alpha_i, \beta_i, \text{pf}_i)\}_{i \in [L]}) \leftarrow A_{FC}(\text{pp}_{PC}, \text{ai}_{FC})$.
 3. If there exists $i, j \in [L]$ such that $\alpha_i = \alpha_j, \beta_i \neq \beta_j$:
 - (a) Choose S to be a size- $(D + 1)$ subset of $\mathbb{F}$ such that $\alpha_i \in S$ and $[Z_S(\tau)]_1 \neq [0]_1$.
 - (b) Set $h_1 := [Z_{S \setminus \{\alpha_i\}}(\tau)]_1$.
 - (c) Set $h_2 := \frac{\text{pf}_i - \text{pf}_j}{\beta_j - \beta_i}$.
 - (d) Output (S, h_1, h_2) .
 4. If $\deg(\text{Lagrange}(\{(\alpha_i, \beta_i)\}_{i \in [L]})) > D$:
 - (a) Let L_0 be the interpolation of the first $D + 1$ distinct points, denoted without loss of generality as $\{\alpha_i\}_{i \in [D+1]}$.
 - (b) Find $i^* \in \{D + 2, \dots, L\}$ such that $\beta_{i^*} \neq L_0(\alpha_{i^*})$.
 - (c) Find $S := \{\alpha_{i_j}\}_{j \in [D+1]}$ from $\{\alpha_i\}_{i \in [D+1] \cup \{i^*\}}$ such that $[\text{Lagrange}(\{(\alpha_{i_j}, \beta_{i_j})\}_{\alpha_{i_j} \in S})]_1 \neq \text{cm}$.
 - (d) Set $L(X) := \text{Lagrange}(\{(\alpha_{i_j}, \beta_{i_j})\}_{\alpha_{i_j} \in S})$.
 - (e) Set $h_1 := \text{cm} - [L(\tau)]_1$.
 - (f) Compute $d_j := \frac{1}{Z_{S \setminus \{\alpha_{i_j}\}}(\alpha_{i_j})}$ for each $j \in [D + 1]$.
 - (g) Set $h_2 := \sum_{j \in [D+1]} d_j \cdot \text{pf}_{i_j}$.
 - (h) Output (S, h_1, h_2) .
 5. Otherwise, output $(\perp, \perp, \perp)$.

Running time. Step 3 takes time at most L^2 , Step 3a takes time at most $2D$, Step 3b takes time at most D , Step 4 takes time L^2 , Step 4a takes time $L + (D + 1)^2$, Step 4b takes time $D \cdot (L - D - 1)$, Step 4c takes time at most $2D + (D + 1)^2$, Step 4d takes time $(D + 1)^2$, Step 4e takes time D , and Step 4g takes time $2D$. Hence, the time complexity $t_{ARSDH} \leq t_{FC} + 2L^2 + L + D \cdot (L + 8) + 2(D + 1)^2$.

Success probability. When A_{FC} is successful, there are two cases:

1. There are two valid openings $(\alpha, \beta_i, \text{pf}_i)$ and $(\alpha, \beta_j, \text{pf}_j)$ where $\beta_i \neq \beta_j$. Step 3 in A_{ARSDH} finds the two inconsistent openings and compute $h_2 := \frac{\text{pf}_i - \text{pf}_j}{\beta_j - \beta_i} = [\frac{1}{\tau - \alpha}]_1$, which implies that $h_2 = [\frac{1}{\tau - \alpha}]_1 = \frac{1}{Z_S(\tau)} \cdot h_1$.
2. $\deg(\text{Lagrange}(\{(\alpha_i, \beta_i)\}_{i \in [L]})) > D$. According to the construction of A_{ARSDH} , the following holds:
 - There must exist $(\alpha_{i^*}, \beta_{i^*})$ for some $i^* \in \{D + 2, \dots, L\}$ such that $\beta_{i^*} \neq L_0(\alpha_{i^*})$, because otherwise $\deg(\text{Lagrange}(\{(\alpha_i, \beta_i)\}_{i \in [L]})) = \deg(L_0) \leq D$.
 - $\deg(\text{Lagrange}(\{(\alpha_i, \beta_i)\}_{i \in [D+1] \cup \{i^*\}})) \geq D + 1$, because otherwise, by the fundamental theorem of algebra, we would have $\text{Lagrange}(\{(\alpha_i, \beta_i)\}_{i \in [D+1] \cup \{i^*\}}) = L_0$.
 - For all size- $(D + 1)$ subsets of $\{(\alpha_i, \beta_i)\}_{i \in [D+1] \cup \{i^*\}}$, their interpolations are pairwise distinct, because otherwise $\deg(\text{Lagrange}(\{(\alpha_i, \beta_i)\}_{i \in [D+1] \cup \{i^*\}})) \leq D$.

Therefore, there are at least $D + 1$ among $D + 2$ possible subsets such that the corresponding interpolation is not consistent with cm , which implies that $h_1 = \text{cm} - [L(\tau)]_1 \neq [0]_1$. Note that by construction,

$$L(\tau) = \sum_{j \in [D+1]} \frac{Z_{S \setminus \{\alpha_{i_j}\}}(\tau)}{Z_{S \setminus \{\alpha_{i_j}\}}(\alpha_{i_j})} \beta_{i_j} = Z_S(\tau) \sum_{j \in [D]} \frac{\beta_{i_j}}{(\tau - \alpha_{i_j}) Z_{S \setminus \{\alpha_{i_j}\}}(\alpha_{i_j})}.$$

This implies that

$$h_2 = \sum_{j \in [D+1]} d_j \cdot \text{pf}_{i_j} = \sum_{j \in [D+1]} \frac{1}{Z_{S \setminus \{\alpha_{i_j}\}}(\alpha_{i_j})} \cdot \frac{\text{cm} - [\beta_{i_j}]_1}{\tau - \alpha_{i_j}} = \frac{1}{Z_S(\tau)} \cdot (\text{cm} - [L(\tau)]_1) = \frac{1}{Z_S(\tau)} \cdot h_1 .$$

□

9.2 Proof of Corollary 9.2 (batch KZG)

For KZG, the alphabet Σ is $\mathbb{F}$, and the space of commitments and proofs is $\mathbb{G}_1$. Further, KZG is triply homomorphic, and can thus be compiled into a batch polynomial commitment scheme $\text{bKZG} = \text{BatchMsg}[\text{KZG}, m]$ using Construction 8.5. Thus, Corollary 9.2 follows from Lemmas 8.2 and 9.1.

Lemma 9.7. *KZG is triply homomorphic (Definition 8.4).*

Proof. Fix pp_{FC} , fix $\text{cm}, \text{cm}' \in \mathbb{G}_1$, $\alpha, \beta, \beta' \in \mathbb{F}$, and $\text{pf}, \text{pf}' \in \mathbb{G}_1$ such that $\text{FC.Check}(\text{pp}_{\text{FC}}, \text{cm}, \alpha, \beta, \text{pf}) = 1$ and $\text{FC.Check}(\text{pp}_{\text{FC}}, \text{cm}', \alpha, \beta', \text{pf}') = 1$. Then

$$\begin{aligned} & e((\text{cm} + \text{cm}') - [\beta + \beta']_1, [1]_2) - e(\text{pf} + \text{pf}', [\tau - \alpha]_2) \\ &= e(\text{cm} - [\beta]_1, [1]_2) + e(\text{cm}' - [\beta']_1, [1]_2) - e(\text{pf}, [\tau - \alpha]_2) - e(\text{pf}', [\tau - \alpha]_2) \\ &= 0 . \end{aligned}$$

□

For completeness, we show the construction of bKZG below; differences with the plain KZG scheme are shown in blue.

Construction 9.8 (Batch KZG).

- $\text{bKZG.Gen}(1^\lambda, D, m)$:
 1. Choose $(p, \mathbb{G}_1, \mathbb{G}_2, [1]_1, [1]_2, \mathbb{G}_T, e)$ where $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are additive groups of prime order p , $[1]_1$ is a generator of $\mathbb{G}_1$, $[1]_2$ is a generator of $\mathbb{G}_2$, and $e: \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is a bilinear pairing.
 2. Sample $\tau \leftarrow \mathbb{Z}_p \setminus \{0\}$.
 3. Output $\text{pp}_{\text{FC}} := ([1]_1, [\tau]_1, [\tau^2]_1, \dots, [\tau^D]_1, [1]_2, [\tau]_2)$.
- $\text{bKZG.Commit}(\text{pp}_{\text{FC}}, f)$:
 1. Parse pp_{FC} as $([1]_1, [\tau]_1, [\tau^2]_1, \dots, [\tau^D]_1, [1]_2, [\tau]_2)$ and f as $(\text{p}_b)_{b \in [m]}$.
 2. For $b \in [m]$: Compute $\text{cm}_b := [\text{p}_b(\tau)]_1$.
 3. Set $\text{cm} := (\text{cm}_b)_{b \in [m]}$.
 4. Set $\text{aux} := f$.
 5. Output (cm, aux) .
- $\langle \mathcal{P}_{\text{bKZG}}(\text{pp}_{\text{FC}}, \text{cm}, \text{aux}, \alpha, \beta), \mathcal{V}_{\text{bKZG}}(\text{pp}_{\text{FC}}, \text{cm}, \alpha, \beta) \rangle$
 1. Both $\mathcal{P}_{\text{bKZG}}$ and $\mathcal{V}_{\text{bKZG}}$ parse cm as $(\text{cm}_b)_{b \in m}$ and β as $(\beta_b)_{b \in m}$.
 2. $\mathcal{P}_{\text{bKZG}}$ parses aux as $(\text{p}_b)_{b \in [m]}$.
 3. $\mathcal{V}_{\text{FC}}$ sends a random challenge $\gamma \leftarrow \mathbb{F}$ to $\mathcal{P}_{\text{FC}}$.

4. $\mathcal{P}_{\text{FC}}$ computes $\text{pf} := [\text{p}_\alpha(\tau)]_1$ where

$$\text{p}_\alpha(X) := \sum_{b \in [m]} \gamma^{b-1} \text{KZG.Open}(\text{pp}_{\text{FC}}, \text{cm}_b, \text{aux}_b, \alpha, \beta_b) = \sum_{b \in [m]} \gamma^{b-1} \frac{\text{p}_b(X) - \text{p}_b(\alpha)}{X - \alpha},$$

and sends pf to $\mathcal{V}_{\text{FC}}$.

5. $\mathcal{V}_{\text{FC}}$ checks if $\text{KZG.Check}(\text{pp}_{\text{FC}}, \sum_{b \in [m]} \gamma^{b-1} \text{cm}_b, \alpha, \sum_{b \in [m]} \gamma^{b-1} \beta_b, \text{pf})$, i.e., if

$$e\left(\sum_{b \in [m]} \gamma^{b-1} (\text{cm}_b - [\beta_b]_1), [1]_2\right) = e(\text{pf}, [\tau - \alpha]_2).$$

Definition 9.9. Let $\mathbb{F}$ be a field and $Z_S(X) := \prod_{s \in S} (X - s)$ for $S \subseteq \mathbb{F}$. The **expected-time adaptive rational strong Diffie–Hellman (ARSDH)** holds with error ϵ_{ARSDH} if for every security parameter λ , adversary runtime bound t_{ARSDH}^* , adversary A_{ARSDH}^* running in expected time t_{ARSDH}^* , and tuple $(p, \mathbb{G}_1, \mathbb{G}_2, [1]_1, [1]_2, \mathbb{G}_T, e)$ where $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are groups of prime order $p(\lambda)$, $[1]_1$ is a generator of $\mathbb{G}_1$, $[1]_2$ is a generator of $\mathbb{G}_2$, and $e: \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is a bilinear pairing,

$$\Pr \left[\begin{array}{l} S \subseteq \mathbb{F}, |S| = D + 1 \\ \wedge h_1, h_2 \in \mathbb{G}_1, h_1 \neq [0]_1 \\ \wedge h_2 = \frac{1}{Z_S(\tau)} \cdot h_1 \end{array} \middle| \begin{array}{l} \tau \leftarrow \mathbb{Z}_p \setminus \{0\} \\ (S, h_1, h_2) \leftarrow A_{\text{ARSDH}}^*([1]_1, [\tau]_1, [\tau^2]_1, \dots, [\tau^D]_1, [1]_2, [\tau]_2) \end{array} \right] \leq \epsilon_{\text{ARSDH}}(\lambda, D, t_{\text{ARSDH}}^*).$$

Remark 9.10. [LPS25, Section 4.1] considers a slightly different construction with a two-round protocol (an initial round in which the verifier sends the evaluation point), and where the evaluation point α must be chosen uniformly at random in $\mathbb{F}$. In our case, we consider a one-round protocol and do not restrict the distribution of the evaluation point (which allows us to compile the batch KZG FC with *any* FIOP).

9.3 Proof of Lemma 9.3 (KZG with linear combinations)

Definition 9.11. Let $\mathbb{F}$ be a field and $Z_S(X) := \prod_{s \in S} (X - s)$ for $S \subseteq \mathbb{F}$. The **expected-time Split Rational Strong Diffie–Hellman (SplitRSDH)** holds with error $\epsilon_{\text{SplitRSDH}} = \epsilon_{\text{SplitRSDH}}(\lambda, n_1, n, d, n_S, (\psi_k)_{k \in [n]}, t_{\text{SplitRSDH}}^*)$ if for every $\lambda \in \mathbb{N}$, $n_1 \in \mathbb{N}$, $n \in \mathbb{N}$, $n_S \in \mathbb{N}$, $\psi_\ell \in \mathbb{F}[X]^{\leq d}$, running time bound $t_{\text{SplitRSDH}}^* \in \mathbb{N}$, and every adversary $A_{\text{SplitRSDH}}$ running in expected time $t_{\text{SplitRSDH}}^*$,

$$\Pr \left[\begin{array}{l} S \subset \mathbb{F} \wedge |S| = n_S \\ \wedge L \in \mathbb{F}[X] \wedge n_1 + d + 1 \leq \deg(L) \leq n_S - 1 \\ \wedge \sum_{\ell \in [n]} \psi_\ell(\tau) \cdot \text{c}_\ell = [L(\tau)]_1 + Z_S(\tau) \cdot \text{pf} \end{array} \middle| \begin{array}{l} \tau \leftarrow \mathbb{F} \setminus \{0\} \\ \text{pp} := (([\tau^i]_1)_{i=0}^{n_1}, ([1]_2, [\tau]_2)) \\ (S, L, (\text{c}_\ell)_{\ell \in [n]}, \text{pf}) \leftarrow A_{\text{SplitRSDH}}(\text{pp}) \end{array} \right] \leq \epsilon_{\text{SplitRSDH}}(\lambda, n_1, n, d, n_S, (\psi_k)_{k \in [n]}, t_{\text{SplitRSDH}}^*).$$

Proof of Lemma 9.3. We reduce the function binding property of $\text{combFC}[\text{KZG}, n]$ to the evaluation binding property of KZG and to SplitRSDH.

Reduction to the evaluation binding of KZG. Consider the following adversary A_{EB} against evaluation binding of the underlying KZG FC:

$A_{\text{combFC}}(\text{pp}, \text{ai}_{\text{FC}})$:

1. Run $(\text{cm}, (((\alpha_i, h_{i,1}, \dots, h_{i,n}), \beta_i, \text{pf}_i))_{i \in [L]})) \leftarrow A_{\text{combFC}}(\text{pp}, \text{ai}_{\text{FC}})$.
2. If $\exists i \neq j : \alpha_i = \alpha_j \wedge \beta_i \neq \beta_j$: output $(\sum_{\ell \in [m]} h_{i,\ell} \text{cm}_\ell, \alpha_i, \beta_i, \beta_j, \text{pf}_i, \text{pf}_j)$.

3. Otherwise, output $\perp$.

Running time A_{EB} runs in time at most $t_{\text{combFC}} + L^2$.

Success probability Here, we assume that A_{combFC} is the adversary from Section 8.2. Thus, we have the guarantee that if $\alpha_i = \alpha_j$, then $(h_{i,1}, \dots, h_{i,n}) = (h_{j,1}, \dots, h_{j,n})$ (the other case is covered by the reduction to the underlying FC in Section 8.2). Hence, if A_{combFC} is successful and A_{combFC} does not output $\perp$, then it wins the evaluation binding game for KZG.

Reduction to SplitRSDH. Consider the following adversary $A_{\text{SplitRSDH}}$ against SplitRSDH with parameters $\psi_k(X)$ for $k \in [m]$:

$A_{\text{SplitRSDH}}(\text{pp})$:

1. Run $(\text{cm}, ((\alpha_i, h_{i,1}, \dots, h_{i,n}), \beta_i, \text{pf}_i))_{i \in [L]} \leftarrow A_{\text{combFC}}(\text{pp})$.
2. If $\exists i \neq j : \alpha_i = \alpha_j \wedge \beta_i \neq \beta_j$: output $\perp$.
3. Set $n_S := L$.
4. Set $S := \{\alpha_i\}_{i \in [n_S]}$.
5. Compute $\text{pf} := \sum_{i \in [n_S]} \frac{1}{Z_{S \setminus \{\alpha_i\}}(\alpha_i)} \text{pf}_i$.
6. Compute $L(X) := \text{Lagrange}(\{(\alpha_i, \beta_i)\}_{i \in [n_S]})$.
7. Output $(S, L, (\text{cm}_\ell)_{\ell \in [n]}, \text{pf})$

Running time: $A_{\text{SplitRSDH}}$ runs in time at most $t_{\text{combFC}} + 3n_S^2$.

Success probability: Assume that A_{combFC} is successful, i.e., $\text{combFC}(\text{cm}, (\alpha_i, h_{i,1}, \dots, h_{i,n}), \beta_i, \text{pf}_i) = 1$ for all $i \in [L]$, but there is no $(\Pi_\ell)_{\ell \in [n]}$ such that $\forall i \in [L] : \sum_{\ell \in [n]} h_{i,\ell} \cdot \alpha_i(\Pi_\ell) = \beta_i$.

We have

$$\begin{aligned}
Z_S(\tau) \cdot \text{pf} &= \sum_{i \in [n_S]} \frac{Z_{S \setminus \{\alpha_i\}}(\tau)(\tau - \alpha_i)}{Z_{S \setminus \{\alpha_i\}}(\alpha_i)} \text{pf}_i \\
&= \sum_{i \in [n_S]} \frac{Z_{S \setminus \{\alpha_i\}}(\tau)}{Z_{S \setminus \{\alpha_i\}}(\alpha_i)} \left(\left(\sum_{\ell \in [n]} h_{i,\ell} \cdot \text{cm}_\ell \right) - [\beta_i]_1 \right) \\
&= \sum_{\ell \in [n]} \left(\sum_{i \in [n_S]} \frac{Z_{S \setminus \{\alpha_i\}}(\tau)}{Z_{S \setminus \{\alpha_i\}}(\alpha_i)} h_{i,\ell} \right) \text{cm}_\ell - [\text{Lagrange}(\{(\alpha_i, \beta_i)\}_{\alpha_i \in S})(\tau)]_1 \\
&= \sum_{\ell \in [n]} \text{Lagrange}(\{(\alpha_i, h_{i,\ell})\}_{i \in [S]})(\tau) \cdot \text{cm}_\ell - [\text{Lagrange}(\{(\alpha_i, \beta_i)\}_{\alpha_i \in S})(\tau)]_1 \\
&= \sum_{\ell \in [n]} \psi_\ell(\tau) \cdot \text{cm}_\ell - [L(\tau)]_1,
\end{aligned}$$

where we make use of the fact that $\text{Lagrange}(\{(\alpha_i, h_{i,\ell})\}_{i \in [S]})(X) \equiv \psi_\ell(X)$ in the last step.

For $A_{\text{SplitRSDH}}$'s output to be a valid SplitRSDH solution, it remains to argue that the degree of $L(X)$ is within the bounds specified by the assumption. By construction, L has degree at most $n_S - 1$. Further, L must have degree at least $n_1 + d + 1$.

In order to show that L has degree at least $n_1 + d + 1$, we need to leverage the specific instantiation of the query class used in Plonk. For a parameter $w \in \mathbb{N}$, $n = 5$ (there are commitments to polynomials

$\mathbf{c}, \mathbf{z}, \mathbf{t}_{\text{lo}}, \mathbf{t}_{\text{mid}}, \mathbf{t}_{\text{hi}}, n_1 = w + 5, d = 3w$ with

$$\begin{aligned}\psi_{\mathbf{c}}(X) &:= 1, \\ \psi_{\mathbf{z}}(X) &:= \alpha(A(X) + \beta X + \gamma)(B(X) + \beta k_1 X)(C(X) + \beta k_2 X + \gamma) + \alpha^2 \frac{\omega(X^w - 1)}{w(X - \omega)}, \\ \psi_{\mathbf{t}_{\text{lo}}}(X) &:= Z_{\mathbb{H}}(X), \\ \psi_{\mathbf{t}_{\text{mid}}}(X) &:= Z_{\mathbb{H}}(X)X^w, \\ \psi_{\mathbf{t}_{\text{hi}}}(X) &:= Z_{\mathbb{H}}(X)X^{2w},\end{aligned}$$

where A, B, C are polynomials of degree at most w , $\alpha, \beta, \gamma, k_1, k_2, \omega \in \mathbb{F}$ are constants.

Assume towards a contradiction that $\deg(L) \leq n_1 + d = 4w + 5$. Then using polynomial division, one can write

$$L(X) = \psi_{\mathbf{c}}(X) \cdot \mathbf{c}(X) + \psi_{\mathbf{z}}(X) \cdot \mathbf{z}(X) + \psi_{\mathbf{t}_{\text{lo}}}(X) \cdot \mathbf{t}_{\text{lo}}(X) + \psi_{\mathbf{t}_{\text{mid}}}(X) \cdot \mathbf{t}_{\text{mid}}(X) + \psi_{\mathbf{t}_{\text{hi}}}(X) \cdot \mathbf{t}_{\text{hi}}(X)$$

with $\deg(\mathbf{c}), \deg(\mathbf{z}), \deg(\mathbf{t}_{\text{mid}}), \deg(\mathbf{t}_{\text{lo}}) \leq w - 1, \deg(\mathbf{t}_{\text{hi}}) \leq w + 5$. This contradicts the assumption that there is no $(\mathbf{c}, \mathbf{z}, \mathbf{t}_{\text{lo}}, \mathbf{t}_{\text{mid}}, \mathbf{t}_{\text{hi}})$ with the above degree bounds such that $\forall i \in [L] : \sum_{\ell \in [n]} h_{i,\ell} \cdot \Pi_{\ell}(\alpha_i) = \beta_i$. $\square$

9.4 Proof of Corollary 9.4 (linearized KZG)

The KZG polynomial commitment scheme satisfies the conditions required by Construction 8.9 (the message space $\mathbb{F}^{\ell}$ and the commitment space $\mathbb{G}_1$ are $\mathbb{F}$ -modules, batch commitments are done element-wise, and commitments are homomorphic). KZG thus gives rise to the linearized KZG functional commitment scheme linKZG , and Corollary 9.4 follows directly from Lemmas 8.3 and 9.3, together with the fact that KZG is evaluation binding under the SDH assumption [KZG10]. Note that ARSDH implies SDH [LPS24].

A Special function binding

We define the notion of *special function binding* (a function binding analogue of special soundness), and show that it implies state-restoration function binding. Special function binding can be seen as a weaker version of standard special soundness: namely, if an FC satisfies special function binding, then it is infeasible for an adversary to produce accepting trees of transcripts for a commitment and a set of query-answer pairs that are not consistent with a message. Special function binding is useful to show state-restoration function binding for interactive functional commitment schemes when some rewinding is needed.

Definition A.1 (Special function binding). *A FC scheme with a k_{FC} -round opening protocol has expected $(a_1, \dots, a_{k_{\text{FC}}})$ -special function binding error ϵ_{FC} if for every security parameter $\lambda \in \mathbb{N}$, message length $\ell \in \mathbb{N}$, number of samples $L \in \mathbb{N}$, auxiliary input distribution $\mathcal{D}_{\text{FC}}$, expected adversary size bound t_{SFB}^* , and adversary A_{SFB} with expected runtime t_{SFB}^* ,*

$$\Pr \left[\begin{array}{l} \forall i \in [L] : T_i \text{ is an all-accepting } (a_1, \dots, a_{k_{\text{FC}}})\text{-tree for } (\text{cm}, \alpha_i, \beta_i) \\ \text{with paths } ((\text{pm}_{i,j,k}, \text{vm}_{i,j,k}))_{j \in [T], k \in [k_{\text{FC}}]} \\ \wedge \left(\begin{array}{l} \forall i \in [L], j \in [T] : \\ \mathcal{V}_{\text{FC}}(\text{pp}_{\text{FC}}, \text{cm}, \alpha_i, \beta_i, ((\text{pm}_{i,j,k}, \text{vm}_{i,j,k}))_{j \in [T], k \in [k_{\text{FC}}]}) = 1 \\ \wedge \nexists \Pi \text{ s.t. } \forall i \in [L], \alpha_i(\Pi) = \beta_i \end{array} \right) \end{array} \middle| \begin{array}{l} \text{pp}_{\text{FC}} \leftarrow \text{FC.Gen}(1^\lambda, \ell) \\ \text{ai}_{\text{FC}} \leftarrow \mathcal{D}_{\text{FC}} \\ (\text{cm}, ((\alpha_i, \beta_i, T_i))_{i \in [L]}) \\ \leftarrow A_{\text{SFB}}(\text{pp}_{\text{FC}}, \text{ai}_{\text{FC}}) \end{array} \right] \\ \leq \epsilon_{\text{SFB}}(\lambda, \ell, L, t_{\text{SFB}}^*) .$$

Lemma A.2 (Special function binding implies state-restoration function binding). *If the commitment scheme FC with a k_{FC} -round opening protocol and randomness complexities $(r_{\text{FC},1}, \dots, r_{\text{FC},k_{\text{FC}}})$ has expected-time $(a_1, \dots, a_{k_{\text{FC}}})$ -special function binding error ϵ_{SFB} , then FC has state-restoration function binding error*

$$\epsilon_{\text{FC}}^{\text{SR}}(\lambda, \ell, L, s_{\text{FC}}, m_{\text{FC}}, t_{\text{FC}}^{\text{SR}}) \leq L \cdot (m_{\text{FC}} + 1) \cdot \left(\sum_{i \in [k_{\text{FC}}]} \frac{a_i - 1}{2^{r_{\text{FC},i}}} \right) + \epsilon_{\text{SFB}}(\lambda, \ell, L, t_{\text{SFB}}^*) ,$$

where $t_{\text{SFB}}^* \leq L \cdot (m_{\text{FC}} + 1) \left(\prod_{i \in [k_{\text{FC}}]} (a_i - 1) \right) (t_{\text{FC}}^{\text{SR}} + \text{poly}(\ell))$.

To prove Lemma A.2, we will make use of the following lemma.

Lemma A.3. *For any k_{FC} -round interactive argument $(\mathcal{P}_{\text{FC}}, \mathcal{V}_{\text{FC}})$ with randomness complexities $(r_{\text{FC},1}, \dots, r_{\text{FC},k_{\text{FC}}})$, for any move budget $m_{\text{FC}} \in \mathbb{N}$, and for any tree arity $(a_1, \dots, a_{k_{\text{FC}}}) \in \mathbb{N}^{k_{\text{FC}}}$, there exists an algorithm SRTreeFinder with rewinding access to the prover such that for every malicious prover $\tilde{\mathcal{P}}_{\text{PC}}$, commitment cm , point α and evaluation β ,*

$$\Pr \left[\begin{array}{l} T \text{ is not an all-accepting} \\ (a_1, \dots, a_{k_{\text{FC}}})\text{-tree for } (\text{cm}, \alpha, \beta) \\ \wedge b = 1 \end{array} \middle| \begin{array}{l} \text{ai}_{\text{FC}} \leftarrow \mathcal{D}_{\text{FC}} \\ \text{For } j \in [k_{\text{FC}}] : \text{rnd}_{\text{FC},j} \leftarrow \mathcal{U}(r_{\text{FC},j}) \\ ((\text{cm}, \alpha, \beta), ((\text{pm}_j, \eta_j, \text{vm}_j))_{j \in [k_{\text{FC}}]}) \\ \xleftarrow{\text{tr}} \text{SRGame}(s_{\text{FC}}, \text{rnd}_{\text{FC}}, \tilde{\mathcal{P}}_{\text{PC}}, \text{pp}_{\text{FC}}, \text{ai}_{\text{FC}}) \\ T \leftarrow \text{SRTreeFinder}^{\tilde{\mathcal{P}}_{\text{PC}}}((\text{cm}, \alpha, \beta), ((\text{pm}_j, \eta_j, \text{vm}_j))_{j \in [k_{\text{FC}}]}, \text{tr}) \\ b \leftarrow \mathcal{V}_{\text{FC}}(\text{cm}, \alpha, \beta, (\text{pm}_j)_{j \in [k_{\text{FC}}]}, (\text{vm}_j)_{j \in [k_{\text{FC}}]}) \end{array} \right] \\ \leq (m_{\text{FC}} + 1) \cdot \left(\sum_{i \in [k_{\text{FC}}]} \frac{a_i - 1}{2^{r_{\text{FC},i}}} \right) ,$$

where SRTreeFinder makes $(m_{\text{FC}} + 1) \prod_{i \in [k_{\text{FC}}]} (a_i - 1)$ calls to $\tilde{\mathcal{P}}_{\text{FC}}$ in expectation. Moreover, the leftmost transcript in T is the state-restoration adversary's final output $((\text{pm}_j, \text{vm}_j))_{j \in [k_{\text{FC}}]}$ in SRGame .

Proof. This is a direct application of [CY24, Lemma 30.5.2] for the FC prover $\tilde{\mathcal{P}}_{\text{FC}}$, the FC verifier $\mathcal{V}_{\text{FC}}$, and the relation

$$\{((\text{cm}, \alpha, \beta), \Pi) \mid \alpha(\Pi) = \beta\}.$$

□

Proof of Lemma A.2. Given an state-restoration function binding adversary $\tilde{\mathcal{P}}_{\text{FC}}^{\text{SR}}$ for FC, we construct the following reducer from state-restoration function binding to special function binding:

$\mathfrak{R}^{\tilde{\mathcal{P}}_{\text{FC}}^{\text{SR}}}(\text{pp}_{\text{FC}}, \text{aux}):$

1. For $i \in [L]$:
 - (a) Construct an SRGame adversary $\tilde{\mathcal{P}}_i$ as follows: when playing the SRGame , $\tilde{\mathcal{P}}_i$ simulates FCSRGame towards $\tilde{\mathcal{P}}_{\text{FC}}^{\text{SR}}$ by forwarding its queries to the SRGame .
 $\tilde{\mathcal{P}}_i$ outputs $(\text{cm}, (\alpha_i, \beta_i, ((\text{pm}_{i,j}, \eta_{i,j}, \text{vm}_{i,j}))_{j \in [k_{\text{FC}}]}))$, i.e., only the i -th point, its evaluation, and the corresponding opening transcript.
 - (b) For $j \in [k_{\text{FC}}]$: Lazily sample a random oracle $\text{rnd}_{\text{FC},j} \leftarrow \mathcal{U}(r_{\text{FC},j})$.
 - (c) Run $(\text{cm}, (\alpha_i, \beta_i, ((\text{pm}_{i,j}, \eta_{i,j}, \text{vm}_{i,j}))_{j \in [k_{\text{FC}}]})) \xleftarrow{\text{tr}_i} \text{SRGame}(s_{\text{FC}}, \text{rnd}_{\text{FC}}, \tilde{\mathcal{P}}_i, \text{pp}_{\text{FC}}, \text{ai}_{\text{FC}})$ by lazily simulating the random oracles $\text{rnd}_{\text{FC},j}$.
 - (d) Run $\text{T}_i \leftarrow \text{SRTreeFinder}^{\tilde{\mathcal{P}}_i}(\text{cm}, (\alpha_i, \beta_i), ((\text{pm}_{i,j}, \eta_{i,j}, \text{vm}_{i,j}))_{j \in [k_{\text{FC}}]}, \text{tr}_i)$.
2. Output $(\text{cm}, ((\alpha_i, \beta_i))_{i \in [L]}, (((\text{pm}_{i,j}, \eta_{i,j}, \text{vm}_{i,j}))_{j \in [k_{\text{FC}}]})_{i \in [L]}, (\text{T}_i)_{i \in [L]})$.

Running time. $\mathfrak{R}$ makes $L \cdot \left(1 + (m_{\text{FC}} + 1) \prod_{i \in [k_{\text{FC}}]} (a_i - 1)\right)$ calls to $\tilde{\mathcal{P}}_{\text{FC}}^{\text{SR}}$ in expectation.

Success probability. We first note that $\mathfrak{R}$ perfectly simulates FCSRGame towards $\tilde{\mathcal{P}}_{\text{FC}}^{\text{SR}}$, and thus the experiments

$$\left[\begin{array}{l} \text{pp} \leftarrow \text{FC.Gen}(1^\lambda, n) \\ \mathcal{D}_{\text{FC}} \leftarrow \mathcal{D}_{\text{FC}} \\ \text{For } r \in [k_{\text{FC}}] : \text{rnd}_{\text{FC},r} \leftarrow \mathcal{U}(r_{\text{FC},r}) \\ \left(\text{cm}, \left(\left(\begin{array}{l} \alpha_i, \beta_i, \\ (\text{pm}_{i,r})_{r \in [k_{\text{FC}}]}, (\eta_{i,r})_{r \in [k_{\text{FC}}]}, (\text{vm}_{i,r})_{r \in [k_{\text{FC}}]} \end{array} \right) \right)_{i \in [L]} \right) \leftarrow \text{FCSRGame}(s_{\text{FC}}, \text{rnd}_{\text{FC}}, \tilde{\mathcal{P}}_{\text{FC}}^{\text{SR}}, \text{pp}, \text{ai}_{\text{FC}}) \end{array} \right]$$

and

$$E_{\text{SFB}} := \left[\begin{array}{l} \text{pp}_{\text{FC}} \leftarrow \text{FC.Gen}(1^\lambda, \ell) \\ \text{ai}_{\text{FC}} \leftarrow \mathcal{D}_{\text{FC}} \\ (\text{cm}, ((\alpha_i, \beta_i))_{i \in [L]}, (((\text{pm}_{i,j}, \eta_{i,j}, \text{vm}_{i,j}))_{j \in [k_{\text{FC}}]})_{i \in [L]}, (\text{T}_i)_{i \in [L]}) \leftarrow \mathfrak{R}^{\tilde{\mathcal{P}}_{\text{FC}}^{\text{SR}}}(\text{pp}_{\text{FC}}, \text{ai}_{\text{FC}}) \end{array} \right]$$

are equivalent.

We bound

$$\begin{aligned} & \Pr \left[\left(\begin{array}{l} \forall i \in [L] : \mathcal{V}_{\text{FC}}(\text{pp}_{\text{FC}}, \text{cm}, \alpha_i, \beta_i, ((\text{pm}_{i,j}, \text{vm}_{i,j}))_{j \in [k_{\text{FC}}]}) = 1 \\ \wedge \nexists \Pi \text{ s.t. } \forall i \in [L], \alpha_i(\Pi) = \beta_i \end{array} \right) \mid E_{\text{SFB}} \right] \\ &= \Pr \left[\begin{array}{l} \forall i \in [L] : \text{T}_i \text{ is an all-accepting } (a_1, \dots, a_{k_{\text{FC}}})\text{-tree for } (\text{cm}, \alpha_i, \beta_i) \\ \wedge \left(\begin{array}{l} \forall i \in [L] : \mathcal{V}_{\text{FC}}(\text{pp}_{\text{FC}}, \text{cm}, \alpha_i, \beta_i, ((\text{pm}_{i,j}, \text{vm}_{i,j}))_{j \in [k_{\text{FC}}]}) = 1 \\ \wedge \nexists \Pi \text{ s.t. } \forall i \in [L], \alpha_i(\Pi) = \beta_i \end{array} \right) \end{array} \mid E_{\text{SFB}} \right] \end{aligned}$$

$$\begin{aligned}
& + \Pr \left[\begin{array}{l} \exists i \in [L] : T_i \text{ is not an all-accepting } (a_1, \dots, a_{k_{\text{FC}}})\text{-tree for } (\text{cm}, \alpha_i, \beta_i) \\ \wedge \left(\begin{array}{l} \forall i \in [L] : \mathcal{V}_{\text{FC}}(\text{pp}_{\text{FC}}, \text{cm}, \alpha_i, \beta_i, ((\text{pm}_{i,j}, \text{vm}_{i,j}))_{j \in [k_{\text{FC}}]}) = 1 \\ \wedge \nexists \Pi \text{ s.t. } \forall i \in [L], \alpha_i(\Pi) = \beta_i \end{array} \right) \end{array} \middle| E_{\text{SFB}} \right] \\
\leq & \Pr \left[\begin{array}{l} \forall i \in [L] : T_i \text{ is an all-accepting } (a_1, \dots, a_{k_{\text{FC}}})\text{-tree for } (\text{cm}, \alpha_i, \beta_i) \\ \wedge \left(\begin{array}{l} \forall i \in [L] : \mathcal{V}_{\text{FC}}(\text{pp}_{\text{FC}}, \text{cm}, \alpha_i, \beta_i, ((\text{pm}_{i,j}, \text{vm}_{i,j}))_{j \in [k_{\text{FC}}]}) = 1 \\ \wedge \nexists \Pi \text{ s.t. } \forall i \in [L], \alpha_i(\Pi) = \beta_i \end{array} \right) \end{array} \middle| E_{\text{SFB}} \right] \\
& + \Pr \left[\begin{array}{l} \exists i \in [L] : T_i \text{ is not an all-accepting } (a_1, \dots, a_{k_{\text{FC}}})\text{-tree for } (\text{cm}, \alpha_i, \beta_i) \\ \wedge \forall i \in [L] : \mathcal{V}_{\text{FC}}(\text{pp}_{\text{FC}}, \text{cm}, \alpha_i, \beta_i, ((\text{pm}_{i,j}, \text{vm}_{i,j}))_{j \in [k_{\text{FC}}]}) = 1 \end{array} \middle| E_{\text{SFB}} \right] .
\end{aligned}$$

The first probability is bounded by $\epsilon_{\text{SFB}}(\lambda, \ell, L, t_{\text{SFB}}^*)$. For the second term, we have

$$\begin{aligned}
& \Pr \left[\begin{array}{l} \exists i \in [L] : T_i \text{ is not an all-accepting } (a_1, \dots, a_{k_{\text{FC}}})\text{-tree for } (\text{cm}, \alpha_i, \beta_i) \\ \wedge \forall i \in [L] : \mathcal{V}_{\text{FC}}(\text{pp}_{\text{FC}}, \text{cm}, \alpha_i, \beta_i, ((\text{pm}_{i,j}, \text{vm}_{i,j}))_{j \in [k_{\text{FC}}]}) = 1 \end{array} \middle| E_{\text{SFB}} \right] \\
\leq & \Pr \left[\begin{array}{l} \exists i \in [L] : \left(\begin{array}{l} T_i \text{ is not an all-accepting } (a_1, \dots, a_{k_{\text{FC}}})\text{-tree for } (\text{cm}, \alpha_i, \beta_i) \\ \wedge \mathcal{V}_{\text{FC}}(\text{pp}_{\text{FC}}, \text{cm}, \alpha_i, \beta_i, ((\text{pm}_{i,j}, \text{vm}_{i,j}))_{j \in [k_{\text{FC}}]}) = 1 \end{array} \right) \end{array} \middle| E_{\text{SFB}} \right] \\
\leq & \sum_{i \in [L]} \Pr \left[\begin{array}{l} T_i \text{ is not an all-accepting } (a_1, \dots, a_{k_{\text{FC}}})\text{-tree for } (\text{cm}, \alpha_i, \beta_i) \\ \wedge \mathcal{V}_{\text{FC}}(\text{pp}_{\text{FC}}, \text{cm}, \alpha_i, \beta_i, ((\text{pm}_{i,j}, \text{vm}_{i,j}))_{j \in [k_{\text{FC}}]}) = 1 \end{array} \middle| E_{\text{SFB}} \right] \\
\leq & L \cdot (m_{\text{FC}} + 1) \cdot \left(\sum_{i \in [k_{\text{FC}}]} \frac{a_i - 1}{2^{r_{\text{FC}, i}}} \right) .
\end{aligned}$$

□

B Comparing function binding to other properties for KZG

We discuss how function binding relates to other properties of the KZG polynomial commitment schemes introduced in prior works. We show in Appendix B.1 that function binding implies strong correctness [KZG10]. In Appendix B.2, we show that function binding is implied by interpolation binding, and that interpolation binding holds under the ARSDH assumption (the same assumption we use to show that KZG is function binding in Section 9.1).

B.1 Function binding implies strong correctness

Definition B.1 (Strong correctness [KZG10]). *A polynomial commitment scheme PC has strong correctness error ϵ_{FCSC} if for every security parameter $\lambda \in \mathbb{N}$, polynomial degree bound $D \in \mathbb{N}$, auxiliary input distribution $\mathcal{D}$, adversary size bound $t_{\text{FCSC}} \in \mathbb{N}$, and t_{FCSC} -size circuit A_{FCSC} ,*

$$\Pr \left[\begin{array}{l} d > D \\ \wedge \forall i \in [d+1], p(\alpha_i) = \beta_i \\ \wedge \forall i \in [d+1], \text{PC.Check}(\text{pp}_{\text{PC}}, \text{cm}, \alpha_i, \beta_i, \text{pf}_i) = 1 \\ \wedge \deg(\text{Lagrange}(\{(\alpha_i, \beta_i)\}_{i \in [d+1]})) = d \\ \wedge \deg(\text{Lagrange}(\{(\alpha_i, \text{pf}_i)\}_{i \in [d+1]})) = d-1 \end{array} \mid \begin{array}{l} \text{pp}_{\text{PC}} \leftarrow \text{PC.Gen}(1^\lambda, D) \\ \text{ai} \leftarrow \mathcal{D} \\ (d, p, \text{cm}, \text{state}) \leftarrow A_{\text{FCSC}}(\text{pp}_{\text{PC}}, \text{ai}) \\ \{\alpha_i\}_{i \in [d+1]} \leftarrow \mathbb{F} \text{ with } \alpha_i \neq \alpha_j, \forall i \neq j \in [d+1] \\ \{(\beta_i, \text{pf}_i)\}_{i \in [d+1]} \leftarrow A_{\text{FCSC}}(\text{pp}_{\text{PC}}, \{\alpha_i\}_{i \in [d+1]}, \text{state}) \end{array} \right] \leq \epsilon_{\text{FCSC}}(\lambda, D, t_{\text{FCSC}}).$$

Lemma B.2. *The KZG polynomial commitment scheme (Construction 9.5) has function binding error $\epsilon_{\text{FC}} = \epsilon_{\text{FC}}(\lambda, D, L, t_{\text{FC}})$. Then KZG has strong correctness error $\epsilon_{\text{FCSC}} = \epsilon_{\text{FCSC}}(\lambda, D, t_{\text{FCSC}})$ such that for every security parameter $\lambda \in \mathbb{N}$, polynomial degree bound $D \in \mathbb{N}$, and adversary size bound $t_{\text{FCSC}} \in \mathbb{N}$,*

$$\epsilon_{\text{FCSC}}(\lambda, D, t_{\text{FCSC}}) \leq \epsilon_{\text{FC}}(\lambda, D, L, t_{\text{FC}}) \quad \text{where} \quad \begin{cases} L = O(t_{\text{FCSC}}) \\ t_{\text{FC}} = O(t_{\text{FCSC}}) \end{cases}.$$

Proof. Fix the security parameter $\lambda \in \mathbb{N}$, polynomial degree bound $D \in \mathbb{N}$, and adversary size bound $t_{\text{FCSC}} \in \mathbb{N}$. For every auxiliary input ai and t_{FCSC} -size adversary A_{FCSC} against strong correctness, consider the following adversary A_{FC} against function binding:

- $A_{\text{FC}}(\text{pp}_{\text{PC}}, \text{ai}_{\text{FC}})$:
1. Run $(d, p, \text{cm}, \text{state}) \leftarrow A_{\text{FCSC}}(\text{pp}_{\text{PC}}, \text{ai})$.
 2. Sample $\{\alpha_i\}_{i \in [d+1]} \leftarrow \mathbb{F}$ with $\alpha_i \neq \alpha_j, \forall i \neq j \in [d+1]$.
 3. Run $\{(\beta_i, \text{pf}_i)\}_{i \in [d+1]} \leftarrow A_{\text{FCSC}}(\text{pp}_{\text{PC}}, \{\alpha_i\}_{i \in [d+1]}, \text{state})$.
 4. Output $(\text{cm}, \{(\alpha_i, \beta_i, \text{pf}_i)\}_{i \in [d+1]})$.

Success probability. When A_{FCSC} successfully breaks strong correctness, the following holds:

- $\forall i \in [d+1], \text{KZG.Check}(\text{pp}_{\text{PC}}, \text{cm}, \alpha_i, \beta_i, \text{pf}_i) = 1$; and
- $d := \deg(\text{Lagrange}(\{(\alpha_i, \beta_i)\}_{i \in [d+1]})) > D$.

Therefore,

$$\Pr \left[\begin{array}{l} d > D \\ \wedge \forall i \in [d+1], p(\alpha_i) = \beta_i \\ \wedge \forall i \in [d+1], \text{KZG.Check}(\text{pp}_{\text{PC}}, \text{cm}, \alpha_i, \beta_i, \text{pf}_i) = 1 \\ \wedge \deg(\text{Lagrange}(\{(\alpha_i, \beta_i)\}_{i \in [d+1]})) = d \\ \wedge \deg(\text{Lagrange}(\{(\alpha_i, \text{pf}_i)\}_{i \in [d+1]})) = d-1 \end{array} \mid \begin{array}{l} \text{pp}_{\text{PC}} \leftarrow \text{KZG.Gen}(1^\lambda, D) \\ \text{ai} \leftarrow \mathcal{D} \\ (d, p, \text{cm}, \text{state}) \leftarrow A_{\text{FCSC}}(\text{pp}_{\text{PC}}, \text{ai}) \\ \{\alpha_i\}_{i \in [d+1]} \leftarrow \mathbb{F} \text{ with } \alpha_i \neq \alpha_j, \forall i \neq j \in [d+1] \\ \{(\beta_i, \text{pf}_i)\}_{i \in [d+1]} \leftarrow A_{\text{FCSC}}(\text{pp}_{\text{PC}}, \{\alpha_i\}_{i \in [d+1]}, \text{state}) \end{array} \right]$$

$$\leq \Pr \left[\begin{array}{l} \forall a \in [L] : \text{KZG.Check}(\text{pp}_{\text{PC}}, \text{cm}, \alpha_i, \beta_i, \text{pf}_i) = 1 \\ \wedge \left(\begin{array}{l} \exists i, j \in [L] : \alpha_i = \alpha_j, \beta_i \neq \beta_j \\ \vee \deg(\text{Lagrange}(\{(\alpha_i, \beta_i)\}_{i \in [L]})) > D \end{array} \right) \end{array} \mid \begin{array}{l} \text{pp}_{\text{PC}} \leftarrow \text{KZG.Gen}(1^\lambda, D) \\ \text{ai}_{\text{FC}} \leftarrow \mathcal{D}_{\text{FC}} \\ (\text{cm}, \{(\alpha_i, \beta_i, \text{pf}_i)\}_{i \in [L]}) \leftarrow A_{\text{FC}}(\text{pp}_{\text{PC}}, \text{ai}_{\text{FC}}) \end{array} \right] \\ \leq \epsilon_{\text{FC}}(\lambda, D, L, t_{\text{FC}}) .$$

Running time. A_{FC} has sample size $L = O(t_{\text{FCSC}})$, and adversary size $t_{\text{FC}} = O(t_{\text{FCSC}})$. $\square$

B.2 Interpolation binding implies function binding

Definition B.3 (Interpolation binding [AJMMS23]). *A polynomial commitment scheme PC has **interpolation binding error** ϵ_{FCIB} if for every security parameter $\lambda \in \mathbb{N}$, polynomial degree bound $D \in \mathbb{N}$, auxiliary input distribution $\mathcal{D}$, adversary size bound $t_{\text{FCIB}} \in \mathbb{N}$, and t_{FCIB} -size circuit A_{FCIB} ,*

$$\Pr \left[\begin{array}{l} \forall i \neq j \in [D+1], \alpha_i \neq \alpha_j \\ \wedge \forall i \in [D+1], \text{PC.Check}(\text{pp}_{\text{PC}}, \text{cm}, \alpha_i, \beta_i, \text{pf}_i) = 1 \\ \wedge \text{cm} \neq \text{PC.Commit}(\text{pp}_{\text{PC}}, \text{Lagrange}(\{(\alpha_i, \beta_i)\}_{i \in [D+1]})) \end{array} \mid \begin{array}{l} \text{pp}_{\text{PC}} \leftarrow \text{PC.Gen}(1^\lambda, D) \\ \text{ai} \leftarrow \mathcal{D} \\ (\text{cm}, \{(\alpha_i, \beta_i, \text{pf}_i)\}_{i \in [D+1]}) \leftarrow A_{\text{FCIB}}(\text{pp}_{\text{PC}}, \text{ai}) \end{array} \right] \\ \leq \epsilon_{\text{FCIB}}(\lambda, D, t_{\text{FCIB}}) .$$

[AJMMS23] proves that the KZG polynomial commitment scheme has interpolation binding in the algebraic group model, under the discrete logarithm (Definition C.9) and strong Diffie–Hellman assumption. We show that interpolation binding implies function binding. For completeness, we also provide the proof for interpolation binding from the ARSDH assumption, which is adapted from the proof for special soundness from the ARSDH assumption, as given in [LPS24].

Lemma B.4. *Let KZG be the polynomial commitment scheme constructed in [KZG10] (Construction 9.5). Assume KZG has evaluation binding error $\epsilon_{\text{FCEB}} = \epsilon_{\text{FCEB}}(\lambda, D, t_{\text{FCEB}})$ and interpolation binding error $\epsilon_{\text{FCIB}} = \epsilon_{\text{FCIB}}(\lambda, D, t_{\text{FCIB}})$. Then, KZG has function binding error $\epsilon_{\text{FC}} = \epsilon_{\text{FC}}(\lambda, D, L, t_{\text{FC}})$ such that for every security parameter $\lambda \in \mathbb{N}$, polynomial degree bound $D \in \mathbb{N}$, sample set size $L \in \mathbb{N}$, and adversary size bound $t_{\text{FC}} \in \mathbb{N}$,*

$$\epsilon_{\text{FC}}(\lambda, D, L, t_{\text{FC}}) \leq \epsilon_{\text{FCEB}}(\lambda, D, t_{\text{FCEB}}) + \epsilon_{\text{FCIB}}(\lambda, D, t_{\text{FCIB}}) ,$$

where $t_{\text{FCEB}} \leq t_{\text{FC}} + L^2$ and $t_{\text{FCIB}} \leq t_{\text{FC}} + L + 3L^2 + D \cdot (L - D + 1)$.

Proof of Lemma B.4. Fix the security parameter $\lambda \in \mathbb{N}$, polynomial degree bound $D \in \mathbb{N}$, sample set size $L \in \mathbb{N}$, and adversary size bound $t_{\text{FC}} \in \mathbb{N}$. For every auxiliary input ai and t_{FC} -size adversary A_{FC} against function binding, consider the following adversaries: A_{FCEB} against evaluation binding and A_{FCIB} against interpolation binding:

$A_{\text{FCEB}}(\text{pp}_{\text{PC}}, \text{ai}_{\text{FC}})$:

1. Run $(\text{cm}, \{(\alpha_i, \beta_i, \text{pf}_i)\}_{i \in [L]}) \leftarrow A_{\text{FC}}(\text{pp}_{\text{PC}}, \text{ai}_{\text{FC}})$.
2. If there exists $i \neq j \in [L]$ such that $\alpha_i = \alpha_j$ and $\beta_i \neq \beta_j$, output $(\text{cm}, \alpha_i, \beta_i, \beta_j, \text{pf}_i, \text{pf}_j)$.
3. Otherwise, output $(\text{cm}, \alpha_1, \beta_1, \beta_1, \text{pf}_1, \text{pf}_1)$.

Step 2 in A_{FCEB} takes time at most L^2 , thus the running time of A_{FCEB} is $t_{\text{FCEB}} \leq t_{\text{FC}} + L^2$.

$A_{\text{FCIB}}(\text{pp}_{\text{PC}}, \text{ai}_{\text{FC}})$:

1. Run $(\text{cm}, \{(\alpha_i, \beta_i, \text{pf}_i)\}_{i \in [L]}) \leftarrow A_{\text{FC}}(\text{pp}_{\text{PC}}, \text{ai}_{\text{FC}})$.
2. If $\deg(\text{Lagrange}(\{(\alpha_i, \beta_i)\}_{i \in [L]})) > D$:

- (a) Let L_0 be the interpolation of the first $D + 1$ distinct points, denoted without loss of generality as $\{\alpha_i\}_{i \in [D+1]}$.
 - (b) Find (α_k, β_k) for some $k \in \{D + 2, \dots, L\}$ such that $\beta_k \neq L_0(\alpha_k)$.
 - (c) Sample $S := \{\alpha'_i\}_{i \in [D+1]}$ from $\{\alpha_i\}_{i \in [D+1] \cup \{k\}}$ such that $[\text{Lagrange}(\{(\alpha, \beta[\alpha])\}_{\alpha \in S})]_1 \neq \text{cm}$.
 - (d) Output $(\text{cm}, \{(\alpha'_i, \beta'_i, \text{pf}'_i)\}_{i \in [D+1]})$.
3. Otherwise, output $(\text{cm}, \perp)$.

In the construction of A_{FCIB} , Step 2 takes time L^2 , Step 2a takes time $L + (D + 1)^2$, Step 2b takes time $D \cdot (L - D - 1)$, and Step 2c takes time at most $2D + (D + 1)^2$. Therefore, the total running time is $t_{\text{FCIB}} \leq t_{\text{FC}} + L + 3L^2 + D \cdot (L - D + 1)$.

When A_{FC} successfully breaks function binding, there are two cases:

- There are two valid openings $(\alpha, \beta_i, \text{pf}_i)$ and $(\alpha, \beta_j, \text{pf}_j)$ where $\beta_i \neq \beta_j$. In this case, A_{FCEB} succeeds.
- $\deg(\text{Lagrange}(\{(\alpha_i, \beta_i)\}_{i \in [L]})) > D$. From the construction of A_{FCIB} , the following holds:
 - There must exist (α_k, β_k) for some $k \in \{D + 2, \dots, L\}$ such that $\beta_k \neq L_0(\alpha_k)$, because otherwise $\deg(\text{Lagrange}(\{(\alpha_i, \beta_i)\}_{i \in [L]})) = \deg(L_0) \leq D$.
 - $\deg(\text{Lagrange}(\{(\alpha_i, \beta_i)\}_{i \in [D+1] \cup \{k\}})) \geq D + 1$, because otherwise, by the fundamental theorem of algebra, we would have $\text{Lagrange}(\{(\alpha_i, \beta_i)\}_{i \in [D+1] \cup \{k\}}) = L_0$.
 - For all size- $(D + 1)$ subsets of $\{(\alpha_i, \beta_i)\}_{i \in [D+1] \cup \{k\}}$, their interpolations are pairwise distinct, because otherwise $\deg(\text{Lagrange}(\{(\alpha_i, \beta_i)\}_{i \in [D+1] \cup \{k\}})) \leq D$.

Therefore, there are at least $D + 1$ among $D + 2$ possible subsets such that the corresponding interpolation is not consistent with cm , which implies that A_{FCIB} succeeds with high probability.

By union bound,

$$\begin{aligned}
& \Pr \left[\begin{array}{l} \forall i \in [L] : \text{KZG.Check}(\text{pp}_{\text{PC}}, \text{cm}, \alpha_i, \beta_i, \text{pf}_i) = 1 \\ \wedge \left(\begin{array}{l} \exists i, j \in [L] : \alpha_i = \alpha_j, \beta_i \neq \beta_j \\ \vee \deg(\text{Lagrange}(\{(\alpha_i, \beta_i)\}_{i \in [L]})) > D \end{array} \right) \end{array} \middle| \begin{array}{l} \text{pp}_{\text{PC}} \leftarrow \text{KZG.Gen}(1^\lambda, D) \\ \text{ai}_{\text{FC}} \leftarrow \mathcal{D}_{\text{FC}} \\ (\text{cm}, \{(\alpha_i, \beta_i, \text{pf}_i)\}_{i \in [L]}) \leftarrow A_{\text{FC}}(\text{pp}_{\text{PC}}, \text{ai}_{\text{FC}}) \end{array} \right] \\
& \leq \Pr \left[\begin{array}{l} \beta_i \neq \beta_j \\ \wedge \text{KZG.Check}(\text{pp}_{\text{PC}}, \text{cm}, \alpha, \beta_i, \text{pf}_i) = 1 \\ \wedge \text{KZG.Check}(\text{pp}_{\text{PC}}, \text{cm}, \alpha, \beta_j, \text{pf}_j) = 1 \end{array} \middle| \begin{array}{l} \text{pp}_{\text{PC}} \leftarrow \text{KZG.Gen}(1^\lambda, D) \\ \text{ai} \leftarrow \mathcal{D} \\ (\text{cm}, \alpha, \beta_i, \beta_j, \text{pf}_i, \text{pf}_j) \leftarrow A_{\text{FCEB}}(\text{pp}_{\text{PC}}, \text{ai}) \end{array} \right] \\
& + \Pr \left[\begin{array}{l} \forall i \neq j \in [D + 1], \alpha'_i \neq \alpha'_j \\ \wedge \forall i \in [D + 1], \text{KZG.Check}(\text{pp}_{\text{PC}}, \text{cm}, \alpha'_i, \beta'_i, \text{pf}'_i) = 1 \\ \wedge \text{cm} \neq \text{KZG.Commit}(\text{pp}_{\text{PC}}, \text{Lagrange}(\{(\alpha_i, \beta'_i)\}_{i \in [D+1]})) \end{array} \middle| \begin{array}{l} \text{pp}_{\text{PC}} \leftarrow \text{KZG.Gen}(1^\lambda, D) \\ \text{ai} \leftarrow \mathcal{D} \\ (\text{cm}, \{(\alpha_i, \beta'_i, \text{pf}'_i)\}_{i \in [D+1]}) \leftarrow A_{\text{FCIB}}(\text{pp}_{\text{PC}}, \text{ai}) \end{array} \right] \\
& \leq \epsilon_{\text{FCEB}}(\lambda, D, t_{\text{FCEB}}) + \epsilon_{\text{FCIB}}(\lambda, D, t_{\text{FCIB}}) .
\end{aligned}$$

Since the relation holds for any auxiliary input distribution $\mathcal{D}_{\text{FC}}$ and t_{FC} -size adversary A_{FC} against function binding, we conclude that $\epsilon_{\text{FC}}(\lambda, D, L, t_{\text{FC}}) \leq \epsilon_{\text{FCEB}}(\lambda, D, t_{\text{FCEB}}) + \epsilon_{\text{FCIB}}(\lambda, D, t_{\text{FCIB}})$, which completes the proof. $\square$

Lemma B.5. Assume the ARSDH assumption holds with error $\epsilon_{\text{ARSDH}} = \epsilon_{\text{ARSDH}}(\lambda, D, t_{\text{ARSDH}})$. Let KZG be the polynomial commitment scheme constructed in [KZG10] (Construction 9.5). Then KZG has interpolation

binding error $\epsilon_{\text{FCIB}} = \epsilon_{\text{FCIB}}(\lambda, D, t_{\text{FCIB}})$ such that for every security parameter $\lambda \in \mathbb{N}$, polynomial degree bound $D \in \mathbb{N}$, and adversary size bound $t_{\text{FCIB}} \in \mathbb{N}$,

$$\epsilon_{\text{FCIB}}(\lambda, D, t_{\text{FCIB}}) \leq \epsilon_{\text{ARSDH}}(\lambda, D, t_{\text{ARSDH}}) ,$$

where $t_{\text{ARSDH}} \leq t_{\text{FCIB}} + (D + 1)^2 + 3(D + 1)$.

Proof of Lemma B.5. Fix the security parameter $\lambda \in \mathbb{N}$, polynomial degree bound $D \in \mathbb{N}$, and adversary size bound $t_{\text{FCIB}} \in \mathbb{N}$. For every auxiliary input distribution $\mathcal{D}$ and t_{FCIB} -size adversary A_{FCIB} against interpolation binding, consider the following adversary A_{ARSDH} against ARSDH:

- $A_{\text{ARSDH}}([1]_1, [\tau]_1, [\tau^2]_1, \dots, [\tau^D]_1, [1]_2, [\tau]_2)$:
1. Sample $\text{ai} \leftarrow \mathcal{D}$ and set $\text{pp}_{\text{PC}} := ([1]_1, [\tau]_1, [\tau^2]_1, \dots, [\tau^D]_1, [1]_2, [\tau]_2)$.
 2. Run $(\text{cm}, \{(\alpha_i, \beta_i, \text{pf}_i)\}_{i \in [D+1]}) \leftarrow A_{\text{FCIB}}(\text{pp}_{\text{PC}}, \text{ai})$.
 3. Set $S := \{\alpha_i\}_{i \in [D+1]}$.
 4. Set $L(X) := \text{Lagrange}(\{(\alpha_i, \beta_i)\}_{i \in [D+1]})$.
 5. Set $h_1 := \text{cm} - [L(\tau)]_1$.
 6. Compute $d_i := \frac{1}{Z_{S \setminus \{\alpha_i\}}(\alpha_i)}$.
 7. Set $h_2 := \sum_{i=1}^{D+1} d_i \cdot \text{pf}_i$.
 8. Output (S, h_1, h_2) .

Running time. In the construction of A_{ARSDH} , Step 4 takes time $(D + 1)^2$, Step 5 takes time at most D , and Step 7 takes time at most $2D$. Therefore, the total running time is $t_{\text{ARSDH}} \leq t_{\text{FCIB}} + (D + 1)^2 + 3(D + 1)$.

Success probability. When A_{FCIB} succeeds, the following holds:

- $\forall i \neq j \in [D + 1], \alpha_i \neq \alpha_j$, which implies that $|S| = D + 1$;
- $\text{cm} \neq \text{KZG.Commit}(\text{pp}_{\text{PC}}, \text{Lagrange}(\{(\alpha_i, \beta_i)\}_{i \in [D+1]})) = [1]_1^{L(\tau)}$, which implies that $h_1, h_2 \in \mathbb{G}_1$ and $h_1 \neq [0]_1$;
- $\forall i \in [D + 1], \text{KZG.Check}(\text{pp}_{\text{PC}}, \text{cm}, \alpha_i, \beta_i, \text{pf}_i) = 1$, which means for every $i \in [D + 1]$, $e(\text{cm} - [\beta_i]_1, [1]_2) = e(\text{pf}_i, [\tau - \alpha_i]_2)$. By the definition of Lagrange interpolation,

$$\begin{aligned} & \frac{1}{Z_S(\tau)} \cdot h_1 \\ &= \frac{1}{Z_S(\tau)} \cdot (\text{cm} - [L(\tau)]_1) \\ &= \frac{1}{Z_S(\tau)} \cdot (\text{cm} - [\sum_{i=1}^{D+1} \beta_i \cdot \frac{Z_S(\tau)d_i}{\tau - \alpha_i}]_1) \\ &= \frac{1}{Z_S(\tau)} \cdot (\sum_{i=1}^{D+1} \frac{Z_S(\tau)d_i}{\tau - \alpha_i} \cdot \text{cm} - [\sum_{i=1}^{D+1} \beta_i \cdot \frac{Z_S(\tau)d_i}{\tau - \alpha_i}]_1) \\ &= \sum_{i=1}^{D+1} d_i \cdot \frac{\text{cm} - \beta_i}{\tau - \alpha_i} \\ &= \sum_{i=1}^{D+1} d_i \cdot \text{pf}_i \\ &= h_2 . \end{aligned}$$

Hence,

$$\begin{aligned}
& \Pr \left[\begin{array}{l} \forall i \neq j \in [D+1], \alpha_i \neq \alpha_j \\ \wedge \forall i \in [D+1], \text{KZG.Check}(\text{pp}_{\text{PC}}, \text{cm}, \alpha_i, \beta_i, \text{pf}_i) = 1 \\ \wedge \text{cm} \neq \text{KZG.Commit}(\text{pp}_{\text{PC}}, \text{Lagrange}(\{(\alpha_i, \beta_i)\}_{i \in [D+1]})) \end{array} \middle| \begin{array}{l} \text{pp}_{\text{PC}} \leftarrow \text{KZG.Gen}(1^\lambda, D) \\ \text{ai} \leftarrow \mathcal{D} \\ (\text{cm}, \{(\alpha_i, \beta_i, \text{pf}_i)\}_{i \in [D+1]}) \leftarrow A_{\text{FCIB}}(\text{pp}_{\text{PC}}, \text{ai}) \end{array} \right] \\
& \leq \Pr \left[\begin{array}{l} S \subseteq \mathbb{F}, |S| = D+1 \\ \wedge h_1, h_2 \in \mathbb{G}_1, h_1 \neq [0]_1 \\ \wedge h_2 = \frac{1}{Z_S(\tau)} \cdot h_1 \end{array} \middle| \begin{array}{l} \text{pp}_{\text{PC}} \leftarrow \text{KZG.Gen}(1^\lambda, D) \\ (S, h_1, h_2) \leftarrow A_{\text{ARSDH}}(\text{pp}_{\text{PC}}) \end{array} \right] \\
& = \Pr \left[\begin{array}{l} S \subseteq \mathbb{F}, |S| = D+1 \\ \wedge h_1, h_2 \in \mathbb{G}_1, h_1 \neq [0]_1 \\ \wedge h_2 = \frac{1}{Z_S(\tau)} \cdot h_1 \end{array} \middle| \begin{array}{l} \tau \leftarrow \mathbb{Z}_p \setminus \{0\} \\ (S, h_1, h_2) \leftarrow A_{\text{ARSDH}}([1]_1, [\tau]_1, [\tau^2]_1, \dots, [\tau^D]_1, [1]_2, [\tau]_2) \end{array} \right] \\
& \leq \epsilon_{\text{ARSDH}}(\lambda, D, t_{\text{ARSDH}}) .
\end{aligned}$$

Since the relation holds for any auxiliary input distribution $\mathcal{D}$ and t_{FCIB} -size adversary A_{FCIB} against interpolation binding, we conclude that $\epsilon_{\text{FCIB}}(\lambda, D, t_{\text{FCIB}}) \leq \epsilon_{\text{ARSDH}}(\lambda, D, t_{\text{ARSDH}})$, which completes the proof. $\square$

C Function binding for polynomial commitment schemes based on DLog

For the sake of completeness, we include full proofs of function binding for two polynomial commitment schemes (PCSs) of interest based on the discrete logarithm assumption: a square-root-sized PCS inspired by [BCGGHJ17; BG18] in Appendix C.1, and a Bulletproofs-style [BCCGP16] in Appendix C.2.2.

C.1 Square-root-sized polynomial commitment scheme

We analyze a non-interactive polynomial commitment scheme with square-root-sized opening proofs, inspired by constructions from Bootle et al. [BCGGHJ17] and Bootle and Groth [BG18]. [BCGGHJ17] show that their construction is knowledge sound in the Ideal Linear Commitment (ILC) model. [BG18] show that their polynomial commitment scheme is “special-sound”, in the sense that it is possible to extract a polynomial from an adversary that outputs valid proofs for sufficiently many distinct evaluation points. We show that this construction satisfies function binding under the discrete logarithm assumption, using similar techniques as [BG18].

Lemma C.1. *Assume the discrete logarithm relation assumption (Definition C.3) holds with error $\epsilon_{\text{DLogRel}} = \epsilon_{\text{DLogRel}}(\lambda, D, \epsilon_{\text{DLogRel}})$. Then for every security parameter $\lambda \in \mathbb{N}$, polynomial degree bound $D \in \mathbb{N}$, sample set size $L \in \mathbb{N}$, and adversary size bound $t_{\text{FC}} \in \mathbb{N}$, $\text{PC}_{\text{sqr}}(\text{Construction C.2})$ has function binding error*

$$\epsilon_{\text{FC}}(\lambda, D, L, t_{\text{FC}}) \leq \epsilon_{\text{DLogRel}}(\lambda, \sqrt{D+1}, t_{\text{DLogRel}}) ,$$

where $t_{\text{DLogRel}} \leq t_{\text{FC}} + L^2 + (D+1)^{\frac{3}{2}} + (L+1)\sqrt{D+1}$.

Throughout this section, we assume that every polynomial p has degree at most D where $D+1 = d^2$ for some $d \in \mathbb{N}$.² For a polynomial p of degree at most D , we define the $d \times d$ matrix P as

$$P := (p_{(i-1)+d(j-1)})_{i,j \in [d]} = \begin{bmatrix} p_0 & \cdots & p_{d(d-1)} \\ \vdots & \ddots & \vdots \\ p_{d-1} & \cdots & p_D \end{bmatrix} ,$$

such that

$$\begin{bmatrix} 1 & x & \cdots & x^{d-1} \end{bmatrix} P \begin{bmatrix} 1 \\ x^d \\ \vdots \\ x^{d(d-1)} \end{bmatrix} = \sum_{i,j \in [d]} p_{(i-1)+d(j-1)} x^{i-1} x^{d(j-1)} = p(x) .$$

Construction C.2. Let $\mathbb{G}$ be a group of prime order $p(\lambda) \geq 2^\lambda$.

- $\text{PC}_{\text{sqr}} \cdot \text{Gen}(1^\lambda, D)$:
 1. Sample random generators $(G_j)_{j \in [d]} \leftarrow \mathbb{G}^d$.
 2. Output $\text{pp}_{\text{PC}} = (G_j)_{j \in [d]}$.
- $\text{PC}_{\text{sqr}} \cdot \text{Commit}(\text{pp}_{\text{PC}}, p)$:
 1. Parse pp_{PC} as $(G_j)_{j \in [d]}$.

²If this condition is not satisfied, one can consider the next $D' > D$ such that $D' + 1$ is square. Alternatively, one can define a variation of this PCS where the coefficient matrix P is rectangular rather than square.

2. Compute $\text{cm} = (\text{cm}_i)_{i \in [d]}$, where $\text{cm}_i := \sum_{j \in [d]} p_{(i-1)+d(j-1)} G_j$ is a Pedersen commitment to the i -th row of P .
 3. Set $\text{aux} := P$.
 4. Output (cm, aux) .
- $\text{PC}_{\text{sqr}}.\text{Open}(\text{pp}_{\text{PC}}, \text{aux}, \alpha)$:
 1. Parse pp_{PC} as $(G_j)_{j \in [d]}$ and aux as P .
 2. Set $\beta := p(\alpha)$.
 3. Compute $\text{pf} := [1 \quad \alpha \quad \dots \quad \alpha^{d-1}] P$, with $\text{pf}_j = \sum_{i \in [d]} p_{(i-1)+d(j-1)} \cdot \alpha^{i-1}$.
 4. Output (β, pf) .
 - $\text{PC}_{\text{sqr}}.\text{Check}(\text{pp}_{\text{PC}}, \text{cm}, \alpha, \beta, \text{pf})$:
 1. Parse pp_{PC} as $(G_j)_{j \in [d]}$.
 2. Check that $\sum_{i \in [d]} \alpha^{i-1} \text{cm}_i = \sum_{j \in [d]} \text{pf}_j G_j$ and $\beta = \sum_{j \in [d]} \text{pf}_j \alpha^{d(j-1)}$.

We now show that PC_{sqr} satisfies function binding under the discrete logarithm relation assumption, which is closely related to the discrete logarithm assumption.

Definition C.3 (Discrete logarithm relation assumption). *The discrete logarithm relation assumption holds with error $\epsilon_{\text{DLogRel}} = \epsilon_{\text{DLogRel}}(\lambda, n, t_{\text{DLogRel}})$ if for every security parameter λ , length parameter $n \in \mathbb{N}$, adversary size $t_{\text{DLogRel}} \in \mathbb{N}$, t_{DLogRel} -sized adversary A_{DLogRel} , and group $\mathbb{G}$ of prime order $p(\lambda) \geq 2^\lambda$,*

$$\Pr \left[\begin{array}{c} \exists i \in [n] : a_i \neq 0 \wedge \\ \sum_{i \in [n]} a_i G_i = 0 \end{array} \mid \begin{array}{c} (G_1, \dots, G_n) \leftarrow \mathbb{G}^n \\ (a_1, \dots, a_n) \leftarrow A_{\text{DLogRel}}(\mathbb{G}, G_1, \dots, G_n) \end{array} \right] \leq \epsilon_{\text{DLogRel}}(\lambda, n, t_{\text{DLogRel}}) .$$

Definition C.4 (Discrete logarithm assumption). *The discrete logarithm assumption holds with error $\epsilon_{\text{DL}} = \epsilon_{\text{DL}}(\lambda, t_{\text{DL}})$ if for every security parameter λ , adversary size $t_{\text{DL}} \in \mathbb{N}$, t_{DL} -sized adversary A_{DL} , and group $\mathbb{G}$ of prime order $p(\lambda) \geq 2^\lambda$ with generator G ,*

$$\Pr \left[xG = H \mid \begin{array}{c} H \leftarrow \mathbb{G} \\ x \leftarrow A_{\text{DL}}(\mathbb{G}, p, G, H) \end{array} \right] \leq \epsilon_{\text{DL}}(\lambda, t_{\text{DL}}) .$$

Asymptotically, the discrete logarithm and the discrete logarithm relation assumption are equivalent; the following lemma bounds the concrete relationship between the two assumptions.

Lemma C.5 (Discrete logarithm relation to discrete logarithm ([JT20])). *Let $\mathbb{G}$ be a group of prime order $p(\lambda) \geq 2^\lambda$ and $n \geq 1$ be an integer. Then for any $\lambda \in \mathbb{N}$, $t_{\text{DLogRel}} \in \mathbb{N}$, we have*

$$\epsilon_{\text{DLogRel}}(\lambda, n, t_{\text{DLogRel}}) \leq \epsilon_{\text{DL}}(\lambda, t_{\text{DL}}) + \frac{1}{2^\lambda} ,$$

where $t_{\text{DL}} = O(t_{\text{DLogRel}} + n)$.

Proof of Lemma C.1. Given a function binding adversary A_{FC} , we construct the following discrete logarithm relation adversary A_{DLogRel} :

- $A_{\text{DLogRel}}(\mathbb{G}, G_1, \dots, G_d)$:
1. Set $\text{pp}_{\text{PC}} := (G_1, \dots, G_d)$.
 2. Sample $\text{ai}_{\text{FC}} \leftarrow \mathcal{D}_{\text{FC}}$

3. $(\text{cm}, (\alpha_a, \beta_a, \text{pf}_a)_{a \in [L]}) \leftarrow A_{\text{FC}}(\text{pp}_{\text{PC}}, \text{ai}_{\text{FC}})$.
4. If there exists $a \neq b \in [L]$ such that $\alpha_a = \alpha_b$ but $\beta_a \neq \beta_b$: Output $(\text{pf}_{a,j} - \text{pf}_{b,j})_{j \in [d]}$.
5. If $\deg(\text{Lagrange}(\{((\alpha_a, \beta_a))_{a \in [L]}\})) > D$:
 - (a) Find d distinct indices $a_1, \dots, a_d \in [L]$ such that $\alpha_{a_1}, \dots, \alpha_{a_d}$ are distinct.
 - (b) Compute $\begin{bmatrix} \text{p}_{1,1} & \dots & \text{p}_{1,d} \\ \vdots & \ddots & \vdots \\ \text{p}_{d,1} & \dots & \text{p}_{d,d} \end{bmatrix} := \begin{bmatrix} 1 & \dots & \alpha_{a_1}^{d-1} \\ \vdots & \ddots & \vdots \\ 1 & \dots & \alpha_{a_d}^{d-1} \end{bmatrix}^{-1} \begin{bmatrix} \text{pf}_{a_1,1} & \dots & \text{pf}_{a_1,d} \\ \vdots & \ddots & \vdots \\ \text{pf}_{a_d,1} & \dots & \text{pf}_{a_d,d} \end{bmatrix}$.
 - (c) Find $a^* \in [L] \setminus \{a_1, \dots, a_d\}$ such that $\exists j \in [d] : \text{pf}_{a^*,j} \neq \sum_{i \in [d]} \text{p}_{i,j} \alpha_{a^*}^{i-1}$.
 - (d) Output $(\text{pf}_{a^*,j} - \sum_{i \in [d]} \text{p}_{i,j} \alpha_{a^*}^{i-1})_{j \in [d]}$.
6. Abort.

Running time. A_{DLogRel} invokes A_{FC} once. In Step 4, A_{DLogRel} searches for disagreeing answers for the same query in a set of size L , which requires at most L^2 time, and outputs a vector of size d . In Step 5, finding d distinct queries requires L steps, inverting the Vandermonde matrix requires at most d^3 steps, finding a^* can be done in $d + L \cdot d$ steps. This yields a total running time of at most

$$t_{\text{FC}} + \max(L^2 + d, L^2 + d^3 + d + L \cdot d) \leq t_{\text{FC}} + L^2 + (D + 1)^{\frac{3}{2}} + (L + 1)\sqrt{D + 1}.$$

Success probability. Suppose A_{FC} wins the function binding game. Then at least one of the conditions tested in Steps 4 and 5 must be true, which means that A_{DLogRel} does not abort in Step 6.

In Step 4, A_{DLogRel} checks whether A_{FC} wins by breaking evaluation binding. In this case, A_{DLogRel} finds a and b such that $\alpha_a = \alpha_b$ but $\beta_a \neq \beta_b$. However since $\text{PC}_{\text{sqr}}.\text{Check}(\text{pp}_{\text{PC}}, \text{cm}, \alpha_a, \beta_a, \text{pf}_a) = 1$ and $\text{PC}_{\text{sqr}}.\text{Check}(\text{pp}_{\text{PC}}, \text{cm}, \alpha_b, \beta_b, \text{pf}_b) = 1$, it must be that $\beta_a = \sum_{j \in [d]} \text{pf}_{a,j} \alpha_a^{d(j-1)}$ and $\beta_b = \sum_{j \in [d]} \text{pf}_{b,j} \alpha_b^{d(j-1)}$, and thus $\text{pf}_{a,j} \neq \text{pf}_{b,j}$ for at least one index j . Since $\sum_{i \in [d]} \alpha_a^{i-1} \text{cm}_i = \sum_{j \in [d]} \text{pf}_{a,j} G_j$ and $\sum_{i \in [d]} \alpha_b^{i-1} \text{cm}_i = \sum_{j \in [d]} \text{pf}_{b,j} G_j$ from the verification equations, $\sum_{j \in [d]} (\text{pf}_{a,j} - \text{pf}_{b,j}) G_j = 0$, and A_{DLogRel} outputs a non-trivial discrete logarithm relation in Step 4.

In Step 5, A_{DLogRel} checks whether A_{FC} wins by breaking degree binding. Since the queries $\alpha_{a_1}, \dots, \alpha_{a_d}$ are distinct, the Vandermonde matrix in Step 5b is invertible. Further, A_{DLogRel} always find an index a^* in step Step 5c. To see why, assume towards a contradiction that $\text{pf}_{a,j} = \sum_{i \in [d]} \text{p}_{i,j} \alpha_a^{i-1}$ for all $j \in [d]$. Then, since $\text{PC}_{\text{sqr}}.\text{Check}(\text{pp}_{\text{PC}}, \text{cm}, \alpha_a, \beta_a, \text{pf}_a) = 1$ (and thus $\beta_a = \sum_{j \in [d]} \text{pf}_{a,j} \alpha_a^{d(j-1)}$) for all $a \in [L]$, we have that for all $a \in [L]$,

$$\begin{aligned} \beta_a &= \sum_{j \in [d]} \text{pf}_{a,j} \alpha_a^{d(j-1)} \\ &= \sum_{j \in [d]} \left(\sum_{i \in [d]} \text{p}_{i,j} \alpha_a^{i-1} \right) \alpha_a^{d(j-1)} \\ &= \sum_{i,j \in [d]} \text{p}_{i,j} \alpha_a^{(i-1)+d(j-1)}, \end{aligned}$$

which contradicts the assumption that $\deg(\text{Lagrange}(\{((\alpha_a, \beta_a))_{a \in [L]}\})) > D$.

Finally, A_{DLogRel} outputs a non-trivial discrete logarithm relation $(\text{pf}_{a^*,j} - \sum_{i \in [d]} \text{p}_{i,j} \alpha_{a^*}^{i-1})_{j \in [d]}$ in Step 5d. To show this, we first note that the i -row $(\text{p}_{i,j})_{j \in [d]}$ is a pre-image of the commitment cm_i , i.e., $\text{cm}_i = \sum_{j \in [d]} \text{p}_{i,j} G_j$ for all $i \in [d]$. More precisely, since $\text{PC}_{\text{sqr}}.\text{Check}(\text{pp}_{\text{PC}}, \text{cm}, \alpha_a, \beta_a, \text{pf}_a) = 1$ (and thus

$\sum_{i \in [d]} \alpha_a^{i-1} \text{cm}_i = \sum_{j \in [d]} \text{pf}_{a,j} G_j$) for all $a \in [L]$, we have that for all $l \in [d]$:

$$\begin{aligned} \sum_{i \in [d]} \alpha_{a_l}^{i-1} \text{cm}_i &= \sum_{j \in [d]} \text{pf}_{a_l,j} G_j \\ &= \sum_{j \in [d]} \left(\sum_{i \in [d]} \text{p}_{i,j} \alpha_{a_l}^{i-1} \right) G_j \\ &= \sum_{i \in [d]} \alpha_{a_l}^{i-1} \sum_{j \in [d]} \text{p}_{i,j} G_j, \end{aligned}$$

and thus

$$\begin{bmatrix} 1 & \dots & \alpha_{a_1}^{d-1} \\ \vdots & \ddots & \vdots \\ 1 & \dots & \alpha_{a_d}^{d-1} \end{bmatrix} \begin{bmatrix} \text{cm}_1 - \sum_{j \in [d]} \text{p}_{1,j} G_j \\ \vdots \\ \text{cm}_d - \sum_{j \in [d]} \text{p}_{d,j} G_j \end{bmatrix} = \mathbf{0}.$$

Since the Vandermonde matrix is invertible, it must be that $\text{cm}_i = \sum_{j \in [d]} \text{p}_{i,j} G_j$ for all $i \in [d]$. This immediately implies that A_{DLogRel} 's output in Step 5d is a non-trivial discrete logarithm relation, since

$$\begin{aligned} \sum_{j \in [d]} (\text{pf}_{a^*,j} - \sum_{i \in [d]} \text{p}_{i,j} \alpha_{a^*}^{i-1}) G_j &= \sum_{j \in [d]} \text{pf}_{a^*,j} G_j - \sum_{i,j \in [d]} \text{p}_{i,j} \alpha_{a^*}^{i-1} G_j \\ &= \sum_{i \in [d]} \alpha_{a^*}^{i-1} \text{cm}_i - \sum_{i,j \in [d]} \text{p}_{i,j} \alpha_{a^*}^{i-1} G_j \\ &= 0. \end{aligned}$$

□

C.2 Bulletproofs-style polynomial commitment scheme

We show that a Bulletproofs-style [BCCGP16] interactive polynomial commitment scheme PC_{BP} with logarithmically many rounds satisfies (state-restoration) function binding under the discrete logarithm assumption.

Lemma C.6 (State-restoration function binding of PC_{BP}). *Assume the expected-time discrete logarithm relation assumption holds with error $\epsilon_{\text{DLogRel}}^* = \epsilon_{\text{DLogRel}}^*(\lambda, D, t_{\text{DLogRel}}^*)$ (Definition C.8), PC_{BP} (Construction C.7) has state-restoration function binding error*

$$\epsilon_{\text{FC}}^{\text{SR}}(\lambda, D, L, s_{\text{FC}}, m_{\text{FC}}, t_{\text{FC}}) \leq L \cdot (m_{\text{FC}} + 1) \cdot \frac{1 + 2 \log(D + 1)}{2^\lambda} + \epsilon_{\text{DLogRel}}^*(\lambda, D + 2, t_{\text{DLogRel}}^*),$$

where $t_{\text{DLogRel}}^* = O(L(m_{\text{FC}} + 1)(2(D + 1)^{\log 3})(t_{\text{FC}} + \text{poly}(D)) + L^2 + L(D + 1)^{\log 3})$.

In Appendix C.2.1, we present an interactive polynomial commitment scheme PC_{BP} inspired by the Bulletproofs protocol [BCCGP16] (we closely follow the description of Bünz et al. [BCMS20, Appendix A]). We show that PC_{BP} satisfies special function binding under the expected-time discrete logarithm relation assumption in Appendix C.2.2. In Appendix C.2.3, we compare the proof technique and the concrete security bound of state-restoration function binding with soundness and knowledge soundness.

C.2.1 Construction

Throughout this section, we assume that every polynomial has degree at most D where $D + 1$ is a power of 2. We write $\vec{p} = (p_0, \dots, p_D)$ for the coefficients of a polynomial $p \in \mathbb{F}[X]^{\leq D}$. We denote by $L((v_1, \dots, v_{2k})) := (v_1, \dots, v_k)$ the operation which returns the left half of a vector, and similarly $R((v_1, \dots, v_{2k}))$ for the right half.

Construction C.7. Let $\mathbb{G}$ be a group of prime order $p(\lambda) \geq 2^\lambda$.

- $\text{PC}_{\text{BP}}.\text{Gen}(1^\lambda, D)$:
 1. Sample random generators $\vec{G} \leftarrow \mathbb{G}^{D+1}$, $H \leftarrow \mathbb{G}$.
 2. Output $\text{pp}_{\text{FC}} = (\vec{G}, H)$.
- $\text{PC}_{\text{BP}}.\text{Commit}(\text{pp}_{\text{FC}}, p)$: Outputs $\text{cm} := \langle \vec{p}, \vec{G} \rangle$.
- $\text{PC}_{\text{BP}}.\text{Open}, \text{PC}_{\text{BP}}.\text{FC}.\text{Check}$: The interactive protocol between $\mathcal{P}_{\text{FC}}$ and $\mathcal{V}_{\text{FC}}$ with common reference string pp_{FC} , instance $(\text{cm}, \alpha, \beta)$ and witness p proceeds as follows:
 1. $\mathcal{V}_{\text{FC}}$ samples $\xi_0 \leftarrow \mathbb{F} \setminus \{0\}$ uniformly at random, and sends it to $\mathcal{P}_{\text{FC}}$.
 2. Both $\mathcal{P}_{\text{FC}}$ and $\mathcal{V}_{\text{FC}}$ set
 - $H' := \xi_0 H$,
 - $\vec{\alpha}^{(0)} := (1, \alpha, \dots, \alpha^D)$, and
 - $\vec{G}^{(0)} := \vec{G}$.
 3. $\mathcal{P}_{\text{FC}}$ sets $\vec{p}^{(0)} := \vec{p}$.
 4. $\mathcal{V}_{\text{FC}}$ sets $C^{(0)} := \text{cm} + \beta \cdot H'$.
 5. For $i \in \{1, \dots, \log(D+1)\}$:
 - (a) $\mathcal{P}_{\text{FC}}$ sends $L^{(i)}$ and $R^{(i)}$ to the verifier, where

$$\begin{aligned} L^{(i)} &:= \langle R(\vec{p}^{(i-1)}), L(\vec{G}^{(i-1)}) \rangle + \langle R(\vec{p}^{(i-1)}), L(\vec{\alpha}^{(i-1)}) \rangle H' \\ R^{(i)} &:= \langle L(\vec{p}^{(i-1)}), R(\vec{G}^{(i-1)}) \rangle + \langle L(\vec{p}^{(i-1)}), R(\vec{\alpha}^{(i-1)}) \rangle H' \end{aligned}$$

- (b) $\mathcal{V}_{\text{FC}}$ samples $\xi_i \leftarrow \mathbb{F} \setminus \{0\}$, computes $C^{(i)} := \xi_i^{-1} L^{(i)} + C^{(i-1)} + \xi_i R^{(i)}$, and sends ξ_i to $\mathcal{P}_{\text{FC}}$.
- (c) $\mathcal{P}_{\text{FC}}$ computes the inputs for the next round as follows:
 - $\vec{p}^{(i)} := L(\vec{p}^{(i-1)}) + \xi_i^{-1} R(\vec{p}^{(i-1)})$,
 - $\vec{\alpha}^{(i)} := L(\vec{\alpha}^{(i-1)}) + \xi_i R(\vec{\alpha}^{(i-1)})$, and
 - $\vec{G}^{(i)} := L(\vec{G}^{(i-1)}) + \xi_i R(\vec{G}^{(i-1)})$.
6. $\mathcal{P}_{\text{FC}}$ sends the constant polynomial $u := \vec{p}^{(\log(D+1))} \in \mathbb{F}$ to $\mathcal{V}_{\text{FC}}$.
7. $\mathcal{V}_{\text{FC}}$ defines $h(X) := \prod_{i=0}^{\log(D+1)-1} (1 + \xi_{\log(D+1)-i} X^{2^i})$ (with coefficient vector $\vec{h}$) and checks that $C^{(\log(D+1))} = u \langle \vec{h}, \vec{G} \rangle + u \cdot h(\alpha) H'$.

C.2.2 Function binding

We show Lemma C.6 by showing that PC_{BP} is special function binding under the expected-time discrete logarithm relation assumption, and applying Lemma A.2.

Finally, we show that PC_{BP} satisfies special function binding under the expected-time discrete logarithm relation assumption.

Definition C.8 (Expected-time discrete logarithm relation assumption). *The expected-time discrete logarithm relation assumption holds with error $\epsilon_{\text{DLogRel}}^*$ if for every security parameter λ , length parameter $n \in \mathbb{N}$, expected t_{DLogRel}^* -time adversary A_{DLogRel} and group $\mathbb{G}$ of prime order $p(\lambda) \geq 2^\lambda$,*

$$\Pr \left[\exists a_i \neq 0 \wedge \sum_{i \in [n]} a_i G_i = 0 \mid \begin{array}{l} G_1, \dots, G_n \leftarrow \mathbb{G} \\ a_1, \dots, a_n \in \mathbb{Z}_p \leftarrow A_{\text{DLogRel}}(\mathbb{G}, G_1, \dots, G_n) \end{array} \right] \leq \epsilon_{\text{DLogRel}}^*(\lambda, n, t_{\text{DLogRel}}^*) .$$

Definition C.9 (Expected-time discrete logarithm assumption). *The expected-time discrete logarithm assumption holds with error ϵ_{DL}^* if for every security parameter λ , expected t_{DL}^* -time adversary A_{DL} , and group $\mathbb{G}$ of prime order $p(\lambda) \geq 2^\lambda$ with generator G ,*

$$\Pr \left[xG = H \mid \begin{array}{l} H \leftarrow \mathbb{G} \\ x \leftarrow A_{\text{DL}}(\mathbb{G}, p, G, H) \end{array} \right] \leq \epsilon_{\text{DL}}^*(\lambda, t_{\text{DL}}^*) .$$

Asymptotically, the expected-time discrete logarithm and the expected-time discrete logarithm relation assumption are equivalent; the following expected-time equivalent of Lemma C.5 bounds the concrete relationship between the two assumptions.

Lemma C.10 (Expected-time discrete logarithm relation $\Rightarrow$ expected-time discrete logarithm). *Let $\mathbb{G}$ be a group of prime order $p(\lambda) \geq 2^\lambda$ and $n \geq 1$ be an integer. Then for any $\lambda \in \mathbb{N}$, $t_{\text{DLogRel}}^* \in \mathbb{N}$, we have*

$$\epsilon_{\text{DLogRel}}^*(\lambda, n, t_{\text{DLogRel}}^*) \leq \epsilon_{\text{DL}}^*(\lambda, t_{\text{DL}}^*) + \frac{1}{2^\lambda} ,$$

where $t_{\text{DL}}^* = O(t_{\text{DLogRel}}^* + n)$.

In order to upper-bound on the expected discrete logarithm error, one can either use reduce to the strict-time discrete logarithm assumption, or use an upper-bound derived using an idealized model [JT20; SSY23].

Lemma C.11 (Special function binding of PC_{BP}). *PC_{BP} has $(2, 3, \dots, 3)$ -special function binding error*

$$\epsilon_{\text{SFB}}(\lambda, D + 1, L, t_{\text{SFB}}^*) \leq \epsilon_{\text{DLogRel}}^*(\lambda, D + 2, t_{\text{DLogRel}}^*) ,$$

where $t_{\text{DLogRel}}^* = O(t_{\text{SFB}}^* + L^2 + L \cdot (D + 1)^{\log 3})$.

Proof. We define the following adversary A_{SFB} , which, when run on the output of a successful special function binding reductor $\mathfrak{R}$, outputs a non-trivial discrete logarithm relation for the generators $G_0, \dots, G_d, H$. In the following, let $d := \log(D + 1)$ denote the number of rounds in the opening protocol.

$A_{\text{SFB}}(\text{pp}_{\text{FC}}, (\text{cm}, ((\alpha_i, \beta_i, T_i))_{i \in [L]}))$:

1. For $i \in [L]$:

- (a) Parse T_i as prover messages $((L_{i,r}^{(j_0, \dots, j_{r-1})}, R_{i,r}^{(j_0, \dots, j_{r-1})}))_{r \in [d], j_0 \in [2], j_1, \dots, j_{r-1} \in [3]}, (u_i^{(j_0, \dots, j_d)})_{j_0 \in [2], j_1, \dots, j_d \in [3]}$ and verifier challenges $(\xi_{i,r}^{(j_0, \dots, j_r)})_{r \in [d], j_0 \in [2], j_1, \dots, j_r \in [3]}$.
- (b) For $(j_0, \dots, j_d) \in \{1, 2\} \times \{1, 2, 3\}^d$:
 - i. Set $J_i^{(j_0, \dots, j_d)} := \sum_{s \in [d]} (\xi_{i,s}^{(j_0, \dots, j_s)})^{-1} L_{i,s}^{(j_0, \dots, j_{s-1})}$

$$+ \text{cm} + \beta_i \xi_{i,0}^{(j_0)} H$$

$$+ \sum_{s \in [d]} \xi_{i,s}^{(j_0, \dots, j_s)} R_{i,s}^{(j_0, \dots, j_{s-1})} .$$

- ii. Set $h_i^{(j_0, \dots, j_d)}(X) := \prod_{s=0}^{\log(D+1)-1} (1 + \xi_{i, \log(D+1)-s}^{(j_0, \dots, j_{\log(D+1)-s})} X^{2^s})$
- iii. Set $p_i^{(j_0, \dots, j_d)}(X) := u_i^{(j_0, \dots, j_d)} \cdot h_i^{(j_0, \dots, j_d)}(X)$, with coefficient vector $\vec{p}_i^{(j_0, \dots, j_d)}$.
- iv. Check that

$$J_i^{(j_0, \dots, j_d)} = \langle \vec{p}_i^{(j_0, \dots, j_d)}, \vec{G} \rangle + p_i^{(j_0, \dots, j_d)}(\alpha_i) \cdot \xi_i^{(j_0)} H ;$$

abort otherwise.

- (c) For $r = d$ to 1: For $(j_0, \dots, j_{r-1}) \in \{1, 2\} \times \{1, 2, 3\}^{r-1}$:
 - i. Check that $\xi_{i,r}^{(j_0, \dots, j_{r-1}, 1)}, \xi_{i,r}^{(j_0, \dots, j_{r-1}, 2)}, \xi_{i,r}^{(j_0, \dots, j_{r-1}, 3)}$ are distinct; abort otherwise.
 - ii. Compute $(\nu_j)_{j \in [3]}$ such that

$$\begin{bmatrix} (\xi_{i,r}^{(j_0, \dots, j_{r-1}, 1)})^{-1} & (\xi_{i,r}^{(j_0, \dots, j_{r-1}, 2)})^{-1} & (\xi_{i,r}^{(j_0, \dots, j_{r-1}, 3)})^{-1} \\ 1 & 1 & 1 \\ \xi_{i,r}^{(j_0, \dots, j_{r-1}, 1)} & \xi_{i,r}^{(j_0, \dots, j_{r-1}, 2)} & \xi_{i,r}^{(j_0, \dots, j_{r-1}, 3)} \end{bmatrix} \begin{bmatrix} \nu_1 \\ \nu_2 \\ \nu_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} .$$

- iii. Set $J_i^{(j_0, \dots, j_{r-1})} := \sum_{j \in [3]} \nu_j J_i^{(j_0, \dots, j_{r-1}, j)}$

$$= \sum_{s \in [r-1]} (\xi_{i,s}^{(j_0, \dots, j_s)})^{-1} L_{i,s}^{(j_0, \dots, j_{s-1})}$$

$$+ \text{cm} + \beta_i \xi_{i,0}^{(j_0)} H$$

$$+ \sum_{s \in [r-1]} \xi_{i,s}^{(j_0, \dots, j_s)} R_{i,s}^{(j_0, \dots, j_{s-1})} .$$
- iv. Set $p_i^{(j_0, \dots, j_{r-1})} := \sum_{j \in [3]} \nu_j p_i^{(j_0, \dots, j_{r-1}, j)} .$

(At this point, each $p_i^{(j_0)}$ satisfies $\text{cm} = \langle \vec{p}_i^{(j_0)}, \vec{G} \rangle + (p_i^{(j_0)}(\alpha_i) - \beta_i) \xi_{i,0}^{(j_0)} H$.)

- 2. If $p_i^{(j_0)} \neq p_j^{(j'_0)}$ for some $i, j \in [L], j_0, j'_0 \in [2]$ (i.e., $\langle \vec{p}_i^{(j_0)}, \vec{G} \rangle + (p_i^{(j_0)}(\alpha_i) - \beta_i) \xi_{i,0}^{(j_0)} H = \langle \vec{p}_j^{(j'_0)}, \vec{G} \rangle + (p_j^{(j'_0)}(\alpha_{j,m}) - \beta_j) \xi_{j,0}^{(j'_0)} H$):
 - (a) Set $p := p_i^{(j_0)} - p_j^{(j'_0)}$.
 - (b) Output $\vec{p}, (p_i^{(j_0)}(\alpha_i) - \beta_i) \xi_{i,0}^{(j_0)} - (p_j^{(j'_0)}(\alpha_j) - \beta_j) \xi_{j,0}^{(j'_0)}$.
- 3. Otherwise:
 - (a) Set $p := p_1^{(1)} = \dots = p_L^{(2)}$.
 - (b) If $p(\alpha_i) \neq \beta_i$ for some $i \in [L]$ (i.e., $(p(\alpha_i) - \beta_i) \xi_{i,0}^{(1)} H = (p(\alpha_i) - \beta_i) \xi_{i,0}^{(2)} H$):
 - Output $(0)_{i \in [D+1]}, (p(\alpha_i) - \beta_i) (\xi_{i,0}^{(1)} - \xi_{i,0}^{(2)})$.
 - (c) Otherwise: abort.

Running time. Each tree of transcripts has $2 \cdot 3^{\log(D+1)} = 2 \cdot (D+1)^{\log 3}$ nodes. In Step 1, A_{SFB} performs at most $L(2 \cdot \log(D+1)^{\log 3} + \sum_{r \in [\log(D+1)]} 2 \cdot 3^r)$ operations. In Step 2, A_{SFB} searches for disagreeing answers for the same query among $2L$ query-answer pairs, which requires $O(L^2)$ operations. Finally, Step 3 only requires a linear scan over L values.

Success probability. Let $((\alpha_i, \beta_i, T_i))_{i \in [L]}$ be the output of a successful $\mathfrak{R}$ adversary. Since each tree is a valid tree, A_{SFB} does not abort in Step 1(b)iv. Furthermore, since the verifier challenges for each round are non-zero, and guaranteed by SRTreeFinder to be distinct, the matrix in Step 1(c)ii is well-defined and invertible.

If the condition in Step 2 is satisfied, at least one coefficient of $p_i^{(j_0)} - p_{j,m}^{(j'_0)}$ is non-zero, and A_{SFB} outputs a valid discrete logarithm relation. Finally, note that the function binding condition

$$(\exists i, j \in [L] : \alpha_i = \alpha_j \wedge \beta_i \neq \beta_j) \vee \deg(\text{Lagrange}(\{(\alpha_i, \beta_i)\})) > D$$

is equivalent to the condition $\exists i \in [L] : p(\alpha_i) \neq \beta_i$ tested in Step 3b, which means that R does not abort in Step 3. In this case, $(p(\alpha_i) - \beta_i)(\xi_{i,0}^{(1)} - \xi_{i,0}^{(2)})$ is non-zero, and A_{SFB} also outputs a valid discrete logarithm relation. $\square$

C.2.3 Comparison with soundness and knowledge soundness

Bulletproofs and Bulletproofs-style protocols are known to satisfy knowledge soundness, which requires the existence of an efficient extractor that, given an instance and rewinding access to a (potentially malicious) prover, is able to efficiently extract a valid witness. The Bulletproofs protocol (and variants thereof), are shown to be (asymptotically) knowledge sound under the expected-time discrete logarithm relation assumption [BCCGP16; BBBPWM18; Tha22]. [JT20] gives a more formal and concrete treatment, and provide knowledge soundness errors for both strict-time and expected-time extractors, also from the discrete logarithm relation assumption. Interestingly, their strict-time knowledge soundness error is significantly worse than the expected-time error; in general, for protocols with superconstant rounds, an expected-time tree finder seems inherent for an efficient reductor or extractor [ACK21; JT20]. In the context of polynomial commitment schemes, knowledge soundness requires the existence of an extractor which, given a commitment cm and an opening proof for a query-answer pair (α, β) , can extract a polynomial p such that (i) $\deg(p) \leq D$, (ii) cm is a commitment to p , and (iii) $p(\alpha) = \beta$ [BMMTV21]. For Bulletproofs-style protocols, knowledge soundness is usually shown via special soundness [BCCGP16; BBBPWM18; JT20] (i.e., there exists an efficient, extractor which outputs a valid witness polynomial when given as input a commitment, a query-answer pair, and a tree of accepting transcripts). The final extractor is a concatenation of a tree finder algorithm (which finds a tree of accepting transcripts with the required arity) and this special soundness extractor.

Note that for $L = 1$, the proof of Lemma C.11 can easily be adapted to show knowledge soundness (rather than aborting in Step 3c, A_{SFB} outputs the valid witness p instead). In particular, both the special function binding and the special soundness adversary requires a tree of the same arity $(2, 3, \dots, 3)$. On the other hand, one might consider soundness, i.e., the property that a malicious prover cannot convince a verifier for an instance that is not in the language (in our case, every polynomial of degree at most D is either not committed to in cm , or does not evaluate to β at α). In particular, a natural minimal property for PC_{BP} is state-restoration soundness. For PC_{BP} , we are not aware of any proof that specifically targets state-restoration soundness, or even soundness; however, note that for $L = 1$, the proof of Lemma C.11 is a proof of state-restoration soundness. One might ask whether (state-restoration) soundness can be shown from either a simpler proof (without using special-soundness-type techniques), or from a tree with smaller arities. The former seems difficult, owing to the superconstant number of rounds in the protocol. The latter also seems challenging (at least when relying solely on the discrete logarithm assumption): the requirement for three distinct challenges in each round is dictated by the structure of the “folding” step, and by the fact that the prover messages at each step of the protocol define a quadratic polynomial evaluated at the verifier’s challenge.

Acknowledgments

Ziyi Guan thanks Ngoc Khanh Nguyen, Kshiteej Sheth, and Weiqiang Yuan for helpful discussions in early stages of this work. The authors are partially supported by the Ethereum Foundation and Polygon Labs.

References

- [ACK21] Thomas Attema, Ronald Cramer, and Lisa Kohl. “A Compressed Σ -Protocol Theory for Lattices”. In: *Proceedings of the 41st Annual International Cryptology Conference*. CRYPTO ’21. 2021, pp. 549–579.
- [AGLMS23] Arasu Arun, Chaya Ganesh, Satya V. Lokam, Tushar Mopuri, and Sriram Sridhar. “Dew: Transparent Constant-sized zkSNARKs”. In: *Proceedings of the 26th International Conference on Practice and Theory in Public Key Cryptography*. PKC ’23. 2023, pp. 542–571.
- [AJMMS23] Ittai Abraham, Philipp Jovanovic, Mary Maller, Sarah Meiklejohn, and Gilad Stern. “Bingo: Adaptivity and Asynchrony in Verifiable Secret Sharing and Distributed Key Generation”. In: *Proceedings of the 43rd Annual International Cryptology Conference*. CRYPTO ’23. 2023, pp. 39–70.
- [AY25] Gal Arnon and Eylon Yogev. “Towards a White-Box Secure Fiat-Shamir Transformation”. In: *Proceedings of the 45th Annual International Cryptology Conference*. CRYPTO ’25. 2025, pp. 27–56.
- [BBBPWM18] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. “Bulletproofs: Short Proofs for Confidential Transactions and More”. In: *Proceedings of the 39th IEEE Symposium on Security and Privacy*. S&P ’18. 2018, pp. 315–334.
- [BBHMR19] James Bartusek, Liron Bronfman, Justin Holmgren, Fermi Ma, and Ron D. Rothblum. “On the (In)security of Kilian-Based SNARGs”. In: *Proceedings of the 17th Theory of Cryptography Conference*. TCC ’19. 2019, pp. 522–551.
- [BCCGP16] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Jens Groth, and Christophe Petit. “Efficient Zero-Knowledge Arguments for Arithmetic Circuits in the Discrete Log Setting”. In: *Proceedings of the 35th Annual International Conference on Theory and Application of Cryptographic Techniques*. EUROCRYPT ’16. 2016, pp. 327–357.
- [BCG20] Jonathan Bootle, Alessandro Chiesa, and Jens Groth. “Linear-Time Arguments with Sublinear Verification from Tensor Codes”. In: *Proceedings of the 18th Theory of Cryptography Conference*. TCC ’20. 2020, pp. 19–46.
- [BCGGHJ17] Jonathan Bootle, Andrea Cerulli, Essam Ghadafi, Jens Groth, Mohammad Hajiabadi, and Sune K. Jakobsen. “Linear-Time Zero-Knowledge Proofs for Arithmetic Circuit Satisfiability”. In: *Proceedings of the 23rd International Conference on the Theory and Applications of Cryptology and Information Security*. ASIACRYPT ’17. 2017, pp. 336–365.
- [BCMS20] Benedikt Bünz, Alessandro Chiesa, Pratyush Mishra, and Nicholas Spooner. “Recursive Proof Composition from Accumulation Schemes”. In: *Proceedings of the 18th Theory of Cryptography Conference*. TCC ’20. 2020, pp. 1–18.
- [BCS16] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. “Interactive Oracle Proofs”. In: *Proceedings of the 14th Theory of Cryptography Conference*. TCC ’16-B. 2016, pp. 31–60.
- [BFS20] Benedikt Bünz, Ben Fisch, and Alan Szepieniec. “Transparent SNARKs from DARK Compilers”. In: *Proceedings of the 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques*. EUROCRYPT ’20. 2020, pp. 677–706.
- [BG08] Boaz Barak and Oded Goldreich. “Universal Arguments and their Applications”. In: *SIAM Journal on Computing* 38.5 (2008). Preliminary version appeared in CCC ’02., pp. 1661–1694.

- [BG18] Jonathan Bootle and Jens Groth. “Efficient Batch Zero-Knowledge Arguments for Low Degree Polynomials”. In: *Proceedings of the 21st IACR International Conference on Practice and Theory of Public-Key Cryptography*. PKC ’18. 2018, pp. 561–588.
- [BMMTV21] Benedikt Bünz, Mary Maller, Pratyush Mishra, Nirvan Tyagi, and Psi Vesely. “Proofs for Inner Pairing Products and Applications”. In: *Proceedings of the 27th International Conference on the Theory and Application of Cryptology and Information Security*. ASIACRYPT ’21. 2021, pp. 65–97.
- [CCHLRR18] Ran Canetti, Yilei Chen, Justin Holmgren, Alex Lombardi, Guy N. Rothblum, and Ron D. Rothblum. *Fiat–Shamir From Simpler Assumptions*. Cryptology ePrint Archive, Report 2018/1004. 2018.
- [CDDGS25] Alessandro Chiesa, Marcel Dall’Agnol, Zijing Di, Ziyi Guan, and Nicholas Spooner. “Quantum Rewinding for IOP-Based Succinct Arguments”. In: *Proceedings of the 23rd Theory of Cryptography Conference*. TCC ’25. 2025.
- [CDGS23] Alessandro Chiesa, Marcel Dall’Agnol, Ziyi Guan, and Nicholas Spooner. *On the Security of Succinct Interactive Arguments from Vector Commitments*. Cryptology ePrint Archive, Report 2023/1737. 2023.
- [CDGSY24] Alessandro Chiesa, Marcel Dall’Agnol, Ziyi Guan, Nicholas Spooner, and Eylon Yogev. “Untangling the Security of Kilian’s Protocol: Upper and Lower Bounds”. In: *Proceedings of the 22nd Theory of Cryptography Conference*. TCC ’24. 2024.
- [CFFQR21] Matteo Campanelli, Antonio Faonio, Dario Fiore, Anaïs Querol, and Hadrián Rodríguez. “Lunar: A Toolbox for More Efficient Universal and Updatable zkSNARKs and Commit-and-Prove Extensions”. In: *Proceedings of the 27th International Conference on the Theory and Application of Cryptology and Information Security*. ASIACRYPT ’21. 2021, pp. 3–33.
- [CGH04] Ran Canetti, Oded Goldreich, and Shai Halevi. “The random oracle methodology, revisited”. In: *Journal of the ACM* 51.4 (2004), pp. 557–594.
- [CHMMVW20] Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Noah Vesely, and Nicholas Ward. “Marlin: Preprocessing zkSNARKs with Universal and Updatable SRS”. In: *Proceedings of the 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques*. EUROCRYPT ’20. 2020, pp. 738–768.
- [CO25] Alessandro Chiesa and Michele Orrù. “A Fiat–Shamir Transformation from Duplex Sponges”. In: *Proceedings of the 23rd Theory of Cryptography Conference*. TCC ’25. 2025, pp. 452–474.
- [CY20] Alessandro Chiesa and Eylon Yogev. “Barriers for Succinct Arguments in the Random Oracle Model”. In: *Proceedings of the 17th Theory of Cryptography Conference*. TCC ’20. 2020, pp. 47–76.
- [CY21a] Alessandro Chiesa and Eylon Yogev. “Subquadratic SNARGs in the Random Oracle Model”. In: *Proceedings of the 41st Annual International Cryptology Conference*. CRYPTO ’21. 2021, pp. 711–741.
- [CY21b] Alessandro Chiesa and Eylon Yogev. “Tight Security Bounds for Micali’s SNARGs”. In: *Proceedings of the 19th Theory of Cryptography Conference*. TCC ’21. 2021, pp. 401–434.
- [CY24] Alessandro Chiesa and Eylon Yogev. *Building Cryptographic Proofs from Hash Functions*. 2024. URL: <https://github.com/hash-based-snargs-book>.
- [FFR24] Antonio Faonio, Dario Fiore, and Luigi Russo. “Real-World Universal zkSNARKs are Non-Malleable”. In: *Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security*. 2024, pp. 3138–3151.
- [FS86] Amos Fiat and Adi Shamir. “How to prove yourself: practical solutions to identification and signature problems”. In: *Proceedings of the 6th Annual International Cryptology Conference*. CRYPTO ’86. 1986, pp. 186–194.

- [GK03] Shafi Goldwasser and Yael Tauman Kalai. “On the (In)security of the Fiat-Shamir Paradigm”. In: *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*. FOCS ’03. 2003, pp. 102–113.
- [GW11] Craig Gentry and Daniel Wichs. “Separating Succinct Non-Interactive Arguments From All Falsifiable Assumptions”. In: *Proceedings of the 43rd Annual ACM Symposium on Theory of Computing*. STOC ’11. 2011, pp. 99–108.
- [GWC19] Ariel Gabizon, Zachary J. Williamson, and Oana Ciobotaru. *PLONK: Permutations over Lagrange-bases for Oecumenical Noninteractive arguments of Knowledge*. Cryptology ePrint Archive, Report 2019/953. 2019.
- [JT20] Joseph Jaeger and Stefano Tessaro. “Expected-Time Cryptography: Generic Techniques and Applications to Concrete Soundness”. In: *Proceedings of the 18th Theory of Cryptography Conference*. TCC ’20. 2020, pp. 414–443.
- [KRS25] Dmitry Khovratovich, Ron D. Rothblum, and Lev Soukhanov. “How to Prove False Statements: Practical Attacks on Fiat-Shamir”. In: *Proceedings of the 45th Annual International Cryptology Conference*. CRYPTO ’25. 2025, pp. 3–26.
- [KZG10] Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. “Constant-Size Commitments to Polynomials and Their Applications”. In: *Proceedings of the 16th International Conference on the Theory and Application of Cryptology and Information Security*. ASIACRYPT ’10. 2010, pp. 177–194.
- [Kil92] Joe Kilian. “A note on efficient zero-knowledge proofs and arguments”. In: *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*. STOC ’92. 1992, pp. 723–732.
- [LM19] Russell W. F. Lai and Giulio Malavolta. “Subvector Commitments with Application to Succinct Arguments”. In: *Proceedings of the 39th Annual International Cryptology Conference*. CRYPTO ’19. 2019, pp. 530–560.
- [LPS24] Helger Lipmaa, Roberto Parisella, and Janno Siim. “Constant-Size zk-SNARKs in ROM from Falsifiable Assumptions”. In: *Proceedings of the 43rd Annual International Conference on the Theory and Applications of Cryptographic Techniques*. EUROCRYPT ’24. 2024, pp. 34–64.
- [LPS25] Helger Lipmaa, Roberto Parisella, and Janno Siim. “On Knowledge-Soundness of Plonk in ROM from Falsifiable Assumptions”. In: *Proceedings of the 45th Annual International Cryptology Conference*. CRYPTO ’25. 2025, pp. 362–395.
- [Lee21] Jonathan Lee. “Dory: Efficient, Transparent Arguments for Generalised Inner Products and Polynomial Commitments”. In: *Proceedings of the 19th Theory of Cryptography Conference*. TCC ’21. 2021, pp. 1–34.
- [Mic00] Silvio Micali. “Computationally Sound Proofs”. In: *SIAM Journal on Computing* 30.4 (2000). Preliminary version appeared in FOCS ’94., pp. 1253–1298.
- [RV09] Guy N. Rothblum and Salil Vadhan. “Are PCPs Inherent in Efficient Arguments?” In: *Proceedings of the 24th IEEE Annual Conference on Computational Complexity*. CCC ’09. 2009, pp. 81–92.
- [SSY23] Gil Segev, Amit Sharabi, and Eylon Yogev. “Rogue-Instance Security for Batch Knowledge Proofs”. In: *Proceedings of the 21st Theory of Cryptography Conference*. TCC ’23. 2023, pp. 121–157.
- [Tha22] Justin Thaler. “Proofs, Arguments, and Zero-Knowledge”. In: *Found. Trends Priv. Secur.* 4.2-4 (2022), pp. 117–660. DOI: 10.1561/33000000030. URL: <https://people.cs.georgetown.edu/jthaler/ProofsArgsAndZK>.
- [Val08] Paul Valiant. “Incrementally Verifiable Computation or Proofs of Knowledge Imply Time/Space Efficiency”. In: *Proceedings of the 5th Theory of Cryptography Conference*. TCC ’08. 2008, pp. 1–18.