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
zihan.yu@epfl.ch
EPFL

May 27, 2025

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 is 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 security proof of Plonk, in the ROM, based on ARSDH (a falsifiable assumption).

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

Contents

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 $\mathbb{x}$, an argument prover aims to convince the argument verifier that there exists a witness $\mathbb{w}$ such that $(\mathbb{x}, \mathbb{w}) \in R$, while sending an argument string much shorter than $|\mathbb{w}|$ 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 the study of SNARGs in the ROM that *additionally rely on cryptographic assumptions*, in order 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; LPS24a].

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.

Special cases are studied in prior work: with vector commitment schemes [Kil92; BG08; CDGS23; CDGSY24]; with linear commitment schemes [LM19]; and with polynomial commitment schemes [CHMMVW20; BFS20; CFFQR21; LPS24a]. These works do not show security of the SNARG in the ROM obtained via the Fiat–Shamir transformation, *leaving a security gap*.[1] Indeed, the Fiat–Shamir transformation requires the interactive argument (in particular, the Funky protocol) to satisfy a soundness property stronger than standard soundness (and similarly for knowledge soundness).[2] Specifically, the Funky protocol must satisfy *state-restoration soundness* [BCS16; CY24] (essentially equivalent to Fiat–Shamir security). We ask:

When does the Funky protocol satisfy state-restoration soundness?

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; LPS24a], 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

[1] [LPS24b], an exception, 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.

[2] For every $\mathsf{k}_{\mathsf{FIOP}}$, there are $\mathsf{k}_{\mathsf{FIOP}}$-round interactive arguments with standard (not state-restoration) soundness ϵ_{ARG} for which the soundness error after the Fiat–Shamir transformation in the ROM is $\mathsf{m}^{\Omega(\mathsf{k})} \cdot \epsilon_{\mathsf{ARG}}$ against m-query adversaries. This does not suffice for asymptotic security when $\mathsf{k}_{\mathsf{FIOP}}$ is superconstant and, even when $\mathsf{k}_{\mathsf{FIOP}}$ is constant, yields poor concrete security.

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 arguments in the standard model by combining FC schemes (functional commitment schemes) and FIOPs (functional IOPs). 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 Plonk (with optimizations) [GWC19] is secure in the ROM based on the ARSDH assumption.

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 $\mathsf{Funky}[\mathsf{FIOP}, \mathsf{FC}]$ the Funky protocol based on: (i) FIOP, a functional IOP with round complexity $\mathsf{k}_{\mathsf{FIOP}}$, proof length ℓ, and query complexity q; (ii) FC, an interactive FC scheme. We upper bound the state-restoration soundness error $\epsilon^{\mathsf{SR}}_{\mathsf{ARG}}$ of $\mathsf{Funky}[\mathsf{FIOP}, \mathsf{FC}]$ in terms of the state-restoration soundness error $\epsilon^{\mathsf{SR}}_{\mathsf{FIOP}}$ of FIOP and the state-restoration function binding error $\epsilon^{\mathsf{SR}}_{\mathsf{FC}}$ of FC.[3] 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* $\epsilon^{\mathsf{SR}}_{\mathsf{ARG}}$ *of* $\mathsf{Funky}[\mathsf{FIOP}, \mathsf{FC}]$ *against* m*-move* t_{ARG}*-size adversaries satisfies the following for every error tolerance* $\epsilon > 0$*:*

$$\epsilon^{\mathsf{SR}}_{\mathsf{ARG}}(\mathsf{m}, t_{\mathsf{ARG}}) \leq \epsilon^{\mathsf{SR}}_{\mathsf{FIOP}}(\mathsf{m} + \mathsf{k}_{\mathsf{FIOP}}) + \mathsf{k}_{\mathsf{FIOP}} \cdot \epsilon^{\mathsf{SR}}_{\mathsf{FC}}(\mathsf{m}_{\mathsf{FC}}, t_{\mathsf{FC}}) + \mathsf{k}_{\mathsf{FIOP}} \cdot \epsilon \quad \textit{where} \quad \begin{cases} \mathsf{m}_{\mathsf{FC}} = O(\frac{\mathsf{k}_{\mathsf{FIOP}} \cdot \ell}{\epsilon} \cdot \mathsf{m}) \\ t_{\mathsf{FC}} = O(\frac{\ell}{\epsilon} \cdot t_{\mathsf{ARG}}) \end{cases} .$$

This implies negligible bounds in the negligible case.

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.

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

- *VC setting.* Theorem 1 implies the same bound on standard (not state-restoration) security as in [CDGS23; CDGSY24] for non-interactive VC schemes.
- *LC 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.
- *PC setting.* [CHMMVW20; BFS20] proves standard (not state-restoration) security based on extractability in the negligible regime. [LPS24a] proves standard (not state-restoration) security for a *less efficient variant* of the protocol (it includes random openings that double the round complexity); moreover, [LPS24a] 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, [LPS24b] 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 (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.

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. Random oracles are replaced by efficient hash functions in practice. 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** $\epsilon_{\mathsf{FC}} = \epsilon_{\mathsf{FC}}(t_{\mathsf{FC}})$ *if for every* t_{FC}*-size adversary that outputs a commitment* cm *and a list of openings* $\big((\alpha_i, \beta_i, \mathsf{pf}_i)\big)_i$ *valid for* cm*, with probability at most* ϵ_{FC}*, there does not exists* 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 $\mathbf{Q}_{\mathsf{Struct}}[\mathbf{Q}, m, (\mathsf{h}_k)_{k\in[n]}]$ 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 (a falsifiable assumption introduced by [LPS24a]).

Lemma 1 (informal)**.** *Assume that the adaptive rational strong Diffie–Hellman (ARSDH) assumption holds with error* $\epsilon_{\mathsf{ARSDH}} = \epsilon_{\mathsf{ARSDH}}(\lambda, t_{\mathsf{ARSDH}})$*. The FC scheme* linKZG *(Construction 8.8) has state-restoration function binding error* $\epsilon^{\mathsf{SR}}_{\mathsf{FC}}$ *against every* t_{FC}*-size where*

$$\epsilon^{\mathsf{SR}}_{\mathsf{FC}}(\mathsf{m}_{\mathsf{FC}}, t_{\mathsf{FC}}) = \frac{(\mathsf{m}_{\mathsf{FC}} + 1)}{2^{\lambda}} + \epsilon_{\mathsf{ARSDH}}(\lambda, t_{\mathsf{ARSDH}}) \ ,$$

where $t_{\mathsf{ARSDH}} = O((\mathsf{m}_{\mathsf{FC}} + 1) \cdot t_{\mathsf{FC}})$*.*

Lemma 1 provides new insights over prior work. [LPS24b] 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

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

Corollary 1. *Assuming ARSDH, 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 an additional falsifiable assumption [LPS24b] (via a direct proof that does not distill a security property for linKZG).

2 Techniques

We overview our techniques. In Section 2.1 we describe challenges that arise when adapting ideas from prior work to the general setting of FC schemes and FIOPs. In Section 2.2, as a warmup, we sketch how we address these challenges for the special case of FPCPs. In Section 2.3 we discuss the general case of FIOPs, including showing state-restoration security. In Section 2.4 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 $\alpha\colon \Sigma^\ell \to \mathbb{D}$ 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 2.4, which arise in Plonk [GWC19].

$$\mathbf{Q}_{\mathsf{Point}} \coloneqq \left\{ \alpha\colon \mathbb{F}^\ell \to \mathbb{F} \;\middle|\; \exists\, i \in [\ell] \text{ s.t. } \alpha(\Pi) = \Pi[i] \right\} \quad , \tag{1}$$

$$\mathbf{Q}_{\mathsf{Lin}} \coloneqq \left\{ \alpha\colon \mathbb{F}^\ell \to \mathbb{F} \;\middle|\; \exists\, \gamma \in \mathbb{F}^\ell \text{ s.t. } \alpha(\Pi) = \sum_{i=1}^{\ell} \gamma[i] \cdot \Pi[i] \right\} \quad , \tag{2}$$

$$\mathbf{Q}_{\mathsf{UniPoly}} \coloneqq \left\{ \alpha\colon \mathbb{F}^\ell \to \mathbb{F} \;\middle|\; \exists\, \gamma \in \mathbb{F} \text{ s.t. } \alpha(\Pi) = \sum_{i=0}^{\ell-1} \gamma^i \cdot \Pi[i] \right\} \quad , \tag{3}$$

$$\mathbf{Q}^{(m,\mathsf{D})}_{\mathsf{MultiPoly}} \coloneqq \left\{ \ell \coloneqq \binom{m+\mathsf{D}}{\mathsf{D}}, \alpha\colon \mathbb{F}^\ell \to \mathbb{F} \;\middle|\; \begin{array}{l} \exists\, \gamma \in \mathbb{F}^m \text{ s.t.} \\ \alpha(\Pi) = \sum_{\substack{\omega \in \{0,\ldots,\mathsf{D}\}^m \\ \sum_{i\in[m]} \omega[i] \le \mathsf{D}}} \gamma^\omega \cdot \Pi[\omega] \\ \text{where } \gamma^\omega \coloneqq \gamma[1]^{\omega[1]} \cdots \gamma[m]^{\omega[m]} \end{array} \right\} \quad . \tag{4}$$

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 $\mathsf{FIOP} = (\mathbf{P}, \mathbf{V})$ with query class $\mathbf{Q}$. The prover $\mathbf{P}$ receives as input an instance-witness tuple $(\mathbb{x}, \mathbb{w})$ and the verifier $\mathbf{V}$ receives as input the instance $\mathbb{x}$. For each round $i \in [\mathsf{k}_{\mathsf{FIOP}}]$, $\mathbf{P}$ outputs a proof string $\Pi \in \Sigma^{\ell_i}$ and $\mathbf{V}$ sends uniformly sampled randomness $\rho \in \{0,1\}^{\mathsf{r}_{\mathsf{FIOP},i}}$; 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 $\mathsf{k}_{\mathsf{FIOP}} = 1$. (See Section 3.2 for the formal definition of an FIOP.)

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

- $\mathsf{FC.Commit}$: On input a message $m \in \Sigma^\ell$, $\mathsf{FC.Commit}$ outputs a commitment cm and auxiliary state aux.
- $\mathsf{FC.Open}$: On input the auxiliary state aux and a query $\alpha \in \mathbf{Q}$, $\mathsf{FC.Open}$ outputs an opening proof pf.
- $\mathsf{FC.Check}$: On input a commitment cm, query $\alpha \in \mathbf{Q}$, answer $\beta \in \mathbb{D}$, and opening proof pf, $\mathsf{FC.Check}$ 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 $\mathbf{Q}_{\mathsf{Point}}$ is simply an IOP (resp., PCP) and an FC scheme for $\mathbf{Q}_{\mathsf{Point}}$ is a VC scheme. The Funky protocol in the case $\mathbf{Q} = \mathbf{Q}_{\mathsf{Point}}$ 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:

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

Reducing the general case of an FPCP and FC scheme for a function class $\mathbf{Q} = \{\alpha \colon \Sigma^{\ell} \to \mathbb{D}\}$ to the aforementioned special case is tempting though problematic. One could map an FPCP string $\Pi \in \Sigma^{\ell}$ to a corresponding PCP string $\Pi' \coloneqq (\alpha(\Pi))_{\alpha \in \mathbf{Q}} \in \mathbb{D}^{\ell'}$ with proof length $\ell' \coloneqq |\mathbf{Q}|$, 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

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

We *cannot afford such a bound*: while for point queries $|\mathbf{Q}_{\mathsf{Point}}| = \ell$, in general $|\mathbf{Q}|$ may be *much larger*, even exponentially larger, than ℓ. For example, for univariate polynomial queries $|\mathbf{Q}_{\mathsf{UniPoly}}| = |\mathbb{F}| \gg \ell$ 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: arguments based on functional PCPs

We sketch how we address the challenges in Section 2.1 for a special case of the Funky protocol: we temporarily focus on the case of a **FPCP** and a **non-interactive FC scheme** for a generic query class $\mathbf{Q}$.

2.2.1 The Funky protocol (special case for FPCPs)

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

1. $\mathcal{P}$ computes the FPCP proof $\Pi \leftarrow \mathbf{P}(\mathbb{x}, \mathbb{w})$, the commitment $(\mathsf{cm}, \mathsf{aux}) \leftarrow \mathsf{FC.Commit}(\Pi)$, and sends cm to $\mathcal{V}$.
2. $\mathcal{V}$ samples FPCP verifier randomness $\rho \leftarrow \{0,1\}^{\mathsf{r}}$ and sends it to $\mathcal{P}$.
3. $\mathcal{P}$ deduces the query set $\mathcal{Q} \subseteq \mathbf{Q}$ that $\mathbf{V}(\mathbb{x}; \rho)$ makes to Π. For each $\alpha \in \mathcal{Q}$, $\mathcal{P}$ sets the query answer $\beta_\alpha \coloneqq \alpha(\Pi)$ and computes an opening proof $\mathsf{pf}_\alpha \coloneqq \mathsf{FC.Open}(\mathsf{aux}, \alpha)$. $\mathcal{P}$ sends $\big((\alpha, \beta_\alpha, \mathsf{pf}_\alpha)\big)_{\alpha \in \mathcal{Q}}$ to $\mathcal{V}$.
4. $\mathcal{V}$ performs the following checks:

(a) for every $\alpha \in \mathcal{Q}$, $\mathsf{FC.Check}(\mathsf{cm}, \alpha, \beta_\alpha, \mathsf{pf}_\alpha) = 1$;
(b) $\mathbf{V}^{[(\alpha,\beta_\alpha)]_{\alpha\in\mathcal{Q}}}(\mathbb{x}; \rho) = 1$.

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

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., $\mathsf{Funky}[\mathsf{PCP}, \mathsf{VC}]$ 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 $\widetilde{\mathcal{P}}$, rewinds $\widetilde{\mathcal{P}}$ for sufficiently many times to recover an underlying PCP string $\widetilde{\Pi}$. Then they show that the probability of $\widetilde{\mathcal{P}}$ convincing the argument verifier $\mathcal{V}$ is close to the probability of $\widetilde{\Pi}$ convincing the PCP verifier $\mathbf{V}$. In particular, when $\widetilde{\mathcal{P}}$ succeeds, one of the following happens: (i) $\mathbf{V}^{\widetilde{\Pi}}$ accepts; (ii) $\widetilde{\mathcal{P}}$ 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 $\widetilde{\Pi}$. 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 $\widetilde{\mathcal{P}}$ 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 $\mathsf{N}+1$ times (each time with fresh randomness). If $\widetilde{\mathcal{P}}$ gives valid opening in the i-th run, record the query set $\mathcal{Q}_i$ and $\widetilde{\mathcal{P}}$'s answer ans_i. For what N does the following hold:*

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

(The first condition implies that $\widetilde{\mathcal{P}}$ 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 $\widetilde{\mathcal{P}}$ gives a valid opening. The probability that j does not appear in $\mathcal{Q}_i$ for any $i \in [\mathsf{N}]$ but is included in $\mathcal{Q}_{\mathsf{N}+1}$ is

$$(1 - \delta_j)^{\mathsf{N}} \cdot \delta_j\ ,$$

which is bounded by $\frac{1}{\mathsf{N}}$ 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 $\frac{\ell}{\mathsf{N}}$, where ℓ is the proof length of the PCP. Setting $\mathsf{N} \coloneqq \frac{\ell}{\epsilon}$ 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 $\mathsf{Funky}[\mathsf{FPCP}, \mathsf{FC}]$ reduces to the soundness error of FPCP and the function binding error of FC. In the VC setting, each time it rewinds the adversary $\widetilde{\mathcal{P}}$, the reductor $\mathfrak{R}$ records in a PCP string $\widetilde{\Pi} \in \Sigma^\ell$ the answers given by $\widetilde{\mathcal{P}}$ (if valid). However, for a general query class $\mathbf{Q}$, recorded query-answer pairs $(\alpha_1, \beta_1), \ldots, (\alpha_{\mathsf{N}}, \beta_{\mathsf{N}}) \in \mathbf{Q} \times \mathbb{D}$ are *constraints* on the set of possible proof strings $\widetilde{\Pi} \in \Sigma^\ell$:

$$S \coloneqq \left\{ \widetilde{\Pi} \in \Sigma^\ell : \begin{array}{c} \alpha_1(\widetilde{\Pi}) = \beta_1 \\ \vdots \\ \alpha_{\mathsf{N}}(\widetilde{\Pi}) = \beta_{\mathsf{N}} \end{array} \right\}\ . \tag{6}$$

When $\mathbf{Q} = \mathbf{Q}_{\mathsf{Point}}$ finding an element in this set is trivial: each query-answer directly specifies the location and value of an entry of $\widetilde{\Pi}$, so the reductor sets the entries of $\widetilde{\Pi}$ 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 $\mathbf{Q} = \mathbf{Q}_{\mathsf{UniPoly}}$, 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 $\alpha\colon \Sigma^\ell \to \mathbb{D}$ that is *one-way*; if the prover uses a random $\widetilde{\Pi} \in \Sigma^\ell$ then no efficient algorithm can, given α and $\alpha(\widetilde{\Pi})$, recover any Π' such that $\alpha(\widetilde{\Pi}) = \alpha(\widetilde{\Pi}')$.

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

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

$\mathfrak{R}^{\widetilde{\mathcal{P}}(\mathsf{aux},\cdot)}(\mathsf{cm})$: # aux is $\widetilde{\mathcal{P}}$'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 $\rho \leftarrow \{0,1\}^{\mathsf{r}}$.
   (b) Run $\widetilde{\mathcal{P}}(\mathsf{aux}, \rho)$ to get the openings $((\alpha, \beta_\alpha, \mathsf{pf}_\alpha))_{\alpha \in \mathcal{Q}}$.
   (c) If $\mathsf{FC.Check}(\mathsf{cm}, \alpha, \beta_\alpha, \mathsf{pf}_\alpha) = 1$ for every $\alpha \in \mathcal{Q}$, then append $((\alpha, \beta_\alpha, \mathsf{pf}_\alpha))_{\alpha \in \mathcal{Q}}$ to K.
3. Output $\widetilde{\Pi} \coloneqq \mathsf{Solver}_{\mathbf{Q}}(K) \in \Sigma^\ell$.

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

How to set the number of rewinds N? In the special case $\mathbf{Q} = \mathbf{Q}_{\mathsf{Point}}$, 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 $\epsilon_{\mathbf{Q}}(\mathsf{N})$ that we call the *tail error* of the query class $\mathbf{Q}$. Informally, $\epsilon_{\mathbf{Q}}(\mathsf{N})$ upper bounds the probability that after $\mathsf{N}+1$ 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 $(\mathsf{N}+1)$-th rewinding.

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

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

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

This definition enables us to obtain an upper bound. We let $\mathcal{D}$ be the output distribution of $\widetilde{\mathcal{P}}$: $\mathcal{D}$ gives the query set and answers $(\mathcal{Q}, \boldsymbol{\beta})$ if $\widetilde{\mathcal{P}}$ provides an accepting opening, otherwise $\mathcal{D}$ outputs $\emptyset$. Hence, fresh

samples from $\mathcal{D}$ corresponds to (fresh) rewinds of $\widetilde{\mathcal{P}}$; and S_i is the set of all FPCP strings that are consistent with $\widetilde{\mathcal{P}}$'s output after the i-th rewinding. This lets us prove that, for every N, the following holds:

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

This leaves us with the task of setting N so that $\epsilon_{\mathbf{Q}}(\mathsf{N})$ 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 $\epsilon_{\mathbf{Q}_{\mathsf{Point}}}(\mathsf{N}) \leq \frac{\ell}{\mathsf{N}}$. However their analysis does not generalize (see Section 2.1). We take a different approach.

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

$$\sum_{i=1}^{\mathsf{N}} \Pr\left[\mathcal{Q}_{i+1} \not\subseteq \cup_{j \in [i]} \mathcal{Q}_j\right] \leq \ell \ . \tag{8}$$

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

$$\forall i, i' \in [\mathsf{N}] \text{ s.t. } i' > i, \Pr\left[\mathcal{Q}_{i+1} \not\subseteq \cup_{j \in [i]} \mathcal{Q}_j\right] \geq \Pr\left[\mathcal{Q}_{i'+1} \not\subseteq \cup_{j \in [i']} \mathcal{Q}_j\right] \ . \tag{9}$$

Therefore:

$$\begin{aligned}
&\Pr\left[\begin{array}{l}\left(\forall\, i, j \in [\mathsf{N}+1]\ \forall\, q \in \mathcal{Q}_i \cap \mathcal{Q}_j\ ,\ \mathsf{ans}_i[q] = \mathsf{ans}_j[q]\right) \\ \wedge\, \mathcal{Q}_{\mathsf{N}+1} \not\subseteq \bigcup_{i \in [\mathsf{N}]} \mathcal{Q}_i\end{array}\right] \\
&\leq \Pr\left[\mathcal{Q}_{\mathsf{N}+1} \not\subseteq \bigcup_{i \in [\mathsf{N}]} \mathcal{Q}_i\right] \\
&\leq \frac{1}{\mathsf{N}} \cdot \sum_{i=1}^{\mathsf{N}} \Pr\left[\mathcal{Q}_{i+1} \not\subseteq \cup_{j \in [i]} \mathcal{Q}_j\right] && \text{(Equation 9)} \\
&\leq \frac{\ell}{\mathsf{N}} \ . && \text{(Equation 8)}
\end{aligned}$$

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

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

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.

2.2.3 Fiat–Shamir security

In this warmup example Fiat–Shamir security is "for free". Let $\mathsf{ARG} := \mathsf{Funky}[\mathsf{FPCP}, \mathsf{FC}]$ and let $\mathsf{NARG} := \mathsf{FS}[\mathsf{ARG}]$ be the non-interactive argument in the ROM after applying the Fiat–Shamir transformation to ARG. 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]):

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

However, in general, it is *not* the case that the soundness error ϵ_{NARG} for $\mathsf{NARG} := \mathsf{FS}[\mathsf{ARG}]$, where ARG is a (public-coin) $\mathsf{k}_{\mathsf{NARG}}$-round interactive argument, is closely related to the soundness ϵ_{ARG} for ARG. Specifically, it holds that $\epsilon_{\mathsf{NARG}}(\mathbb{x}, t_{\mathsf{NARG}}) \leq O(t_{\mathsf{NARG}}^{\mathsf{k}_{\mathsf{NARG}}}) \cdot \epsilon_{\mathsf{ARG}}(\mathbb{x}, t_{\mathsf{NARG}})$, 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.

2.3 Succinct arguments based on public-coin functional IOPs

We discuss the general case: the Fiat–Shamir security of $\mathsf{ARG} := \mathsf{Funky}[\mathsf{FIOP}, \mathsf{FC}]$, 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, 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.

2.3.1 The Funky protocol (general case for every FIOP)

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

1. For every FIOP round $i \in [\mathsf{k}_{\mathsf{FIOP}}]$:
   (a) $\mathcal{P}$ computes the i-th FIOP string $\Pi_i \leftarrow \mathbf{P}(\mathbb{x}, \mathbb{w}, \rho_{i-1})$, the commitment $(\mathsf{cm}_i, \mathsf{aux}_i) \leftarrow \mathsf{FC.Commit}(\Pi_i)$, and sends cm_i to $\mathcal{V}$.
   (b) $\mathcal{V}$ samples the i-th FIOP randomness $\rho_i \leftarrow \{0,1\}^{\mathsf{r}_{\mathsf{FIOP},i}}$ and sends it to $\mathcal{P}$.
2. $\mathcal{P}$ simulates $\mathbf{V}^{(\Pi_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}}(\mathbb{x}, (\rho_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]})$ to deduce the query sets $(\mathcal{Q}_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}$ ($\mathcal{Q}_i$ is $\mathbf{V}$'s queries to Π_i). For every $i \in [\mathsf{k}_{\mathsf{FIOP}}]$ and $\alpha \in \mathcal{Q}_i$:
   (a) $\mathcal{P}$ sets $\beta_{i,\alpha} := \alpha(\Pi_i)$;
   (b) $\mathcal{P}$ computes the opening proof $\mathsf{pf}_{i,\alpha} := \mathsf{FC.Open}(\mathsf{aux}_i, \alpha)$;
   (c) $\mathcal{P}$ sets $\beta_i := (\beta_{i,\alpha})_{\alpha\in\mathcal{Q}_i}$ and $\mathsf{pf}_i := (\mathsf{pf}_{i,\alpha})_{\alpha\in\mathcal{Q}_i}$.
3. $\mathcal{P}$ sends $\big(\big((\alpha, \beta_{i,\alpha}, \mathsf{pf}_{i,\alpha})\big)_{\alpha\in\mathcal{Q}_i}\big)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}$ to $\mathcal{V}$.
4. $\mathcal{V}$ performs the following checks:
   (a) for every $i \in [\mathsf{k}_{\mathsf{FIOP}}]$ and $\alpha \in \mathcal{Q}_i$, $\mathsf{FC.Check}(\mathsf{cm}_i, \alpha, \beta_{i,\alpha}, \mathsf{pf}_{i,\alpha}) = 1$;
   (b) $\mathbf{V}^{[(\alpha,\beta_{i,\alpha})]_{i\in[\mathsf{k}_{\mathsf{FIOP}}],\alpha\in\mathcal{Q}_i}}(\mathbb{x}, (\rho_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}) = 1$.

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.)

2.3.2 State-restoration security of the Funky protocol

We review the notion of state-restoration soundness, and then we describe how we extend the basic ideas from the warmup special case in Section 2.2 to prove Theorem 1.

State-restoration. The *state-restoration game* for a public-coin k-round interactive protocol is as follows. The game samples random functions $(\mathsf{rnd}_i)_{i\in[\mathsf{k}]}$ to be used as the verifier randomness. The adversary has a move budget m, and each move is a list of prover messages $(m_j)_{j\in[i]}$ for some $i \in [\mathsf{k}]$. The game answers a move $(m_j)_{j\in[i]}$ with $\mathsf{rnd}(\mathbb{x}, (m_j)_{j\in[i]})$. Eventually the adversary makes no more moves (by choice or by exhausting the move budget), and outputs $(m_i^\star)_{i\in[\mathsf{k}]}$. The game sets $\rho_i^\star := \mathsf{rnd}_i(\mathbb{x}, (m_j^\star)_{j\in[i]})$ for every $i \in [\mathsf{k}]$. The adversary wins the state-restoration game if and only if the verifier accepts when given the instance $\mathbb{x}$, the randomness $(\rho_i^\star)_{i\in[\mathsf{k}]}$, and prover messages $(m_i^\star)_{i\in[\mathsf{k}]}$.

The state-restoration soundness error $\epsilon^{\mathsf{SR}}_{\mathsf{ARG}}(\mathsf{m}, t_{\mathsf{ARG}})$ 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 $\epsilon^{\mathsf{SR}}_{\mathsf{FIOP}}(\mathsf{m}_{\mathsf{FIOP}})$ depends only on the move budget $\mathsf{m}_{\mathsf{FIOP}}$ of the FIOP state-restoration adversary (rather than also its size).

Proof sketch of Theorem 1 for non-interactive FC schemes. Analogously to standard (not state-restoration) security, we want to reduce the state-restoration security of the Funky protocol to the state-restoration security of the underlying FIOP. Ideally, given an argument state-restoration adversary $\widetilde{\mathcal{P}}^{\mathsf{SR}}$, we want to construct a FIOP state-restoration adversary $\widetilde{\mathbf{P}}^{\mathsf{SR}} := \widetilde{\mathbf{P}}^{\mathsf{SR}}(\widetilde{\mathcal{P}}^{\mathsf{SR}})$ for FIOP where:

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

There are challenges in turning the above rewinding template into a strategy that works.

- *Challenge 1: How to extract for moves?* Consider a move $(\mathbb{x}, (\mathsf{cm}_j)_{j\in[i]})$ made by $\widetilde{\mathcal{P}}^{\mathsf{SR}}$. In order to reconstruct corresponding FIOP strings, $\widetilde{\mathbf{P}}^{\mathsf{SR}}$ should rewind $\widetilde{\mathcal{P}}^{\mathsf{SR}}$ multiple times in search of (valid) openings for the commitments in the move. Each fresh run of $\widetilde{\mathcal{P}}^{\mathsf{SR}}$ yields more moves (which do not contain any openings) and a final output $\big(\mathbb{x}^\star, (\mathsf{cm}^\star_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}, \big(\big((\alpha^\star, \beta^\star_\alpha, \mathsf{pf}^\star_\alpha)\big)_{\alpha\in\mathcal{Q}_i}\big)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}\big)$. If $(\mathbb{x}, (\mathsf{cm}_j)_{j\in[i]}) \neq (\mathbb{x}^\star, (\mathsf{cm}^\star_j)_{j\in[i]})$ then this final output does not contribute any valid openings for the move. Worse, it can be that *no run of $\widetilde{\mathcal{P}}^{\mathsf{SR}}$ yields any progress for* $(\mathbb{x}, (\mathsf{cm}_j)_{j\in[i]})$ ($\widetilde{\mathcal{P}}^{\mathsf{SR}}$ never opens the commitments for this particular move).

  Solution: If $\widetilde{\mathbf{P}}^{\mathsf{SR}}$ cannot reconstruct FIOP strings for $(\mathbb{x}, (\mathsf{cm}_j)_{j\in[i]})$ then it suffices for $\widetilde{\mathbf{P}}^{\mathsf{SR}}$ to use *empty FIOP strings* for this move because $\widetilde{\mathcal{P}}^{\mathsf{SR}}$ does not use this move to win SRGame. Hence, it suffices for $\widetilde{\mathbf{P}}^{\mathsf{SR}}$ to provide to $\widetilde{\mathcal{P}}^{\mathsf{SR}}$ random and consistent answers. (To ensure that the mapping of moves from one game to the other is injective, the commitments $(\mathsf{cm}_j)_{j\in[i]}$ are included in the salt string.)

- *Challenge 2: How to extract for the (final) output?* The final output $\big(\mathbb{x}^\star, (\mathsf{cm}^\star_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}, \big(\big((\alpha^\star, \beta^\star_\alpha, \mathsf{pf}^\star_\alpha)\big)_{\alpha\in\mathcal{Q}_i}\big)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}\big)$ of $\widetilde{\mathcal{P}}^{\mathsf{SR}}$ contains valid openings for the commitments, so the above issue does not arise here. Instead there is a different issue: $\widetilde{\mathcal{P}}^{\mathsf{SR}}$ halts with the final output, so how can $\widetilde{\mathbf{P}}^{\mathsf{SR}}$ rewind $\widetilde{\mathcal{P}}^{\mathsf{SR}}$ from this point onward to collect additional valid openings?

  Solution: We augment $\widetilde{\mathcal{P}}^{\mathsf{SR}}$ to output all partial moves induced by the final output. We consider an extended argument state-restoration adversary $\hat{\mathcal{P}}^{\mathsf{SR}}$ that works the same as $\widetilde{\mathcal{P}}^{\mathsf{SR}}$ except that, before halting, additionally makes these moves (ordered by increasing i):

$$\big\{(\mathbb{x}, (\mathsf{cm}_j)_{j\in[i]})\big\}_{i\in[\mathsf{k}_{\mathsf{FIOP}}]} \quad .$$

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

Taking the above into account, we construct $\widetilde{\mathbf{P}}^{\mathsf{SR}}$ as follows.

$\widetilde{\mathbf{P}}^{\mathsf{SR}}$:

1. Simulate SRGame using $\hat{\mathcal{P}}^{\mathsf{SR}}$.
2. For every move $\mathsf{mv} = (\mathbb{x}, (\mathsf{cm}_j)_{j\in[i]})$ made by $\hat{\mathcal{P}}^{\mathsf{SR}}$:
   (a) Let aux be the internal state of $\hat{\mathcal{P}}^{\mathsf{SR}}$ after outputting the move.
   (b) Repeat N times:
      i. Continue the simulation of $\hat{\mathcal{P}}^{\mathsf{SR}}(\mathsf{aux})$ using fresh randomness to answer $\widetilde{\mathcal{P}}^{\mathsf{SR}}$'s moves.
      ii. If $\hat{\mathcal{P}}^{\mathsf{SR}}$ provides answers and valid openings consistent with mv, record the answers.
   (c) For every $j \in [i]$, construct $(\widetilde{\Pi}_j)_{j\in[i]}$ using $\mathsf{Solver}_{\mathbf{Q}}$ from the recorded answers.
   (d) Forward $(\mathbb{x}, (\widetilde{\Pi}_j)_{j\in[i]})$ to $\mathsf{FIOPSRGame}$ to get the next verifier randomness and answer the move mv output by $\hat{\mathcal{P}}^{\mathsf{SR}}$.
3. Let $(\widetilde{\Pi}^{\star}_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}$ be the FIOP strings corresponds to the final output $(\mathsf{cm}^{\star}_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}$ of $\widetilde{\mathcal{P}}^{\mathsf{SR}}$.
4. Output $(\mathbb{x}, (\widetilde{\Pi}^{\star}_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]})$.

We are left to argue that $\widetilde{\mathbf{P}}^{\mathsf{SR}}$ is "as good as" $\widetilde{\mathcal{P}}^{\mathsf{SR}}$. For this, we extend to state-restoration games the ideas that we introduced in Section 2.2.2. If $\widetilde{\mathcal{P}}^{\mathsf{SR}}$ succeeds then one of the following happens: (i) $\widetilde{\mathbf{P}}^{\mathsf{SR}}$ succeeds; (ii) $\widetilde{\mathcal{P}}^{\mathsf{SR}}$ violates function binding of FC; or (iii) $\widetilde{\mathbf{P}}^{\mathsf{SR}}$ fails to extract enough information about the FIOP strings underlying the final output of $\widetilde{\mathcal{P}}^{\mathsf{SR}}$. The first event can be bounded by the FIOP state-restoration soundness error. The second event can be bounded by the function binding error of FC. The last event can be bounded via tail errors as defined in Section 2.2.2. Overall, we bound the state-restoration soundness error $\epsilon^{\mathsf{SR}}_{\mathsf{ARG}}$ of ARG as follows:

$$\epsilon^{\mathsf{SR}}_{\mathsf{ARG}}(\mathsf{m}, t_{\mathsf{ARG}}) \leq \epsilon^{\mathsf{SR}}_{\mathsf{FIOP}}(\mathsf{m}_{\mathsf{FIOP}}) + \mathsf{k}_{\mathsf{FIOP}} \cdot \epsilon_{\mathsf{FC}}(t_{\mathsf{FC}}) + \mathsf{k}_{\mathsf{FIOP}} \cdot \epsilon_{\mathbf{Q}}(\mathsf{N}) \quad \text{where} \quad \begin{cases} \mathsf{m}_{\mathsf{FIOP}} = \mathsf{m} + \mathsf{k}_{\mathsf{FIOP}} \\ t_{\mathsf{FC}} = O\left(\mathsf{N} \cdot t_{\mathsf{ARG}} + t_{\mathbf{Q}}\right) \end{cases}.$$

The round complexity $\mathsf{k}_{\mathsf{FIOP}}$ of the FIOP multiplicatively affects two terms: (a) the term $\mathsf{k}_{\mathsf{FIOP}} \cdot \epsilon_{\mathsf{FC}}(t_{\mathsf{FC}})$, as the argument state-restoration adversary $\widetilde{\mathcal{P}}^{\mathsf{SR}}$ 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 $\mathsf{Solver}_{\mathbf{Q}}$; (b) the term $\mathsf{k}_{\mathsf{FIOP}} \cdot \epsilon_{\mathbf{Q}}(\mathsf{N})$, as each extracted FIOP string in the final output incurs the tail error $\epsilon_{\mathbf{Q}}(\mathsf{N})$.

Interactive FC schemes. FC schemes with an interactive opening phase are widely used (e.g., [BBBPWM18; Lee21; AGLMS22]). An interactive FC scheme FC is a tuple $(\mathsf{FC.Commit}, \mathcal{P}_{\mathsf{FC}}, \mathcal{V}_{\mathsf{FC}})$ where $\mathcal{P}_{\mathsf{FC}}$ and $\mathcal{V}_{\mathsf{FC}}$ interact with each other for 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 $\mathsf{FCSRGame}$ (Definition 3.19) and binding property (Definition 3.20).

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

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

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

In this case, $\widetilde{\mathcal{P}}^{\mathsf{SR}}_{\mathsf{FC},i}$ makes the move $(\mathsf{cm}_u, (\mathcal{Q}_u, \boldsymbol{\beta}_u), (\mathsf{pm}_{u,j})_{j\in[c]})$ in the FC state-restoration game $\mathsf{FCSRGame}$ for every $u \in [\mathsf{k}_{\mathsf{FIOP}}]$. After this, $\widetilde{\mathcal{P}}^{\mathsf{SR}}_{\mathsf{FC},i}$ runs $\mathsf{Solver}_{\mathbf{Q}}$ 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 $\widetilde{\mathcal{P}}^{\mathsf{SR}}_{\mathsf{FC},i}$ wins if $\mathsf{Solver}_{\mathbf{Q}}$ does not find any such Π_i, since this means that $\widetilde{\mathcal{P}}^{\mathsf{SR}}_{\mathsf{FC},i}$ broke function binding of FC. We prove that if Case ii happens when $\widetilde{\mathcal{P}}^{\mathsf{SR}}$ successfully wins the argument state-restoration game SRGame then there exists $i \in [\mathsf{k}_{\mathsf{FIOP}}]$ such that $\widetilde{\mathcal{P}}^{\mathsf{SR}}_{\mathsf{FC},i}$ wins in the FC state-restoration game $\mathsf{FCSRGame}$.

Note that since $\widetilde{\mathcal{P}}^{\mathsf{SR}}_{\mathsf{FC},i}$ rewinds $\widetilde{\mathcal{P}}^{\mathsf{SR}}$ for N times, and in the worst case each move by $\widetilde{\mathcal{P}}^{\mathsf{SR}}$ results in $\mathsf{k}_{\mathsf{FIOP}}$ moves by $\widetilde{\mathcal{P}}^{\mathsf{SR}}_{\mathsf{FC},i}$, the total number of moves by $\widetilde{\mathcal{P}}^{\mathsf{SR}}_{\mathsf{FC},i}$ is $O(\mathsf{N}\cdot\mathsf{k}_{\mathsf{FIOP}}\cdot(\mathsf{m}+\mathsf{k}_{\mathsf{FC}}))$. (Similarly to how we address Challenge 2 in Section 2.3.2 in $\widetilde{\mathbf{P}}^{\mathsf{SR}}$, we augment $\widetilde{\mathcal{P}}^{\mathsf{SR}}$ to output all partial moves corresponding to the FC opening phase in its final output, which increases the number of moves by $\widetilde{\mathcal{P}}^{\mathsf{SR}}$ from m to $\mathsf{m}+\mathsf{k}_{\mathsf{FC}}$.)

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

$$\epsilon^{\mathsf{SR}}_{\mathsf{ARG}}(\mathsf{m}, t_{\mathsf{ARG}}) \le \epsilon^{\mathsf{SR}}_{\mathsf{FIOP}}(\mathsf{m}_{\mathsf{FIOP}}) + \mathsf{k}_{\mathsf{FIOP}}\cdot\epsilon^{\mathsf{SR}}_{\mathsf{FC}}(\mathsf{m}_{\mathsf{FC}}, t_{\mathsf{FC}}) + \mathsf{k}_{\mathsf{FIOP}}\cdot\epsilon_{\mathbf{Q}}(\mathsf{N}) \quad \text{where} \quad \begin{cases} \mathsf{m}_{\mathsf{FIOP}} = \mathsf{m} + \mathsf{k}_{\mathsf{FIOP}} \\ \mathsf{m}_{\mathsf{FC}} = O(\mathsf{N}\cdot\mathsf{k}_{\mathsf{FIOP}}\cdot(\mathsf{m}+\mathsf{k}_{\mathsf{FC}})) \\ t_{\mathsf{FC}} = O\left(\mathsf{N}\cdot t_{\mathsf{ARG}} + t_{\mathbf{Q}}\right) \end{cases}.$$

where $\epsilon^{\mathsf{SR}}_{\mathsf{FC}}$ 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.

2.4 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 (suitably) homomorphic PC scheme PC, of which the PC scheme KZG in [KZG10] is an example. These reductions may be useful towards designing other optimized FC schemes.

- Every homomorphic polynomial commitment scheme PC for $\mathbf{Q}_{\mathsf{UniPoly}}$ gives rise to an optimized batched polynomial commitment scheme $\mathsf{bPC} = \mathsf{BatchMsg}[\mathsf{PC}, s]$ for the query class $\mathbf{Q}_{\mathsf{BatchMsg}}[\mathbf{Q}_{\mathsf{UniPoly}}, s]$. We show that if PC is state-restoration function binding, then so is bPC (Section 8.1).
- Every homomorphic batch polynomial commitment scheme bPC for the query class $\mathbf{Q}_{\mathsf{BatchMsg}}[\mathbf{Q}_{\mathsf{UniPoly}}, m{+}1]$ gives rise to an optimized linearized polynomial commitment scheme $\mathsf{linPC} = \mathsf{Lin}[\mathsf{bPC}, m, (\mathsf{h}_k)_{k\in[n]}]$ for the query class $\mathbf{Q}_{\mathsf{Struct}}[\mathbf{Q}_{\mathsf{UniPoly}}, m, (\mathsf{h}_k)_{k\in[n]}]$ of structured polynomial evaluation queries, via the linearization trick (also known as Maller's trick) [GWC19; LPS24b; FFR24]. We show that if bPC is state-restoration function binding, then so is linPC (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. Since KZG is homomorphic we can apply the black-box reductions outlined above, establishing that

$$\mathsf{linKZG} = \mathsf{Lin}\left[\mathsf{BatchMsg}[\mathsf{KZG}, m+1], m, (\mathsf{h}_k)_{k\in[n]}\right]$$

is state-restoration function binding (see Corollaries 9.2 and 9.3 for details). In contrast, [LPS24b] 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 proposed for KZG in Appendix B.

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

- Let $\mathsf{PlonkIOP}$ be the FIOP for the query class $\mathbf{Q}_{\mathsf{Struct}}[\mathbf{Q}_{\mathsf{UniPoly}}, m, (\mathsf{h}_k)_{k\in[n]}]$ described in [GWC19].
- Let linKZG be the FC scheme for the query class $\mathbf{Q}_{\mathsf{Struct}}[\mathbf{Q}_{\mathsf{UniPoly}}, m, (\mathsf{h}_k)_{k\in[n]}]$ discussed above.
- Let $\mathsf{iPlonk} \coloneqq \mathsf{Funky}[\mathsf{PlonkIOP}, \mathsf{linKZG}]$ 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 $\epsilon^{\mathsf{SR}}_{\mathsf{FIOP}}$ of $\mathsf{PlonkIOP}$, (ii) the state-restoration function binding error $\epsilon^{\mathsf{SR}}_{\mathsf{FC}}$ of linKZG, and (iii) the tail error of $\mathbf{Q}_{\mathsf{Struct}}[\mathbf{Q}_{\mathsf{UniPoly}}, m, (\mathsf{h}_k)_{k\in[n]}]$. We discuss each in turn.

- $\mathsf{PlonkIOP}$ 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 $\epsilon^{\mathsf{SR}}_{\mathsf{FC}}$ for linKZG (our Lemma 1).
- We prove that $\epsilon_{\mathbf{Q}_{\mathsf{Struct}}[\mathbf{Q}, m, (\mathsf{h}_k)_{k\in[n]}]}(\mathsf{N}) \leq \frac{(m+n)(\mathsf{D}+1)}{\mathsf{N}}$ (see Lemma 5.10).

[LPS24b] 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 establishes the Fiat–Shamir security of iPlonk from the ARSDH assumption alone.

Remark 2.1. 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 a log-round interactive polynomial commitment scheme 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 Q** *is a set of functions of the form* $\alpha \colon \Sigma^\ell \to \mathbb{D}$*, 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 $\mathsf{ARG} = (\mathcal{G}, \mathcal{P}, \mathcal{V})$ satisfying the following properties.

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

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

Definition 3.4. $\mathsf{ARG} = (\mathcal{G}, \mathcal{P}, \mathcal{V})$ *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* $t_{\mathsf{ARG}} \in \mathbb{N}$*, and* t_{ARG}*-size circuit* $\widetilde{\mathcal{P}}$*,*

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

Definition 3.5. $\mathsf{ARG} = (\mathcal{G}, \mathcal{P}, \mathcal{V})$ *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* $t_{\mathsf{ARG}} \in \mathbb{N}$*, and* t_{ARG}*-size circuit* $\widetilde{\mathcal{P}}$*,*

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

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

Above, $b \xleftarrow{\mathsf{tr}} \langle \widetilde{\mathcal{P}}(\mathsf{aux}), \mathcal{V}(\mathsf{pp}, \mathbb{x}) \rangle$ denotes the fact that tr is the transcript of the interaction (i.e., the messages exchanged between $\widetilde{\mathcal{P}}$ and $\mathcal{V}$). Moreover, $\mathcal{E}^{\widetilde{\mathcal{P}}}$ means that $\mathcal{E}$ has black-box access to (each next-message function of) $\widetilde{\mathcal{P}}$; in particular $\mathcal{E}$ can send verifier messages to $\widetilde{\mathcal{P}}$ in order to obtain the next message of $\widetilde{\mathcal{P}}$ (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* $f \colon \{0,1\}^* \to \{0,1\}^n$*. Moreover, for* $m \in \mathbb{N}$ *and* $(n_1, \ldots, n_m) \in \mathbb{N}^m$*,* $\mathcal{U}((n_1, \ldots, n_m)) \coloneqq \mathcal{U}(n_1) \times \cdots \times \mathcal{U}(n_m)$*.*

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

$\mathsf{SRGame}(s, \mathsf{rnd}, \widetilde{\mathcal{P}}^{\mathsf{SR}}, \mathsf{pp}, \mathsf{ai})$*:*

1. *Repeat the following until* $\widetilde{\mathcal{P}}^{\mathsf{SR}}(\mathsf{pp}, \mathsf{ai})$ *decides to exit the loop:*
   (a) $\widetilde{\mathcal{P}}^{\mathsf{SR}}$ *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* $\rho_i \coloneqq \mathsf{rnd}_i(\mathbb{x}, (m_j)_{j\in[i]}, (\sigma_j)_{j\in[i]})$*.*
   (c) *Send* ρ_i *to* $\widetilde{\mathcal{P}}^{\mathsf{SR}}$*.*
2. $\widetilde{\mathcal{P}}^{\mathsf{SR}}$ *outputs* $(\mathbb{x}, (m_i)_{i\in[\mathsf{k}]}, (\sigma_i)_{i\in[\mathsf{k}]})$*.*
3. *For every* $i \in [\mathsf{k}]$*, set* $\rho_i \coloneqq \mathsf{rnd}_i(\mathbb{x}, (m_j)_{j\in[i]}, (\sigma_j)_{j\in[i]})$*.*
4. *Output* $(\mathbb{x}, (m_i)_{i\in[\mathsf{k}]}, (\sigma_i)_{i\in[\mathsf{k}]}, (\rho_i)_{i\in[\mathsf{k}]})$*.*

We denote by $\mathsf{tr}^{\mathsf{SR}}$ *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* $\mathsf{tr}^{\mathsf{SR}}$ *in an execution of the argument state-restoration game* SRGame *using the following notation:*

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

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

Definition 3.8. $\mathsf{ARG} = (\mathcal{G}, \mathcal{P}, \mathcal{V})$ *has* **state-restoration soundness error** $\epsilon^{\mathsf{SR}}_{\mathsf{ARG}}$ *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* $\mathsf{m} \in \mathbb{N}$*, circuit size bound* $t_{\mathsf{ARG}} \in \mathbb{N}$*, and* m*-move* t_{ARG}*-size circuit* $\widetilde{\mathcal{P}}^{\mathsf{SR}}$*,*

$$\Pr\left[\begin{array}{l|l} |\mathbb{x}| \le n & \mathsf{pp} \leftarrow \mathcal{G}(1^\lambda, n) \\ \wedge\, \mathbb{x} \notin L(R) & \mathsf{ai} \leftarrow \mathcal{D} \\ \wedge\, \mathcal{V}(\mathbb{x}, (m_i)_{i\in[\mathsf{k}]}, (\rho_i)_{i\in[\mathsf{k}]}) = 1 & \mathsf{rnd} \coloneqq (\mathsf{rnd}_i)_{i\in[\mathsf{k}]} \leftarrow \mathcal{U}((\mathsf{r}_i)_{i\in[\mathsf{k}]}) \\ & (\mathbb{x}, (m_i)_{i\in[\mathsf{k}]}, (\sigma_i)_{i\in[\mathsf{k}]}, (\rho_i)_{i\in[\mathsf{k}]}) \leftarrow \mathsf{SRGame}(s, \mathsf{rnd}, \widetilde{\mathcal{P}}^{\mathsf{SR}}, \mathsf{pp}, \mathsf{ai}) \end{array}\right]$$
$$\le \epsilon^{\mathsf{SR}}_{\mathsf{ARG}}(\lambda, n, s, \mathsf{m}, t_{\mathsf{ARG}}) \ .$$

Definition 3.9. $\mathsf{ARG} = (\mathcal{G}, \mathcal{P}, \mathcal{V})$ *has* **state-restoration knowledge soundness error** $\kappa^{\mathsf{SR}}_{\mathsf{ARG}}$ **with extraction time** $t_{\mathcal{E}_{\mathsf{SR}}}$ *if there exists a probabilistic algorithm* $\mathcal{E}_{\mathsf{SR}}$ *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* $\mathsf{m} \in \mathbb{N}$*, circuit size bound* $t_{\mathsf{ARG}} \in \mathbb{N}$*, and* m*-move* t_{ARG}*-size deterministic state-restoration prover* $\widetilde{\mathcal{P}}^{\mathsf{SR}}$*,*

$$\Pr\left[\begin{array}{l|l} |\mathbb{x}| \le n & \mathsf{pp} \leftarrow \mathcal{G}(1^\lambda, n) \\ \wedge\, (\mathbb{x}, \mathbb{w}) \notin R & \mathsf{ai} \leftarrow \mathcal{D} \\ \wedge\, \mathcal{V}(\mathbb{x}, (m_i)_{i\in[\mathsf{k}]}, (\rho_i)_{i\in[\mathsf{k}]}) = 1 & \mathsf{rnd} \coloneqq (\mathsf{rnd}_i)_{i\in[\mathsf{k}]} \leftarrow \mathcal{U}((\mathsf{r}_i)_{i\in[\mathsf{k}]}) \\ & (\mathbb{x}, (m_i)_{i\in[\mathsf{k}]}, (\sigma_i)_{i\in[\mathsf{k}]}, (\rho_i)_{i\in[\mathsf{k}]}) \xleftarrow{\mathsf{tr}^{\mathsf{SR}}} \mathsf{SRGame}(s, \mathsf{rnd}, \widetilde{\mathcal{P}}^{\mathsf{SR}}, \mathsf{pp}, \mathsf{ai}) \\ & \mathbb{w} \leftarrow \mathcal{E}^{\widetilde{\mathcal{P}}^{\mathsf{SR}}(\mathsf{pp},\mathsf{ai})}_{\mathsf{SR}}(\mathbb{x}, (m_i)_{i\in[\mathsf{k}]}, (\sigma_i)_{i\in[\mathsf{k}]}, (\rho_i)_{i\in[\mathsf{k}]}, \mathsf{tr}^{\mathsf{SR}}) \end{array}\right]$$
$$\le \kappa^{\mathsf{SR}}_{\mathsf{ARG}}(\lambda, n, s, \mathsf{m}, t_{\mathsf{ARG}}) \ ;$$

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

We consider several 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 $\mathsf{r}_{\mathsf{max}} \coloneqq \max_{i\in[\mathsf{k}]}\{\mathsf{r}_i\}$.

- 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 $\mathsf{pc}_{\mathsf{max}} := \max_{i \in [\mathsf{k}]}\{\mathsf{pc}_i\}$;
- the argument generator running time is $t_{\mathcal{G}}$;
- the argument prover running time is $t_{\mathcal{P}}$;
- the argument verifier running time is $t_{\mathcal{V}}$;
- the public parameter complexity $\mathsf{p}_{\mathsf{ARG}}$ 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 $\mathbf{Q}_1, \ldots, \mathbf{Q}_{\mathsf{k}_{\mathsf{FIOP}}}$ is a tuple $\mathsf{FIOP} = (\mathbf{P}, \mathbf{V})$ 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 $\mathsf{k}_{\mathsf{FIOP}}$ rounds, and in each round $i \in [\mathsf{k}_{\mathsf{FIOP}}]$ 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, and the verifier replies with a random string $\rho_i \in \{0,1\}^{\mathsf{r}_{\mathsf{FIOP},i}}$. The verifier has access to $(\Pi_1, \Pi_2, \ldots, \Pi_{\mathsf{k}_{\mathsf{FIOP}}})$ through queries in $\mathbf{Q}_1, \ldots, \mathbf{Q}_{\mathsf{k}_{\mathsf{FIOP}}}$. In more detail, the answer of a query $\alpha \in \mathbf{Q}_i$ to Π_i for some $i \in [\mathsf{k}_{\mathsf{FIOP}}]$ 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. $\mathsf{FIOP} = (\mathbf{P}, \mathbf{V})$ *has* **perfect completeness** *if, for every instance-witness pair $(\mathbb{x}, \mathbb{w}) \in R$,*

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

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

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

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

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

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

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

$\mathsf{FIOPSRGame}(s_{\mathsf{FIOP}}, \mathsf{rnd}_{\mathsf{FIOP}}, \widetilde{\mathbf{P}}^{\mathsf{SR}}, \mathbf{ai})$*:*

1. *Repeat the following until* $\widetilde{\mathbf{P}}^{\mathsf{SR}}(\mathbf{ai})$ *decides to exit the loop:*
   (a) $\widetilde{\mathbf{P}}^{\mathsf{SR}}$ *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* $\{0,1\}^{s_{\mathsf{FIOP}}}$.
   (b) *Set* $\rho_i \coloneqq \mathsf{rnd}_{\mathsf{FIOP},i}(\mathbb{x}, (m_j)_{j\in[i]}, (\gamma_j)_{j\in[i]})$.
   (c) *Send* ρ_i *to* $\widetilde{\mathbf{P}}^{\mathsf{SR}}$.
2. $\widetilde{\mathbf{P}}^{\mathsf{SR}}$ *outputs* $(\mathbb{x}, (m_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}, (\gamma_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]})$.
3. *For every* $i \in [\mathsf{k}_{\mathsf{FIOP}}]$*, set* $\rho_i \coloneqq \mathsf{rnd}_{\mathsf{FIOP},i}(\mathbb{x}, (m_j)_{j\in[i]}, (\gamma_j)_{j\in[i]})$.
4. *Output* $(\mathbb{x}, (m_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}, (\gamma_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}, (\rho_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]})$.

We denote by $\mathsf{tr}^{\mathsf{SR}}_{\mathsf{FIOP}}$ *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* $\mathsf{tr}^{\mathsf{SR}}_{\mathsf{FIOP}}$ *in an execution of the FIOP state-restoration game* FIOPSRGame *using the following notation:*

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

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

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

$$\Pr\left[\begin{array}{l|l} |\mathbb{x}| \le n & \mathbf{ai} \leftarrow \mathbf{D} \\ \wedge\, \mathbb{x} \notin L(R) & \mathsf{rnd}_{\mathsf{FIOP}} \coloneqq (\mathsf{rnd}_{\mathsf{FIOP},i})_{i\in[\mathsf{k}_{\mathsf{FIOP}}]} \leftarrow \mathcal{U}((\mathsf{r}_{\mathsf{FIOP},i})_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}) \\ \wedge\, \mathbf{V}(\mathbb{x}, (m_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}, (\rho_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}) = 1 & (\mathbb{x}, (m_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}, (\gamma_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}, (\rho_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}) \\ & \quad \leftarrow \mathsf{FIOPSRGame}(s_{\mathsf{FIOP}}, \mathsf{rnd}_{\mathsf{FIOP}}, \widetilde{\mathbf{P}}^{\mathsf{SR}}, \mathbf{ai}) \end{array}\right]$$
$$\le \epsilon^{\mathsf{SR}}_{\mathsf{FIOP}}(n, s_{\mathsf{FIOP}}, \mathsf{m}_{\mathsf{FIOP}}) \ .$$

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

$$\Pr\left[\begin{array}{l|l} |\mathbb{x}| \le n & \mathbf{ai} \leftarrow \mathbf{D} \\ \wedge\, (\mathbb{x}, \mathbb{w}) \notin R & \mathsf{rnd}_{\mathsf{FIOP}} \coloneqq (\mathsf{rnd}_{\mathsf{FIOP},i})_{i\in[\mathsf{k}_{\mathsf{FIOP}}]} \leftarrow \mathcal{U}((\mathsf{r}_{\mathsf{FIOP},i})_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}) \\ \wedge\, \mathbf{V}(\mathbb{x}, (m_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}, (\rho_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}) = 1 & (\mathbb{x}, (m_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}, (\gamma_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}, (\rho_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}) \\ & \quad \xleftarrow{\mathsf{tr}^{\mathsf{SR}}_{\mathsf{FIOP}}} \mathsf{FIOPSRGame}(s_{\mathsf{FIOP}}, \mathsf{rnd}_{\mathsf{FIOP}}, \widetilde{\mathbf{P}}^{\mathsf{SR}}, \mathbf{ai}) \\ & \mathbb{w} \leftarrow \mathbf{E}^{\widetilde{\mathbf{P}}^{\mathsf{SR}}(\mathbf{ai})}_{\mathsf{SR}}(\mathbb{x}, (m_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}, (\gamma_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}, (\rho_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}, \mathsf{tr}^{\mathsf{SR}}_{\mathsf{FIOP}}) \end{array}\right]$$
$$\le \kappa^{\mathsf{SR}}_{\mathsf{FIOP}}(n, s_{\mathsf{FIOP}}, \mathsf{m}_{\mathsf{FIOP}}) \ .$$

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

We consider several efficiency measures for an FIOP:

- the round complexity $\mathsf{k}_{\mathsf{FIOP}}$ 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 $\ell_{\mathsf{max}} \coloneqq \max_{i \in [\mathsf{k_{FIOP}}]}\{\ell_i\}$;
- 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 $\mathsf{q}_{\mathsf{max}} \coloneqq \max_{i \in [\mathsf{k_{FIOP}}]}\{\mathsf{q}_i\}$;
- the randomness complexity $\mathsf{r_{FIOP}}$ is the total number of random bits used by the FIOP verifier, $\mathsf{r}_{\mathsf{FIOP},i}$ is the number of random bits that the FIOP verifier uses in round i, and $\mathsf{r}_{\mathsf{FIOP},\mathsf{max}} \coloneqq \max_{i \in [\mathsf{k_{FIOP}}]}\{\mathsf{r}_{\mathsf{FIOP},i}\}$.

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 $\mathbf{Q} = \{\alpha \colon \Sigma^\ell \to \mathbb{D}\}$ is a tuple of algorithms

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

that satisfies the following syntax.

- $\mathsf{FC.Gen}(1^\lambda, \ell) \to \mathsf{pp_{FC}}$: On input a security parameter $\lambda \in \mathbb{N}$ and message length $\ell \in \mathbb{N}$, $\mathsf{FC.Gen}$ samples public parameter $\mathsf{pp_{FC}}$.
- $\mathsf{FC.Commit}(\mathsf{pp_{FC}}, \Pi) \to (\mathsf{cm}, \mathsf{aux})$: On input a public parameter $\mathsf{pp_{FC}}$ and a message vector $\Pi \in \Sigma^\ell$, $\mathsf{FC.Commit}$ produces a commitment cm and the corresponding auxiliary state aux.
- $\mathsf{FC.Open}(\mathsf{pp_{FC}}, \mathsf{aux}, \alpha, \beta) \to \mathsf{pf}$: On input a public parameter $\mathsf{pp_{FC}}$, an auxiliary state aux, a query $\alpha \in \mathbf{Q}$ and a purported evaluation $\beta \in \mathbb{D}$, $\mathsf{FC.Open}$ outputs an opening proof pf for the claim "$\beta = \alpha(\Pi)$".
- $\mathsf{FC.Check}(\mathsf{pp_{FC}}, \mathsf{cm}, \alpha, \beta, \mathsf{pf}) \to \{0,1\}$: On input a public parameter $\mathsf{pp_{FC}}$, a commitment cm, a query $\alpha \in \mathbf{Q}$, an answer $\beta \in \mathbb{D}$, and an opening proof pf, $\mathsf{FC.Check}$ 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)**.** $\mathsf{FC} = (\mathsf{FC.Gen}, \mathsf{FC.Commit}, \mathsf{FC.Open}, \mathsf{FC.Check})$ *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* $\mathsf{L} \in \mathbb{N}$*, auxiliary input distribution* $\mathcal{D}_{\mathsf{FC}}$*, adversary size bound* t_{FC}*, and* t_{FC}*-size circuit* A_{FC}*,*

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

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:

- $\epsilon_{\mathsf{FC}}(\cdot, \ell, \mathsf{L}, t_{\mathsf{FC}})$ is non-increasing (larger security parameters decrease an adversary's success);
- $\epsilon_{\mathsf{FC}}(\lambda, \cdot, \mathsf{L}, t_{\mathsf{FC}})$ is non-decreasing (longer message vector decrease an adversary's success);
- $\epsilon_{\mathsf{FC}}(\lambda, \ell, \cdot, t_{\mathsf{FC}})$ is non-decreasing (The solution space with a set of constraints is smaller than that of its subset); and
- $\epsilon_{\mathsf{FC}}(\lambda, \ell, \mathsf{L}, \cdot)$ 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 $\epsilon_{\mathsf{FC}}(\lambda, \ell_{\mathsf{max}}, \mathsf{L}_{\mathsf{max}}, t_{\mathsf{FC}})$, when $\ell_{\mathsf{max}} = \max_i \{ \ell_i \}$ and $\mathsf{L}_{\mathsf{max}} = \max_j \{ \mathsf{L}_j \}$, with

$$\max_{i,j} \{\epsilon_{\mathsf{FC}}(\lambda, \ell_i, \mathsf{L}_j, t_{\mathsf{FC}})\} \quad .$$

3.3.2 Interactive functional commitment schemes

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

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

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

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

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

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

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

$\mathsf{FCSRGame}(s_{\mathsf{FC}}, \mathsf{rnd}_{\mathsf{FC}}, \widetilde{\mathcal{P}}^{\mathsf{SR}}_{\mathsf{FC}}, \mathsf{pp}_{\mathsf{FC}}, \mathsf{ai}_{\mathsf{FC}})$*:*

1. *Repeat the following until* $\widetilde{\mathcal{P}}^{\mathsf{SR}}_{\mathsf{FC}}(\mathsf{pp}_{\mathsf{FC}}, \mathsf{ai}_{\mathsf{FC}})$ *decides to exit the loop:*
   (a) $\widetilde{\mathcal{P}}^{\mathsf{SR}}_{\mathsf{FC}}$ *outputs* $(\mathsf{cm}, \alpha, \beta, (\mathsf{pm}_j)_{j \in [i]}, (\eta_j)_{j \in [i]})$*, where* cm *is a commitment,* α *is a query in* $\mathbf{Q}$*,* β *is a corresponding answer,* $(\mathsf{pm}_j)_{j \in [i]}$ *are prover messages, and* $(\eta_j)_{j \in [i]}$ *are salt strings in* $\{0,1\}^{s_{\mathsf{FC}}}$*.*
   (b) *Set* $\mathsf{vm}_i \coloneqq \mathsf{rnd}_{\mathsf{FC},i}(\mathsf{cm}, \alpha, \beta, (\mathsf{pm}_j)_{j \in [i]}, (\eta_j)_{j \in [i]})$*.*
   (c) *Send* vm_i *to* $\widetilde{\mathcal{P}}^{\mathsf{SR}}_{\mathsf{FC}}$*.*
2. $\widetilde{\mathcal{P}}^{\mathsf{SR}}_{\mathsf{FC}}$ *outputs* $(\mathsf{cm}, ((\alpha_i, \beta_i, (\mathsf{pm}_{i,j})_{j \in [\mathsf{k}_{\mathsf{FC}}]}, (\eta_{i,j})_{j \in [\mathsf{k}_{\mathsf{FC}}]}))_{i \in [\mathsf{L}]})$*.*
3. *For every* $i \in [\mathsf{L}], j \in [\mathsf{k}_{\mathsf{FC}}]$*, set* $\mathsf{vm}_{i,j} \coloneqq \mathsf{rnd}_{\mathsf{FC},j}(\mathsf{cm}, \alpha_i, \beta_i, (\mathsf{pm}_{i,k})_{k \in [j]}, (\eta_{i,k})_{k \in [j]})$*.*
4. *Output* $(\mathsf{cm}, ((\alpha_i, \beta_i, ((\mathsf{pm}_{i,j}, \eta_{i,j}, \mathsf{vm}_{i,j}))_{j \in [\mathsf{k}_{\mathsf{FC}}]}))_{i \in [\mathsf{L}]})$*.*

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

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

We say that $\widetilde{\mathcal{P}}_{\mathsf{FC}}^{\mathsf{SR}}$ *is* m_{FC}*-move if* $\widetilde{\mathcal{P}}_{\mathsf{FC}}^{\mathsf{SR}}$ *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** $\epsilon_{\mathsf{FC}}^{\mathsf{SR}}$ *if for every security parameter* $\lambda \in \mathbb{N}$*, message length* $\ell \in \mathbb{N}$*, sample set size* $\mathsf{L} \in \mathbb{N}$*, auxiliary input distribution* $\mathcal{D}_{\mathsf{FC}}$*, salt size* $s_{\mathsf{FC}} \in \mathbb{N}$*, move budget* $\mathsf{m}_{\mathsf{FC}} \in \mathbb{N}$*, circuit size bound* $t_{\mathsf{FC}} \in \mathbb{N}$*, and* m_{FC}*-move* t_{FC}*-size circuit* $\widetilde{\mathcal{P}}_{\mathsf{FC}}^{\mathsf{SR}}$*,*

$$\Pr\left[\begin{array}{l|l} \forall\, i \in [\mathsf{L}] \ : \ \mathcal{V}_{\mathsf{FC}}(\mathsf{pp}_{\mathsf{FC}}, \mathsf{cm}, \alpha_i, \beta_i, ((\mathsf{pm}_{i,j}, \mathsf{vm}_{i,j}))_{j\in[\mathsf{k}_{\mathsf{FC}}]}) = 1 & \mathsf{pp}_{\mathsf{FC}} \leftarrow \mathsf{FC.Gen}(1^\lambda, \ell) \\ \wedge \nexists\, \Pi \text{ s.t. } \forall\, i \in [L], \alpha_i(\Pi) = \beta_i & \mathsf{ai}_{\mathsf{FC}} \leftarrow \mathcal{D}_{\mathsf{FC}} \\ & \mathsf{rnd}_{\mathsf{FC}} := (\mathsf{rnd}_{\mathsf{FC},i})_{i\in[\mathsf{k}_{\mathsf{FC}}]} \leftarrow \mathcal{U}((\mathsf{r}_{\mathsf{FC},i})_{i\in[\mathsf{k}_{\mathsf{FC}}]}) \\ & \left(\mathsf{cm}, \left(\left(\begin{array}{l} \alpha_i, \beta_i, \\ ((\mathsf{pm}_{i,j}, \eta_{i,j}, \mathsf{vm}_{i,j}))_{j\in[\mathsf{k}_{\mathsf{FC}}]} \end{array}\right)\right)_{i\in[\mathsf{L}]}\right) \\ & \quad \leftarrow \mathsf{FCSRGame}(s_{\mathsf{FC}}, \mathsf{rnd}_{\mathsf{FC}}, \widetilde{\mathcal{P}}_{\mathsf{FC}}^{\mathsf{SR}}, \mathsf{pp}_{\mathsf{FC}}, \mathsf{ai}_{\mathsf{FC}}) \end{array}\right]$$

$$\leq \epsilon_{\mathsf{FC}}^{\mathsf{SR}}(\lambda, \ell, \mathsf{L}, s_{\mathsf{FC}}, \mathsf{m}_{\mathsf{FC}}, t_{\mathsf{FC}}) \ .$$

3.3.3 Batched functional commitment schemes

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

$$\mathsf{Batch}[\mathbf{Q}, s] := \left\{\alpha' \colon \Sigma^\ell \to \mathbb{D}^s \;\middle|\; \exists\, \alpha_1, \ldots, \alpha_s \in \mathbf{Q} \text{ s.t. } \alpha'(\Pi) = (\alpha_i(\Pi))_{i\in[s]}\right\} \ .$$

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

- $\mathsf{bFC.Gen}(1^\lambda, \ell) \to \mathsf{pp}_{\mathsf{bFC}}$: On input a security parameter $\lambda \in \mathbb{N}$ and message length $\ell \in \mathbb{N}$, $\mathsf{bFC.Gen}$ samples public parameter $\mathsf{pp}_{\mathsf{bFC}}$.
- $\mathsf{bFC.Commit}(\mathsf{pp}_{\mathsf{bFC}}, \Pi) \to (\mathsf{cm}, \mathsf{aux})$: On input a public parameter $\mathsf{pp}_{\mathsf{bFC}}$ and a message vector $\Pi \in \Sigma^\ell$, $\mathsf{bFC.Commit}$ produces a commitment cm and the corresponding auxiliary state aux.
- $\mathsf{bFC.Open}(\mathsf{pp}_{\mathsf{bFC}}, \mathsf{aux}, \mathcal{Q}, \boldsymbol{\beta}) \to \mathsf{pf}$: On input a public parameter $\mathsf{pp}_{\mathsf{bFC}}$, an auxiliary state aux, a query set $\mathcal{Q} \subseteq \mathbf{Q}$ and a corresponding answer set $\boldsymbol{\beta} \in \mathbb{D}^{|\mathcal{Q}|}$, $\mathsf{bFC.Open}$ outputs an opening proof pf for the claim "$\forall \alpha \in \mathcal{Q}$, $\boldsymbol{\beta}^{(\alpha)} = \alpha(\Pi)$".
- $\mathsf{bFC.Check}(\mathsf{pp}_{\mathsf{bFC}}, \mathsf{cm}, \mathcal{Q}, \boldsymbol{\beta}, \mathsf{pf}) \to \{0, 1\}$: On input a public parameter $\mathsf{pp}_{\mathsf{bFC}}$, a commitment cm, a query set $\mathcal{Q} \subseteq \mathbf{Q}$, an answer set $\boldsymbol{\beta} \in \mathbb{D}^{|\mathcal{Q}|}$, and an opening proof pf, $\mathsf{bFC.Check}$ determines if pf is a valid proof for $\boldsymbol{\beta}$ being the answer of the query set $\mathcal{Q}$ on the vector committed in cm.

Definition 3.22 ((Batched) function binding)**.** $\mathsf{bFC} = (\mathsf{bFC.Gen}, \mathsf{bFC.Commit}, \mathsf{bFC.Open}, \mathsf{bFC.Check})$ *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* $s \in \mathbb{N}$*, sample set size* $\mathsf{L} \in \mathbb{N}$*, auxiliary input distribution* $\mathcal{D}_{\mathsf{bFC}}$*, adversary size bound* t_{bFC}*, and* t_{bFC}*-size circuit* A_{bFC}*,*

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

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:

- $\epsilon_{\mathsf{bFC}}(\cdot, \ell, s, \mathsf{L}, t_{\mathsf{bFC}})$ is non-increasing (larger security parameters decrease an adversary's success);
- $\epsilon_{\mathsf{bFC}}(\lambda, \cdot, s, \mathsf{L}, t_{\mathsf{bFC}})$ is non-decreasing (longer message vector decrease an adversary's success);
- $\epsilon_{\mathsf{bFC}}(\lambda, \ell, \cdot, \mathsf{L}, t_{\mathsf{bFC}})$ is non-decreasing (The solution space with a set of constraints is smaller than that of its subset);
- $\epsilon_{\mathsf{bFC}}(\lambda, \ell, s, \cdot, t_{\mathsf{bFC}})$ is non-decreasing (The solution space with a set of constraints is smaller than that of its subset); and
- $\epsilon_{\mathsf{bFC}}(\lambda, \ell, s, \mathsf{L}, \cdot)$ 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 $\epsilon_{\mathsf{bFC}}(\lambda, \ell_{\mathsf{max}}, s_{\mathsf{max}}, \mathsf{L}_{\mathsf{max}}, t_{\mathsf{bFC}})$, when $\ell_{\mathsf{max}} = \max_i \{\, \ell_i \,\}$, $s_{\mathsf{max}} = \max_j \{\, s_j \,\}$ and $\mathsf{L}_{\mathsf{max}} = \max_k \{\, \mathsf{L}_k \,\}$, with

$$\max_{i,j,k} \{\epsilon_{\mathsf{bFC}}(\lambda, \ell_i, s_j, \mathsf{L}_k, t_{\mathsf{bFC}})\} \ .$$

Interactive batched functional commitment schemes. An interactive batched FC scheme $\mathsf{bFC} = (\mathsf{bFC.Gen}, \mathsf{bFC.Commit}, \mathcal{P}_{\mathsf{bFC}}, \mathcal{V}_{\mathsf{bFC}})$ for a query class $\mathbf{Q} = \{\alpha \colon \Sigma^\ell \to \mathbb{D}\}$ is an interactive FC scheme for $\mathsf{Batch}[\mathbf{Q}, s]$. In particular, $\mathsf{bFC.Gen}$ and $\mathsf{bFC.Commit}$ 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 $\mathsf{pp}_{\mathsf{bFC}}$, an instance $(\mathsf{cm}, \mathcal{Q}, \boldsymbol{\beta})$ and an auxiliary state aux, the prover and verifier engage in a public-coin $\mathsf{k}_{\mathsf{bFC}}$-round interactive protocol:

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

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

- $\mathcal{P}_{\mathsf{bFC}}$'s j-th message.
  1. Compute the j-th FC prover message pm_j and auxiliary state:
  $$(\mathsf{pm}_j, \mathsf{aux}_j) \leftarrow \begin{cases} \mathcal{P}_{\mathsf{bFC}}(\mathsf{pp}_{\mathsf{bFC}}, \mathsf{aux}, \mathcal{Q}, \boldsymbol{\beta}) & \text{if } j = 1 \\ \mathcal{P}_{\mathsf{bFC}}(\mathsf{aux}_{j-1}, \mathsf{vm}_{j-1}) & \text{if } j > 1 \end{cases} .$$
  2. Send pm_j to $\mathcal{V}_{\mathsf{bFC}}$.

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

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

$\mathsf{bFCSRGame}(s_{\mathsf{bFC}}, \mathsf{rnd}_{\mathsf{bFC}}, \widetilde{\mathcal{P}}^{\mathsf{SR}}_{\mathsf{bFC}}, \mathsf{pp}_{\mathsf{bFC}}, \mathsf{ai}_{\mathsf{bFC}})$*:*

1. *Repeat the following until* $\mathcal{P}^{\mathsf{SR}}_{\mathsf{bFC}}(\mathsf{pp}_{\mathsf{bFC}}, \mathsf{ai}_{\mathsf{bFC}})$ *decides to exit the loop:*
   (a) $\widetilde{\mathcal{P}}^{\mathsf{SR}}_{\mathsf{bFC}}$ *outputs* $(\mathsf{cm}, \mathcal{Q}, \boldsymbol{\beta}, (\mathsf{pm}_j)_{j\in[i]}, (\eta_j)_{j\in[i]})$*, where* cm *is a commitment,* $\mathcal{Q}$ *is a query set from* $\mathbf{Q}$*,* $\boldsymbol{\beta}$ *is the claimed answer,* $(\mathsf{pm}_j)_{j\in[i]}$ *are prover messages, and* $(\eta_j)_{j\in[i]}$ *are salt strings in* $\{0,1\}^{s_{\mathsf{bFC}}}$*.*
   (b) *Set* $\mathsf{vm}_i \coloneqq \mathsf{rnd}_{\mathsf{bFC},i}(\mathsf{cm}, \mathcal{Q}, \boldsymbol{\beta}, (\mathsf{pm}_j)_{j\in[i]}, (\eta_j)_{j\in[i]})$*.*
   (c) *Send* vm_i *to* $\widetilde{\mathcal{P}}^{\mathsf{SR}}_{\mathsf{bFC}}$*.*
2. $\widetilde{\mathcal{P}}^{\mathsf{SR}}_{\mathsf{bFC}}$ *outputs* $(\mathsf{cm}, ((\mathcal{Q}_i, \boldsymbol{\beta}_i, (\mathsf{pm}_{i,j})_{j\in[\mathsf{k}_{\mathsf{bFC}}]}, (\eta_{i,j})_{j\in[\mathsf{k}_{\mathsf{bFC}}]}))_{i\in[\mathsf{L}]})$*.*
3. *For every* $i \in [\mathsf{L}], j \in [\mathsf{k}_{\mathsf{bFC}}]$*, set* $\mathsf{vm}_{i,j} \coloneqq \mathsf{rnd}_{\mathsf{bFC},j}(\mathsf{cm}, \mathcal{Q}_i, \boldsymbol{\beta}_i, (\mathsf{pm}_{i,k})_{k\in[j]}, (\eta_{i,k})_{k\in[j]})$*.*
4. *Output* $(\mathsf{cm}, ((\mathcal{Q}_i, \boldsymbol{\beta}_i, ((\mathsf{pm}_{i,j}, \eta_{i,j}, \mathsf{vm}_{i,j}))_{j\in[\mathsf{k}_{\mathsf{bFC}}]}))_{i\in[\mathsf{L}]})$*.*

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

$$(\mathsf{cm}, ((\mathcal{Q}_i, \boldsymbol{\beta}_i, ((\mathsf{pm}_{i,j}, \eta_{i,j}, \mathsf{vm}_{i,j}))_{j\in[\mathsf{k}_{\mathsf{bFC}}]}))_{i\in[\mathsf{L}]}) \xleftarrow{\mathsf{tr}^{\mathsf{SR}}_{\mathsf{bFC}}} \mathsf{bFCSRGame}(s_{\mathsf{bFC}}, \mathsf{rnd}_{\mathsf{bFC}}, \widetilde{\mathcal{P}}^{\mathsf{SR}}_{\mathsf{bFC}}, \mathsf{pp}_{\mathsf{bFC}}, \mathsf{ai}_{\mathsf{bFC}}) \ .$$

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

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

$$\Pr\left[\begin{array}{l|l} \left(\begin{array}{l} \forall\, i \in [\mathsf{L}] : \\ |\mathcal{Q}_i| = s \\ \wedge\, \mathcal{V}_{\mathsf{bFC}}(\mathsf{pp}_{\mathsf{bFC}}, \mathsf{cm}, \mathcal{Q}_i, \boldsymbol{\beta}_i, ((\mathsf{pm}_{i,j}, \mathsf{vm}_{i,j}))_{j\in[\mathsf{k}_{\mathsf{bFC}}]}) = 1 \end{array}\right) & \begin{array}{l} \mathsf{pp}_{\mathsf{bFC}} \leftarrow \mathsf{bFC.Gen}(1^\lambda, \ell) \\ \mathsf{ai}_{\mathsf{bFC}} \leftarrow \mathcal{D}_{\mathsf{bFC}} \\ \mathsf{rnd}_{\mathsf{bFC}} \coloneqq (\mathsf{rnd}_{\mathsf{bFC},i})_{i\in[\mathsf{k}_{\mathsf{bFC}}]} \leftarrow \mathcal{U}((\mathsf{r}_{\mathsf{bFC},i})_{i\in[\mathsf{k}_{\mathsf{bFC}}]}) \\ \left(\mathsf{cm}, \left(\left(\begin{array}{l} \mathcal{Q}_i, \boldsymbol{\beta}_i, \\ ((\mathsf{pm}_{i,j}, \eta_{i,j}, \mathsf{vm}_{i,j}))_{j\in[\mathsf{k}_{\mathsf{bFC}}]} \end{array}\right)\right)_{i\in[\mathsf{L}]}\right) \\ \quad \leftarrow \mathsf{bFCSRGame}(s_{\mathsf{bFC}}, \mathsf{rnd}_{\mathsf{bFC}}, \widetilde{\mathcal{P}}^{\mathsf{SR}}_{\mathsf{bFC}}, \mathsf{pp}_{\mathsf{bFC}}, \mathsf{ai}_{\mathsf{bFC}}) \end{array} \\ \wedge\, \nexists\, \Pi \text{ s.t. } \forall\, i \in [\mathsf{L}], \alpha \in \mathcal{Q}_i,\ \alpha(\Pi) = \boldsymbol{\beta}_i^{(\alpha)} & \end{array}\right]$$
$$\leq \epsilon^{\mathsf{SR}}_{\mathsf{bFC}}(\lambda, \ell, s, \mathsf{L}, s_{\mathsf{bFC}}, \mathsf{m}_{\mathsf{bFC}}, t_{\mathsf{bFC}}) \ .$$

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:

- $\mathsf{bFC.Gen}(1^\lambda, \ell)$: Output $\mathsf{pp}_{\mathsf{bFC}} \leftarrow \mathsf{FC.Gen}(1^\lambda, \ell)$.
- $\mathsf{bFC.Commit}(\mathsf{pp}_{\mathsf{bFC}}, \Pi)$: Output $(\mathsf{cm}, \mathsf{aux}) \leftarrow \mathsf{FC.Commit}(\mathsf{pp}_{\mathsf{bFC}}, \Pi)$.
- $\mathsf{bFC.Open}(\mathsf{pp}_{\mathsf{bFC}}, \mathsf{aux}, \mathcal{Q}, \boldsymbol{\beta})$:
  1. Parse $\mathcal{Q}$ as $(\alpha_i)_{i\in[|\mathcal{Q}|]}$ and $\boldsymbol{\beta}$ as $(\beta_i)_{i\in[|\mathcal{Q}|]}$
  2. For $i \in [|\mathcal{Q}|]$: compute $\mathsf{pf}_i \leftarrow \mathsf{FC.Open}(\mathsf{pp}_{\mathsf{bFC}}, \mathsf{aux}, \alpha_i, \beta_i)$.
  3. Output $(\mathsf{pf}_i)_{i\in[|\mathcal{Q}|]}$.
- $\mathsf{bFC.Check}(\mathsf{pp}_{\mathsf{bFC}}, \mathsf{cm}, \mathcal{Q}, \boldsymbol{\beta}, \mathsf{pf})$:
  1. Parse $\mathcal{Q}$ as $(\alpha_i)_{i\in[|\mathcal{Q}|]}$, $\boldsymbol{\beta}$ as $(\beta_i)_{i\in[|\mathcal{Q}|]}$ and pf as $(\mathsf{pf}_i)_{i\in[|\mathcal{Q}|]}$.
  2. For $i \in [|\mathcal{Q}|]$, compute $b_i \leftarrow \mathsf{FC.Check}(\mathsf{pp}_{\mathsf{bFC}}, \mathsf{cm}, \alpha_i, \beta_i, \mathsf{pf}_i)$.
  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 query class* $\mathbf{Q}$ *with function binding error* $\epsilon_{\mathsf{FC}} = \epsilon_{\mathsf{FC}}(\lambda, \ell, \mathsf{L}, t_{\mathsf{FC}})$*. Let* bFC *be defined as in Construction 3.26. Then,* bFC *has batched function binding error* $\epsilon_{\mathsf{bFC}} = \epsilon_{\mathsf{bFC}}(\lambda, \ell, s, \mathsf{L}, t_{\mathsf{bFC}})$ *such that*

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

where $t_{\mathsf{FC}} \leq t_{\mathsf{bFC}} + s \cdot \mathsf{L}$.

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

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

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

Moreover,

$$\Pr\left[\begin{array}{l|l} \forall i \in [\mathsf{L}] : |\mathcal{Q}_i| = s \ \wedge \ \mathsf{bFC.Check}(\mathsf{pp}_{\mathsf{bFC}}, \mathsf{cm}, \mathcal{Q}_i, \boldsymbol{\beta}_i, \mathsf{pf}_i) = 1 & \mathsf{pp}_{\mathsf{bFC}} \leftarrow \mathsf{bFC.Gen}(1^\lambda, \ell) \\ \wedge \nexists \Pi \text{ s.t. } \forall i \in [\mathsf{L}], \alpha \in \mathcal{Q}_i, \ \alpha(\Pi) = \boldsymbol{\beta}_i^{(\alpha)} & \mathsf{ai}_{\mathsf{bFC}} \leftarrow \mathcal{D}_{\mathsf{bFC}} \\ & (\mathsf{cm}, ((\mathcal{Q}_i, \boldsymbol{\beta}_i, \mathsf{pf}_i))_{i\in[\mathsf{L}]}) \leftarrow A_{\mathsf{bFC}}(\mathsf{pp}_{\mathsf{bFC}}, \mathsf{ai}_{\mathsf{bFC}}) \end{array}\right]$$

$$\leq \Pr\left[\begin{array}{l|l} \left(\begin{array}{l} \forall i \in [\mathsf{L}], j \in [s] : \\ \mathsf{FC.Check}(\mathsf{pp}_{\mathsf{FC}}, \mathsf{cm}, \alpha_{i,j}, \beta_{i,j}, \mathsf{pf}_{i,j}) = 1 \end{array}\right) & \mathsf{pp}_{\mathsf{FC}} \leftarrow \mathsf{FC.Gen}(1^\lambda, \ell) \\ \wedge \nexists \Pi \text{ s.t. } \forall i \in [\mathsf{L}], j \in [s], \ \alpha_{i,j}(\Pi) = \boldsymbol{\beta}_{i,j} & \mathsf{ai}_{\mathsf{FC}} \leftarrow \mathcal{D}_{\mathsf{FC}} \\ & (\mathsf{cm}, ((\alpha_{i,j}, \beta_{i,j}, \mathsf{pf}_{i,j}))_{i\in[\mathsf{L}],j\in[s]}) \leftarrow A_{\mathsf{FC}}(\mathsf{pp}_{\mathsf{FC}}, \mathsf{ai}_{\mathsf{FC}}) \end{array}\right]$$

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

□

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:

- $\mathsf{bFC.Gen}(1^\lambda, \ell)$: Output $\mathsf{pp}_{\mathsf{bFC}} \leftarrow \mathsf{FC.Gen}(1^\lambda, \ell)$.
- $\mathsf{bFC.Commit}(\mathsf{pp}_{\mathsf{bFC}}, \Pi)$: Output $(\mathsf{cm}, \mathsf{aux}) \leftarrow \mathsf{FC.Commit}(\mathsf{pp}_{\mathsf{bFC}}, \Pi)$.
- $\langle \mathcal{P}_{\mathsf{bFC}}(\mathsf{pp}_{\mathsf{bFC}}, \mathsf{aux}, \mathcal{Q}, \boldsymbol{\beta}), \mathcal{V}_{\mathsf{bFC}}(\mathsf{pp}_{\mathsf{bFC}}, \mathsf{cm}, \mathcal{Q}, \boldsymbol{\beta}) \rangle$:

1. Both $\mathcal{P}_{\mathsf{bFC}}$ and $\mathcal{V}_{\mathsf{bFC}}$ parse $\mathcal{Q}$ as $(\alpha_i)_{i\in[|\mathcal{Q}|]}$ and $\boldsymbol{\beta}$ as $(\beta_i)_{i\in[|\mathcal{Q}|]}$.
2. For $j \in [\mathsf{k}_{\mathsf{bFC}}]$:
   (a) $\mathcal{P}_{\mathsf{bFC}}$'s j-th message.
      i. For $i \in [|\mathcal{Q}|]$, compute
$$(\mathsf{pm}_{i,j}, \mathsf{aux}_{i,j}) \leftarrow \begin{cases} \mathcal{P}_{\mathsf{FC}}(\mathsf{pp}_{\mathsf{bFC}}, \mathsf{aux}, \alpha_i, \beta_i) & \text{if } j = 1 \\ \mathcal{P}_{\mathsf{FC}}(\mathsf{aux}_{i,j-1}, \mathsf{vm}_{i,j-1}) & \text{if } j > 1 \end{cases} .$$
      ii. Send $(\mathsf{pm}_{i,j})_{i\in[|\mathcal{Q}|]}$ to $\mathcal{V}_{\mathsf{bFC}}$.
   (b) $\mathcal{V}_{\mathsf{bFC}}$'s j-th message.
      i. For $i \in [|\mathcal{Q}|]$, sample $\mathsf{vm}_{i,j} \leftarrow \{0,1\}^{\mathsf{r}_{\mathsf{bFC},j}}$.
      ii. Send $(\mathsf{vm}_{i,j})_{i\in[|\mathcal{Q}|]}$ to $\mathcal{P}_{\mathsf{bFC}}$.
3. For $i \in [|\mathcal{Q}|]$, compute $b_i = \mathcal{V}_{\mathsf{FC}}(\mathsf{pp}_{\mathsf{bFC}}, \mathsf{cm}, \alpha_i, \beta_i, ((\mathsf{pm}_{i,j}, \mathsf{vm}_{i,j}))_{j\in[\mathsf{k}_{\mathsf{bFC}}]})$.
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* $\epsilon^{\mathsf{SR}}_{\mathsf{FC}} = \epsilon^{\mathsf{SR}}_{\mathsf{FC}}(\lambda, \ell, \mathsf{L}, s_{\mathsf{FC}}, \mathsf{m}_{\mathsf{FC}}, t_{\mathsf{FC}})$. *Let* bFC *be defined as in Construction 3.28. Then,* bFC *has batched state-restoration function binding error* $\epsilon^{\mathsf{SR}}_{\mathsf{bFC}} = \epsilon^{\mathsf{SR}}_{\mathsf{bFC}}(\lambda, \ell, s, \mathsf{L}, s_{\mathsf{bFC}}, \mathsf{m}_{\mathsf{bFC}}, t_{\mathsf{bFC}})$ *such that*

$$\epsilon^{\mathsf{SR}}_{\mathsf{bFC}}(\lambda, \ell, s, \mathsf{L}, s_{\mathsf{bFC}}, \mathsf{m}_{\mathsf{bFC}}, t_{\mathsf{bFC}}) \le \epsilon^{\mathsf{SR}}_{\mathsf{FC}}(\lambda, \ell, s \cdot \mathsf{L}, \frac{s_{\mathsf{bFC}}}{s}, s \cdot \mathsf{m}_{\mathsf{bFC}}, t_{\mathsf{FC}}) \ ,$$

where $t_{\mathsf{FC}} \le t_{\mathsf{bFC}} + s \cdot \mathsf{L}$.

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

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

Note that $\widetilde{\mathcal{P}}^{\mathsf{SR}}_{\mathsf{FC}}$ has size $t_{\mathsf{FC}} \le t_{\mathsf{bFC}} + s \cdot \mathsf{L}$, salt size $s_{\mathsf{FC}} \le \frac{s_{\mathsf{bFC}}}{s}$, and move budget $\mathsf{m}_{\mathsf{FC}} \le s \cdot \mathsf{m}_{\mathsf{bFC}}$.

Moreover,

$$\Pr\left[\begin{array}{l} \left(\begin{array}{l} \forall\, i \in [\mathsf{L}] : \\ |\mathcal{Q}_i| = s \\ \wedge\, \mathcal{V}_{\mathsf{bFC}}\left(\begin{array}{l} \mathsf{pp}_{\mathsf{bFC}}, \mathsf{cm}, \\ \mathcal{Q}_i, \boldsymbol{\beta}_i, \\ ((\mathsf{pm}_{i,j}, \mathsf{vm}_{i,j}))_{j\in[\mathsf{k}_{\mathsf{bFC}}]} \end{array}\right) = 1 \end{array}\right) \\ \wedge\, \nexists\, \Pi \text{ s.t. } \forall\, i \in [\mathsf{L}], \alpha \in \mathcal{Q}_i,\ \alpha(\Pi) = \boldsymbol{\beta}_i^{(\alpha)} \end{array} \middle| \begin{array}{l} \mathsf{pp}_{\mathsf{bFC}} \leftarrow \mathsf{bFC.Gen}(1^\lambda, \ell) \\ \mathsf{ai}_{\mathsf{bFC}} \leftarrow \mathcal{D}_{\mathsf{bFC}} \\ \mathsf{rnd}_{\mathsf{bFC}} \coloneqq (\mathsf{rnd}_{\mathsf{bFC},i})_{i\in[\mathsf{k}_{\mathsf{bFC}}]} \leftarrow \mathcal{U}((\mathsf{r}_{\mathsf{bFC},i})_{i\in[\mathsf{k}_{\mathsf{bFC}}]}) \\ \left(\mathsf{cm}, \left(\left(\begin{array}{l} \mathcal{Q}_i, \boldsymbol{\beta}_i, \\ ((\mathsf{pm}_{i,j}, \eta_{i,j}, \mathsf{vm}_{i,j}))_{j\in[\mathsf{k}_{\mathsf{bFC}}]} \end{array}\right)\right)_{i\in[\mathsf{L}]}\right) \\ \quad \leftarrow \mathsf{bFCSRGame}(s_{\mathsf{bFC}}, \mathsf{rnd}_{\mathsf{bFC}}, \widetilde{\mathcal{P}}^{\mathsf{SR}}_{\mathsf{bFC}}, \mathsf{pp}_{\mathsf{bFC}}, \mathsf{ai}_{\mathsf{bFC}}) \end{array}\right]$$

$$\leq \Pr\left[\begin{array}{l|l} \left(\begin{array}{l} \forall\, i \in [\mathsf{L}], u \in [s] : \\ \mathcal{V}_{\mathsf{FC}}\left(\begin{array}{l} \mathsf{pp}_{\mathsf{FC}}, \mathsf{cm}, \\ \alpha_{u,i}, \beta_{u,i}, \\ ((\mathsf{pm}_{u,i,j}, \mathsf{vm}_{u,i,j}))_{j\in[\mathsf{k}_{\mathsf{bFC}}]} \end{array}\right) = 1 \end{array}\right) & \begin{array}{l} \mathsf{pp}_{\mathsf{FC}} \leftarrow \mathsf{FC.Gen}(1^\lambda, \ell) \\ \mathsf{ai}_{\mathsf{FC}} \leftarrow \mathcal{D}_{\mathsf{FC}} \\ \mathsf{rnd}_{\mathsf{FC}} \coloneqq (\mathsf{rnd}_{\mathsf{FC},i})_{i\in[\mathsf{k}_{\mathsf{FC}}]} \leftarrow \mathcal{U}((\mathsf{r}_{\mathsf{FC},i})_{i\in[\mathsf{k}_{\mathsf{FC}}]}) \\ \left(\mathsf{cm}, \left(\left(\begin{array}{l} \alpha_{u,i}, \beta_{u,i}, \\ ((\mathsf{pm}_{u,i,j}, \eta_{u,i,j}, \mathsf{vm}_{u,i,j}))_{j\in[\mathsf{k}_{\mathsf{FC}}]} \end{array}\right)\right)_{i\in[\mathsf{L}],u\in[s]}\right) \\ \quad \leftarrow \mathsf{FCSRGame}(s_{\mathsf{FC}}, \mathsf{rnd}_{\mathsf{FC}}, \widetilde{\mathcal{P}}^{\mathsf{SR}}_{\mathsf{FC}}, \mathsf{pp}_{\mathsf{FC}}, \mathsf{ai}_{\mathsf{FC}}) \end{array} \\ \wedge \nexists\, \Pi \text{ s.t. } \forall\, i \in [\mathsf{L}], u \in [s],\ \alpha_{u,i}(\Pi) = \boldsymbol{\beta}_{u,i} & \end{array}\right]$$

$$\leq \epsilon^{\mathsf{SR}}_{\mathsf{FC}}(\lambda, \ell, s\cdot\mathsf{L}, \frac{s_{\mathsf{bFC}}}{s}, s\cdot\mathsf{m}_{\mathsf{bFC}}, t_{\mathsf{FC}}) \ .$$

□

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 $\mathcal{D}_{\mathsf{FC}}$, circuit size bound $t_{\mathsf{FC}} \in \mathbb{N}$, and t_{FC}-size circuit $\widetilde{\mathcal{P}}^{\mathsf{SR}}_{\mathsf{FC}}$,*

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

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

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

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

$$\epsilon^{\mathsf{SR}}_{\mathsf{FC}}(\lambda, \ell, \mathsf{L}, s_{\mathsf{FC}}, \mathsf{m}_{\mathsf{FC}}, t^{\mathsf{SR}}_{\mathsf{FC}}) \leq \mathsf{L}\cdot\kappa^{\mathsf{SR}}_{\mathsf{KS}}(\lambda, \ell, s_{\mathsf{FC}}, \mathsf{m}_{\mathsf{FC}}, t^{\mathsf{SR}}_{\mathsf{KS}}) + \epsilon_{\mathsf{B}}(\lambda, \ell, t_{\mathsf{B}}) \ ,$$

where $t^{\mathsf{SR}}_{\mathsf{KS}} \leq t^{\mathsf{SR}}_{\mathsf{FC}} + \mathsf{L}\ell$ and $t_{\mathsf{B}} \leq t^{\mathsf{SR}}_{\mathsf{FC}} + \mathsf{L}t_{\mathcal{E}_{\mathsf{SR}}} + \mathsf{L}(\ell + s_{\mathsf{FC}}\mathsf{m}_{\mathsf{FC}})$.

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

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

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

Consider the following adversary A_{B} against the binding property of FC. For each $i \in [\mathsf{L}]$, A_{B} simulates $\mathsf{FCSRGame}$ towards $(\widetilde{\mathcal{P}}^{\mathsf{SR}}_{\mathsf{FC}})^{(i)}$, and then runs the knowledge extractor $\mathcal{E}_{\mathsf{SR}}$ with rewinding access to $(\widetilde{\mathcal{P}}^{\mathsf{SR}}_{\mathsf{FC}})^{(i)}$, taking as input the instance $(\mathsf{cm}, \alpha_i, \beta_i)$, prover messages $(\mathsf{pm}_{i,j})_{j\in[\mathsf{k}_{\mathsf{FC}}]}$ and salts $(\eta_{i,j})_{j\in[\mathsf{k}_{\mathsf{FC}}]}$. The i-th extraction will yield a valid witness Π_i, except with probability $\kappa^{\mathsf{SR}}_{\mathsf{KS}}(\lambda, \ell, s_{\mathsf{FC}}, \mathsf{m}_{\mathsf{FC}}, t^{\mathsf{SR}}_{\mathsf{KS}})$. If all extractions succeeds, A_{B} searches for indices $i, j \in [\mathsf{L}]$ 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 $i \in [\mathsf{L}]$. If all L extractions succeed, this implies that there exist $i, j \in [\mathsf{L}]$ such that $\Pi_i \neq \Pi_j$, and A_{B} is successful. □

4 The Funky protocol

Consider the following two ingredients:

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

The construction of $\mathsf{ARG} := \mathsf{Funky}[\mathsf{FIOP}, \mathsf{bFC}]$ 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: $\mathsf{pp}_{\mathsf{bFC}} \leftarrow \mathsf{bFC.Gen}(1^\lambda, \ell_{\mathsf{max}}(n))$.
2. Set public parameter for the interactive argument: $\mathsf{pp} := \mathsf{pp}_{\mathsf{bFC}}$.
3. Output pp.

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

1. **$\mathcal{P}$'s commitments and $\mathcal{V}$'s query challenges.**
   For $i \in [\mathsf{k}_{\mathsf{FIOP}}]$:
   (a) $\mathcal{P}$'s i-th commitment.
      i. Compute the i-th FIOP string $\Pi_i \in \Sigma^{\ell_i}$ and auxiliary state:
$$(\Pi_i, \mathbf{aux}_i) \leftarrow \begin{cases} \mathbf{P}(\mathbb{x}, \mathbb{w}) & \text{if } i = 1 \\ \mathbf{P}(\mathbf{aux}_{i-1}, \rho_{i-1}) & \text{if } i > 1 \end{cases}.$$
      ii. Compute a functional commitment to the FIOP string: $(\mathsf{cm}_i, \mathsf{aux}_i) \leftarrow \mathsf{bFC.Commit}(\mathsf{pp}, \Pi_i)$.
      iii. Send cm_i to $\mathcal{V}$.
   (b) $\mathcal{V}$'s i-th query challenge.
      i. Sample the i-th FIOP verifier randomness $\rho_i \leftarrow \{0,1\}^{\mathsf{r}_{\mathsf{FIOP},i}}$.
      ii. Send ρ_i to $\mathcal{P}$.

2. **$\mathcal{P}$'s response.**
   (a) Run the FIOP verifier $\mathbf{V}^{\Pi_1, \ldots, \Pi_{\mathsf{k}_{\mathsf{FIOP}}}}(\mathbb{x}; \rho_1, \ldots, \rho_{\mathsf{k}_{\mathsf{FIOP}}})$ to deduce $\mathcal{Q}_1, \ldots, \mathcal{Q}_{\mathsf{k}_{\mathsf{FIOP}}}$, where $\mathcal{Q}_i \subseteq \mathbf{Q}_i$ is the query set of $\mathbf{V}$ to Π_i.
   (b) For every $i \in [\mathsf{k}_{\mathsf{FIOP}}]$, $\alpha \in \mathcal{Q}_i$, compute the evaluation $\beta_i^{(\alpha)} := \alpha(\Pi_i)$.
   (c) Set $\boldsymbol{\beta}_i := (\beta_i^{(\alpha)})_{\alpha \in \mathcal{Q}_i}$.
   (d) Send $((\mathcal{Q}_i, \boldsymbol{\beta}_i))_{i \in [\mathsf{k}_{\mathsf{FIOP}}]}$ to $\mathcal{V}$.

3. **Interaction for the opening proofs.**
   For every $j \in [\mathsf{k}_{\mathsf{bFC}}]$:

(a) $\mathcal{P}$'s j-th message.

i. For every $i \in [\mathsf{k}_{\mathsf{FIOP}}]$, compute the j-th FC prover message $\mathsf{pm}_{i,j}$ and auxiliary state:

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

ii. Send $(\mathsf{pm}_{i,j})_{i \in [\mathsf{k}_{\mathsf{FIOP}}]}$ to $\mathcal{V}$.

(b) $\mathcal{V}$'s j-th message.

i. For every $i \in [\mathsf{k}_{\mathsf{FIOP}}]$, sample the j-th FC verifier randomness $\mathsf{vm}_{i,j} \leftarrow \{0,1\}^{\mathsf{r}_{\mathsf{bFC},j}}$.

ii. Send $(\mathsf{vm}_{i,j})_{i \in [\mathsf{k}_{\mathsf{FIOP}}]}$ to $\mathcal{P}$.

Set $\mathsf{pf}_i \coloneqq ((\mathsf{pm}_{i,j}, \mathsf{vm}_{i,j}))_{j \in [\mathsf{k}_{\mathsf{bFC}}]}$.

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

Check that the following holds:

(a) $\mathbf{V}^{([\mathcal{Q}_i, \boldsymbol{\beta}_i])_{i \in [\mathsf{k}_{\mathsf{FIOP}}]}}(\mathbb{x}; (\rho_i)_{i \in [\mathsf{k}_{\mathsf{FIOP}}]}) = 1$;

(b) For every $i \in [\mathsf{k}_{\mathsf{FIOP}}]$, $\mathcal{V}_{\mathsf{bFC}}(\mathsf{pp}, \mathsf{cm}_i, \mathcal{Q}_i, \boldsymbol{\beta}_i, \mathsf{pf}_i) = 1$.

The protocol has $2\mathsf{k}_{\mathsf{FIOP}} + 1 + 2\mathsf{k}_{\mathsf{bFC}}$ messages: the first $2\mathsf{k}_{\mathsf{FIOP}} + 1$ simulate the FIOP, and the rest simulates the bFC. For notational simplicity, we view the $(2\mathsf{k}_{\mathsf{FIOP}} + 1)$-st message (Item 2d) of the prover and the $(2\mathsf{k}_{\mathsf{FIOP}} + 2)$-nd message (Item 3a for $j = 1$) as one single message. Hence, the protocol has $2\mathsf{k}_{\mathsf{FIOP}} + 2\mathsf{k}_{\mathsf{bFC}}$ in total and the prover $\mathcal{P}$ and the verifier $\mathcal{V}$ interact for $\mathsf{k}_{\mathsf{FIOP}} + \mathsf{k}_{\mathsf{bFC}}$ 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 $\mathsf{p}_{\mathsf{ARG}} \coloneqq |\mathsf{pp}_{\mathsf{bFC}}|$ bits;
- the round complexity $\mathsf{k} \coloneqq \mathsf{k}_{\mathsf{FIOP}} + \mathsf{k}_{\mathsf{bFC}}$;
- the prover-to-verifier communication complexity, in bits, is

$$\mathsf{pc} \coloneqq \sum_{i \in [\mathsf{k}_{\mathsf{FIOP}}]} |\mathsf{cm}_i| + \mathsf{q} \cdot (\log |\mathbf{Q}_i| + \log |\mathbb{D}|) + \sum_{i \in [\mathsf{k}_{\mathsf{FIOP}}]} \sum_{j \in [\mathsf{k}_{\mathsf{bFC}}]} \left|\mathsf{pm}_{i,j}\right| \ ,$$

moreover,

$$\begin{aligned} \mathsf{pc}_i &\coloneqq |\mathsf{cm}_i| \ \text{ for } i \in [\mathsf{k}_{\mathsf{FIOP}}] \ , \\ \mathsf{pc}_{\mathsf{k}_{\mathsf{FIOP}}+1} &\coloneqq \mathsf{q} \cdot (\log |\mathbf{Q}_i| + \log |\mathbb{D}|) + \sum_{i \in [\mathsf{k}_{\mathsf{FIOP}}]} \left|\mathsf{pm}_{i,1}\right| \ , \\ \mathsf{pc}_{\mathsf{k}_{\mathsf{FIOP}}+j} &\coloneqq \sum_{i \in [\mathsf{k}_{\mathsf{FIOP}}]} \left|\mathsf{pm}_{i,j}\right| \ \text{ for } 2 \leq j \leq \mathsf{k}_{\mathsf{bFC}} \ ; \end{aligned}$$

- the randomness complexity, in bits, is

$$\mathsf{r} \coloneqq \sum_{i \in [\mathsf{k}_{\mathsf{FIOP}}]} \mathsf{r}_{\mathsf{FIOP},i} + \mathsf{k}_{\mathsf{FIOP}} \cdot \sum_{j \in [\mathsf{k}_{\mathsf{bFC}}]} \mathsf{r}_{\mathsf{bFC},j} \ ,$$

moreover,

$$\begin{aligned} \mathsf{r}_i &\coloneqq \mathsf{r}_{\mathsf{FIOP},i} \text{ for } i \in [\mathsf{k}_{\mathsf{FIOP}}] \ , \\ \mathsf{r}_{\mathsf{k}_{\mathsf{FIOP}}+j} &\coloneqq \mathsf{k}_{\mathsf{FIOP}} \cdot \mathsf{r}_{\mathsf{bFC},j} \text{ for } j \in [\mathsf{k}_{\mathsf{bFC}}] \ ; \end{aligned}$$

- the argument generator running time is $t_{\mathcal{G}} \coloneqq t_{\mathsf{bFC.Gen}}$.
- the argument prover running time is $t_{\mathcal{P}} \coloneqq t_{\mathbf{P}} + \mathsf{k}_{\mathsf{FIOP}} \cdot t_{\mathsf{bFC.Commit}} + \mathsf{k}_{\mathsf{FIOP}} \cdot t_{\mathcal{P}_{\mathsf{bFC}}} + t_{\mathbf{V}}$;
- the argument verifier running time is $t_{\mathcal{V}} \coloneqq t_{\mathbf{V}} + \mathsf{k}_{\mathsf{FIOP}} \cdot t_{\mathcal{V}_{\mathsf{bFC}}}$.

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 $\mathsf{Solver}_{\mathbf{Q}}$ 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*

$$\cap_{i=1}^{n}\{\Pi \in \Sigma^{\ell} \colon \forall i \in [n], \alpha_i(\Pi) = \boldsymbol{\beta}_i\} \ .$$

We denote by $t_{\mathbf{Q}}(\ell, n)$ the running time of $\mathsf{Solver}_{\mathbf{Q}}$ 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*

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

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 $\mathcal{D}$,*

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

where $\mathsf{Sampler}$ *works as follows:*

$\mathsf{Sampler}(n, \mathcal{D})$*:*
1. *Set $S_0 \coloneqq \Sigma^{\ell}$.*
2. *For $i \in [n+1]$:*
   (a) *Sample $(\mathcal{Q}_i, \boldsymbol{\beta}_i) \leftarrow \mathcal{D}$.*
   (b) *Set $S_i \coloneqq S_{i-1} \cap \{\Pi \in \Sigma^{\ell} \colon \forall \alpha \in \mathcal{Q}_i, \alpha(\Pi) = \boldsymbol{\beta}_i^{(\alpha)}\}$.*
3. *Output $(S_1, \ldots, S_{n+1})$.*

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:

$$[S_{m+1} \neq \emptyset \wedge S_{m+1} \neq S_m \mid (S_1 = (\mathcal{Q}_1, \boldsymbol{\beta}_1), \ldots, S_{N+1} = (\mathcal{Q}_{N+1}, \boldsymbol{\beta}_{N+1})) \leftarrow \mathsf{Sampler}(N, \mathcal{D})] \ .$$

Note that $\chi_m = 1$ implies that $\mathcal{Q}_{m+1} \setminus \cup_{i=1}^{m} \mathcal{Q}_i \neq \emptyset$. Hence,

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

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

We conclude that

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

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

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

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

The running time of $\mathsf{Solver}_{\mathbf{Q}}$ 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 $\mathbf{Q}_{\mathsf{Lin}}$ of linear queries.

Lemma 5.5. *The query class $\mathbf{Q}_{\mathsf{Lin}}$ (Eq. 2) has the following:*
- *tail error $\epsilon_{\mathbf{Q}_{\mathsf{Lin}}}(\ell, k, N) \leq \frac{\ell}{N}$;*
- *solving time $t_{\mathbf{Q}_{\mathsf{Lin}}}(\ell, n) \leq n \cdot \ell^2$;*

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

$$
[S_{m+1} \neq \emptyset \wedge S_{m+1} \neq S_m \mid (S_1 = (\mathcal{Q}_1, \boldsymbol{\beta}_1), \ldots, S_{N+1} = (\mathcal{Q}_{N+1}, \boldsymbol{\beta}_{N+1})) \leftarrow \mathsf{Sampler}(N, \mathcal{D})] \ .
$$

For each $i \in [N+1]$, $(\mathcal{Q}_i, \boldsymbol{\beta}_i)$ defines a linear equation system. Note that $\chi_m = 1$ implies that $\mathcal{Q}_{m+1}$ lies outside the linear span of $\cup_{i=1}^{m} \mathcal{Q}_m$. 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,

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

We conclude that

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

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

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

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

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

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

□

5.3 Point queries

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

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

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

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

$$[S_{m+1} \neq \emptyset \wedge S_{m+1} \neq S_m \mid (S_1 = (\mathcal{Q}_1, \boldsymbol{\beta}_1), \ldots, S_{N+1} = (\mathcal{Q}_{N+1}, \boldsymbol{\beta}_{N+1})) \leftarrow \mathsf{Sampler}(N, \mathcal{D})] \ .$$

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,

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

We conclude that

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

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

$\mathsf{Solver}_{\mathbf{Q}_{\mathsf{Point}}}(((\alpha_i, \beta_i))_{i\in[n]})$:
1. Set $\Pi \coloneqq (\perp)^{\ell}$.
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 $\mathsf{Solver}_{\mathbf{Q}_{\mathsf{Point}}}$ can be upper-bounded by $n + \ell$. Hence,

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

□

5.4 Univariate polynomial evaluation queries

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

Lemma 5.7. *The query class $\mathbf{Q}_{\mathsf{UniPoly}}$ (Eq. 3) has the following:*
- *tail error $\epsilon_{\mathbf{Q}_{\mathsf{UniPoly}}}(\ell, k, N) \leq \frac{\ell}{N}$;*
- *solving time $t_{\mathbf{Q}_{\mathsf{UniPoly}}}(\ell, n) \leq n \cdot \ell$;*

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

$$[S_{m+1} \neq \emptyset \wedge S_{m+1} \neq S_m \mid (S_1 = (\mathcal{Q}_1, \boldsymbol{\beta}_1), \ldots, S_{N+1} = (\mathcal{Q}_{N+1}, \boldsymbol{\beta}_{N+1})) \leftarrow \mathsf{Sampler}(N, \mathcal{D})] \ .$$

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

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

Hence,

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

We conclude that

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

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

$\mathsf{Solver}_{\mathbf{Q}_{\mathsf{UniPoly}}}(((\alpha_i, \beta_i))_{i\in[n]})$:
1. If $n \leq \ell$, compute $\mathsf{Lagrange}(((\alpha_i, \beta_i))_{i\in[\ell]})$ and output its coefficient vector.
2. Compute $f \coloneqq \mathsf{Lagrange}(((\alpha_i, \beta_i))_{i\in[\ell]})$.

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 $\mathsf{Solver}_{\mathbf{Q}_{\mathsf{UniPoly}}}$ can be upper-bounded by $n \cdot \ell$. Hence,

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

□

5.5 Multivariate polynomial evaluation queries

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

Lemma 5.8. *The query class $\mathbf{Q}^{(m,\mathsf{D})}_{\mathsf{MultiPoly}}$ (Equation 4) has the following:*
- *tail error $\epsilon_{\mathbf{Q}^{(m,\mathsf{D})}_{\mathsf{MultiPoly}}}(\ell, k, N) \leq \frac{\ell}{N}$;*
- *solving time $t_{\mathbf{Q}^{(m,\mathsf{D})}_{\mathsf{MultiPoly}}}(\ell, n) \leq n \cdot \ell^2$;*

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

$$[S_{m+1} \neq \emptyset \wedge S_{m+1} \neq S_m \mid (S_1 = (\mathcal{Q}_1, \boldsymbol{\beta}_1), \ldots, S_{N+1} = (\mathcal{Q}_{N+1}, \boldsymbol{\beta}_{N+1})) \leftarrow \mathsf{Sampler}(N, \mathcal{D})] \ .$$

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

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

We conclude that

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

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

$\mathsf{Solver}_{\mathbf{Q}^{(m,\mathsf{D})}_{\mathsf{MultiPoly}}}(((\alpha_i, \beta_i))_{i\in[n]})$:
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 $\mathsf{Solver}_{\mathbf{Q}_{\mathsf{MultiPoly}}^{(m,\mathsf{D})}}$ can be upper-bounded by $n \cdot \ell^2$. Hence,

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

□

5.6 Structured polynomial evaluation queries

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

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

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

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

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

Proof. We compute the tail error and solving time for $\mathbf{Q}_{\mathsf{Struct}}[\mathbf{Q}, m, (\mathsf{h}_k)_{k\in[n]}]^{(m,n,\mathsf{D},(\mathsf{h}_i)_{i\in[m]})}$ 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:

$$[S_{m+1} \neq \emptyset \wedge S_{m+1} \neq S_m \mid (S_1 = (\mathcal{Q}_1, \boldsymbol{\beta}_1), \ldots, S_{N+1} = (\mathcal{Q}_{N+1}, \boldsymbol{\beta}_{N+1})) \leftarrow \mathsf{Sampler}(N, \mathcal{D})] \ .$$

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

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

Consequently, $\chi_j = 1$ implies that $\mathcal{Q}_{j+1}$ lies outside the linear span of $\cup_{i=1}^{j} \mathcal{Q}_j$. 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

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

We conclude that

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

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

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

1. Construct the linear equation system:

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

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 $\mathsf{Solver}_{\mathbf{Q}_{\mathsf{Struct}}[\mathbf{Q},m,(\mathsf{h}_k)_{k\in[n]}]^{(m,n,\mathsf{D},(\mathsf{h}_i)_{i\in[m]})}}$ can be upper-bounded by $u \cdot \ell^2$. Hence,

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

□

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

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

$$\mathbf{Q}_{\mathsf{Func}}^{(\mathsf{D})} \coloneqq \mathbb{F}^{\leq \mathsf{D}}[X_1, \ldots, X_\ell]$$

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

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

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

$$[S_{m+1} \neq \emptyset \wedge S_{m+1} \neq S_m \mid (S_1 = (\mathcal{Q}_1, \boldsymbol{\beta}_1), \ldots, S_{N+1} = (\mathcal{Q}_{N+1}, \boldsymbol{\beta}_{N+1})) \leftarrow \mathsf{Sampler}(N, \mathcal{D})] \ .$$

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

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

Hence

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

We conclude that

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

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

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

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

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

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

□

6 State-restoration security reduction

We prove a security reduction lemma for the state-restoration security of $\mathsf{ARG} := \mathsf{Funky}[\mathsf{FIOP}, \mathsf{bFC}]$.

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

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

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

$$\big(\mathbb{x}, \big((\mathsf{cm}_i, \sigma_i)\big)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}, \big((\mathcal{Q}_i, \boldsymbol{\beta}_i)\big)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}, ((\mathsf{pm}_{i,j}, \eta_{i,j}))_{i\in[\mathsf{k}_{\mathsf{FIOP}}],j\in[\mathsf{k}_{\mathsf{bFC}}]}\big)$$

such that all its *partial moves* are included in $\mathsf{tr}^{\mathsf{SR}}$, which can be ensured by increasing the move budget from m to $\hat{\mathsf{m}} := \mathsf{m} + \mathsf{k}_{\mathsf{FIOP}} + \mathsf{k}_{\mathsf{bFC}}$. Specifically, the following moves are included in $\mathsf{tr}^{\mathsf{SR}}$:
- for every $i \in [\mathsf{k}_{\mathsf{FIOP}}]$, $(\mathbb{x}, ((\mathsf{cm}_j, \sigma_j))_{j\in[i]})$;
- for every $i \in [\mathsf{k}_{\mathsf{bFC}}]$, $(\mathbb{x}, ((\mathsf{cm}_j, \sigma_j))_{j\in[\mathsf{k}_{\mathsf{FIOP}}]}, \big((\mathcal{Q}_j, \boldsymbol{\beta}_j)\big)_{j\in[\mathsf{k}_{\mathsf{FIOP}}]}, ((\mathsf{pm}_{j,u}, \eta_{j,u}))_{j\in[\mathsf{k}_{\mathsf{FIOP}}],u\in[i]})$.

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

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

$$\Pr\left[\begin{array}{l} \mathbf{V}^{(\widetilde{\Pi}_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}}(\mathbb{x}; (\rho_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}) \neq 1 \\ \wedge\, \mathbf{V}^{([\mathcal{Q}_i, \boldsymbol{\beta}_i])_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}}(\mathbb{x}; (\rho_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}) = 1 \\ \wedge \left(\begin{array}{l} \forall\, i \in [\mathsf{k}_{\mathsf{FIOP}}] : \\ \mathcal{V}_{\mathsf{bFC}}\left(\begin{array}{l} \mathsf{pp}, \mathsf{cm}_i, \mathcal{Q}_i, \boldsymbol{\beta}_i, \\ ((\mathsf{pm}_{i,j}, \mathsf{vm}_{i,j}))_{j\in[\mathsf{k}_{\mathsf{bFC}}]} \end{array}\right) = 1 \end{array}\right) \end{array} \;\middle|\; \begin{array}{l} \mathsf{pp} \leftarrow \mathcal{G}(1^\lambda, n) \\ \mathsf{ai} \leftarrow \mathcal{D} \\ \mathsf{rnd} := (\mathsf{rnd}_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}+\mathsf{k}_{\mathsf{bFC}}]} \leftarrow \mathcal{U}((\mathsf{r}_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}+\mathsf{k}_{\mathsf{bFC}}]}) \\ \left(\begin{array}{l} \mathbb{x}, \big((\mathsf{cm}_i, \sigma_i)\big)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}, \big((\mathcal{Q}_i, \boldsymbol{\beta}_i)\big)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}, \\ ((\mathsf{pm}_{i,j}, \eta_{i,j}))_{i\in[\mathsf{k}_{\mathsf{FIOP}}],j\in[\mathsf{k}_{\mathsf{bFC}}]}, \\ (\rho_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}, (\mathsf{vm}_{i,j})_{i\in[\mathsf{k}_{\mathsf{FIOP}}],j\in[\mathsf{k}_{\mathsf{bFC}}]} \end{array}\right) \\ \quad \xleftarrow{\mathsf{tr}^{\mathsf{SR}}} \mathsf{SRGame}(s, \mathsf{rnd}, \widetilde{\mathcal{P}}^{\mathsf{SR}}, \mathsf{pp}, \mathsf{ai}) \\ \textit{Parse } \mathsf{tr}^{\mathsf{SR}} \textit{ as } ((\mathsf{mv}_a, \mathsf{res}_a))_{a\in[\hat{\mathsf{m}}]} \\ \textit{For every } a \in [\hat{\mathsf{m}}] : \\ \quad \big(\widetilde{\Pi}_i^{(a)}\big)_{i\in[\mathsf{round}(\mathsf{mv}_a)]} \leftarrow \mathfrak{R}_{\mathsf{m}}^{\widetilde{\mathcal{P}}^{\mathsf{SR}}(\mathsf{pp},\mathsf{ai})}(a, (\mathsf{res}_i)_{i\in[a-1]}) \\ (\widetilde{\Pi}_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]} \leftarrow \mathfrak{R}\left(\begin{array}{l} \mathsf{tr}^{\mathsf{SR}}, (\mathbb{x}, ((\mathsf{cm}_i, \sigma_i))_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}), \\ (\widetilde{\Pi}_i^{(a)})_{a\in[\hat{\mathsf{m}}],i\in[\mathsf{round}(\mathsf{mv}_a)]} \end{array}\right) \end{array}\right]$$

$$\leq \sum_{i\in[\mathsf{k}_{\mathsf{FIOP}}]} \left(\epsilon^{\mathsf{SR}}_{\mathsf{bFC}}(\lambda, \ell_i, \mathsf{q}_i, \mathsf{N}+1, s_{\mathsf{bFC}}, \mathsf{m}_{\mathsf{bFC}}, t_{\mathsf{bFC}}) + \epsilon_{\mathbf{Q}_i}(\ell_i, \mathsf{q}_i, \mathsf{N})\right) \;,$$

where
- $s_{\mathsf{bFC}} \leq s + \log((\mathsf{k}_{\mathsf{FIOP}} \cdot \mathsf{N} + 1) \cdot (\mathsf{m} + \mathsf{k}_{\mathsf{bFC}}))$*;*
- $\mathsf{m}_{\mathsf{bFC}} \leq (\mathsf{k}_{\mathsf{FIOP}} \cdot \mathsf{N} + 1) \cdot (\mathsf{m} + \mathsf{k}_{\mathsf{bFC}})$*; and*
- $t_{\mathsf{bFC}} \leq O(\mathsf{N} \cdot (t_{\mathsf{ARG}} + \hat{\mathsf{m}} + t_{\mathcal{V}_{\mathsf{bFC}}})) + t_{\mathbf{Q}}(\ell_{\mathsf{max}}, \mathsf{N} \cdot \mathsf{q}_{\mathsf{max}})$.

6.1 Construction of the security reductors

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

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

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

1. Simulate SRGame until $\widetilde{\mathcal{P}}^{\mathsf{SR}}$ makes the a-th move mv_a, answering the first $a-1$ moves with $(\mathsf{res}_i)_{i\in[a-1]}$.
2. If $\mathsf{round}(\mathsf{mv}_a) > \mathsf{k}_{\mathsf{FIOP}}$, output $\perp$; otherwise parse mv_a as $(\mathbb{x}^*, ((\mathsf{cm}_i^*, \sigma_i^*))_{i\in[\mathsf{round}(\mathsf{mv}_a)]})$.
3. For every $i \in [\mathsf{round}(\mathsf{mv}_a)]$, initialize a set of query-answer pairs $K_i^{(a)} := \emptyset$.
4. *Rewind to obtain query-answer pairs.* Sample $\mathsf{N}_a \leftarrow [\mathsf{N}]$ and repeat the following N_a times.
   (a) For every $i \in [\mathsf{k}]$, initialize an empty partial function $\mathsf{rnd}_i \colon \{0,1\}^* \to \{0,1\}^{\mathsf{r}_i}$ where
   $$\mathsf{rnd}_i(\mathsf{mv}) := \begin{cases} \mathsf{res}_v & \text{if } \mathsf{mv} \text{ is the } v\text{-th move of the simulation with } v \in [a-1] \text{ and } \mathsf{round}(\mathsf{mv}) = i \\ \perp & \text{otherwise} \end{cases} .$$
   (b) Continue simulating SRGame using $(\mathsf{rnd}_i)_{i\in[\mathsf{k}]}$ to obtain the final output
   $$\left(\mathbb{x}, \left((\mathsf{cm}_i, \sigma_i)\right)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}, \left((\mathcal{Q}_i, \boldsymbol{\beta}_i)\right)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}, ((\mathsf{pm}_{i,j}, \eta_{i,j}))_{i\in[\mathsf{k}_{\mathsf{FIOP}}], j\in[\mathsf{k}_{\mathsf{bFC}}]}\right) .$$
   When $\widetilde{\mathcal{P}}^{\mathsf{SR}}$ makes a move mv for round $i \in [\mathsf{k}]$: if $\mathsf{rnd}_i(\mathsf{mv}) = \perp$ then sample $\rho \in \{0,1\}^{\mathsf{r}_i}$ and set $\mathsf{rnd}_i(\mathsf{mv}) := \rho$; either way, answer mv with $\mathsf{rnd}_i(\mathsf{mv})$.
   (c) For every $j \in [\mathsf{k}_{\mathsf{bFC}}]$, let $(\mathsf{vm}_{i,j})_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}$ be the response of SRGame for the move
   $$\left(\mathbb{x}, \left((\mathsf{cm}_i, \sigma_i)\right)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}, \left((\mathcal{Q}_i, \boldsymbol{\beta}_i)\right)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}, ((\mathsf{pm}_{i,u}, \eta_{i,u}))_{i\in[\mathsf{k}_{\mathsf{FIOP}}], u\in[j]}\right) .$$
   (d) For every $i \in [\mathsf{round}(\mathsf{mv}_a)]$: if $\mathcal{V}_{\mathsf{bFC}}(\mathsf{pp}, \mathsf{cm}_i^*, \mathcal{Q}_i, \boldsymbol{\beta}_i, ((\mathsf{pm}_{i,j}, \mathsf{vm}_{i,j}))_{j\in[\mathsf{k}_{\mathsf{bFC}}]}) = 1$, add $(\mathcal{Q}_i, \boldsymbol{\beta}_i)$ to $K_i^{(a)}$.
5. *Solve for the FIOP strings.* For every $i \in [\mathsf{round}(\mathsf{mv}_a)]$:
   (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 $\widetilde{\Pi}_i^{(a)} \leftarrow \mathsf{Solver}_{\mathbf{Q}_i}(((\alpha_j, \beta_j))_{j\in[u]})$.
6. Output $(\widetilde{\Pi}_i^{(a)})_{i\in[\mathsf{round}(\mathsf{mv}_a)]}$.

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

$$\begin{aligned} t_{\mathfrak{R}_{\mathsf{m}}} &\le \mathsf{N} \cdot (t_{\mathsf{ARG}} + \mathsf{k}_{\mathsf{FIOP}} \cdot t_{\mathcal{V}_{\mathsf{bFC}}}) + \sum_{i\in[\mathsf{k}_{\mathsf{FIOP}}]} t_{\mathbf{Q}}(\ell_i, \mathsf{N} \cdot \mathsf{q}_i) \\ &\le \mathsf{N} \cdot (t_{\mathsf{ARG}} + \mathsf{k}_{\mathsf{FIOP}} \cdot t_{\mathcal{V}_{\mathsf{bFC}}}) + \mathsf{k}_{\mathsf{FIOP}} \cdot t_{\mathbf{Q}}(\ell_{\max}, \mathsf{N} \cdot \mathsf{q}_{\max}) . \end{aligned}$$

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

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

1. For every $i \in [\mathsf{k}_{\mathsf{FIOP}}]$:
   (a) Let a be the index of the first move mv in $\mathsf{tr}^{\mathsf{SR}}$ such that $\mathsf{round}(\mathsf{mv}) \le \mathsf{k}_{\mathsf{FIOP}}$ and mv is of the form $\big(\mathbb{x}, \big((\mathsf{cm}_j, \sigma_j)\big)_{j\in[i]}, \big((\mathsf{cm}'_j, \sigma'_j)\big)_{j\in[u]\setminus[i]}\big)$ for some $u \ge i$. Set $\widetilde{\Pi}_i \coloneqq \widetilde{\Pi}_i^{(a)}$.
   (b) If there is no such move, set $\widetilde{\Pi}_i \coloneqq \perp$.
2. Output $(\widetilde{\Pi}_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}$.

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

$$t_{\mathfrak{R}} \le \mathsf{k}_{\mathsf{FIOP}} \cdot \hat{\mathsf{m}} \le \mathsf{k}_{\mathsf{FIOP}} \cdot (\mathsf{m} + \mathsf{k}_{\mathsf{FIOP}} + \mathsf{k}_{\mathsf{bFC}}) \ .$$

6.2 Proof of Lemma 6.1

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

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

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

$$\left[\begin{array}{l}
\mathbf{V}^{(\widetilde{\Pi}_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}}(\mathbb{x}; (\rho_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}) \ne 1 \\
\wedge\, \mathbf{V}^{([\mathcal{Q}_i,\boldsymbol{\beta}_i])_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}}(\mathbb{x}; (\rho_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}) = 1 \\
\wedge \left(\begin{array}{l} \forall\, i \in [\mathsf{k}_{\mathsf{FIOP}}] : \\ \mathcal{V}_{\mathsf{bFC}}\left(\begin{array}{l} \mathsf{pp}, \mathsf{cm}_i, \mathcal{Q}_i, \boldsymbol{\beta}_i, \\ ((\mathsf{pm}_{i,j}, \mathsf{vm}_{i,j}))_{j\in[\mathsf{k}_{\mathsf{bFC}}]} \end{array}\right) = 1 \end{array}\right)
\end{array}\right] .$$

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

1. There exits $i \in [\mathsf{k}_{\mathsf{FIOP}}]$ such that $S_i^* = \emptyset$.
2. There exits $i \in [\mathsf{k}_{\mathsf{FIOP}}]$ such that $S_i^* \ne \emptyset$ and $S_i^* \ne S_i$.

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

$$\Pr\left[\ \exists\, i \in [\mathsf{k}_{\mathsf{FIOP}}], \left(\begin{array}{l} S_i^* = \emptyset \\ \wedge\, \mathcal{V}_{\mathsf{bFC}}\left(\begin{array}{l} \mathsf{pp}, \mathsf{cm}_i, \mathcal{Q}_i, \boldsymbol{\beta}_i, \\ ((\mathsf{pm}_{i,j}, \mathsf{vm}_{i,j}))_{j\in[\mathsf{k}_{\mathsf{bFC}}]} \end{array}\right) = 1 \end{array}\right)\ \right]\ ; \tag{10}$$

and

$$\Pr\left[\ \exists\, i \in [\mathsf{k}_{\mathsf{FIOP}}], \left(\begin{array}{l} S_i^* \neq \emptyset \\ \wedge\, S_i^* \neq S_i \\ \wedge\, \mathcal{V}_{\mathsf{bFC}}\left(\begin{array}{l} \mathsf{pp}, \mathsf{cm}_i, \mathcal{Q}_i, \boldsymbol{\beta}_i, \\ ((\mathsf{pm}_{i,j}, \mathsf{vm}_{i,j}))_{j\in[\mathsf{k}_{\mathsf{bFC}}]} \end{array}\right) = 1 \end{array}\right)\ \right]\ . \tag{11}$$

6.2.1 Bound for Equation 10 from state-restoration function binding

We prove that the probability in Equation 10 can be bounded by the state-restoration function binding error of bFC. First, we construct FC state-restoration adversaries $(\widetilde{\mathcal{P}}^{\mathsf{SR}}_{\mathsf{bFC},i})_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}$ using the argument state-restoration adversary $\widetilde{\mathcal{P}}^{\mathsf{SR}}$.

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

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

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

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

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

$$(\mathbb{x}', ((\mathsf{cm}'_u, \sigma'_u))_{u\in[\mathsf{k}_{\mathsf{FIOP}}]}, \left((\mathcal{Q}'_u, \boldsymbol{\beta}'_u)\right)_{u\in[\mathsf{k}_{\mathsf{FIOP}}]}, ((\mathsf{pm}'_{u,j}, \eta'_{u,j}))_{u\in[\mathsf{k}_{\mathsf{FIOP}}], j\in[c]}) \text{ for some } c \in [\mathsf{k}_{\mathsf{bFC}}]\ :$$

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

$$\left(\mathbb{x}, ((\mathsf{cm}_u, \sigma_u))_{u\in[\mathsf{k}_{\mathsf{FIOP}}]}, ((\mathcal{Q}_u, \boldsymbol{\beta}_u))_{u\in[\mathsf{k}_{\mathsf{FIOP}}]}, ((\mathsf{pm}_{u,j}, \eta_{u,j}, \mathsf{vm}_{u,j}))_{u\in[\mathsf{k}_{\mathsf{FIOP}}], j\in[\mathsf{k}_{\mathsf{bFC}}]}\right)\ .$$

5. Let a be the index of the first move mv_a in $\mathsf{tr}^{\mathsf{SR}}$ such that $\mathsf{round}(\mathsf{mv}_a) \le \mathsf{k}_{\mathsf{FIOP}}$ and mv_a is of the form $\left(\mathbb{x}, \left((\mathsf{cm}_j, \sigma_j)\right)_{j\in[i]}, \left((\mathsf{cm}'_j, \sigma'_j)\right)_{j\in[u]\setminus[i]}\right)$ for some $u \ge i$.
6. Simulate SRGame until $\widetilde{\mathcal{P}}^{\mathsf{SR}}$ outputs the a-th move by answering the first $a-1$ moves with $(\mathsf{res}_u)_{u\in[a-1]}$ obtained from $\mathsf{tr}^{\mathsf{SR}}$.
7. Initialize $K_i := \emptyset$.
8. *Rewind to obtain query-answer pairs.* Sample $\mathsf{N}_i \leftarrow [\mathsf{N}]$ and repeat the following for N_i times.
   (a) Continue the simulation of SRGame as follows and obtain the final output
   $$\left(\mathbb{x}^*, \left((\mathsf{cm}^*_u, \sigma^*_u)\right)_{u\in[\mathsf{k}_{\mathsf{FIOP}}]}, \left((\mathcal{Q}^*_u, \boldsymbol{\beta}^*_u)\right)_{u\in[\mathsf{k}_{\mathsf{FIOP}}]}, ((\mathsf{pm}^*_{u,j}, \eta^*_{u,j}))_{u\in[\mathsf{k}_{\mathsf{FIOP}}],j\in[\mathsf{k}_{\mathsf{bFC}}]}\right) .$$
      i. For every $u \in [\mathsf{k}]$, reinitialize the function $\mathsf{rnd}_u \colon \{0,1\}^* \to \{0,1\}^{\mathsf{r}_u}$ where
      $$\mathsf{rnd}_u(\mathsf{mv}) := \begin{cases} \mathsf{res}_v & \text{if } \mathsf{mv} \text{ is the } v\text{-th move of the simulation with } v \in [a-1] \text{ and } \mathsf{round}(\mathsf{mv}) = u \\ \perp & \text{otherwise} \end{cases} .$$
      ii. When $\widetilde{\mathcal{P}}^{\mathsf{SR}}$ outputs a move of the form $\mathsf{mv} := (\mathbb{x}', ((\mathsf{cm}'_j, \sigma'_j))_{j\in[u]})$ for some $u \in [\mathsf{k}_{\mathsf{FIOP}}]$: if $\mathsf{rnd}_u(\mathsf{mv}) = \perp$ then sample $\rho \in \{0,1\}^{\mathsf{r}_u}$ and set $\mathsf{rnd}_u(\mathsf{mv}) := \rho$; either way, answer mv with $\mathsf{rnd}_u(\mathsf{mv})$.
      iii. When $\widetilde{\mathcal{P}}^{\mathsf{SR}}$ outputs a move mv of the form
      $$(\mathbb{x}', ((\mathsf{cm}'_u, \sigma'_u))_{u\in[\mathsf{k}_{\mathsf{FIOP}}]}, \left((\mathcal{Q}'_u, \boldsymbol{\beta}'_u)\right)_{u\in[\mathsf{k}_{\mathsf{FIOP}}]}, ((\mathsf{pm}'_{u,j}, \eta'_{u,j}))_{u\in[\mathsf{k}_{\mathsf{FIOP}}],j\in[c]}) \text{ for some } c \in [\mathsf{k}_{\mathsf{bFC}}] .$$
      – If $\mathsf{rnd}_{\mathsf{k}_{\mathsf{FIOP}}+c}(\mathsf{mv}) = \perp$ then set $\mathsf{rnd}_{\mathsf{k}_{\mathsf{FIOP}}+c}(\mathsf{mv}) := (\mathsf{vm}'_{u,c})_{u\in[\mathsf{k}_{\mathsf{FIOP}}]}$ where, for every $u \in [\mathsf{k}_{\mathsf{FIOP}}]$, $\mathsf{vm}'_{u,c}$ is obtained as follows:
        * Set $\mathsf{mv}_{\mathsf{bFC}} := (\mathsf{cm}'_u, \mathcal{Q}'_u, \boldsymbol{\beta}'_u, (\mathsf{pm}'_{u,j})_{j\in[c]}, (\eta'_{u,j})_{j\in[c]})$.
        * If $\mathsf{mv}_{\mathsf{bFC}} \in I$, modify $\eta'_{u,c}$ to obtain a new move $\mathsf{mv}'_{\mathsf{bFC}}$ not in I; add $\mathsf{mv}_{\mathsf{bFC}}$ to I.
        * Make the move $\mathsf{mv}_{\mathsf{bFC}}$ in $\mathsf{FCSRGame}$ to get the answer $\mathsf{vm}'_{u,c}$.
      – Answer mv with $\mathsf{rnd}_{\mathsf{k}_{\mathsf{FIOP}}+c}(\mathsf{mv})$.
   (b) For every $c \in [\mathsf{k}_{\mathsf{bFC}}]$, let $(\mathsf{vm}^*_{u,c})_{u\in[\mathsf{k}_{\mathsf{FIOP}}]}$ be the response of SRGame for move
   $$\left(\mathbb{x}^*, \left((\mathsf{cm}^*_u, \sigma^*_u)\right)_{u\in[\mathsf{k}_{\mathsf{FIOP}}]}, \left((\mathcal{Q}^*_u, \boldsymbol{\beta}^*_u)\right)_{u\in[\mathsf{k}_{\mathsf{FIOP}}]}, ((\mathsf{pm}^*_{u,j}, \eta^*_{u,j}))_{u\in[\mathsf{k}_{\mathsf{FIOP}}],j\in[c]}\right) .$$
   (c) If $\mathcal{V}_{\mathsf{bFC}}(\mathsf{pp}, \mathsf{cm}_i, \mathcal{Q}^*_i, \boldsymbol{\beta}^*_i, ((\mathsf{pm}^*_{i,c}, \mathsf{vm}^*_{i,c}))_{c\in[\mathsf{k}_{\mathsf{bFC}}]}) = 1$, add $(\mathcal{Q}^*_i, \boldsymbol{\beta}^*_i, ((\mathsf{pm}^*_{i,c}, \eta^*_{i,c}))_{c\in[\mathsf{k}_{\mathsf{bFC}}]})$ to K_i.
9. *Solve for the FIOP strings.* Run $\widetilde{\Pi}_i \leftarrow \mathsf{Solver}_{\mathbf{Q}_i}(((\alpha_j, \beta_j))_{j\in[u]})$ where $((\alpha_j, \beta_j))_{j\in[u]}$ are all query-answer pairs in K_i.
10. If $\widetilde{\Pi}_i \ne \perp$, output the "dummy" tuple $(\mathsf{cm}_1, \perp)$.
11. If $\widetilde{\Pi}_i = \perp$, add $(\mathcal{Q}_i, \boldsymbol{\beta}_i, ((\mathsf{pm}_{i,j}, \eta_{i,c}))_{c\in[\mathsf{k}_{\mathsf{bFC}}]})$ to K_i and output (cm_i, K_i).

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

- *Salt size s_{bFC} of $\widetilde{\mathcal{P}}^{\mathsf{SR}}_{\mathsf{bFC},i}$.* The algorithm $\widetilde{\mathcal{P}}^{\mathsf{SR}}_{\mathsf{bFC},i}$ relies on having enough distinct salts, specifically at most $(\mathsf{k}_{\mathsf{FIOP}} \cdot \mathsf{N} + 1) \cdot (\mathsf{m} + \mathsf{k}_{\mathsf{bFC}})$ 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 most $s + \log((\mathsf{k}_{\mathsf{FIOP}} \cdot \mathsf{N} + 1) \cdot (\mathsf{m} + \mathsf{k}_{\mathsf{bFC}}))$.

- *Move budget* $\mathsf{m}_{\mathsf{bFC}}$. In Step 3c and Step 8(a)iii, whenever $\widetilde{\mathcal{P}}^{\mathsf{SR}}$ makes a move of the form
  $$(\mathbb{x}', ((\mathsf{cm}'_u, \sigma'_u))_{u\in[\mathsf{k}_{\mathsf{FIOP}}]}, ((\mathcal{Q}'_u, \beta'_u))_{u\in[\mathsf{k}_{\mathsf{FIOP}}]}, ((\mathsf{pm}'_{u,j}, \eta'_{u,j}))_{u\in[\mathsf{k}_{\mathsf{FIOP}}],j\in[c]})$$
  for some $c \in [\mathsf{k}_{\mathsf{bFC}}]$, $\widetilde{\mathcal{P}}^{\mathsf{SR}}_{\mathsf{bFC},i}$ may query $\mathsf{bFCSRGame}$ for the answer. Hence the move budget $\mathsf{m}_{\mathsf{bFC}}$ is at most $(\mathsf{k}_{\mathsf{FIOP}} \cdot \mathsf{N} + 1) \cdot (\mathsf{m} + \mathsf{k}_{\mathsf{bFC}})$.

- *Running time* t_{bFC}.
  - Step 3 takes time at most $t_{\mathsf{ARG}} + \hat{\mathsf{m}}$.
  - Step 5 takes time $\hat{\mathsf{m}}$.
  - Step 8 takes time $\mathsf{N} \cdot (t_{\mathsf{ARG}} + \hat{\mathsf{m}} + t_{\mathcal{V}_{\mathsf{bFC}}})$.
  - Step 9 takes time $t_{\mathbf{Q}}(\ell_i, \mathsf{N} \cdot \mathsf{q}_i)$.

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

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

  By Definition 3.25,

$$\begin{aligned} &\Pr\left[\ \exists\, i \in [\mathsf{k}_{\mathsf{FIOP}}], \left(\begin{array}{l} S_i^* = \emptyset \\ \wedge\, \mathcal{V}_{\mathsf{bFC}}\left(\begin{array}{l} \mathsf{pp}, \mathsf{cm}_i, \mathcal{Q}_i, \boldsymbol{\beta}_i, \\ ((\mathsf{pm}_{i,j}, \mathsf{vm}_{i,j}))_{j \in [\mathsf{k}_{\mathsf{bFC}}]} \end{array}\right) = 1 \end{array}\right)\ \right] \\ &\leq \sum_{i \in [\mathsf{k}_{\mathsf{FIOP}}]} \Pr\left[\begin{array}{l} S_i^* = \emptyset \\ \wedge\, \mathcal{V}_{\mathsf{bFC}}\left(\begin{array}{l} \mathsf{pp}, \mathsf{cm}_i, \mathcal{Q}_i, \boldsymbol{\beta}_i, \\ ((\mathsf{pm}_{i,j}, \mathsf{vm}_{i,j}))_{j \in [\mathsf{k}_{\mathsf{bFC}}]} \end{array}\right) = 1 \end{array}\right] \\ &\leq \sum_{i \in [\mathsf{k}_{\mathsf{FIOP}}]} \Pr\left[\begin{array}{l} \left(\begin{array}{l} \forall\, j \in [\mathsf{N}+1]: \\ |\mathcal{Q}_j| \leq \mathsf{q}_{\mathsf{max}} \\ \wedge\, \mathcal{V}_{\mathsf{bFC}}\left(\begin{array}{l} \mathsf{pp}, \mathsf{cm}, \mathcal{Q}_j, \boldsymbol{\beta}_j, \\ ((\mathsf{pm}_{j,u}, \mathsf{vm}_{j,u}))_{u \in [\mathsf{k}_{\mathsf{bFC}}]} \end{array}\right) = 1 \end{array}\right) \\ \wedge\, \nexists\, \Pi \text{ s.t. } \forall\, j \in [\mathsf{N}+1], \alpha \in \mathcal{Q}_j,\ \alpha(\Pi) = \boldsymbol{\beta}_j^{(\alpha)} \end{array}\ \middle|\ \begin{array}{l} \mathsf{pp}_{\mathsf{bFC}} \leftarrow \mathsf{bFC.Gen}(1^\lambda, \ell_{\mathsf{max}}) \\ \mathsf{ai}_{\mathsf{bFC}} \leftarrow \mathcal{D}_{\mathsf{bFC}} \\ \mathsf{rnd}_{\mathsf{bFC}} := (\mathsf{rnd}_{\mathsf{bFC},i})_{i \in [\mathsf{k}_{\mathsf{bFC}}]} \leftarrow \mathcal{U}((\mathsf{r}_{\mathsf{bFC},i})_{i \in [\mathsf{k}_{\mathsf{bFC}}]}) \\ \left(\mathsf{cm}, \left(\left(\left(\begin{array}{l} \mathcal{Q}_i, \boldsymbol{\beta}_i, \\ \left(\left(\begin{array}{l} \mathsf{pm}_{i,j}, \\ \eta_{i,j}, \\ \mathsf{vm}_{i,j} \end{array}\right)\right)_{j \in [\mathsf{k}_{\mathsf{bFC}}]} \end{array}\right)\right)\right)_{i \in [\mathsf{N}+1]}\right) \\ \quad \leftarrow \mathsf{bFCSRGame}(s_{\mathsf{bFC}}, \mathsf{rnd}_{\mathsf{bFC}}, \widetilde{\mathcal{P}}^{\mathsf{SR}}_{\mathsf{bFC},i}, \mathsf{pp}_{\mathsf{bFC}}, \mathsf{ai}_{\mathsf{bFC}}) \end{array}\right] \\ &\leq \sum_{i \in [\mathsf{k}_{\mathsf{FIOP}}]} \epsilon^{\mathsf{SR}}_{\mathsf{bFC}}(\lambda, \ell_i, \mathsf{q}_i, \mathsf{N}+1, s_{\mathsf{bFC}}, \mathsf{m}_{\mathsf{bFC}}, t_{\mathsf{bFC}}) \ . \end{aligned}$$

where $s_{\mathsf{bFC}}, \mathsf{m}_{\mathsf{bFC}}, t_{\mathsf{bFC}}$ satisfy the same conditions as in Lemma 6.1.

6.2.2 Bound for Equation 11 from tail errors

We bound the probability in Equation 11 using tail errors.

Based on Construction 6.3, for every $i \in [\mathsf{k}_{\mathsf{FIOP}}]$, set a_i such that $\widetilde{\Pi}_i := \widetilde{\Pi}_i^{(a_i)}$. We define a q_i-admissible distribution $\mathcal{D}_{(i,a_i,(\mathsf{res}_j)_{j \in [a_i - 1]})}$ as follows.

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

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

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

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

$$\left(\mathbb{x}, \left((\mathsf{cm}_j, \sigma_j)\right)_{j\in[\mathsf{k}_{\mathsf{FIOP}}]}, \left((\mathcal{Q}_j, \boldsymbol{\beta}_j)\right)_{j\in[\mathsf{k}_{\mathsf{FIOP}}]}, ((\mathsf{pm}_{j,u}, \eta_{j,u}))_{j\in[\mathsf{k}_{\mathsf{FIOP}}],u\in[\mathsf{k}_{\mathsf{bFC}}]}\right) \ .$$

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

$$\left(\mathbb{x}, \left((\mathsf{cm}_j, \sigma_j)\right)_{j\in[\mathsf{k}_{\mathsf{FIOP}}]}, \left((\mathcal{Q}_j, \boldsymbol{\beta}_j)\right)_{j\in[\mathsf{k}_{\mathsf{FIOP}}]}, ((\mathsf{pm}_{j,c}, \eta_{j,c}))_{i\in[\mathsf{k}_{\mathsf{FIOP}}],c\in[u]}\right) \ .$$

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

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

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

We conclude that

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

7 State-restoration security of the Funky protocol

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

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

Then $\mathsf{ARG} = (\mathcal{G}, \mathcal{P}, \mathcal{V}) \coloneqq \mathsf{Funky}[\mathsf{FIOP}, \mathsf{bFC}]$ (Construction 4.1) is a $(2\mathsf{k}_{\mathsf{FIOP}} + 1 + 2\mathsf{k}_{\mathsf{bFC}})$-message public-coin interactive argument system for R with state-restoration soundness error $\epsilon^{\mathsf{SR}}_{\mathsf{ARG}}$ and state-restoration knowledge soundness error $\kappa^{\mathsf{SR}}_{\mathsf{ARG}}$ that satisfy

$$\epsilon^{\mathsf{SR}}_{\mathsf{ARG}}(\lambda, n, s, \mathsf{m}, t_{\mathsf{ARG}}) \leq \epsilon^{\mathsf{SR}}_{\mathsf{FIOP}}(n, s + \lambda, \mathsf{m}_{\mathsf{FIOP}}) + \sum_{i \in [\mathsf{k}_{\mathsf{FIOP}}]} \left(\epsilon^{\mathsf{SR}}_{\mathsf{bFC}}(\lambda, \ell_i, \mathsf{q}_i, \mathsf{N}+1, s_{\mathsf{bFC}}, \mathsf{m}_{\mathsf{bFC}}, t_{\mathsf{bFC}}) + \epsilon_{\mathbf{Q}_i}(\ell_i, \mathsf{q}_i, \mathsf{N})\right) \text{ and}$$

$$\kappa^{\mathsf{SR}}_{\mathsf{ARG}}(\lambda, n, s, \mathsf{m}, t_{\mathsf{ARG}}) \leq \kappa^{\mathsf{SR}}_{\mathsf{FIOP}}(n, s + \lambda, \mathsf{m}_{\mathsf{FIOP}}) + \sum_{i \in [\mathsf{k}_{\mathsf{FIOP}}]} \left(\epsilon^{\mathsf{SR}}_{\mathsf{bFC}}(\lambda, \ell_i, \mathsf{q}_i, \mathsf{N}+1, s_{\mathsf{bFC}}, \mathsf{m}_{\mathsf{bFC}}, t_{\mathsf{bFC}}) + \epsilon_{\mathbf{Q}_i}(\ell_i, \mathsf{q}_i, \mathsf{N})\right) \ ,$$

where

- $\mathsf{m}_{\mathsf{FIOP}} \leq \mathsf{m} + \mathsf{k}_{\mathsf{FIOP}}$*;*
- $t_{\mathsf{FIOP}} \leq (\mathsf{m} + \mathsf{k}_{\mathsf{FIOP}}) \cdot (\mathsf{N} \cdot (t_{\mathsf{ARG}} + \mathsf{k}_{\mathsf{FIOP}} \cdot t_{\mathcal{V}_{\mathsf{bFC}}}) + \mathsf{k}_{\mathsf{FIOP}} \cdot t_{\mathbf{Q}}(\ell_{\mathsf{max}}, \mathsf{N} \cdot \mathsf{q}_{\mathsf{max}}))$*;*
- $s_{\mathsf{bFC}} \leq s + \log((\mathsf{k}_{\mathsf{FIOP}} \cdot \mathsf{N} + 1) \cdot (\mathsf{m} + \mathsf{k}_{\mathsf{bFC}}))$*;*
- $\mathsf{m}_{\mathsf{bFC}} \leq (\mathsf{k}_{\mathsf{FIOP}} \cdot \mathsf{N} + 1) \cdot (\mathsf{m} + \mathsf{k}_{\mathsf{bFC}})$*; and*
- $t_{\mathsf{bFC}} \leq O(\mathsf{N} \cdot (t_{\mathsf{ARG}} + \hat{\mathsf{m}} + t_{\mathcal{V}_{\mathsf{bFC}}})) + t_{\mathbf{Q}}(\ell_{\mathsf{max}}, \mathsf{N} \cdot \mathsf{q}_{\mathsf{max}})$.

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

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

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

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

$$\epsilon^{\mathsf{SR}}_{\mathsf{ARG}}(\lambda, n, s, \mathsf{m}, t_{\mathsf{ARG}}) \leq \epsilon^{\mathsf{SR}}_{\mathsf{FIOP}}(n, s + \lambda, \mathsf{m}_{\mathsf{FIOP}}) + \mathrm{negl}(\lambda) \text{ and}$$
$$\kappa^{\mathsf{SR}}_{\mathsf{ARG}}(\lambda, n, s, \mathsf{m}, t_{\mathsf{ARG}}) \leq \kappa^{\mathsf{SR}}_{\mathsf{FIOP}}(n, s + \lambda, \mathsf{m}_{\mathsf{FIOP}}) + \mathrm{negl}(\lambda) \ .$$

Proof. Suppose $\mathsf{k}_{\mathsf{FIOP}} = \mathsf{p}_1(\lambda)$ for some polynomial p_1. Let $\mathsf{p}_2(\lambda)$ be an arbitrary polynomial. Choose $\mathsf{N} = \mathsf{poly}(\lambda)$ such that $\epsilon_{\mathbf{Q}_i}(\ell, \mathsf{q}, \mathsf{N}) \leq \frac{1}{2\mathsf{p}_1(\lambda)\mathsf{p}_2(\lambda)}$ for all $i \in [\mathsf{k}_{\mathsf{FIOP}}]$. Hence, $\mathsf{m}_{\mathsf{bFC}} = \mathsf{poly}(\lambda)$ and $t_{\mathsf{bFC}} = \mathsf{poly}(\lambda)$. Therefore,

$$\begin{aligned}\epsilon^{\mathsf{SR}}_{\mathsf{ARG}}(\lambda, n, s, \mathsf{m}, t_{\mathsf{ARG}}) &\leq \epsilon^{\mathsf{SR}}_{\mathsf{FIOP}}(n, s + \lambda, \mathsf{m}_{\mathsf{FIOP}}) + \mathrm{negl}(\lambda) + \frac{1}{2\mathsf{p}_2(\lambda)} \\ &< \epsilon^{\mathsf{SR}}_{\mathsf{FIOP}}(n, s + \lambda, \mathsf{m}_{\mathsf{FIOP}}) + \mathrm{negl}(\lambda) + \frac{1}{\mathsf{p}_2(\lambda)} \ .\end{aligned}$$

Since p_2 is an arbitrary polynomial, we conclude that

$$\epsilon^{\mathsf{SR}}_{\mathsf{ARG}}(\lambda, n, s, \mathsf{m}, t_{\mathsf{ARG}}) \leq \epsilon^{\mathsf{SR}}_{\mathsf{FIOP}}(n, s + \lambda, \mathsf{m}_{\mathsf{FIOP}}) + \mathrm{negl}(\lambda) \ .$$

An analogous argument holds for $\kappa^{\mathsf{SR}}_{\mathsf{ARG}}$. □

7.1 Construction of the FIOP state-restoration adversary

We construct an FIOP state-restoration adversary $\widetilde{\mathbf{P}}^{\mathsf{SR}}$ we use in the proof of Theorem 7.1 using the argument state-restoration adversary $\widetilde{\mathcal{P}}^{\mathsf{SR}}$.

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.

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

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

$\widetilde{\mathbf{P}}^{\mathsf{SR}}(\mathbf{ai})$:
1. Parse $\mathbf{ai}$ as $(\mathsf{pp}, \mathsf{ai})$.
2. Simulate SRGame with $\widetilde{\mathcal{P}}^{\mathsf{SR}}(\mathsf{pp}, \mathsf{ai})$ as follows.
   (a) For every $i \in [\mathsf{k}]$, initialize an empty partial function $\mathsf{rnd}_i \colon \{0,1\}^* \to \{0,1\}^{\mathsf{r}_i}$.
   (b) For $a \in [\hat{\mathsf{m}}]$, when $\widetilde{\mathcal{P}}^{\mathsf{SR}}$ makes its a-th move mv_a:
      i. If mv_a has the form $(\mathbb{x}', ((\mathsf{cm}'_j, \sigma'_j))_{j\in[i]})$ for some $i \in [\mathsf{k}_{\mathsf{FIOP}}]$:
         A. If $\mathsf{rnd}_i(\mathsf{mv}_a) = \bot$:
            – Run $(\widetilde{\Pi}_j^{(a)})_{j\in[i]} \coloneqq \mathfrak{R}_{\mathsf{m}}^{\widetilde{\mathcal{P}}^{\mathsf{SR}}(\mathsf{pp},\mathsf{ai})}(a, (\mathsf{res}_i)_{i\in[a-1]})$ (Construction 6.2).
            – For every $j \in [i]$, set the FIOP salt string $\gamma_j \coloneqq (\mathsf{cm}'_j, \sigma'_j)$.
            – Make the FIOP move $(\mathbb{x}', (\widetilde{\Pi}_j^{(a)})_{j\in[i]}, (\gamma_j)_{j\in[i]})$ in FIOPSRGame to get an answer ρ_a.
            – Set $\mathsf{rnd}_i(\mathsf{mv}_a) \coloneqq \rho_a$.
         B. Answer mv with $\mathsf{rnd}_i(\mathsf{mv})$.
      ii. Otherwise, mv_a has the form

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

         A. If $\mathsf{rnd}_{\mathsf{k}_{\mathsf{FIOP}}+j}(\mathsf{mv}_a) = \bot$, sample $\rho \in \{0,1\}^{\mathsf{r}_{\mathsf{k}_{\mathsf{FIOP}}+j}}$ and set $\mathsf{rnd}_{\mathsf{k}_{\mathsf{FIOP}}+j}(\mathsf{mv}_a) \coloneqq \rho$.
         B. Set $(\widetilde{\Pi}_i^{(a)})_{i\in[\mathsf{round}(\mathsf{mv}_a)]} \coloneqq \bot$ and answer mv_a with $\mathsf{rnd}_{\mathsf{k}_{\mathsf{FIOP}}+j}(\mathsf{mv}_a)$.
3. Let $\mathsf{tr}^{\mathsf{SR}}$ be the move-response trace of SRGame and let the final output of $\widetilde{\mathcal{P}}^{\mathsf{SR}}$ be:

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

4. Run $(\widetilde{\Pi}_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]} \leftarrow \mathfrak{R}\big(\mathsf{tr}^{\mathsf{SR}}, (\mathbb{x}, ((\mathsf{cm}_i, \sigma_i))_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}), (\widetilde{\Pi}_i^{(a)})_{a\in[\hat{\mathsf{m}}],i\in[\mathsf{round}(\mathsf{mv}_a)]}\big)$.
5. For every $i \in [\mathsf{k}_{\mathsf{FIOP}}]$, set the FIOP salt string $\gamma_i \coloneqq (\mathsf{cm}_i, \sigma_i)$.
6. Output $(\mathbb{x}, (\widetilde{\Pi}_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}, (\gamma_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]})$.

We compute the efficiency parameters for $\widetilde{\mathbf{P}}^{\mathsf{SR}}$.

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

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

7.2 State-restoration soundness

We wish to upper-bound the following expression:

$$\Pr\left[\begin{array}{l} \\ |\mathbb{x}| \le n \\ \wedge\, \mathbb{x} \notin L(R) \\ \wedge\, \mathcal{V}\left(\begin{array}{l} \mathsf{pp}, \mathbb{x}, (\mathsf{cm}_i)_{i\in[\mathsf{k_{FIOP}}]}, \\ \big((\mathcal{Q}_i, \boldsymbol{\beta}_i)\big)_{i\in[\mathsf{k_{FIOP}}]}, (\rho_i)_{i\in[\mathsf{k_{FIOP}}]}, \\ ((\mathsf{pm}_{i,j}, \mathsf{vm}_{i,j}))_{i\in[\mathsf{k_{FIOP}}],j\in[\mathsf{k_{bFC}}]} \end{array}\right) = 1 \\ \\ \end{array} \;\middle|\; \begin{array}{l} \mathsf{pp} \leftarrow \mathcal{G}(1^\lambda, n) \\ \mathsf{ai} \leftarrow \mathcal{D} \\ \mathsf{rnd} := (\mathsf{rnd}_i)_{i\in[\mathsf{k_{FIOP}}+\mathsf{k_{bFC}}]} \leftarrow \mathcal{U}((\mathsf{r}_i)_{i\in[\mathsf{k_{FIOP}}+\mathsf{k_{bFC}}]}) \\ \left(\begin{array}{l} \mathbb{x}, \big((\mathsf{cm}_i, \sigma_i)\big)_{i\in[\mathsf{k_{FIOP}}]}, \big((\mathcal{Q}_i, \boldsymbol{\beta}_i)\big)_{i\in[\mathsf{k_{FIOP}}]}, \\ ((\mathsf{pm}_{i,j}, \eta_{i,j}))_{i\in[\mathsf{k_{FIOP}}],j\in[\mathsf{k_{bFC}}]}, \\ (\rho_i)_{i\in[\mathsf{k_{FIOP}}]}, (\mathsf{vm}_{i,j})_{i\in[\mathsf{k_{FIOP}}],j\in[\mathsf{k_{bFC}}]} \end{array}\right) \\ \quad \xleftarrow{\mathsf{tr}^{\mathsf{SR}}} \mathsf{SRGame}(s, \mathsf{rnd}, \widetilde{\mathcal{P}}^{\mathsf{SR}}, \mathsf{pp}, \mathsf{ai}) \end{array}\right]. \tag{12}$$

We augment the experiment in Equation 12 by running $\mathfrak{R}_{\mathsf{m}}$ (Construction 6.2) for every $a \in [\hat{\mathsf{m}}]$ and $\mathfrak{R}$ as follows:

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

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

Note that $(\mathbb{x}, \big((\mathsf{cm}_i, \sigma_i)\big)_{i\in[\mathsf{k_{FIOP}}]})$ is fully contained in $\mathsf{tr}^{\mathsf{SR}}$. By the law of total probability,

$$\Pr\left[\begin{array}{l} |\mathbb{x}| \le n \\ \wedge\, \mathbb{x} \notin L(R) \\ \wedge\, \mathcal{V}\left(\begin{array}{l} \mathsf{pp}, \mathbb{x}, (\mathsf{cm}_i)_{i\in[\mathsf{k_{FIOP}}]}, \\ \big((\mathcal{Q}_i, \boldsymbol{\beta}_i)\big)_{i\in[\mathsf{k_{FIOP}}]}, (\rho_i)_{i\in[\mathsf{k_{FIOP}}]}, \\ ((\mathsf{pm}_{i,j}, \mathsf{vm}_{i,j}))_{i\in[\mathsf{k_{FIOP}}],j\in[\mathsf{k_{bFC}}]} \end{array}\right) = 1 \end{array}\right]$$

$$= \Pr\left[\begin{array}{l} |\mathbb{x}| \le n \\ \wedge\, \mathbb{x} \notin L(R) \\ \wedge\, \mathbf{V}^{([\mathcal{Q}_i,\boldsymbol{\beta}_i])_{i\in[\mathsf{k_{FIOP}}]}}(\mathbb{x}; (\rho_i)_{i\in[\mathsf{k_{FIOP}}]}) = 1 \\ \wedge\, \left(\begin{array}{l} \forall\, i \in [\mathsf{k_{FIOP}}] : \\ \mathcal{V}_{\mathsf{bFC}}\left(\begin{array}{l} \mathsf{pp}, \mathsf{cm}_i, \mathcal{Q}_i, \boldsymbol{\beta}_i, \\ ((\mathsf{pm}_{i,j}, \mathsf{vm}_{i,j}))_{j\in[\mathsf{k_{bFC}}]} \end{array}\right) = 1 \end{array}\right) \end{array}\right]$$

$$\le \Pr\left[\begin{array}{l} |\mathbb{x}| \le n \\ \wedge\, \mathbb{x} \notin L(R) \\ \wedge\, \mathbf{V}^{(\widetilde{\Pi}_i)_{i\in[\mathsf{k_{FIOP}}]}}(\mathbb{x}; (\rho_i)_{i\in[\mathsf{k_{FIOP}}]}) = 1 \\ \wedge\, \mathbf{V}^{([\mathcal{Q}_i,\beta_i])_{i\in[\mathsf{k_{FIOP}}]}}(\mathbb{x}; (\rho_i)_{i\in[\mathsf{k_{FIOP}}]}) = 1 \\ \wedge\, \left(\begin{array}{l} \forall\, i \in [\mathsf{k_{FIOP}}] : \\ \mathcal{V}_{\mathsf{bFC}}\left(\begin{array}{l} \mathsf{pp}, \mathsf{cm}_i, \mathcal{Q}_i, \boldsymbol{\beta}_i, \\ ((\mathsf{pm}_{i,j}, \mathsf{vm}_{i,j}))_{j\in[\mathsf{k_{bFC}}]} \end{array}\right) = 1 \end{array}\right) \end{array}\right] + \Pr\left[\begin{array}{l} |\mathbb{x}| \le n \\ \wedge\, \mathbb{x} \notin L(R) \\ \wedge\, \mathbf{V}^{(\widetilde{\Pi}_i)_{i\in[\mathsf{k_{FIOP}}]}}(\mathbb{x}; (\rho_i)_{i\in[\mathsf{k_{FIOP}}]}) \neq 1 \\ \wedge\, \mathbf{V}^{([\mathcal{Q}_i,\boldsymbol{\beta}_i])_{i\in[\mathsf{k_{FIOP}}]}}(\mathbb{x}; (\rho_i)_{i\in[\mathsf{k_{FIOP}}]}) = 1 \\ \wedge\, \left(\begin{array}{l} \forall\, i \in [\mathsf{k_{FIOP}}] : \\ \mathcal{V}_{\mathsf{bFC}}\left(\begin{array}{l} \mathsf{pp}, \mathsf{cm}_i, \mathcal{Q}_i, \boldsymbol{\beta}_i, \\ ((\mathsf{pm}_{i,j}, \mathsf{vm}_{i,j}))_{j\in[\mathsf{k_{bFC}}]} \end{array}\right) = 1 \end{array}\right) \end{array}\right].$$

We bound the latter term and then the former term.

Bound from security reduction. According to Lemma 6.1,

$$\begin{aligned}
&\Pr\left[\begin{array}{l}
|\mathbb{x}| \le n \\
\wedge\, \mathbb{x} \notin L(R) \\
\wedge\, \mathbf{V}^{(\widetilde{\Pi}_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}}(\mathbb{x}; (\rho_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}) \neq 1 \\
\wedge\, \mathbf{V}^{([\mathcal{Q}_i,\boldsymbol{\beta}_i])_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}}(\mathbb{x}; (\rho_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}) = 1 \\
\wedge \left(\begin{array}{l} \forall\, i \in [\mathsf{k}_{\mathsf{FIOP}}] : \\ \mathcal{V}_{\mathsf{bFC}}\left(\begin{array}{l} \mathsf{pp}, \mathsf{cm}_i, \mathcal{Q}_i, \boldsymbol{\beta}_i, \\ ((\mathsf{pm}_{i,j}, \mathsf{vm}_{i,j}))_{j\in[\mathsf{k}_{\mathsf{bFC}}]} \end{array} \right) = 1 \end{array} \right)
\end{array}\right] \\
&\le \Pr\left[\begin{array}{l}
\mathbf{V}^{(\widetilde{\Pi}_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}}(\mathbb{x}; (\rho_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}) \neq 1 \\
\wedge\, \mathbf{V}^{([\mathcal{Q}_i,\beta_i])_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}}(\mathbb{x}; (\rho_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}) = 1 \\
\wedge \left(\begin{array}{l} \forall\, i \in [\mathsf{k}_{\mathsf{FIOP}}], \alpha \in \mathcal{Q}_i, \\ \mathcal{V}_{\mathsf{bFC}}\left(\begin{array}{l} \mathsf{pp}, \mathsf{cm}_i, \alpha, \beta_i^{(\alpha)}, \\ ((\mathsf{pm}_{i,j}, \mathsf{vm}_{i,j}))_{j\in[\mathsf{k}_{\mathsf{bFC}}]} \end{array} \right) = 1 \end{array} \right)
\end{array}\right] \\
&\le \sum_{i\in[\mathsf{k}_{\mathsf{FIOP}}]} \left(\epsilon^{\mathsf{SR}}_{\mathsf{bFC}}(\lambda, \ell_i, \mathsf{q}_i, \mathsf{N}+1, s_{\mathsf{bFC}}, \mathsf{m}_{\mathsf{bFC}}, t_{\mathsf{bFC}}) + \epsilon_{\mathbf{Q}_i}(\ell_i, \mathsf{q}_i, \mathsf{N})\right) \ .
\end{aligned}$$

Bound from FIOP state-restoration soundness. Let $\mathbf{D}$ be as in Construction 7.3 and let $\widetilde{\mathbf{P}}^{\mathsf{SR}}$ be the FIOP state-restoration adversary in Construction 7.4. According to Definition 3.15,

$$\begin{aligned}
&\Pr\left[\begin{array}{l}
|\mathbb{x}| \le n \\
\wedge\, \mathbb{x} \notin L(R) \\
\wedge\, \mathbf{V}^{(\widetilde{\Pi}_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}}(\mathbb{x}; (\rho_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}) = 1 \\
\wedge\, \mathbf{V}^{([\mathcal{Q}_i,\beta_i])_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}}(\mathbb{x}; (\rho_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}) = 1 \\
\wedge \left(\begin{array}{l} \forall\, i \in [\mathsf{k}_{\mathsf{FIOP}}] : \\ \mathcal{V}_{\mathsf{bFC}}\left(\begin{array}{l} \mathsf{pp}, \mathsf{cm}_i, \mathcal{Q}_i, \boldsymbol{\beta}_i, \\ ((\mathsf{pm}_{i,j}, \mathsf{vm}_{i,j}))_{j\in[\mathsf{k}_{\mathsf{bFC}}]} \end{array} \right) = 1 \end{array} \right)
\end{array}\right] \\
&\le \Pr\left[\begin{array}{l|l}
|\mathbb{x}| \le n & \mathbf{ai} \leftarrow \mathbf{D} \\
\wedge\, \mathbb{x} \notin L(R) & \mathsf{rnd}_{\mathsf{FIOP}} := (\mathsf{rnd}_{\mathsf{FIOP},i})_{i\in[\mathsf{k}_{\mathsf{FIOP}}]} \leftarrow \mathcal{U}((\mathsf{r}_{\mathsf{FIOP},i})_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}) \\
\wedge\, \mathbf{V}(\mathbb{x}, (\widetilde{\Pi}_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}, (\rho_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}) = 1 & (\mathbb{x}, (\widetilde{\Pi}_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}, (\gamma_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}, (\rho_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}) \\
& \quad \leftarrow \mathsf{FIOPSRGame}(s_{\mathsf{FIOP}}, \mathsf{rnd}_{\mathsf{FIOP}}, \widetilde{\mathbf{P}}^{\mathsf{SR}}, \mathbf{ai})
\end{array}\right] \\
&\le \epsilon^{\mathsf{SR}}_{\mathsf{FIOP}}(n, s_{\mathsf{FIOP}}, \mathsf{m}_{\mathsf{FIOP}}) \ .
\end{aligned}$$

7.3 State-restoration knowledge soundness

We wish to upper-bound the following expression:

$$\Pr\left[\begin{array}{l} \\ \\ \\ |\mathbb{x}| \le n \\ \wedge (\mathbb{x}, \mathbb{w}) \notin R \\ \wedge \mathcal{V}\left(\begin{array}{l} \mathsf{pp}, \mathbb{x}, (\mathsf{cm}_i)_{i\in[\mathsf{k_{FIOP}}]}, \\ \big((\mathcal{Q}_i, \boldsymbol{\beta}_i)\big)_{i\in[\mathsf{k_{FIOP}}]}, (\rho_i)_{i\in[\mathsf{k_{FIOP}}]}, \\ ((\mathsf{pm}_{i,j}, \mathsf{vm}_{i,j}))_{i\in[\mathsf{k_{FIOP}}],j\in[\mathsf{k_{bFC}}]} \end{array}\right) = 1 \\ \\ \\ \end{array} \;\middle|\; \begin{array}{l} \mathsf{pp} \leftarrow \mathcal{G}(1^\lambda, n) \\ \mathsf{ai} \leftarrow \mathcal{D} \\ \mathsf{rnd} := (\mathsf{rnd}_i)_{i\in[\mathsf{k_{FIOP}}+\mathsf{k_{bFC}}]} \leftarrow \mathcal{U}((\mathsf{r}_i)_{i\in[\mathsf{k_{FIOP}}+\mathsf{k_{bFC}}]}) \\ \left(\begin{array}{l} \mathbb{x}, \big((\mathsf{cm}_i, \sigma_i)\big)_{i\in[\mathsf{k_{FIOP}}]}, \big((\mathcal{Q}_i, \boldsymbol{\beta}_i)\big)_{i\in[\mathsf{k_{FIOP}}]}, \\ ((\mathsf{pm}_{i,j}, \eta_{i,j}))_{i\in[\mathsf{k_{FIOP}}],j\in[\mathsf{k_{bFC}}]}, \\ (\rho_i)_{i\in[\mathsf{k_{FIOP}}]}, (\mathsf{vm}_{i,j})_{i\in[\mathsf{k_{FIOP}}],j\in[\mathsf{k_{bFC}}]} \end{array}\right) \\ \quad \xleftarrow{\mathsf{tr}^{\mathsf{SR}}} \mathsf{SRGame}(s, \mathsf{rnd}, \widetilde{\mathcal{P}}^{\mathsf{SR}}, \mathsf{pp}, \mathsf{ai}) \\ \mathbb{w} \leftarrow \mathcal{E}_{\mathsf{SR}}^{\widetilde{\mathcal{P}}^{\mathsf{SR}}(\mathsf{pp},\mathsf{ai})}\left(\begin{array}{l} \mathbb{x}, \big((\mathsf{cm}_i, \sigma_i)\big)_{i\in[\mathsf{k_{FIOP}}]}, \\ \big((\mathcal{Q}_i, \boldsymbol{\beta}_i)\big)_{i\in[\mathsf{k_{FIOP}}]}, \\ ((\mathsf{pm}_{i,j}, \eta_{i,j}))_{i\in[\mathsf{k_{FIOP}}],j\in[\mathsf{k_{bFC}}]}, \\ (\rho_i)_{i\in[\mathsf{k_{FIOP}}]}, (\mathsf{vm}_{i,j})_{i\in[\mathsf{k_{FIOP}}],j\in[\mathsf{k_{bFC}}]}, \\ \mathsf{tr}^{\mathsf{SR}} \end{array}\right) \end{array}\right]. \tag{13}$$

We augment the experiment in Equation 13 by running $\mathfrak{R}_{\mathsf{m}}$ (Construction 6.2) for every $a \in [\hat{\mathsf{m}}]$ and $\mathfrak{R}$ as follows:

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

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

Note that $(\mathbb{x}, \big((\mathsf{cm}_i, \sigma_i)\big)_{i\in[\mathsf{k_{FIOP}}]})$ is fully contained in $\mathsf{tr}^{\mathsf{SR}}$. By the law of total probability,

$$\Pr\left[\begin{array}{l} |\mathbb{x}| \le n \\ \wedge (\mathbb{x}, \mathbb{w}) \notin R \\ \wedge \mathcal{V}\left(\begin{array}{l} \mathsf{pp}, \mathbb{x}, (\mathsf{cm}_i)_{i\in[\mathsf{k_{FIOP}}]}, \\ \big((\mathcal{Q}_i, \boldsymbol{\beta}_i)\big)_{i\in[\mathsf{k_{FIOP}}]}, (\rho_i)_{i\in[\mathsf{k_{FIOP}}]}, \\ ((\mathsf{pm}_{i,j}, \mathsf{vm}_{i,j}))_{i\in[\mathsf{k_{FIOP}}],j\in[\mathsf{k_{bFC}}]} \end{array}\right) = 1 \end{array}\right]$$

$$= \Pr\left[\begin{array}{l} |\mathbb{x}| \le n \\ \wedge (\mathbb{x}, \mathbb{w}) \notin R \\ \wedge \mathbf{V}^{([\mathcal{Q}_i,\beta_i])_{i\in[\mathsf{k_{FIOP}}]}}(\mathbb{x}; (\rho_i)_{i\in[\mathsf{k_{FIOP}}]}) = 1 \\ \wedge \left(\begin{array}{l} \forall\, i \in [\mathsf{k_{FIOP}}] : \\ \mathcal{V}_{\mathsf{bFC}}\left(\begin{array}{l} \mathsf{pp}, \mathsf{cm}_i, \mathcal{Q}_i, \boldsymbol{\beta}_i, \\ ((\mathsf{pm}_{i,j}, \mathsf{vm}_{i,j}))_{j\in[\mathsf{k_{bFC}}]} \end{array}\right) = 1 \end{array}\right) \end{array}\right]$$

$$\leq \Pr\left[\begin{array}{l} |\mathbb{x}| \leq n \\ \wedge\, (\mathbb{x}, \mathbb{w}) \notin R \\ \wedge\, \mathbf{V}^{(\widetilde{\Pi}_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}}(\mathbb{x}; (\rho_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}) = 1 \\ \wedge\, \mathbf{V}^{([\mathcal{Q}_i,\boldsymbol{\beta}_i])_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}}(\mathbb{x}; (\rho_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}) = 1 \\ \wedge \left(\begin{array}{l} \forall\, i \in [\mathsf{k}_{\mathsf{FIOP}}] : \\ \mathcal{V}_{\mathsf{bFC}}\left(\begin{array}{l} \mathsf{pp}, \mathsf{cm}_i, \mathcal{Q}_i, \boldsymbol{\beta}_i, \\ ((\mathsf{pm}_{i,j}, \mathsf{vm}_{i,j}))_{j\in[\mathsf{k}_{\mathsf{bFC}}]} \end{array} \right) = 1 \end{array} \right) \end{array}\right] + \Pr\left[\begin{array}{l} |\mathbb{x}| \leq n \\ \wedge\, (\mathbb{x}, \mathbb{w}) \notin R \\ \wedge\, \mathbf{V}^{(\widetilde{\Pi}_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}}(\mathbb{x}; (\rho_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}) \neq 1 \\ \wedge\, \mathbf{V}^{([\mathcal{Q}_i,\boldsymbol{\beta}_i])_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}}(\mathbb{x}; (\rho_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}) = 1 \\ \wedge \left(\begin{array}{l} \forall\, i \in [\mathsf{k}_{\mathsf{FIOP}}] : \\ \mathcal{V}_{\mathsf{bFC}}\left(\begin{array}{l} \mathsf{pp}, \mathsf{cm}_i, \mathcal{Q}_i, \boldsymbol{\beta}_i, \\ ((\mathsf{pm}_{i,j}, \mathsf{vm}_{i,j}))_{j\in[\mathsf{k}_{\mathsf{bFC}}]} \end{array} \right) = 1 \end{array} \right) \end{array}\right] .$$

We bound the latter term and then the former term.

Bound from security reduction. According to Lemma 6.1:

$$\begin{aligned} &\Pr\left[\begin{array}{l} |\mathbb{x}| \leq n \\ \wedge\, (\mathbb{x}, \mathbb{w}) \notin R \\ \wedge\, \mathbf{V}^{(\widetilde{\Pi}_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}}(\mathbb{x}; (\rho_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}) = 1 \\ \wedge\, \mathbf{V}^{([\mathcal{Q}_i,\boldsymbol{\beta}_i])_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}}(\mathbb{x}; (\rho_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}) = 1 \\ \wedge \left(\begin{array}{l} \forall\, i \in [\mathsf{k}_{\mathsf{FIOP}}] : \\ \mathcal{V}_{\mathsf{bFC}}\left(\begin{array}{l} \mathsf{pp}, \mathsf{cm}_i, \mathcal{Q}_i, \boldsymbol{\beta}_i, \\ ((\mathsf{pm}_{i,j}, \mathsf{vm}_{i,j}))_{j\in[\mathsf{k}_{\mathsf{bFC}}]} \end{array} \right) = 1 \end{array} \right) \end{array}\right] \\ &\leq \Pr\left[\begin{array}{l} \mathbf{V}^{(\widetilde{\Pi}_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}}(\mathbb{x}; (\rho_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}) \neq 1 \\ \wedge\, \mathbf{V}^{([\mathcal{Q}_i,\boldsymbol{\beta}_i])_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}}(\mathbb{x}; (\rho_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}) = 1 \\ \wedge \left(\begin{array}{l} \forall\, i \in [\mathsf{k}_{\mathsf{FIOP}}] : \\ \mathcal{V}_{\mathsf{bFC}}\left(\begin{array}{l} \mathsf{pp}, \mathsf{cm}_i, \mathcal{Q}_i, \boldsymbol{\beta}_i, \\ ((\mathsf{pm}_{i,j}, \mathsf{vm}_{i,j}))_{j\in[\mathsf{k}_{\mathsf{bFC}}]} \end{array} \right) = 1 \end{array} \right) \end{array}\right] \\ &\leq \sum_{i\in[\mathsf{k}_{\mathsf{FIOP}}]} \left(\epsilon^{\mathsf{SR}}_{\mathsf{bFC}}(\lambda, \ell_i, \mathsf{q}_i, \mathsf{N}+1, s_{\mathsf{bFC}}, \mathsf{m}_{\mathsf{bFC}}, t_{\mathsf{bFC}}) + \epsilon_{\mathbf{Q}_i}(\ell_i, \mathsf{q}_i, \mathsf{N})\right) . \end{aligned}$$

Bound from FIOP state-restoration knowledge soundness. Let $\mathbf{D}$ be defined as in Construction 7.3, $\widetilde{\mathbf{P}}^{\mathsf{SR}}$ be the FIOP state-restoration adversary constructed in Construction 7.4.

We construct the state-restoration knowledge extractor $\mathcal{E}_{\mathsf{SR}}$ for ARG.

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

$\mathcal{E}^{\widetilde{\mathcal{P}}^{\mathsf{SR}}(\mathsf{pp},\mathsf{ai})}_{\mathsf{SR}}(\mathbb{x}, ((\mathsf{cm}_i, \sigma_i))_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}, ((\mathcal{Q}_i, \boldsymbol{\beta}_i))_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}, (\rho_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}, (\mathsf{pm}_{i,j}, \eta_{i,j}, \mathsf{vm}_{i,j})_{i\in[\mathsf{k}_{\mathsf{FIOP}}], j\in[\mathsf{k}_{\mathsf{bFC}}]}, \mathsf{tr}^{\mathsf{SR}})$:

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

$$(\mathbb{x}, (\widetilde{\Pi}_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}, (\gamma_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}, (\rho_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}) .$$

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

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

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

According to Definition 3.16,

$$\Pr\left[\begin{array}{l} |\mathbb{x}| \leq n \\ \wedge\, (\mathbb{x}, \mathbb{w}) \notin R \\ \wedge\, \mathbf{V}^{(\widetilde{\Pi}_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}}(\mathbb{x}; (\rho_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}) = 1 \\ \wedge\, \mathbf{V}^{([\mathcal{Q}_i, \boldsymbol{\beta}_i])_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}}(\mathbb{x}; (\rho_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}) = 1 \\ \wedge\, \left(\begin{array}{l} \forall\, i \in [\mathsf{k}_{\mathsf{FIOP}}] : \\ \mathcal{V}_{\mathsf{bFC}}\left(\begin{array}{l} \mathsf{pp}, \mathsf{cm}_i, \mathcal{Q}_i, \boldsymbol{\beta}_i, \\ ((\mathsf{pm}_{i,j}, \mathsf{vm}_{i,j}))_{j\in[\mathsf{k}_{\mathsf{bFC}}]} \end{array} \right) = 1 \end{array} \right) \end{array}\right]$$

$$\leq \Pr\left[\begin{array}{l} |\mathbb{x}| \leq n \\ \wedge\, (\mathbb{x}, \mathbb{w}) \notin R \\ \wedge\, \mathbf{V}(\mathbb{x}, (\widetilde{\Pi}_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}, (\rho_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}) = 1 \end{array} \middle| \begin{array}{l} \mathbf{ai} \leftarrow \mathbf{D} \\ \mathsf{rnd}_{\mathsf{FIOP}} := (\mathsf{rnd}_{\mathsf{FIOP},i})_{i\in[\mathsf{k}_{\mathsf{FIOP}}]} \leftarrow \mathcal{U}((\mathsf{r}_{\mathsf{FIOP},i})_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}) \\ (\mathbb{x}, (\widetilde{\Pi}_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}, (\gamma_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}, (\rho_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}) \\ \quad \xleftarrow{\mathsf{tr}_{\mathsf{FIOP}}^{\mathsf{SR}}} \mathsf{FIOPSRGame}(s_{\mathsf{FIOP}}, \mathsf{rnd}_{\mathsf{FIOP}}, \widetilde{\mathbf{P}}^{\mathsf{SR}}, \mathbf{ai}) \\ \mathbb{w} \leftarrow \mathbf{E}_{\mathsf{SR}}^{\widetilde{\mathbf{P}}^{\mathsf{SR}}(\mathbf{ai})}(\mathbb{x}, (\widetilde{\Pi}_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}, (\gamma_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}, (\rho_i)_{i\in[\mathsf{k}_{\mathsf{FIOP}}]}, \mathsf{tr}_{\mathsf{FIOP}}^{\mathsf{SR}}) \end{array}\right]$$

$$\leq \kappa_{\mathsf{FIOP}}^{\mathsf{SR}}(n, s_{\mathsf{FIOP}}, \mathsf{m}_{\mathsf{FIOP}}) \ .$$

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 $\mathbf{Q}_{\mathsf{BatchMsg}}[\mathbf{Q}, s]$, where a query $\alpha \in \mathbf{Q}$ is applied to a batch of messages.

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

$$\mathbf{Q}_{\mathsf{BatchMsg}}[\mathbf{Q}, s] := \left\{ \alpha' \colon \Sigma^{s\cdot\ell} \to \mathbb{D}^s \;\middle|\; \exists\, \alpha \in \mathbf{Q} :\ \textit{s.t. } \alpha'(\Pi) = (\alpha((\Pi[i\ell+1], \ldots, \Pi[i\ell+\ell])))_{i\in\{0,\cdots,s-1\}} \right\} \ .$$

Every suitably homomorphic functional commitment FC for some query class $\mathbf{Q}$ gives rise to a natural (optimized) batched functional commitment $\mathsf{BatchMsg}[\mathsf{FC}]$ for the batched query class $\mathbf{Q}_{\mathsf{BatchMsg}}[\mathbf{Q}, s]$, we give a detailed construction in Construction 8.6. We show that if FC is state-restoration function binding, then so is $\mathsf{BatchMsg}[\mathsf{FC}]$:

Lemma 8.2 (state-restoration function binding of $\mathsf{BatchMsg}[\mathsf{FC}, s]$)**.** *Let FC be a non-interactive, triply homomorphic FC scheme(Definition 8.5) with expected-time function binding error $\epsilon^{\star}_{\mathsf{FC}} = \epsilon^{\star}_{\mathsf{FC}}(\lambda, \ell, \mathsf{L}, t^{\star}_{\mathsf{FC}})$. Then for any batch size $s \in \mathbb{N}$, the batch FC scheme $\mathsf{BatchMsg}[\mathsf{FC}, s]$ for the query class $\mathbf{Q}_{\mathsf{BatchMsg}}[\mathbf{Q}, s]$ has state-restoration function binding error $\epsilon^{\mathsf{SR}}_{\mathsf{FC}}$ such that for every security parameter $\lambda \in \mathbb{N}$, message length $\ell \in \mathbb{N}$, sample set size $\mathsf{L} \in \mathbb{N}$, salt size $s_{\mathsf{FC}} \in \mathbb{N}$, move budget $\mathsf{m}_{\mathsf{FC}} \in \mathbb{N}$, circuit size bound $t_{\mathsf{bFC}} \in \mathbb{N}$,*

$$\epsilon^{\mathsf{SR}}_{\mathsf{bFC}}(\lambda, s\cdot\ell, \mathsf{L}, s_{\mathsf{FC}}, \mathsf{m}_{\mathsf{FC}}, t_{\mathsf{bFC}}) \le \mathsf{L}\cdot(\mathsf{m}_{\mathsf{FC}}+1)\cdot\frac{s-1}{2^{\lambda}} + \epsilon^{\star}_{\mathsf{FC}}(\lambda, \ell, \mathsf{L}, t^{\star}_{\mathsf{FC}}) \ ,$$

where $t^{\star}_{\mathsf{FC}} \le \mathsf{L}(\mathsf{m}_{\mathsf{FC}}+1)(s-1)\,(t_{\mathsf{bFC}} + \mathsf{poly}(s\cdot\ell)) + s\cdot t_{\mathbf{Q}}(\ell, \mathsf{L}) + \mathsf{L}\cdot(s^3+s^2)$.

Linearization trick. We consider the query class $\mathbf{Q}_{\mathsf{Struct}}[\mathbf{Q}, m, (\mathsf{h}_k)_{k\in[n]}]$ (Definition 5.9), where queries are non-linear combinations (parametrized by $(\mathsf{h}_k)_{k\in[n]}$) 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].

Given a batch functional commitment bFC, we construct an efficient functional commitment $\mathsf{Lin}[\mathsf{bFC}]$ for $\mathbf{Q}_{\mathsf{Struct}}[\mathbf{Q}, m, (\mathsf{h}_k)_{k\in[n]}]$ in Construction 8.8 by applying the linearization trick, and we show that if bFC is state-restoration function binding, then so is $\mathsf{Lin}[\mathsf{bFC}, m, (\mathsf{h}_k)_{k\in[n]}]$:

Lemma 8.3 (state-restoration function binding for linearized FCs)**.** *Let bFC be a batch polynomial commitment scheme with state-restoration function binding error $\epsilon^{\mathsf{SR}}_{\mathsf{bFC}} = \epsilon^{\mathsf{SR}}_{\mathsf{bFC}}(\lambda, (m+n)\ell, \mathsf{L}, s_{\mathsf{FC}}, \mathsf{m}_{\mathsf{FC}}, t_{\mathsf{bFC}})$. For any $m \in \mathbb{N}$ and public polynomials $(\mathsf{h}_k)_{k\in[n]}$, the linearized functional commitment scheme $\mathsf{linFC}[\mathsf{bFC}, m, (\mathsf{h}_k)_{k\in[n]}]$ has state-restoration function binding error $\epsilon^{\mathsf{SR}}_{\mathsf{linFC}}$ such that for every security parameter $\lambda \in \mathbb{N}$, polynomial degree bound $\mathsf{D} \in \mathbb{N}$, sample set size $\mathsf{L} \in \mathbb{N}$, and adversary size bound $t_{\mathsf{linFC}} \in \mathbb{N}$,*

$$\epsilon^{\mathsf{SR}}_{\mathsf{linFC}}(\lambda, (m+n)(\mathsf{D}+1), \mathsf{L}, s_{\mathsf{FC}}, \mathsf{m}_{\mathsf{FC}}, t_{\mathsf{linFC}}) \le \epsilon^{\mathsf{SR}}_{\mathsf{bFC}}(\lambda, (m+1)(\mathsf{D}+1), \mathsf{L}, s_{\mathsf{FC}}, \mathsf{m}_{\mathsf{FC}}, t_{\mathsf{bFC}}) \ ,$$

where $t_{\mathsf{bFC}} \le t_{\mathsf{linFC}}$.

Remark 8.4. Note that we place no restrictions on the public polynomials $(\mathsf{h}_k)_{k\in[n]}$ in Definition 5.9 and Lemma 8.3. This is in contrast to previous analyses of the linearization trick [FFR24], which required the public polynomials to be linearly independent. This restriction is a by-product of requiring extractability of the commitment scheme; since we merely target function binding, we are able to lift this restriction.

8.1 Proof of Lemma 8.2 (batched-messages FC)

Definition 8.5. *Let* FC *be a non-interactive functional commitment scheme for a query set of functions* $\alpha \in \Sigma^{\ell} \to \mathbb{D}$*. 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* $\mathsf{cm}, \mathsf{aux}, \mathsf{aux}', \alpha, \beta, \beta', \mathsf{pf}, \mathsf{pf}'$*:*

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

Construction 8.6 (FC for batched messages)**.** Let FC be a triply homomorphic functional commitment for the query class $\mathbf{Q}$. For any $s \in \mathbb{N}$, we construct a functional commitment $\mathsf{bFC} := \mathsf{BatchMsg}[\mathsf{FC}, s]$ for the query class $\mathbf{Q}_{\mathsf{BatchMsg}}[\mathbf{Q}, s]$ as follows:

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

In order to prove that $\mathsf{BatchMsg}[\mathsf{FC}, s]$ satisfies state-restoration function binding (Definition 3.25), we prove that it is s-special function binding (Definition A.1). Lemma 8.2 follows from Lemmas 8.7 and A.2.

Lemma 8.7. *Let* FC *be a functional commitment scheme with function binding error* $\epsilon_{\mathsf{FC}} = \epsilon_{\mathsf{FC}}(\lambda, \ell, \mathsf{L}, t_{\mathsf{FC}})$*. Then for any batch size* $s \in \mathbb{N}$*,* $\mathsf{BatchMsg}[\mathsf{FC}, s]$ *has* s*-special function binding error* $\epsilon_{\mathsf{bSFB}} = \epsilon_{\mathsf{bSFB}}(\lambda, \ell, s \cdot \mathsf{L}, t^{\star}_{\mathsf{bSFB}})$ *such that for every security parameter* $\lambda \in \mathbb{N}$*, message size* $\ell \in \mathbb{N}$*, sample set size* $\mathsf{L} \in \mathbb{N}$*, and expected adversary size bound* $t^{\star}_{\mathsf{bSFB}} \in \mathbb{N}$*,*

$$\epsilon_{\mathsf{bSFB}}(\lambda, s \cdot \ell, \mathsf{L}, t^{\star}_{\mathsf{bSFB}}) \leq \epsilon_{\mathsf{FC}}(\lambda, \ell, \mathsf{L}, t_{\mathsf{FC}}) \ ,$$

where $t_{\mathsf{FC}} \leq t^{\star}_{\mathsf{bSFB}} + s \cdot t_{\mathbf{Q}}(\ell, \mathsf{L}) + \mathsf{L} \cdot (s^3 + s^2)$.

Proof of Lemma 8.7. Given an s-special function binding adversary A_{bSFB} against bFC, we construct the following function binding adversary A_{FC} against FC:

$A_{\mathsf{FC}}(\mathsf{pp}, \mathsf{ai}_{\mathsf{FC}})$:
1. Run $(\mathsf{cm}, ((\alpha_i, \beta_i, \mathsf{T}_i))_{i \in [\mathsf{L}]}) \leftarrow A_{\mathsf{bSFB}}(\mathsf{pp}, \mathsf{ai}_{\mathsf{FC}})$
2. If there is no Π such that $\forall\, i \in [\mathsf{L}] : \alpha_i(\Pi) = \beta_i$, output $\perp$.
3. Parse cm as $(\mathsf{cm}_b)_{b \in [s]}$.
4. For $i \in [\mathsf{L}]$:
   (a) Parse β_i as $(\beta_{i,b})_{b \in [s]}$.

(b) Parse T_i as verifier challenges $(\gamma_i^{(b)})_{b\in[s]}$ and final prover messages $(\mathsf{pf}_i^{(b)})_{b\in[s]}$.

(c) Compute $V_i \coloneqq \begin{bmatrix} 1 & \gamma_i^{(1)} & \cdots & (\gamma_i^{(1)})^{s-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \gamma_i^{(1)} & \cdots & (\gamma_i^{(s)})^{s-1} \end{bmatrix}$.

(d) If V_i is invertible, compute $V_i^{-1} = \begin{bmatrix} \bar{\gamma}_{i,1,1} & \cdots & \bar{\gamma}_{i,1,s-1} \\ \vdots & \ddots & \vdots \\ \bar{\gamma}_{i,s,1} & \cdots & \bar{\gamma}_{i,s,s} \end{bmatrix}$, and output $\perp$ otherwise.

(e) Set $\begin{bmatrix} \mathsf{pf}'_{i,1} \\ \vdots \\ \mathsf{pf}'_{i,s} \end{bmatrix} \coloneqq V_i^{-1} \begin{bmatrix} \mathsf{pf}_i^{(1)} \\ \vdots \\ \mathsf{pf}_i^{(s)} \end{bmatrix}$.

5. For $j \in [s]$:

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

6. Output $\perp$.

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

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

$$\mathsf{FC.Check}(\sum_{b\in[s]} (\gamma_i^{(k)})^{b-1} \cdot \mathsf{cm}_b, \alpha, \sum_{b\in[s]} (\gamma_i^{(k)})^{b-1} \cdot \beta_i^{(b)}, \mathsf{pf}_i^{(k)}) = 1 \ .$$

By construction, we have that for all $i \in [\mathsf{L}]$ and $j \in [s]$,

$$\sum_{k\in[s]} \bar{\gamma}_{i,j,k} \sum_{b\in[s]} (\gamma_i^{(k)})^{b-1} \cdot \mathsf{cm}_b = \mathsf{cm}_j \ ,$$
$$\sum_{k\in[s]} \bar{\gamma}_{i,j,k} \sum_{b\in[s]} (\gamma_i^{(k)})^{b-1} \cdot \beta_{i,b} = \beta_{i,j} \ .$$

By triple homomorphism, this implies that for all $i \in [\mathsf{L}]$ and $j \in [s]$:

$$\begin{aligned} &\mathsf{FC.Check}(\mathsf{cm}_j, \alpha_i, \beta_{i,j}, \mathsf{pf}'_{i,j}) \\ &= \mathsf{FC.Check}(\sum_{k\in[s]} \bar{\gamma}_{i,j,k} \sum_{b\in[s]} (\gamma_i^{(k)})^{b-1} \cdot \mathsf{cm}_b, \alpha_i, \sum_{k\in[s]} \bar{\gamma}_{i,j,k} \sum_{b\in[s]} (\gamma_i^{(k)})^{b-1} \cdot \beta_{i,b}, \sum_{k\in[s]} \bar{\gamma}_{i,j,k} \cdot \mathsf{pf}_i^{(k)}) \\ &= 1 \ , \end{aligned}$$

i.e., $\mathsf{pf}'_{i,j}$ is an accepting proof for $(\mathsf{cm}_j, \alpha_i, \beta_{i,j})$.

Further, the function binding condition

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

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

$$\neg(\exists\, j \in [s] : \nexists\, \Pi_j : \forall\, i \in [\mathsf{L}] : \alpha_i(\Pi_j) = \beta_{i,j})$$

$$\equiv \forall\, j \in [s] : \exists\, \Pi_j : \forall\, i \in [\mathsf{L}] : \alpha_i(\Pi_j) = \beta_{i,j}$$
$$\Rightarrow \exists\, (\Pi_b)_{b\in[s]} : \forall\, i \in [\mathsf{L}] : \forall\, j \in [s] : \alpha_i(\Pi_j) = \beta_{i,j}\ ,$$

which contradicts our assumption on the success of A_{bSFB}. □

8.2 Proof of Lemma 8.3 (linearization trick)

We recall a generalized formulation of the linearization trick (also known as Maller's trick) [GWC19] [LPS24b, Section 4.2], [FFR24]. More specifically, given a batched functional commitment scheme bFC, we construct a functional commitment scheme $\mathsf{Lin}[\mathsf{bFC}]$ for $\mathbf{Q}_{\mathsf{Struct}}[\mathbf{Q}, m, (\mathsf{h}_k)_{k\in[n]}]$ (Definition 5.9).

Construction 8.8. Let $\mathbb{F}$ be a field, $m, n \in \mathbb{N}$, and let $\mathbf{Q}$ be a query class over the alphabet Σ with length ℓ. Further, let bFC be a batch functional commitment for the query class $\mathbf{Q}_{\mathsf{BatchMsg}}[\mathbf{Q}, m+1]$, where both Σ^ℓ and the commitment space $\mathbb{C}$ are $\mathbb{F}$-modules, and such that (i) batch commitments are done element-wise, i.e., $\mathsf{bFC.Commit}(\mathsf{pp}_{\mathsf{FC}}, (\Pi_b)_{b\in[s]}) = (\mathsf{Commit}(\mathsf{pp}_{\mathsf{FC}}, \Pi_b))_{b\in[s]}$, and (ii) commitments are homomorphic, i.e., $\forall\, \Pi, \Pi' : \mathsf{Commit}(\mathsf{pp}_{\mathsf{FC}}, \Pi + \Pi') = \mathsf{Commit}(\mathsf{pp}_{\mathsf{FC}}, \Pi) + \mathsf{Commit}(\mathsf{pp}_{\mathsf{FC}}, \Pi')$.

For any public polynomials $\mathsf{h}_1, \ldots, \mathsf{h}_n \in \mathbb{F}[X_1, \ldots, X_m]^{\leq \mathsf{D}_\mathsf{h}}$, we construct a functional commitment scheme $\mathsf{linFC} := \mathsf{Lin}[\mathsf{bFC}]$ for the query class $\mathbf{Q}_{\mathsf{Struct}}[\mathbf{Q}, m, (\mathsf{h}_k)_{k\in[n]}]$ as follows:

- $\mathsf{linFC.Gen}(1^\lambda, m, n, \ell)$: Output $\mathsf{pp}_{\mathsf{FC}} \leftarrow \mathsf{bFC.Gen}(1^\lambda, m+1, \ell)$.
- $\mathsf{linFC.Commit}(\mathsf{pp}_{\mathsf{FC}}, \Pi)$:
  1. Parse Π as $((\mathsf{f}_k)_{k\in[m]}, (\mathsf{g}_\ell)_{\ell\in[n]})$
  2. For $k \in [m]$: Compute $(\mathsf{cm}_{\mathsf{f}_k}, \mathsf{aux}_{\mathsf{f}_k}) \leftarrow \mathsf{Commit}(\mathsf{pp}_{\mathsf{FC}}, \mathsf{f}_k)$.
  3. For $\ell \in [n]$: Compute $(\mathsf{cm}_{\mathsf{g}_\ell}, \mathsf{aux}_{\mathsf{g}_\ell}) \leftarrow \mathsf{Commit}(\mathsf{pp}_{\mathsf{FC}}, \mathsf{g}_\ell)$.
  4. Output $\mathsf{cm} := ((\mathsf{cm}_{\mathsf{f}_k})_{k\in[m]}, (\mathsf{cm}_{\mathsf{g}_\ell})_{\ell\in[n]})$ and $\mathsf{aux} := ((\mathsf{aux}_{\mathsf{f}_k})_{k\in[m]}, (\mathsf{aux}_{\mathsf{g}_\ell})_{\ell\in[n]})$
- $\langle \mathcal{P}_{\mathsf{linFC}}(\mathsf{pp}_{\mathsf{FC}}, \mathsf{cm}, \mathsf{aux}, \alpha, \beta, \Pi), \mathcal{V}_{\mathsf{linFC}}(\mathsf{pp}_{\mathsf{FC}}, \mathsf{cm}, \mathsf{aux}, \alpha, \beta)\rangle$:
  1. $\mathcal{P}_{\mathsf{linFC}}$ and $\mathcal{V}_{\mathsf{linFC}}$ both parse β as $(\beta_1, \ldots, \beta_m, \beta')$.
  2. Run $\langle \mathcal{P}_{\mathsf{bFC}}, \mathcal{V}_{\mathsf{bFC}}\rangle$ for the commitment $(\mathsf{cm}_{\mathsf{f}_1}, \ldots, \mathsf{cm}_{\mathsf{f}_m}, \sum_{\ell\in[n]} \mathsf{h}_\ell(\beta_1, \ldots, \beta_m) \cdot \mathsf{cm}_{\mathsf{g}_\ell})$, the query α, evaluations $(\beta_1, \ldots, \beta_m, \beta')$, and for the witness $(\mathsf{f}_1, \ldots, \mathsf{f}_m, \sum_{\ell\in[n]} \mathsf{h}_\ell(\beta_1, \ldots, \beta_m) \cdot \mathsf{g}_\ell)$.

Proof of Lemma 8.3. Given an adversary $\widetilde{\mathcal{P}}^{\mathsf{SR}}_{\mathsf{linFC}}$ against the state-restoration function binding of linFC, we construct the following state-restoration function binding adversary $\widetilde{\mathcal{P}}^{\mathsf{SR}}_{\mathsf{bFC}}$ against bFC.

$\widetilde{\mathcal{P}}^{\mathsf{SR}}_{\mathsf{bFC}}(\mathsf{pp}_{\mathsf{FC}}, \mathsf{ai}_{\mathsf{FC}})$:

1. Simulate $\mathsf{FCSRGame}$ for $\widetilde{\mathcal{P}}^{\mathsf{SR}}_{\mathsf{linFC}}$ with salt size s_{FC}, public parameters $\mathsf{pp}_{\mathsf{FC}}$, and auxiliary input $\mathsf{ai}_{\mathsf{FC}}$ as follows. Repeat the following until $\widetilde{\mathcal{P}}^{\mathsf{SR}}_{\mathsf{linFC}}(\mathsf{pp}_{\mathsf{FC}}, \mathsf{ai}_{\mathsf{FC}})$ decides to exit the loop:
   (a) $\widetilde{\mathcal{P}}^{\mathsf{SR}}_{\mathsf{linFC}}$ makes a move $(((\mathsf{cm}_{\mathsf{f}_k})_{k\in[m]}, (\mathsf{cm}_{\mathsf{g}_\ell})_{\ell\in[n]}), \alpha, (\beta_j)_{j\in[m+1]}, (\mathsf{pm}_j)_{j\in[i]}, (\eta_j)_{j\in[i]})$.
   (b) Make a move $((\mathsf{cm}_{\mathsf{f}_1}, \ldots, \mathsf{cm}_{\mathsf{f}_m}, \sum_{\ell\in[n]} \mathsf{h}_\ell(\beta_1, \ldots, \beta_m)\cdot\mathsf{cm}_{\mathsf{g}_\ell}), \alpha, (\beta_j)_{j\in[m+1]}, (\mathsf{pm}_j)_{j\in[i]}, (\eta_j)_{j\in[i]})$ in $\widetilde{\mathcal{P}}^{\mathsf{SR}}_{\mathsf{bFC}}$'s $\mathsf{FCSRGame}$ to get vm_i.
   (c) Answer $\widetilde{\mathcal{P}}^{\mathsf{SR}}_{\mathsf{linFC}}$'s move with vm_i.
2. Get the $\mathsf{FCSRGame}$ output $(\mathsf{cm}, ((\alpha_i, \beta_i, ((\mathsf{pm}_{i,j}, \eta_{i,j}, \mathsf{vm}_{i,j}))_{j\in[\mathsf{k}_{\mathsf{FC}}]}))_{i\in[\mathsf{L}]}) \leftarrow \widetilde{\mathcal{P}}^{\mathsf{SR}}_{\mathsf{linFC}}(\mathsf{pp}_{\mathsf{FC}}, \mathsf{ai}_{\mathsf{FC}})$.
3. Parse cm as $(\mathsf{cm}_{\mathsf{f}_1}, \ldots, \mathsf{cm}_{\mathsf{f}_m}, \mathsf{cm}_{\mathsf{g}_1}, \ldots, \mathsf{cm}_{\mathsf{g}_n})$ and β_i as $(\beta_{i,1}, \ldots, \beta_{i,m}, \beta'_i)$ for all $i \in [\mathsf{L}]$.
4. Set $\mathsf{cm}' := (\mathsf{cm}_{\mathsf{f}_1}, \ldots, \mathsf{cm}_{\mathsf{f}_m}, \sum_{\ell\in[n]} \mathsf{h}_\ell(\beta_1, \ldots, \beta_m) \cdot \mathsf{cm}_{\mathsf{g}_\ell})$.
5. Output $(\mathsf{cm}', ((\alpha_i, (\beta_{i,1}, \ldots, \beta_{i,m}, \beta'_i), ((\mathsf{pm}_{i,j}, \eta_{i,j}, \mathsf{vm}_{i,j}))_{j\in[\mathsf{k}_{\mathsf{FC}}]}))_{i\in[\mathsf{L}]})$

Success probability. If $\widetilde{\mathcal{P}}^{\mathsf{SR}}_{\mathsf{linFC}}$ wins the state-restoration function binding game, then

$$\forall\, \mathsf{f}_1, \ldots, \mathsf{f}_m, \mathsf{g}_1, \ldots, \mathsf{g}_n \in \Sigma^\ell : \exists\, i \in [\mathsf{L}] : \sum_{k\in[n]} \mathsf{h}_k(\alpha_i(\mathsf{f}_1), \ldots, \alpha_i(\mathsf{f}_m)) \cdot \alpha_i(\mathsf{g}_k) \neq \beta_i \ , \text{ and}$$

$\mathcal{V}_{\mathsf{bFC}}(\mathsf{pp}_{\mathsf{FC}}, \mathsf{cm}', \alpha_i, \beta_i, ((\mathsf{pm}_{i,j}, \mathsf{vm}_{i,j}))_{j\in[\mathsf{k}_{\mathsf{FC}}]}) = \mathcal{V}_{\mathsf{linFC}}(\mathsf{pp}_{\mathsf{FC}}, \mathsf{cm}, \alpha_i, \beta_i, ((\mathsf{pm}_{i,j}, \mathsf{vm}_{i,j}))_{j\in[\mathsf{k}_{\mathsf{FC}}]}) = 1$ for all $i \in [\mathsf{L}]$.

We distinguish two cases:

1. $\exists\, \mathsf{f}_1, \ldots, \mathsf{f}_m \in \Sigma^\ell : \forall\, i \in [\mathsf{L}] : \forall\, k \in [n] : \alpha_i(\mathsf{f}_k) = \beta_{i,k}$, i.e., there exist f_k which are consistent with all query-evaluation pairs. This implies that for these $\mathsf{f}_1, \ldots, \mathsf{f}_m$,

   $$\forall\, \mathsf{g}_1, \ldots, \mathsf{g}_n \in \Sigma^\ell : \exists\, i \in [\mathsf{L}] : \sum_{k\in[n]} \mathsf{h}_k(\alpha_i(\mathsf{f}_1), \ldots, \alpha_i(\mathsf{f}_m)) \cdot \alpha_i(\mathsf{g}_k) \neq \beta_i \ .$$

   Since $\forall\, i \in [\mathsf{L}] : \forall\, k \in [n] : \alpha_i(\mathsf{f}_k) = \beta_{i,k}$, this is equivalent to

   $$\forall\, \mathsf{g}_1, \ldots, \mathsf{g}_n \in \Sigma^\ell : \exists\, i \in [\mathsf{L}] : \sum_{k\in[n]} \mathsf{h}_k(\beta_{i,1}, \ldots, \beta_{i,m}) \cdot \alpha_i(\mathsf{g}_k) \neq \beta_i \ ,$$

   which directly implies the function binding condition for the batch FC (since the last function in the batch will not be consistent with all query-evaluation pairs).

2. $\forall\, \mathsf{f}_1, \ldots, \mathsf{f}_m \in \Sigma^\ell : \exists\, i \in [\mathsf{L}] : \exists\, k \in [n] : \alpha_i(\mathsf{f}_k) \neq \beta_{i,k}$. This also implies the function binding condition for the batch FC, since the sub-batch containing the first m entries will satisfy the function binding condition.

□

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 9.5) holds with error $\epsilon_{\mathsf{ARSDH}} = \epsilon_{\mathsf{ARSDH}}(\lambda, \mathsf{D}, t_{\mathsf{ARSDH}})$. The KZG polynomial commitment scheme (Construction 9.4) has function binding error ϵ_{FC} such that for every security parameter $\lambda \in \mathbb{N}$, polynomial degree bound $\mathsf{D} \in \mathbb{N}$, sample set size $\mathsf{L} \in \mathbb{N}$, and adversary size bound $t_{\mathsf{FC}} \in \mathbb{N}$,*

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

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

In Section 9.2, we then show that the batched PC $\mathsf{bKZG} = \mathsf{BatchMsg}[\mathsf{KZG}, s]$ 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 9.7) holds with error $\epsilon^{\star}_{\mathsf{ARSDH}} = \epsilon^{\star}_{\mathsf{ARSDH}}(\lambda, \mathsf{D}, t^{\star}_{\mathsf{ARSDH}})$. Then for any batch size $s \in \mathbb{N}$, the batch polynomial commitment scheme $\mathsf{bKZG} = \mathsf{BatchMsg}[\mathsf{KZG}, s]$ has state-restoration function binding error $\epsilon^{\mathsf{SR}}_{\mathsf{FC}}$ such that for every security parameter $\lambda \in \mathbb{N}$, polynomial degree bound $\mathsf{D} \in \mathbb{N}$, sample set size $\mathsf{L} \in \mathbb{N}$, and adversary size bound $t_{\mathsf{bFC}} \in \mathbb{N}$,*

$$\epsilon^{\mathsf{SR}}_{\mathsf{bFC}}(\lambda, (\mathsf{D}+1)s, \mathsf{L}, s_{\mathsf{FC}}, \mathsf{m}_{\mathsf{FC}}, t_{\mathsf{bFC}}) \leq \mathsf{L} \cdot (\mathsf{m}_{\mathsf{FC}}+1) \cdot \frac{s-1}{2^{\lambda}} + \epsilon^{\star}_{\mathsf{ARSDH}}(\lambda, \mathsf{D}, t^{\star}_{\mathsf{ARSDH}}) \ ,$$

where $t^{\star}_{\mathsf{ARSDH}} \leq \mathsf{L}(\mathsf{m}_{\mathsf{FC}}+1)(s-1)\left(t_{\mathsf{bFC}} + \mathsf{poly}(s \cdot \mathsf{D}, \mathsf{L})\right) + s \cdot ((\mathsf{D}+1)^2 + (\mathsf{D}+1) \cdot \mathsf{L}) + \mathsf{L} \cdot (s^3 + s^2)$.

In Section 9.3, we show that the linearized KZG functional commitment $\mathsf{linKZG} = \mathsf{linFC}[\mathsf{bKZG}, m, (\mathsf{h}_k)_{k\in[n]}]$ (as used in the Plonk construction [GWC19]) is state-restoration function binding.

Corollary 9.3 (Linearized KZG is state-restoration function binding)**.** *Suppose the expected-time ARSDH assumption (Definition 9.7) holds with error $\epsilon^{\star}_{\mathsf{ARSDH}} = \epsilon^{\star}_{\mathsf{ARSDH}}(\lambda, \mathsf{D}, t^{\star}_{\mathsf{ARSDH}})$. Then for any $m \in \mathbb{N}$ and public polynomials $(\mathsf{h}_k)_{k\in[n]}$, the linearized KZG functional commitment scheme $\mathsf{linKZG} = \mathsf{linFC}[\mathsf{bKZG}, m, (\mathsf{h}_k)_{k\in[n]}]$ (Construction 8.8) has state-restoration function binding error $\epsilon^{\mathsf{SR}}_{\mathsf{FC}}$ such that for every security parameter $\lambda \in \mathbb{N}$, polynomial degree bound $\mathsf{D} \in \mathbb{N}$, sample set size $\mathsf{L} \in \mathbb{N}$, and adversary size bound $t_{\mathsf{linFC}} \in \mathbb{N}$,*

$$\epsilon^{\mathsf{SR}}_{\mathsf{linFC}}(\lambda, (m+n)(\mathsf{D}+1), \mathsf{L}, s_{\mathsf{FC}}, \mathsf{m}_{\mathsf{FC}}, t_{\mathsf{linFC}}) \leq \mathsf{L} \cdot (\mathsf{m}_{\mathsf{FC}}+1) \cdot \frac{m}{2^{\lambda}} + \epsilon^{\star}_{\mathsf{ARSDH}}(\lambda, \mathsf{D}, t^{\star}_{\mathsf{ARSDH}}) \ ,$$

where

$$t^{\star}_{\mathsf{ARSDH}} \leq \mathsf{L}(\mathsf{m}_{\mathsf{FC}}+1) \cdot m \cdot (t_{\mathsf{linFC}} + \mathsf{poly}((m+1)\mathsf{D}, \mathsf{L})) + (m+1) \cdot ((\mathsf{D}+1)^2 + (\mathsf{D}+1) \cdot \mathsf{L}) + \mathsf{L} \cdot ((m+1)^3 + (m+1)^2) \ .$$

9.1 Proof of Lemma 9.1 (KZG)

Construction 9.4 (KZG)**.** The FC scheme KZG for the query class $\mathbf{Q}_{\mathsf{UniPoly}}$ is defined as follows.

- $\mathsf{KZG.Gen}(1^{\lambda}, \mathsf{D})$:

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 $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a bilinear pairing.
2. Sample $\tau \leftarrow \mathbb{Z}_p \setminus \{0\}$.
3. Output $\mathsf{pp}_{\mathsf{PC}} \coloneqq ([1]_1, [\tau]_1, [\tau^2]_1, \ldots, [\tau^{\mathsf{D}}]_1, [1]_2, [\tau]_2)$.

- $\mathsf{KZG.Commit}(\mathsf{pp}_{\mathsf{PC}}, \mathsf{p})$:
  1. Parse $\mathsf{pp}_{\mathsf{PC}}$ as $([1]_1, [\tau]_1, [\tau^2]_1, \ldots, [\tau^{\mathsf{D}}]_1, [1]_2, [\tau]_2)$.
  2. Compute $\mathsf{cm} \coloneqq [\mathsf{p}(\tau)]_1$.
  3. Set $\mathsf{aux} \coloneqq \mathsf{p}$.
  4. Output $(\mathsf{cm}, \mathsf{aux})$.

- $\mathsf{KZG.Open}(\mathsf{pp}_{\mathsf{PC}}, \mathsf{aux}, \alpha, \beta)$:
  1. Parse $\mathsf{pp}_{\mathsf{PC}}$ as $([1]_1, [\tau]_1, [\tau^2]_1, \ldots, [\tau^{\mathsf{D}}]_1, [1]_2, [\tau]_2)$ and aux as p.
  2. Compute $\mathsf{p}_\alpha(X) \coloneqq \frac{\mathsf{p}(X)-\beta}{X-\alpha}$.
  3. Compute $\mathsf{pf} \coloneqq [\mathsf{p}_\alpha(\tau)]_1$.
  4. Output pf.

- $\mathsf{KZG.Check}(\mathsf{pp}_{\mathsf{PC}}, \mathsf{cm}, \alpha, \beta, \mathsf{pf})$:
  1. Parse $\mathsf{pp}_{\mathsf{PC}}$ as $([1]_1, [\tau]_1, [\tau^2]_1, \ldots, [\tau^{\mathsf{D}}]_1, [1]_2, [\tau]_2)$.
  2. Check that $e(\mathsf{cm} - [\beta]_1, [1]_2) = e(\mathsf{pf}, [\tau - \alpha]_2)$.

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

Definition 9.5 (ARSDH assumption [LPS24a])**.** *Let $\mathbb{F}$ be a field and $Z_S(X) \coloneqq \prod_{s\in S}(X - s)$ for $S \subseteq \mathbb{F}$, the* **adaptive rational strong Diffie-Hellman (ARSDH)** *holds with error $\epsilon_{\mathsf{ARSDH}}$ 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 $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a bilinear pairing,*

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

Proof of Lemma 9.1. Fix the security parameter $\lambda \in \mathbb{N}$, polynomial degree bound $\mathsf{D} \in \mathbb{N}$, sample set size $\mathsf{L} \in \mathbb{N}$, and adversary size bound $t_{\mathsf{FC}} \in \mathbb{N}$. For every auxiliary input distribution $\mathcal{D}_{\mathsf{FC}}$ and t_{FC}-size adversary A_{FC} against function binding, consider the following adversary A_{ARSDH} against ARSDH. Below, $\mathsf{Lagrange}(\{(x_i, y_i)\}_{i\in[n]})$ denotes the unique polynomial of degree at most $n-1$ that interpolates all points $\{(x_i, y_i)\}_{i\in[n]}$.

$A_{\mathsf{ARSDH}}([1]_1, [\tau]_1, [\tau^2]_1, \ldots, [\tau^{\mathsf{D}}]_1, [1]_2, [\tau]_2)$:
1. Sample $\mathsf{ai}_{\mathsf{FC}} \leftarrow \mathcal{D}_{\mathsf{FC}}$ and set $\mathsf{pp}_{\mathsf{PC}} \coloneqq ([1]_1, [\tau]_1, [\tau^2]_1, \ldots, [\tau^{\mathsf{D}}]_1, [1]_2, [\tau]_2)$.
2. Run $(\mathsf{cm}, \{(\alpha_i, \beta_i, \mathsf{pf}_i)\}_{i\in[\mathsf{L}]}) \leftarrow A_{\mathsf{FC}}(\mathsf{pp}_{\mathsf{PC}}, \mathsf{ai}_{\mathsf{FC}})$.
3. If there exists $i, j \in [\mathsf{L}]$ such that $\alpha_i = \alpha_j$, $\beta_i \neq \beta_j$:
   (a) Choose S to be a size-$(\mathsf{D}+1)$ subset of $\mathbb{F}$ such that $\alpha_i \in S$ and $[Z_S(\tau)]_1 \neq [0]_1$.

(b) Set $h_1 \coloneqq [Z_{S\setminus\{\alpha_i\}}(\tau)]_1$.
(c) Set $h_2 \coloneqq \frac{\mathsf{pf}_i - \mathsf{pf}_j}{\beta_j - \beta_i}$.
(d) Output (S, h_1, h_2).

4. If $\deg(\mathsf{Lagrange}(\{(\alpha_i, \beta_i)\}_{i\in[\mathsf{L}]})) > \mathsf{D}$:
   (a) Let L_0 be the interpolation of the first $\mathsf{D}+1$ distinct points, denoted without loss of generality as $\{\alpha_i\}_{i\in[\mathsf{D}+1]}$.
   (b) Find (α_k, β_k) for $k \in \{\mathsf{D}+2, \ldots, \mathsf{L}\}$ such that $\beta_k \neq L_0(\alpha_k)$.
   (c) Sample $S \coloneqq \{\alpha'_i\}_{i\in[\mathsf{D}+1]}$ from $\{\alpha_i\}_{i\in[\mathsf{D}+1]\cup\{k\}}$ such that $[\mathsf{Lagrange}(\{(\alpha, \beta[\alpha])\}_{\alpha\in S})]_1 \neq \mathsf{cm}$.
   (d) Set $L(X) \coloneqq \mathsf{Lagrange}(\{\alpha'_i, \beta'_i\}_{i\in[\mathsf{D}+1]})$.
   (e) Set $h_1 \coloneqq \mathsf{cm} - [L(\tau)]_1$.
   (f) Compute $d_i \coloneqq \frac{1}{Z_{S\setminus\{\alpha'_i\}}(\alpha'_i)}$ for each $i \in [\mathsf{D}+1]$.
   (g) Set $h_2 \coloneqq \sum_{i=1}^{\mathsf{D}+1} d_i \cdot \mathsf{pf}'_i$.
   (h) Output (S, h_1, h_2).
5. Otherwise, output $(\bot, \bot, \bot)$.

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

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

1. There are two valid openings $(\alpha, \beta_i, \mathsf{pf}_i)$ and $(\alpha, \beta_j, \mathsf{pf}_j)$ where $\beta_i \neq \beta_j$. Step 3 in A_{ARSDH} finds the two inconsistent openings and compute $h_2 \coloneqq \frac{\mathsf{pf}_i - \mathsf{pf}_j}{\beta_j - \beta_i} = [\frac{1}{\tau-\alpha}]_1$, which implies that $h_2 = [\frac{1}{\tau-\alpha}]_1 = \frac{1}{Z_S(\tau)}\cdot h_1$.

2. $\deg(\mathsf{Lagrange}(\{(\alpha_i, \beta_i)\}_{i\in[\mathsf{L}]})) > \mathsf{D}$. According to the construction of A_{ARSDH}, the following holds:

   - There must exist (α_k, β_k) for some $k \in \{\mathsf{D}+2, \ldots, \mathsf{L}\}$ such that $\beta_k \neq L_0(\alpha_k)$, because otherwise $\deg(\mathsf{Lagrange}(\{(\alpha_i, \beta_i)\}_{i\in[\mathsf{L}]})) = \deg(L_0) \leq \mathsf{D}$.
   - $\deg(\mathsf{Lagrange}(\{(\alpha_i, \beta_i)\}_{i\in[\mathsf{D}+1]\cup\{k\}})) \geq \mathsf{D}+1$, because otherwise, by the fundamental theorem of algebra, we would have $\mathsf{Lagrange}(\{(\alpha_i, \beta_i)\}_{i\in[\mathsf{D}+1]\cup\{k\}}) = L_0$.
   - For all size-$(\mathsf{D}+1)$ subsets of $\{(\alpha_i, \beta_i)\}_{i\in[\mathsf{D}+1]\cup\{k\}}$, their interpolations are pairwise distinct, because otherwise $\deg(\mathsf{Lagrange}(\{(\alpha_i, \beta_i)\}_{i\in[\mathsf{D}+1]\cup\{k\}})) \leq \mathsf{D}$.

   Therefore, there are at least $\mathsf{D}+1$ among $\mathsf{D}+2$ possible subsets such that the corresponding interpolation is not consistent with cm, which implies that $h_1 = \mathsf{cm} - [L(\tau)]_1 \neq [0]_1$, and $h_2 = \sum_{i=1}^{\mathsf{D}+1} d_i \cdot \mathsf{pf}'_i = \frac{1}{Z_S(\tau)}\cdot(\mathsf{cm} - [L(\tau)]_1) = \frac{1}{Z_S(\tau)}\cdot h_1$.

□

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 $\mathsf{bKZG} = \mathsf{BatchMsg}[\mathsf{KZG}, s]$ using Construction 8.6. Thus, Corollary 9.2 follows from Lemmas 8.2 and 9.1.

Lemma 9.6. KZG *is triply homomorphic (Definition 8.5).*

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

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

□

Definition 9.7 (Expected-time ARSDH assumption)**.** *Let $\mathbb{F}$ be a field and $Z_S(X) \coloneqq \prod_{s \in S}(X - s)$ for $S \subseteq \mathbb{F}$, the* **expected-time adaptive rational strong Diffie-Hellman (ARSDH)** *holds with error $\epsilon^\star_{\mathsf{ARSDH}}$ if for every security parameter λ, adversary runtime bound $t^\star_{\mathsf{ARSDH}}$, adversary $A^\star_{\mathsf{ARSDH}}$ running in expected time $t^\star_{\mathsf{ARSDH}}$, 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 $e \colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a bilinear pairing,*

$$\begin{aligned}
&\Pr\left[\begin{array}{l|l} S \subseteq \mathbb{F},\ |S| = \mathsf{D} + 1 & \tau \leftarrow \mathbb{Z}_p \setminus \{0\} \\ \wedge\, h_1, h_2 \in \mathbb{G}_1, h_1 \neq [0]_1 & (S, h_1, h_2) \leftarrow A^\star_{\mathsf{ARSDH}}([1]_1, [\tau]_1, [\tau^2]_1, \ldots, [\tau^{\mathsf{D}}]_1, [1]_2, [\tau]_2) \\ \wedge\, h_2 = \frac{1}{Z_S(\tau)} \cdot h_1 & \end{array}\right] \\
&\leq \epsilon^\star_{\mathsf{ARSDH}}(\lambda, \mathsf{D}, t^\star_{\mathsf{ARSDH}}) \ .
\end{aligned}$$

Remark 9.8. [LPS24b, 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 Corollary 9.3 (linearized KZG)

The bKZG batch polynomial commitment scheme for the query class $\mathbf{Q}_{\mathsf{BatchMsg}}[\mathbf{Q}_{\mathsf{UniPoly}}, m + 1]$ satisfies the conditions required by Construction 8.8 (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). bKZG thus gives rise to the linearized KZG functional commitment scheme linKZG, and Corollary 9.3 follows directly from Lemma 8.3 and Corollary 9.2.

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 satifies 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* $(a_1, \ldots, a_{\mathsf{k}_{\mathsf{FC}}})$*-special function binding error* ϵ_{FC} *if for every security parameter* $\lambda \in \mathbb{N}$*, message length* $\ell \in \mathbb{N}$*, number of samples* $\mathsf{L} \in \mathbb{N}$*, auxiliary input distribution* $\mathcal{D}_{\mathsf{FC}}$*, expected adversary size bound* $t^{\star}_{\mathsf{SFB}}$*, and adversary* A_{SFB} *with expected runtime* $t^{\star}_{\mathsf{SFB}}$*,*

$$\Pr\left[\begin{array}{l} \forall\, i \in [\mathsf{L}] : \mathsf{T}_i \textit{ is a } (a_1, \ldots, a_{\mathsf{k}_{\mathsf{FC}}})\textit{-tree for } (\mathsf{cm}, \alpha_i, \beta_i) \\ \wedge \left(\begin{array}{l} \forall\, i \in [\mathsf{L}] \,:\, \mathcal{V}_{\mathsf{FC}}(\mathsf{pp}_{\mathsf{FC}}, \mathsf{cm}, \alpha_i, \beta_i, ((\mathsf{pm}_{i,j}, \mathsf{vm}_{i,j}))_{j \in [\mathsf{k}_{\mathsf{FC}}]}) = 1 \\ \wedge\, \nexists\, \Pi \textit{ s.t. } \forall\, i \in [\mathsf{L}], \alpha_i(\Pi) = \beta_i \end{array} \right) \end{array} \;\middle|\; \begin{array}{l} \mathsf{pp}_{\mathsf{FC}} \leftarrow \mathsf{FC.Gen}(1^{\lambda}, \ell) \\ \mathsf{ai}_{\mathsf{FC}} \leftarrow \mathcal{D}_{\mathsf{FC}} \\ (\mathsf{cm}, ((\alpha_i, \beta_i, \mathsf{T}_i))_{i \in [\mathsf{L}]}) \\ \quad \leftarrow A_{\mathsf{SFB}}(\mathsf{pp}_{\mathsf{FC}}, \mathsf{ai}_{\mathsf{FC}}) \end{array} \right]$$
$$\leq \epsilon_{\mathsf{SFB}}(\lambda, \ell, \mathsf{L}, t^{\star}_{\mathsf{SFB}}) \ .$$

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* $(\mathsf{r}_{\mathsf{FC},1}, \ldots \mathsf{r}_{\mathsf{FC},\mathsf{k}_{\mathsf{FC}}})$ *has expected-time* $(a_1, \ldots, a_{\mathsf{k}_{\mathsf{FC}}})$*-special function binding error* ϵ_{SFB}*, then* FC *has state-restoration function binding error*

$$\epsilon^{\mathsf{SR}}_{\mathsf{FC}}(\lambda, \ell, \mathsf{L}, s_{\mathsf{FC}}, \mathsf{m}_{\mathsf{FC}}, t^{\mathsf{SR}}_{\mathsf{FC}}) \leq \mathsf{L} \cdot (\mathsf{m}_{\mathsf{FC}} + 1) \cdot \left(\sum_{i \in [\mathsf{k}_{\mathsf{FC}}]} \frac{a_i - 1}{2^{\mathsf{r}_{\mathsf{FC},i}}} \right) + \epsilon_{\mathsf{SFB}}(\lambda, \ell, \mathsf{L}, t^{\star}_{\mathsf{SFB}}) \ ,$$

where $t^{\star}_{\mathsf{SFB}} \leq \mathsf{L} \cdot (\mathsf{m}_{\mathsf{FC}} + 1) \left(\prod_{i \in [\mathsf{k}_{\mathsf{FC}}]} (a_i - 1) \right) (t^{\mathsf{SR}}_{\mathsf{FC}} + \mathsf{poly}(\ell))$.

To prove Lemma A.2, we will make use of the following lemma.

Lemma A.3. *For any* k_{FC}*-round interactive argument* $(\mathcal{P}_{\mathsf{FC}}, \mathcal{V}_{\mathsf{FC}})$ *with randomness complexities* $(\mathsf{r}_{\mathsf{FC},1}, \ldots, \mathsf{r}_{\mathsf{FC},\mathsf{k}_{\mathsf{FC}}})$*, for any move budget* $\mathsf{m}_{\mathsf{FC}} \in \mathbb{N}$*, and for any tree arity* $(a_1, \ldots, a_{\mathsf{k}_{\mathsf{FC}}}) \in \mathbb{N}^{\mathsf{k}_{\mathsf{FC}}}$*, there exists an algorithm* $\mathsf{SRTreeFinder}$ *with rewinding access to the prover such that for every malicious prover* $\widetilde{\mathcal{P}}_{\mathsf{PC}}$*, commitment* cm*, point* α *and evaluation* β*,*

$$\Pr\left[\begin{array}{l} \mathsf{T} \textit{ is not an } (a_1, \ldots, a_{\mathsf{k}_{\mathsf{FC}}})\textit{-tree for } (\mathsf{cm}, \alpha, \beta) \\ \wedge\, \mathcal{V}_{\mathsf{FC}}(\mathsf{cm}, \alpha, \beta, (\mathsf{pm}_j)_{j \in [\mathsf{k}_{\mathsf{FC}}]}, (\mathsf{vm}_j)_{j \in [\mathsf{k}_{\mathsf{FC}}]}) = 1 \end{array} \;\middle|\; \begin{array}{l} \mathsf{ai}_{\mathsf{FC}} \leftarrow \mathcal{D}_{\mathsf{FC}} \\ \textit{For } j \in [\mathsf{k}_{\mathsf{FC}}] : \mathsf{rnd}_{\mathsf{FC},j} \leftarrow \mathcal{U}(\mathsf{r}_{\mathsf{FC},j}) \\ ((\mathsf{cm}, \alpha, \beta), ((\mathsf{pm}_j, \eta_j, \mathsf{vm}_j))_{j \in [\mathsf{k}_{\mathsf{FC}}]}) \\ \quad \xleftarrow{\mathsf{tr}} \mathsf{SRGame}(s_{\mathsf{FC}}, \mathsf{rnd}_{\mathsf{FC}}, \widetilde{\mathcal{P}}_{\mathsf{PC}}, \mathsf{pp}_{\mathsf{FC}}, \mathsf{ai}_{\mathsf{FC}}) \\ \mathsf{T} \leftarrow \mathsf{SRTreeFinder}^{\widetilde{\mathcal{P}}_{\mathsf{PC}}}((\mathsf{cm}, \alpha, \beta), ((\mathsf{pm}_j, \eta_j, \mathsf{vm}_j))_{j \in [\mathsf{k}_{\mathsf{FC}}]}, \mathsf{tr}) \end{array} \right]$$
$$\leq (\mathsf{m}_{\mathsf{FC}} + 1) \cdot \left(\sum_{i \in [\mathsf{k}_{\mathsf{FC}}]} \frac{a_i - 1}{2^{\mathsf{r}_{\mathsf{FC},i}}} \right) \ ,$$

where $\mathsf{SRTreeFinder}$ *makes* $(\mathsf{m}_{\mathsf{FC}} + 1) \prod_{i \in [\mathsf{k}_{\mathsf{FC}}]} (a_i - 1)$ *calls to* $\widetilde{\mathcal{P}}_{\mathsf{PC}}$ *in expectation.*

Proof. This is a direct application of [CY24, Lemma 30.5.2] for the FC prover $\widetilde{\mathcal{P}}_{\mathsf{PC}}$, the FC verifier $\mathcal{V}_{\mathsf{FC}}$, and the relation

$$\{((\mathsf{cm}, \alpha, \beta), \Pi) \mid \alpha(\Pi) = \beta\} \ .$$

□

Proof of Lemma A.2. Given an state-restoration function binding adversary $\widetilde{\mathcal{P}}^{\mathsf{SR}}_{\mathsf{FC}}$ for FC, we construct the following reductor from state-restoration function binding to special function binding:

$\mathfrak{R}^{\widetilde{\mathcal{P}}^{\mathsf{SR}}_{\mathsf{FC}}}(\mathsf{pp}_{\mathsf{FC}}, \mathsf{aux})$:

1. For $i \in [\mathsf{L}]$:
   (a) Construct an SRGame adversary $\widetilde{\mathcal{P}}_i$ as follows: when playing the SRGame, $\widetilde{\mathcal{P}}_i$ simulates FCSRGame towards $\widetilde{\mathcal{P}}^{\mathsf{SR}}_{\mathsf{FC}}$ by forwarding its queries to the SRGame.
   $\widetilde{\mathcal{P}}_i$ outputs $(\mathsf{cm}, (\alpha_i, \beta_i, ((\mathsf{pm}_{i,j}, \eta_{i,j}, \mathsf{vm}_{i,j}))_{j\in[\mathsf{k}_{\mathsf{FC}}]}))$, i.e., only the i-th point, its evaluation, and the corresponding opening transcript.
   (b) For $j \in [\mathsf{k}_{\mathsf{FC}}]$: Lazily sample a random oracle $\mathsf{rnd}_{\mathsf{FC},j} \leftarrow \mathcal{U}(\mathsf{r}_{\mathsf{FC},j})$.
   (c) Run $(\mathsf{cm}, (\alpha_i, \beta_i, ((\mathsf{pm}_{i,j}, \eta_{i,j}, \mathsf{vm}_{i,j}))_{j\in[\mathsf{k}_{\mathsf{FC}}]})) \xleftarrow{\mathsf{tr}_i} \mathsf{SRGame}(s_{\mathsf{FC}}, \mathsf{rnd}_{\mathsf{FC}}, \widetilde{\mathcal{P}}_i, \mathsf{pp}_{\mathsf{FC}}, \mathsf{ai}_{\mathsf{FC}})$ by lazily simulating the random oracles $\mathsf{rnd}_{\mathsf{FC},j}$.
   (d) Run $\mathsf{T}_i \leftarrow \mathsf{SRTreeFinder}^{\widetilde{\mathcal{P}}_i}(\mathsf{cm}, (\alpha_i, \beta_i), ((\mathsf{pm}_{i,j}, \eta_{i,j}, \mathsf{vm}_{i,j}))_{j\in[\mathsf{k}_{\mathsf{FC}}]}, \mathsf{tr}_i)$.
2. Output $(\mathsf{cm}, ((\alpha_i, \beta_i))_{i\in[\mathsf{L}]}, (((\mathsf{pm}_{i,j}, \eta_{i,j}, \mathsf{vm}_{i,j}))_{j\in[\mathsf{k}_{\mathsf{FC}}]})_{i\in[\mathsf{L}]}, (\mathsf{T}_i)_{i\in[\mathsf{L}]})$.

Running time. $\mathfrak{R}$ makes $\mathsf{L} \cdot \left(1 + (\mathsf{m}_{\mathsf{FC}} + 1) \prod_{i\in[\mathsf{k}_{\mathsf{FC}}]}(a_i - 1)\right)$ calls to $\widetilde{\mathcal{P}}^{\mathsf{SR}}_{\mathsf{FC}}$ in expectation.

Success probability. We first note that $\mathfrak{R}$ perfectly simulates FCSRGame towards $\widetilde{\mathcal{P}}^{\mathsf{SR}}_{\mathsf{FC}}$, and thus the experiments

$$\left[\begin{array}{l} \mathsf{pp} \leftarrow \mathsf{FC.Gen}(1^\lambda, n) \\ \mathcal{D}_{\mathsf{FC}} \leftarrow \mathcal{D}_{\mathsf{FC}} \\ \text{For } r \in [\mathsf{k}_{\mathsf{FC}}] : \mathsf{rnd}_{\mathsf{FC},r} \leftarrow \mathcal{U}(\mathsf{r}_{\mathsf{FC},r}) \\ \left(\mathsf{cm}, \left(\left(\begin{array}{l} \alpha_i, \beta_i, \\ (\mathsf{pm}_{i,r})_{r\in[\mathsf{k}_{\mathsf{FC}}]}, (\eta_{i,r})_{r\in[\mathsf{k}_{\mathsf{FC}}]}, (\mathsf{vm}_{i,r})_{r\in[\mathsf{k}_{\mathsf{FC}}]} \end{array}\right)\right)_{i\in[\mathsf{L}]}\right) \leftarrow \mathsf{FCSRGame}(s_{\mathsf{FC}}, \mathsf{rnd}_{\mathsf{FC}}, \widetilde{\mathcal{P}}^{\mathsf{SR}}_{\mathsf{FC}}, \mathsf{pp}, \mathsf{ai}_{\mathsf{FC}}) \end{array}\right]$$

and

$$E_{\mathsf{SFB}} := \left[\begin{array}{l} \mathsf{pp}_{\mathsf{FC}} \leftarrow \mathsf{FC.Gen}(1^\lambda, \ell) \\ \mathsf{ai}_{\mathsf{FC}} \leftarrow \mathcal{D}_{\mathsf{FC}} \\ (\mathsf{cm}, ((\alpha_i, \beta_i))_{i\in[\mathsf{L}]}, (((\mathsf{pm}_{i,j}, \eta_{i,j}, \mathsf{vm}_{i,j}))_{j\in[\mathsf{k}_{\mathsf{FC}}]})_{i\in[\mathsf{L}]}, (\mathsf{T}_i)_{i\in[\mathsf{L}]}) \leftarrow \mathfrak{R}^{\widetilde{\mathcal{P}}^{\mathsf{SR}}_{\mathsf{FC}}}(\mathsf{pp}_{\mathsf{FC}}, \mathsf{ai}_{\mathsf{FC}}) \end{array}\right]$$

are equivalent.
We bound

$$\Pr\left[\begin{array}{l} \forall\, i \in [\mathsf{L}] \ : \ \mathcal{V}_{\mathsf{FC}}((\mathsf{cm}, (\alpha_i, \beta_i)), (\mathsf{pm}_{i,r})_{r\in[\mathsf{k}_{\mathsf{FC}}]}, (\mathsf{vm}_{i,r})_{r\in[\mathsf{k}_{\mathsf{FC}}]}) = 1 \\ \wedge \left(\begin{array}{l} \forall\, i \in [\mathsf{L}] \ : \ \mathcal{V}_{\mathsf{FC}}(\mathsf{pp}_{\mathsf{FC}}, \mathsf{cm}, \alpha_i, \beta_i, ((\mathsf{pm}_{i,j}, \mathsf{vm}_{i,j}))_{j\in[\mathsf{k}_{\mathsf{FC}}]}) = 1 \\ \wedge \nexists\, \Pi \text{ s.t. } \forall\, i \in [\mathsf{L}], \alpha_i(\Pi) = \beta_i \end{array}\right) \end{array} \ \middle| \ E_{\mathsf{SFB}}\right]$$

$$= \Pr\left[\begin{array}{l} \forall\, i \in [\mathsf{L}] \ : \ \mathsf{T}_i \text{ is a } (a_1, \ldots, a_{\mathsf{k}_{\mathsf{FC}}})\text{-tree for } (\mathsf{cm}, \alpha_i, \beta_i) \\ \wedge \forall\, i \in [\mathsf{L}] \ : \ \mathcal{V}_{\mathsf{FC}}((\mathsf{cm}, (\alpha_i, \beta_i)), (\mathsf{pm}_{i,r})_{r\in[\mathsf{k}_{\mathsf{FC}}]}, (\mathsf{vm}_{i,r})_{r\in[\mathsf{k}_{\mathsf{FC}}]}) = 1 \\ \wedge \left(\begin{array}{l} \forall\, i \in [\mathsf{L}] \ : \ \mathcal{V}_{\mathsf{FC}}(\mathsf{pp}_{\mathsf{FC}}, \mathsf{cm}, \alpha_i, \beta_i, ((\mathsf{pm}_{i,j}, \mathsf{vm}_{i,j}))_{j\in[\mathsf{k}_{\mathsf{FC}}]}) = 1 \\ \wedge \nexists\, \Pi \text{ s.t. } \forall\, i \in [\mathsf{L}], \alpha_i(\Pi) = \beta_i \end{array}\right) \end{array} \ \middle| \ E_{\mathsf{SFB}}\right]$$

$$
\begin{aligned}
&+ \Pr\left[\begin{array}{l}\exists i \in [\mathsf{L}] : \mathsf{T}_i \text{ is not a } (a_1,\ldots,a_{\mathsf{k_{FC}}})\text{-tree for } (\mathsf{cm},\alpha_i,\beta_i)\\ \wedge \forall i \in [\mathsf{L}] : \mathcal{V}_{\mathsf{FC}}((\mathsf{cm},(\alpha_i,\beta_i)),(\mathsf{pm}_{i,r})_{r\in[\mathsf{k_{FC}}]},(\mathsf{vm}_{i,r})_{r\in[\mathsf{k_{FC}}]}) = 1\\ \wedge \left(\begin{array}{l}\forall i \in [\mathsf{L}] : \mathcal{V}_{\mathsf{FC}}(\mathsf{pp}_{\mathsf{FC}},\mathsf{cm},\alpha_i,\beta_i,((\mathsf{pm}_{i,j},\mathsf{vm}_{i,j}))_{j\in[\mathsf{k_{FC}}]}) = 1\\ \wedge \nexists \Pi \text{ s.t. } \forall i \in [\mathsf{L}], \alpha_i(\Pi) = \beta_i\end{array}\right)\end{array}\;\middle|\; E_{\mathsf{SFB}}\right]\\
\leq \Pr&\left[\begin{array}{l}\forall i \in [\mathsf{L}] : \mathsf{T}_i \text{ is a } (a_1,\ldots,a_{\mathsf{k_{FC}}})\text{-tree for } (\mathsf{cm},\alpha_i,\beta_i)\\ \wedge \left(\begin{array}{l}\forall i \in [\mathsf{L}] : \mathcal{V}_{\mathsf{FC}}(\mathsf{pp}_{\mathsf{FC}},\mathsf{cm},\alpha_i,\beta_i,((\mathsf{pm}_{i,j},\mathsf{vm}_{i,j}))_{j\in[\mathsf{k_{FC}}]}) = 1\\ \wedge \nexists \Pi \text{ s.t. } \forall i \in [\mathsf{L}], \alpha_i(\Pi) = \beta_i\end{array}\right)\end{array}\;\middle|\; E_{\mathsf{SFB}}\right]\\
&+ \Pr\left[\begin{array}{l}\exists i \in [\mathsf{L}] : \mathsf{T}_i \text{ is not a } (a_1,\ldots,a_{\mathsf{k_{FC}}})\text{-tree for } (\mathsf{cm},\alpha_i,\beta_i)\\ \wedge \forall i \in [\mathsf{L}] : \mathcal{V}_{\mathsf{FC}}((\mathsf{cm},(\alpha_i,\beta_i)),(\mathsf{pm}_{i,r})_{r\in[\mathsf{k_{FC}}]},(\mathsf{vm}_{i,r})_{r\in[\mathsf{k_{FC}}]}) = 1\end{array}\;\middle|\; E_{\mathsf{SFB}}\right] .
\end{aligned}
$$

The first probability is bounded by $\epsilon_{\mathsf{SFB}}(\lambda,\ell,\mathsf{L},t^{\star}_{\mathsf{SFB}})$. For the second term, we have

$$
\begin{aligned}
&\Pr\left[\begin{array}{l}\exists i \in [\mathsf{L}] : \mathsf{T}_i \text{ is not a } (a_1,\ldots,a_{\mathsf{k_{FC}}})\text{-tree for } (\mathsf{cm},\alpha_i,\beta_i)\\ \wedge \forall i \in [\mathsf{L}] : \mathcal{V}_{\mathsf{FC}}((\mathsf{cm},(\alpha_i,\beta_i)),(\mathsf{pm}_{i,r})_{r\in[\mathsf{k_{FC}}]},(\mathsf{vm}_{i,r})_{r\in[\mathsf{k_{FC}}]}) = 1\end{array}\;\middle|\; E_{\mathsf{SFB}}\right]\\
&\leq \Pr\left[\begin{array}{l}\exists i \in [\mathsf{L}] : \left(\begin{array}{l}\mathsf{T}_i \text{ is not a } (a_1,\ldots,a_{\mathsf{k_{FC}}})\text{-tree for } (\mathsf{cm},\alpha_i,\beta_i)\\ \wedge \mathcal{V}_{\mathsf{FC}}((\mathsf{cm},(\alpha_i,\beta_i)),(\mathsf{pm}_{i,r})_{r\in[\mathsf{k_{FC}}]},(\mathsf{vm}_{i,r})_{r\in[\mathsf{k_{FC}}]}) = 1\end{array}\right)\end{array}\;\middle|\; E_{\mathsf{SFB}}\right]\\
&\leq \sum_{i\in[\mathsf{L}]} \Pr\left[\begin{array}{l}\mathsf{T}_i \text{ is not a } (a_1,\ldots,a_{\mathsf{k_{FC}}})\text{-tree for } (\mathsf{cm},\alpha_i,\beta_i)\\ \wedge \mathcal{V}_{\mathsf{FC}}((\mathsf{cm},(\alpha_i,\beta_i)),(\mathsf{pm}_{i,r})_{r\in[\mathsf{k_{FC}}]},(\mathsf{vm}_{i,r})_{r\in[\mathsf{k_{FC}}]}) = 1\end{array}\;\middle|\; E_{\mathsf{SFB}}\right]\\
&= \mathsf{L}\cdot \Pr\left[\begin{array}{l}\mathsf{T}_i \text{ is not a } (a_1,\ldots,a_{\mathsf{k_{FC}}})\text{-tree for } (\mathsf{cm},\alpha_i,\beta_i)\\ \wedge \mathcal{V}_{\mathsf{FC}}((\mathsf{cm},(\alpha_i,\beta_i)),(\mathsf{pm}_{i,r})_{r\in[\mathsf{k_{FC}}]},(\mathsf{vm}_{i,r})_{r\in[\mathsf{k_{FC}}]}) = 1\end{array}\;\middle|\; E\right]\\
&\leq \mathsf{L}\cdot(\mathsf{m_{FC}}+1)\cdot\left(\sum_{i\in[\mathsf{k_{FC}}]}\frac{a_i-1}{2^{\mathsf{r_{FC},}i}}\right) ,
\end{aligned}
$$

where

$$
E \coloneqq \left[\begin{array}{l}\mathsf{pp}_{\mathsf{FC}} \leftarrow \mathsf{FC.Gen}(1^{\lambda},\ell)\\ \mathsf{ai}_{\mathsf{FC}} \leftarrow \mathcal{D}_{\mathsf{FC}}\\ \left(\mathsf{cm},\left(\left(\begin{array}{l}\alpha_i,\beta_i,\\ (\mathsf{pm}_{i,r})_{r\in[\mathsf{k_{FC}}]},(\eta_{i,r})_{r\in[\mathsf{k_{FC}}]},(\mathsf{vm}_{i,r})_{r\in[\mathsf{k_{FC}}]}\end{array}\right)\right)_{i\in[\mathsf{L}]}\right) \xleftarrow{\mathsf{tr}_i} \mathsf{SRGame}(s_{\mathsf{FC}},\mathsf{rnd}_{\mathsf{FC}},\widetilde{\mathcal{P}}_i,\mathsf{pp},\mathsf{ai}_{\mathsf{FC}})\\ \mathsf{T}_i \leftarrow \mathsf{SRTreeFinder}^{\widetilde{\mathcal{P}}_i}((\mathsf{cm},\alpha_i,\beta_i),((\mathsf{pm}_{i,j},\eta_{i,j},\mathsf{vm}_{i,j}))_{j\in[\mathsf{k_{FC}}]},\mathsf{tr}_i)\end{array}\right]
$$

□

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* $\mathsf{D} \in \mathbb{N}$*, auxiliary input distribution* $\mathcal{D}$*, adversary size bound* $t_{\mathsf{FCSC}} \in \mathbb{N}$*, and* t_{FCSC}*-size circuit* A_{FCSC}*,*

$$\Pr\left[\begin{array}{l|l} d > \mathsf{D} & \mathsf{pp}_{\mathsf{PC}} \leftarrow \mathsf{PC.Gen}(1^\lambda, \mathsf{D}) \\ \wedge \forall\, i \in [d+1], \mathsf{p}(\alpha_i) = \beta_i & \mathsf{ai} \leftarrow \mathcal{D} \\ \wedge \forall\, i \in [d+1], \mathsf{PC.Check}(\mathsf{pp}_{\mathsf{PC}}, \mathsf{cm}, \alpha_i, \beta_i, \mathsf{pf}_i) = 1 & (d, \mathsf{p}, \mathsf{cm}, \mathsf{state}) \leftarrow A_{\mathsf{FCSC}}(\mathsf{pp}_{\mathsf{PC}}, \mathsf{ai}) \\ \wedge\ \deg(\mathsf{Lagrange}(\{(\alpha_i, \beta_i)\}_{i\in[d+1]})) = d & \{\alpha_i\}_{i\in[d+1]} \leftarrow \mathbb{F} \textit{ with } \alpha_i \neq \alpha_j, \forall\, i \neq j \in [d+1] \\ \wedge\ \deg(\mathsf{Lagrange}(\{(\alpha_i, \mathsf{pf}_i)\}_{i\in[d+1]})) = d-1 & \{(\beta_i, \mathsf{pf}_i)\}_{i\in[d+1]} \leftarrow A_{\mathsf{FCSC}}(\mathsf{pp}_{\mathsf{PC}}, \{\alpha_i\}_{i\in[d+1]}, \mathsf{state}) \end{array}\right]$$
$$\leq \epsilon_{\mathsf{FCSC}}(\lambda, \mathsf{D}, t_{\mathsf{FCSC}}) \ .$$

Lemma B.2. *The* KZG *polynomial commitment scheme (Construction 9.4) has function binding error* $\epsilon_{\mathsf{FC}} = \epsilon_{\mathsf{FC}}(\lambda, \mathsf{D}, \mathsf{L}, t_{\mathsf{FC}})$*. Then* KZG *has strong correctness error* $\epsilon_{\mathsf{FCSC}} = \epsilon_{\mathsf{FCSC}}(\lambda, \mathsf{D}, t_{\mathsf{FCSC}})$ *such that for every security parameter* $\lambda \in \mathbb{N}$*, polynomial degree bound* $\mathsf{D} \in \mathbb{N}$*, and adversary size bound* $t_{\mathsf{FCSC}} \in \mathbb{N}$*,*

$$\epsilon_{\mathsf{FCSC}}(\lambda, \mathsf{D}, t_{\mathsf{FCSC}}) \leq \epsilon_{\mathsf{FC}}(\lambda, \mathsf{D}, \mathsf{L}, t_{\mathsf{FC}}) \quad \textit{where} \quad \begin{cases} \mathsf{L} = O(t_{\mathsf{FCSC}}) \\ t_{\mathsf{FC}} = O(t_{\mathsf{FCSC}}) \end{cases} .$$

Proof. Fix the security parameter $\lambda \in \mathbb{N}$, polynomial degree bound $\mathsf{D} \in \mathbb{N}$, and adversary size bound $t_{\mathsf{FCSC}} \in \mathbb{N}$. For every auxiliary input ai and t_{FCSC}-size adversary A_{FCSC} against strong correctness, consider the following adversary A_{FC} against function binding:

$A_{\mathsf{FC}}(\mathsf{pp}_{\mathsf{PC}}, \mathsf{ai}_{\mathsf{FC}})$:
1. Run $(d, \mathsf{p}, \mathsf{cm}, \mathsf{state}) \leftarrow A_{\mathsf{FCSC}}(\mathsf{pp}_{\mathsf{PC}}, \mathsf{ai})$.
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 $\{(\beta_i, \mathsf{pf}_i)\}_{i\in[d+1]} \leftarrow A_{\mathsf{FCSC}}(\mathsf{pp}_{\mathsf{PC}}, \{\alpha_i\}_{i\in[d+1]}, \mathsf{state})$.
4. Output $(\mathsf{cm}, \{(\alpha_i, \beta_i, \mathsf{pf}_i)\}_{i\in[d+1]})$.

Success probability. When A_{FCSC} successfully breaks strong correctness, the following holds:
- $\forall\, i \in [d+1], \mathsf{KZG.Check}(\mathsf{pp}_{\mathsf{PC}}, \mathsf{cm}, \alpha_i, \beta_i, \mathsf{pf}_i) = 1$; and
- $d := \deg(\mathsf{Lagrange}(\{(\alpha_i, \beta_i)\}_{i\in[d+1]})) > \mathsf{D}$.

Therefore,

$$\Pr\left[\begin{array}{l|l} d > \mathsf{D} & \mathsf{pp}_{\mathsf{PC}} \leftarrow \mathsf{KZG.Gen}(1^\lambda, \mathsf{D}) \\ \wedge \forall\, i \in [d+1], \mathsf{p}(\alpha_i) = \beta_i & \mathsf{ai} \leftarrow \mathcal{D} \\ \wedge \forall\, i \in [d+1], \mathsf{KZG.Check}(\mathsf{pp}_{\mathsf{PC}}, \mathsf{cm}, \alpha_i, \beta_i, \mathsf{pf}_i) = 1 & (d, \mathsf{p}, \mathsf{cm}, \mathsf{state}) \leftarrow A_{\mathsf{FCSC}}(\mathsf{pp}_{\mathsf{PC}}, \mathsf{ai}) \\ \wedge\ \deg(\mathsf{Lagrange}(\{(\alpha_i, \beta_i)\}_{i\in[d+1]})) = d & \{\alpha_i\}_{i\in[d+1]} \leftarrow \mathbb{F} \text{ with } \alpha_i \neq \alpha_j, \forall\, i \neq j \in [d+1] \\ \wedge\ \deg(\mathsf{Lagrange}(\{(\alpha_i, \mathsf{pf}_i)\}_{i\in[d+1]})) = d-1 & \{(\beta_i, \mathsf{pf}_i)\}_{i\in[d+1]} \leftarrow A_{\mathsf{FCSC}}(\mathsf{pp}_{\mathsf{PC}}, \{\alpha_i\}_{i\in[d+1]}, \mathsf{state}) \end{array}\right]$$

$$\leq \Pr\left[\begin{array}{l|l} \forall a \in [\mathsf{L}] : \mathsf{KZG.Check}(\mathsf{pp}_{\mathsf{PC}}, \mathsf{cm}, \alpha_i, \beta_i, \mathsf{pf}_i) = 1 & \mathsf{pp}_{\mathsf{PC}} \leftarrow \mathsf{KZG.Gen}(1^\lambda, \mathsf{D}) \\ \wedge \left(\begin{array}{l} \exists i, j \in [\mathsf{L}] : \alpha_i = \alpha_j, \beta_i \neq \beta_j \\ \veebar \ \deg(\mathsf{Lagrange}(\{(\alpha_i, \beta_i)\}_{i\in[\mathsf{L}]})) > \mathsf{D} \end{array}\right) & \begin{array}{l} \mathsf{ai}_{\mathsf{FC}} \leftarrow \mathcal{D}_{\mathsf{FC}} \\ (\mathsf{cm}, \{(\alpha_i, \beta_i, \mathsf{pf}_i)\}_{i\in[\mathsf{L}]}) \leftarrow A_{\mathsf{FC}}(\mathsf{pp}_{\mathsf{PC}}, \mathsf{ai}_{\mathsf{FC}}) \end{array} \end{array}\right]$$
$$\leq \epsilon_{\mathsf{FC}}(\lambda, \mathsf{D}, \mathsf{L}, t_{\mathsf{FC}}) \ .$$

Running time. A_{FC} has sample size $\mathsf{L} = O(t_{\mathsf{FCSC}})$, and adversary size $t_{\mathsf{FC}} = O(t_{\mathsf{FCSC}})$. □

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* $\mathsf{D} \in \mathbb{N}$*, auxiliary input distribution* $\mathcal{D}$*, adversary size bound* $t_{\mathsf{FCIB}} \in \mathbb{N}$*, and* t_{FCIB}*-size circuit* A_{FCIB}*,*

$$\Pr\left[\begin{array}{l|l} \forall\, i \neq j \in [\mathsf{D}+1], \alpha_i \neq \alpha_j & \mathsf{pp}_{\mathsf{PC}} \leftarrow \mathsf{PC.Gen}(1^\lambda, \mathsf{D}) \\ \wedge \forall\, i \in [\mathsf{D}+1], \mathsf{PC.Check}(\mathsf{pp}_{\mathsf{PC}}, \mathsf{cm}, \alpha_i, \beta_i, \mathsf{pf}_i) = 1 & \mathsf{ai} \leftarrow \mathcal{D} \\ \wedge\, \mathsf{cm} \neq \mathsf{PC.Commit}(\mathsf{pp}_{\mathsf{PC}}, \mathsf{Lagrange}(\{(\alpha_i, \beta_i)\}_{i\in[\mathsf{D}+1]})) & (\mathsf{cm}, \{(\alpha_i, \beta_i, \mathsf{pf}_i)\}_{i\in[\mathsf{D}+1]}) \leftarrow A_{\mathsf{FCIB}}(\mathsf{pp}_{\mathsf{PC}}, \mathsf{ai}) \end{array}\right]$$
$$\leq \epsilon_{\mathsf{FCIB}}(\lambda, \mathsf{D}, t_{\mathsf{FCIB}}) \ .$$

[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 [LPS24a].

Lemma B.4. *Let* KZG *be the polynomial commitment scheme constructed in [KZG10] (Construction 9.4). Assume* KZG *has evaluation binding error* $\epsilon_{\mathsf{FCEB}} = \epsilon_{\mathsf{FCEB}}(\lambda, \mathsf{D}, t_{\mathsf{FCEB}})$ *and interpolation binding error* $\epsilon_{\mathsf{FCIB}} = \epsilon_{\mathsf{FCIB}}(\lambda, \mathsf{D}, t_{\mathsf{FCIB}})$*. Then,* KZG *has function binding error* $\epsilon_{\mathsf{FC}} = \epsilon_{\mathsf{FC}}(\lambda, \mathsf{D}, \mathsf{L}, t_{\mathsf{FC}})$ *such that for every security parameter* $\lambda \in \mathbb{N}$*, polynomial degree bound* $\mathsf{D} \in \mathbb{N}$*, sample set size* $\mathsf{L} \in \mathbb{N}$*, and adversary size bound* $t_{\mathsf{FC}} \in \mathbb{N}$*,*

$$\epsilon_{\mathsf{FC}}(\lambda, \mathsf{D}, \mathsf{L}, t_{\mathsf{FC}}) \leq \epsilon_{\mathsf{FCEB}}(\lambda, \mathsf{D}, t_{\mathsf{FCEB}}) + \epsilon_{\mathsf{FCIB}}(\lambda, \mathsf{D}, t_{\mathsf{FCIB}}) \ ,$$

where $t_{\mathsf{FCEB}} \leq t_{\mathsf{FC}} + \mathsf{L}^2$ *and* $t_{\mathsf{FCIB}} \leq t_{\mathsf{FC}} + \mathsf{L} + 3\mathsf{L}^2 + \mathsf{D} \cdot (\mathsf{L} - \mathsf{D} + 1)$.

Proof of Lemma B.4. Fix the security parameter $\lambda \in \mathbb{N}$, polynomial degree bound $\mathsf{D} \in \mathbb{N}$, sample set size $\mathsf{L} \in \mathbb{N}$, and adversary size bound $t_{\mathsf{FC}} \in \mathbb{N}$. 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:

$A_{\mathsf{FCEB}}(\mathsf{pp}_{\mathsf{PC}}, \mathsf{ai}_{\mathsf{FC}})$:
1. Run $(\mathsf{cm}, \{(\alpha_i, \beta_i, \mathsf{pf}_i)\}_{i\in[\mathsf{L}]}) \leftarrow A_{\mathsf{FC}}(\mathsf{pp}_{\mathsf{PC}}, \mathsf{ai}_{\mathsf{FC}})$.
2. If there exists $i \neq j \in [\mathsf{L}]$ such that $\alpha_i = \alpha_j$ and $\beta_i \neq \beta_j$, output $(\mathsf{cm}, \alpha_i, \beta_i, \beta_j, \mathsf{pf}_i, \mathsf{pf}_j)$.
3. Otherwise, output $(\mathsf{cm}, \alpha_1, \beta_1, \beta_1, \mathsf{pf}_1, \mathsf{pf}_1)$.

Step 2 in A_{FCEB} takes time at most L^2, thus the running time of A_{FCEB} is $t_{\mathsf{FCEB}} \leq t_{\mathsf{FC}} + \mathsf{L}^2$.

$A_{\mathsf{FCIB}}(\mathsf{pp}_{\mathsf{PC}}, \mathsf{ai}_{\mathsf{FC}})$:
1. Run $(\mathsf{cm}, \{(\alpha_i, \beta_i, \mathsf{pf}_i)\}_{i\in[\mathsf{L}]}) \leftarrow A_{\mathsf{FC}}(\mathsf{pp}_{\mathsf{PC}}, \mathsf{ai}_{\mathsf{FC}})$.
2. If $\deg(\mathsf{Lagrange}(\{(\alpha_i, \beta_i)\}_{i\in[\mathsf{L}]})) > \mathsf{D}$:

(a) Let L_0 be the interpolation of the first $\mathsf{D}+1$ distinct points, denoted without loss of generality as $\{\alpha_i\}_{i\in[\mathsf{D}+1]}$.
(b) Find (α_k, β_k) for some $k \in \{\mathsf{D}+2, \ldots, \mathsf{L}\}$ such that $\beta_k \neq L_0(\alpha_k)$.
(c) Sample $S \coloneqq \{\alpha'_i\}_{i\in[\mathsf{D}+1]}$ from $\{\alpha_i\}_{i\in[\mathsf{D}+1]\cup\{k\}}$ such that $[\mathsf{Lagrange}(\{(\alpha, \beta[\alpha])\}_{\alpha\in S})]_1 \neq \mathsf{cm}$.
(d) Output $(\mathsf{cm}, \{(\alpha'_i, \beta'_i, \mathsf{pf}'_i)\}_{i\in[\mathsf{D}+1]})$.

3. Otherwise, output $(\mathsf{cm}, \perp)$.

In the construction of A_{FCIB}, Step 2 takes time L^2, Step 2a takes time $\mathsf{L} + (\mathsf{D}+1)^2$, Step 2b takes time $\mathsf{D}\cdot(\mathsf{L}-\mathsf{D}-1)$, and Step 2c takes time at most $2\mathsf{D} + (\mathsf{D}+1)^2$. Therefore, the total running time is $t_{\mathsf{FCIB}} \leq t_{\mathsf{FC}} + \mathsf{L} + 3\mathsf{L}^2 + \mathsf{D}\cdot(\mathsf{L}-\mathsf{D}+1)$.

When A_{FC} successfully breaks function binding, there are two cases:

- There are two valid openings $(\alpha, \beta_i, \mathsf{pf}_i)$ and $(\alpha, \beta_j, \mathsf{pf}_j)$ where $\beta_i \neq \beta_j$. In this case, A_{FCEB} succeeds.
- $\deg(\mathsf{Lagrange}(\{(\alpha_i, \beta_i)\}_{i\in[\mathsf{L}]})) > \mathsf{D}$. From the construction of A_{FCIB}, the following holds:
  - There must exist (α_k, β_k) for some $k \in \{\mathsf{D}+2, \ldots, \mathsf{L}\}$ such that $\beta_k \neq L_0(\alpha_k)$, because otherwise $\deg(\mathsf{Lagrange}(\{(\alpha_i, \beta_i)\}_{i\in[\mathsf{L}]})) = \deg(L_0) \leq \mathsf{D}$.
  - $\deg(\mathsf{Lagrange}(\{(\alpha_i, \beta_i)\}_{i\in[\mathsf{D}+1]\cup\{k\}})) \geq \mathsf{D}+1$, because otherwise, by the fundamental theorem of algebra, we would have $\mathsf{Lagrange}(\{(\alpha_i, \beta_i)\}_{i\in[\mathsf{D}+1]\cup\{k\}}) = L_0$.
  - For all size-$(\mathsf{D}+1)$ subsets of $\{(\alpha_i, \beta_i)\}_{i\in[\mathsf{D}+1]\cup\{k\}}$, their interpolations are pairwise distinct, because otherwise $\deg(\mathsf{Lagrange}(\{(\alpha_i, \beta_i)\}_{i\in[\mathsf{D}+1]\cup\{k\}})) \leq \mathsf{D}$.

  Therefore, there are at least $\mathsf{D}+1$ among $\mathsf{D}+2$ 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|l} \forall i \in [\mathsf{L}] : \mathsf{KZG.Check}(\mathsf{pp}_{\mathsf{PC}}, \mathsf{cm}, \alpha_i, \beta_i, \mathsf{pf}_i) = 1 \\ \wedge \left(\begin{array}{l} \exists i, j \in [\mathsf{L}] : \alpha_i = \alpha_j, \beta_i \neq \beta_j \\ \veebar\ \deg(\mathsf{Lagrange}(\{(\alpha_i, \beta_i)\}_{i\in[\mathsf{L}]})) > \mathsf{D} \end{array}\right) & \begin{array}{l} \mathsf{pp}_{\mathsf{PC}} \leftarrow \mathsf{KZG.Gen}(1^\lambda, \mathsf{D}) \\ \mathsf{ai}_{\mathsf{FC}} \leftarrow \mathcal{D}_{\mathsf{FC}} \\ (\mathsf{cm}, \{(\alpha_i, \beta_i, \mathsf{pf}_i)\}_{i\in[\mathsf{L}]}) \leftarrow A_{\mathsf{FC}}(\mathsf{pp}_{\mathsf{PC}}, \mathsf{ai}_{\mathsf{FC}}) \end{array} \end{array}\right] \\
&\leq \Pr\left[\begin{array}{l|l} \beta_i \neq \beta_j & \mathsf{pp}_{\mathsf{PC}} \leftarrow \mathsf{KZG.Gen}(1^\lambda, \mathsf{D}) \\ \wedge \mathsf{KZG.Check}(\mathsf{pp}_{\mathsf{PC}}, \mathsf{cm}, \alpha, \beta_i, \mathsf{pf}_i) = 1 & \mathsf{ai} \leftarrow \mathcal{D} \\ \wedge \mathsf{KZG.Check}(\mathsf{pp}_{\mathsf{PC}}, \mathsf{cm}, \alpha, \beta_j, \mathsf{pf}_j) = 1 & \begin{pmatrix} \mathsf{cm}, \alpha, \beta_i, \\ \beta_j, \mathsf{pf}_i, \mathsf{pf}_j \end{pmatrix} \leftarrow A_{\mathsf{FCEB}}(\mathsf{pp}_{\mathsf{PC}}, \mathsf{ai}) \end{array}\right] \\
&\quad + \Pr\left[\begin{array}{l|l} \forall i \neq j \in [\mathsf{D}+1], \alpha'_i \neq \alpha'_j & \mathsf{pp}_{\mathsf{PC}} \leftarrow \mathsf{KZG.Gen}(1^\lambda, \mathsf{D}) \\ \wedge \forall i \in [\mathsf{D}+1], \mathsf{KZG.Check}(\mathsf{pp}_{\mathsf{PC}}, \mathsf{cm}, \alpha'_i, \beta'_i, \mathsf{pf}'_i) = 1 & \mathsf{ai} \leftarrow \mathcal{D} \\ \wedge \mathsf{cm} \neq \mathsf{KZG.Commit}(\mathsf{pp}_{\mathsf{PC}}, \mathsf{Lagrange}(\{(\alpha_i, \beta'_i)\}_{i\in[\mathsf{D}+1]})) & (\mathsf{cm}, \{(\alpha_i, \beta'_i, \mathsf{pf}'_i)\}_{i\in[\mathsf{D}+1]}) \leftarrow A_{\mathsf{FCIB}}(\mathsf{pp}_{\mathsf{PC}}, \mathsf{ai}) \end{array}\right] \\
&\leq \epsilon_{\mathsf{FCEB}}(\lambda, \mathsf{D}, t_{\mathsf{FCEB}}) + \epsilon_{\mathsf{FCIB}}(\lambda, \mathsf{D}, t_{\mathsf{FCIB}}) \ .
\end{aligned}
$$

Since the relation holds for any auxiliary input distribution $\mathcal{D}_{\mathsf{FC}}$ and t_{FC}-size adversary A_{FC} against function binding, we conclude that $\epsilon_{\mathsf{FC}}(\lambda, \mathsf{D}, \mathsf{L}, t_{\mathsf{FC}}) \leq \epsilon_{\mathsf{FCEB}}(\lambda, \mathsf{D}, t_{\mathsf{FCEB}}) + \epsilon_{\mathsf{FCIB}}(\lambda, \mathsf{D}, t_{\mathsf{FCIB}})$, which completes the proof. □

Lemma B.5. *Assume the ARSDH assumption holds with error $\epsilon_{\mathsf{ARSDH}} = \epsilon_{\mathsf{ARSDH}}(\lambda, \mathsf{D}, t_{\mathsf{ARSDH}})$. Let KZG be the polynomial commitment scheme constructed in [KZG10] (Construction 9.4). Then KZG has interpolation*

binding error $\epsilon_{\mathsf{FCIB}} = \epsilon_{\mathsf{FCIB}}(\lambda, \mathsf{D}, t_{\mathsf{FCIB}})$ *such that for every security parameter* $\lambda \in \mathbb{N}$*, polynomial degree bound* $\mathsf{D} \in \mathbb{N}$*, and adversary size bound* $t_{\mathsf{FCIB}} \in \mathbb{N}$*,*

$$\epsilon_{\mathsf{FCIB}}(\lambda, \mathsf{D}, t_{\mathsf{FCIB}}) \leq \epsilon_{\mathsf{ARSDH}}(\lambda, \mathsf{D}, t_{\mathsf{ARSDH}}) \ ,$$

where $t_{\mathsf{ARSDH}} \leq t_{\mathsf{FCIB}} + (\mathsf{D}+1)^2 + 3(\mathsf{D}+1)$.

Proof of Lemma B.5. Fix the security parameter $\lambda \in \mathbb{N}$, polynomial degree bound $\mathsf{D} \in \mathbb{N}$, and adversary size bound $t_{\mathsf{FCIB}} \in \mathbb{N}$. 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:

$A_{\mathsf{ARSDH}}([1]_1, [\tau]_1, [\tau^2]_1, \ldots, [\tau^{\mathsf{D}}]_1, [1]_2, [\tau]_2)$:
1. Sample $\mathsf{ai} \leftarrow \mathcal{D}$ and set $\mathsf{pp}_{\mathsf{PC}} \coloneqq ([1]_1, [\tau]_1, [\tau^2]_1, \ldots, [\tau^{\mathsf{D}}]_1, [1]_2, [\tau]_2)$.
2. Run $(\mathsf{cm}, \{(\alpha_i, \beta_i, \mathsf{pf}_i)\}_{i\in[\mathsf{D}+1]}) \leftarrow A_{\mathsf{FCIB}}(\mathsf{pp}_{\mathsf{PC}}, \mathsf{ai})$.
3. Set $S \coloneqq \{\alpha_i\}_{i\in[\mathsf{D}+1]}$.
4. Set $L(X) \coloneqq \mathsf{Lagrange}(\{\alpha_i, \beta_i\}_{i\in[\mathsf{D}+1]})$.
5. Set $h_1 \coloneqq \mathsf{cm} - [L(\tau)]_1$.
6. Compute $d_i \coloneqq \frac{1}{Z_{S\setminus\{\alpha_i\}}(\alpha_i)}$.
7. Set $h_2 \coloneqq \sum_{i=1}^{\mathsf{D}+1} d_i \cdot \mathsf{pf}_i$.
8. Output (S, h_1, h_2).

Running time. In the construction of A_{ARSDH}, Step 4 takes time $(\mathsf{D}+1)^2$, Step 5 takes time at most D, and Step 7 takes time at most $2\mathsf{D}$. Therefore, the total running time is $t_{\mathsf{ARSDH}} \leq t_{\mathsf{FCIB}} + (\mathsf{D}+1)^2 + 3(\mathsf{D}+1)$.

Success probability. When A_{FCIB} succeeds, the following holds:
- $\forall\, i \neq j \in [\mathsf{D}+1], \alpha_i \neq \alpha_j$, which implies that $|S| = \mathsf{D}+1$;
- $\mathsf{cm} \neq \mathsf{KZG.Commit}(\mathsf{pp}_{\mathsf{PC}}, \mathsf{Lagrange}(\{(\alpha_i, \beta_i)\}_{i\in[\mathsf{D}+1]})) = [1]_1^{L(\tau)}$, which implies that $h_1, h_2 \in \mathbb{G}_1$ and $h_1 \neq [0]_1$;
- $\forall\, i \in [\mathsf{D}+1], \mathsf{KZG.Check}(\mathsf{pp}_{\mathsf{PC}}, \mathsf{cm}, \alpha_i, \beta_i, \mathsf{pf}_i) = 1$, which means for every $i \in [\mathsf{D}+1]$, $e(\mathsf{cm} - [\beta_i]_1, [1]_2) = e(\mathsf{pf}_i, [\tau - \alpha_i]_2)$. By the definition of Lagrange interpolation,

$$\begin{aligned}
&\frac{1}{Z_S(\tau)} \cdot h_1 \\
&= \frac{1}{Z_S(\tau)} \cdot (\mathsf{cm} - [L(\tau)]_1) \\
&= \frac{1}{Z_S(\tau)} \cdot (\mathsf{cm} - [\sum_{i=1}^{\mathsf{D}+1} \beta_i \cdot \frac{Z_S(\tau) d_i}{\tau - \alpha_i}]_1) \\
&= \frac{1}{Z_S(\tau)} \cdot (\sum_{i=1}^{\mathsf{D}+1} \frac{Z_S(\tau) d_i}{\tau - \alpha_i} \cdot \mathsf{cm} - [\sum_{i=1}^{\mathsf{D}+1} \beta_i \cdot \frac{Z_S(\tau) d_i}{\tau - \alpha_i}]_1) \\
&= \sum_{i=1}^{\mathsf{D}+1} d_i \cdot \frac{\mathsf{cm} - \beta_i}{\tau - \alpha_i} \\
&= \sum_{i=1}^{\mathsf{D}+1} d_i \cdot \mathsf{pf}_i \\
&= h_2 \ .
\end{aligned}$$

Hence,

$$\Pr\left[\begin{array}{l|l} \forall i \neq j \in [\mathsf{D}+1], \alpha_i \neq \alpha_j & \mathsf{pp}_{\mathsf{PC}} \leftarrow \mathsf{KZG.Gen}(1^\lambda, \mathsf{D}) \\ \wedge \forall i \in [\mathsf{D}+1], \mathsf{KZG.Check}(\mathsf{pp}_{\mathsf{PC}}, \mathsf{cm}, \alpha_i, \beta_i, \mathsf{pf}_i) = 1 & \mathsf{ai} \leftarrow \mathcal{D} \\ \wedge \mathsf{cm} \neq \mathsf{KZG.Commit}(\mathsf{pp}_{\mathsf{PC}}, \mathsf{Lagrange}(\{(\alpha_i, \beta_i)\}_{i\in[\mathsf{D}+1]})) & (\mathsf{cm}, \{(\alpha_i, \beta_i, \mathsf{pf}_i)\}_{i\in[\mathsf{D}+1]}) \leftarrow A_{\mathsf{FCIB}}(\mathsf{pp}_{\mathsf{PC}}, \mathsf{ai}) \end{array}\right]$$

$$\leq \Pr\left[\begin{array}{l|l} S \subseteq \mathbb{F},\ |S| = \mathsf{D}+1 & \mathsf{pp}_{\mathsf{PC}} \leftarrow \mathsf{KZG.Gen}(1^\lambda, \mathsf{D}) \\ \wedge h_1, h_2 \in \mathbb{G}_1, h_1 \neq [0]_1 & (S, h_1, h_2) \leftarrow A_{\mathsf{ARSDH}}(\mathsf{pp}_{\mathsf{PC}}) \\ \wedge h_2 = \frac{1}{Z_S(\tau)} \cdot h_1 & \end{array}\right]$$

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

$$\leq \epsilon_{\mathsf{ARSDH}}(\lambda, \mathsf{D}, t_{\mathsf{ARSDH}}) \ .$$

Since the relation holds for any auxiliary input distribution $\mathcal{D}$ and t_{FCIB}-size adversary A_{FCIB} against interpolation binding, we conclude that $\epsilon_{\mathsf{FCIB}}(\lambda, \mathsf{D}, t_{\mathsf{FCIB}}) \leq \epsilon_{\mathsf{ARSDH}}(\lambda, \mathsf{D}, t_{\mathsf{ARSDH}})$, which completes the proof. □

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 Bulleproofs-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* $\epsilon_{\mathsf{DLogRel}} = \epsilon_{\mathsf{DLogRel}}(\lambda, \mathsf{D}, \epsilon_{\mathsf{DLogRel}})$. *Then for every security parameter* $\lambda \in \mathbb{N}$*, polynomial degree bound* $\mathsf{D} \in \mathbb{N}$*, sample set size* $\mathsf{L} \in \mathbb{N}$*, and adversary size bound* $t_{\mathsf{FC}} \in \mathbb{N}$*,* $\mathsf{PC}_{\mathsf{sqrt}}$ *(Construction C.2) has function binding error*

$$\epsilon_{\mathsf{FC}}(\lambda, \mathsf{D}, \mathsf{L}, t_{\mathsf{FC}}) \leq \epsilon_{\mathsf{DLogRel}}(\lambda, \sqrt{\mathsf{D}+1}, t_{\mathsf{DLogRel}}) \ ,$$

where $t_{\mathsf{DLogRel}} \leq t_{\mathsf{FC}} + \mathsf{L}^2 + (\mathsf{D}+1)^{\frac{3}{2}} + (\mathsf{L}+1)\sqrt{\mathsf{D}+1}$.

Throughout this section, we assume that every polynomial p has degree at most D where $\mathsf{D}+1 = d^2$ for some $d \in \mathbb{N}$. [4] For a polynomial p of degree at most D, we define the $d \times d$ matrix P as

$$\mathsf{P} \coloneqq (\mathsf{p}_{(i-1)+d(j-1)})_{i,j\in[d]} = \begin{bmatrix} \mathsf{p}_0 & \cdots & \mathsf{p}_{d(d-1)} \\ \vdots & \ddots & \vdots \\ \mathsf{p}_{d-1} & \cdots & \mathsf{p}_{\mathsf{D}} \end{bmatrix} \ ,$$

such that

$$\begin{bmatrix} 1 & x & \ldots & x^{d-1} \end{bmatrix} \mathsf{P} \begin{bmatrix} 1 \\ x^d \\ \vdots \\ x^{d(d-1)} \end{bmatrix} = \sum_{i,j\in[d]} \mathsf{p}_{(i-1)+d(j-1)} x^{i-1} x^{d(j-1)} = \mathsf{p}(x) \ .$$

Construction C.2. Let $\mathbb{G}$ be a group of prime order $p(\lambda) \geq 2^\lambda$.

- $\mathsf{PC}_{\mathsf{sqrt}}.\mathsf{Gen}(1^\lambda, \mathsf{D})$:
  1. Sample random generators $(G_j)_{j\in[d]} \leftarrow \mathbb{G}^d$.
  2. Output $\mathsf{pp}_{\mathsf{PC}} = (G_j)_{j\in[d]}$.
- $\mathsf{PC}_{\mathsf{sqrt}}.\mathsf{Commit}(\mathsf{pp}_{\mathsf{PC}}, \mathsf{p})$:
  1. Parse $\mathsf{pp}_{\mathsf{PC}}$ as $(G_j)_{j\in[d]}$.

[4] If this condition is not satisfied, one can consider the next $\mathsf{D}' > \mathsf{D}$ such that $\mathsf{D}'+1$ is square. Alternatively, one can define a variation of this PCS where the coefficient matrix P is rectangular rather than square.

2. Compute $\mathsf{cm} = (\mathsf{cm}_i)_{i\in[d]}$, where $\mathsf{cm}_i \coloneqq \sum_{j\in[d]} \mathsf{p}_{(i-1)+d(j-1)} G_j$ is a Pedersen commitment to the i-th row of P.
3. Set $\mathsf{aux} \coloneqq \mathsf{P}$.
4. Output $(\mathsf{cm}, \mathsf{aux})$.

- $\mathsf{PC}_{\mathsf{sqrt}}.\mathsf{Open}(\mathsf{pp}_{\mathsf{PC}}, \mathsf{aux}, \alpha)$:
  1. Parse $\mathsf{pp}_{\mathsf{PC}}$ as $(G_j)_{j\in[d]}$ and aux as P.
  2. Set $\beta \coloneqq \mathsf{p}(\alpha)$.
  3. Compute $\mathsf{pf} \coloneqq \begin{bmatrix}1 & \alpha & \dots & \alpha^{d-1}\end{bmatrix} \mathsf{P}$, with $\mathsf{pf}_j = \sum_{i\in[d]} \mathsf{p}_{(i-1)+d(j-1)} \cdot \alpha^{i-1}$.
  4. Output (β, pf).

- $\mathsf{PC}_{\mathsf{sqrt}}.\mathsf{Check}(\mathsf{pp}_{\mathsf{PC}}, \mathsf{cm}, \alpha, \beta, \mathsf{pf})$:
  1. Parse $\mathsf{pp}_{\mathsf{PC}}$ as $(G_j)_{j\in[d]}$.
  2. Check that $\sum_{i\in[d]} \alpha^{i-1}\mathsf{cm}_i = \sum_{j\in[d]} \mathsf{pf}_j G_j$ and $\beta = \sum_{j\in[d]} \mathsf{pf}_j \alpha^{d(j-1)}$.

We now show that $\mathsf{PC}_{\mathsf{sqrt}}$ 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 $\epsilon_{\mathsf{DLogRel}} = \epsilon_{\mathsf{DLogRel}}(\lambda, n, t_{\mathsf{DLogRel}})$ if for every security parameter λ, length parameter $n \in \mathbb{N}$, adversary size $t_{\mathsf{DLogRel}} \in \mathbb{N}$, t_{DLogRel}-sized adversary A_{DLogRel}, and group $\mathbb{G}$ of prime order $p(\lambda) \geq 2^\lambda$,*

$$\Pr\left[\begin{array}{l|l} \exists\, i \in [n] : a_i \neq 0 \wedge & (G_1, \dots, G_n) \leftarrow \mathbb{G}^n \\ \sum_{i\in[n]} a_i G_i = 0 & (a_1, \dots, a_n) \leftarrow A_{\mathsf{DLogRel}}(\mathbb{G}, G_1, \dots, G_n) \end{array}\right] \leq \epsilon_{\mathsf{DLogRel}}(\lambda, n, t_{\mathsf{DLogRel}}) \ .$$

Definition C.4 (Discrete logarithm assumption)**.** *The discrete logarithm assumption holds with error $\epsilon_{\mathsf{DL}} = \epsilon_{\mathsf{DL}}(\lambda, t_{\mathsf{DL}})$ if for every security parameter λ, adversary size $t_{\mathsf{DL}} \in \mathbb{N}$, t_{DL}-sized adversary A_{DL}, and group $\mathbb{G}$ of prime order $p(\lambda) \geq 2^\lambda$ with generator G,*

$$\Pr\left[\begin{array}{l|l} xG = H & H \leftarrow \mathbb{G} \\ & x \leftarrow A_{\mathsf{DL}}(\mathbb{G}, p, G, H) \end{array}\right] \leq \epsilon_{\mathsf{DL}}(\lambda, t_{\mathsf{DL}}) \ .$$

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 $\lambda \in \mathbb{N}$, $t_{\mathsf{DLogRel}} \in \mathbb{N}$, we have*

$$\epsilon_{\mathsf{DLogRel}}(\lambda, n, t_{\mathsf{DLogRel}}) \leq \epsilon_{\mathsf{DL}}(\lambda, t_{\mathsf{DL}}) + \frac{1}{2^\lambda} \ ,$$

where $t_{\mathsf{DL}} = O(t_{\mathsf{DLogRel}} + n)$.

Proof of Lemma C.1. Given a function binding adversary A_{FC}, we construct the following discrete logarithm relation adversary A_{DLogRel}:

$A_{\mathsf{DLogRel}}(\mathbb{G}, G_1, \dots G_d)$:
1. Set $\mathsf{pp}_{\mathsf{PC}} \coloneqq (G_1, \dots, G_d)$.
2. Sample $\mathsf{ai}_{\mathsf{FC}} \leftarrow \mathcal{D}_{\mathsf{FC}}$

3. $(\mathsf{cm}, (\alpha_a, \beta_a, \mathsf{pf}_a)_{a\in[\mathsf{L}]}) \leftarrow A_{\mathsf{FC}}(\mathsf{pp}_{\mathsf{PC}}, \mathsf{ai}_{\mathsf{FC}})$.
4. If there exists $a \neq b \in [\mathsf{L}]$ such that $\alpha_a = \alpha_b$ but $\beta_a \neq \beta_b$: Output $(\mathsf{pf}_{a,j} - \mathsf{pf}_{b,j})_{j\in[d]}$.
5. If $\deg(\mathsf{Lagrange}(\{((\alpha_a, \beta_a))_{a\in[\mathsf{L}]}\})) > \mathsf{D}$:
   (a) Find d distinct indices $a_1, \ldots, a_d \in [\mathsf{L}]$ such that $\alpha_{a_1}, \ldots, \alpha_{a_d}$ are distinct.
   (b) Compute $\begin{bmatrix} \mathsf{p}_{1,1} & \cdots & \mathsf{p}_{1,d} \\ \vdots & \ddots & \vdots \\ \mathsf{p}_{d,1} & \cdots & \mathsf{p}_{d,d} \end{bmatrix} := \begin{bmatrix} 1 & \cdots & \alpha_{a_1}^{d-1} \\ \vdots & \ddots & \vdots \\ 1 & \cdots & \alpha_{a_d}^{d-1} \end{bmatrix}^{-1} \begin{bmatrix} \mathsf{pf}_{a_1,1} & \cdots & \mathsf{pf}_{a_1,d} \\ \vdots & \ddots & \vdots \\ \mathsf{pf}_{a_d,1} & \cdots & \mathsf{pf}_{a_d,d} \end{bmatrix}$.
   (c) Find $a^* \in [\mathsf{L}] \setminus \{a_1, \ldots, a_d\}$ such that $\exists\, j \in [d] : \mathsf{pf}_{a^*,j} \neq \sum_{i\in[d]} \mathsf{p}_{i,j}\alpha_{a^*}^{i-1}$.
   (d) Output $(\mathsf{pf}_{a^*,j} - \sum_{i\in[d]} \mathsf{p}_{i,j}\alpha_{a^*}^{i-1})_{j\in[d]}$.
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 $d + \mathsf{L} \cdot d$ steps. This yields a total running time of at most

$$t_{\mathsf{FC}} + \max(\mathsf{L}^2 + d, \mathsf{L}^2 + d^3 + d + \mathsf{L} \cdot d) \leq t_{\mathsf{FC}} + \mathsf{L}^2 + (\mathsf{D}+1)^{\frac{3}{2}} + (\mathsf{L}+1)\sqrt{\mathsf{D}+1} \ .$$

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 $\mathsf{PC}_{\mathsf{sqrt}}.\mathsf{Check}(\mathsf{pp}_{\mathsf{PC}}, \mathsf{cm}, \alpha_a, \beta_a, \mathsf{pf}_a) = 1$ and $\mathsf{PC}_{\mathsf{sqrt}}.\mathsf{Check}(\mathsf{pp}_{\mathsf{PC}}, \mathsf{cm}, \alpha_b, \beta_b, \mathsf{pf}_b) = 1$, it must be that $\beta_a = \sum_{j\in[d]} \mathsf{pf}_{a,j}\alpha_a^{d(j-1)}$ and $\beta_b = \sum_{j\in[d]} \mathsf{pf}_{b,j}\alpha_b^{d(j-1)}$, and thus $\mathsf{pf}_{a,j} \neq \mathsf{pf}_{b,j}$ for at least one index j. Since $\sum_{i\in[d]} \alpha_a^{i-1}\mathsf{cm}_i = \sum_{j\in[d]} \mathsf{pf}_{a,j}G_j$ and $\sum_{i\in[d]} \alpha_b^{i-1}\mathsf{cm}_i = \sum_{j\in[d]} \mathsf{pf}_{b,j}G_j$ from the verification equations, $\sum_{j\in[d]}(\mathsf{pf}_{a,j} - \mathsf{pf}_{b,j})G_j = 0$, 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 $\alpha_{a_1}, \ldots, \alpha_{a_d}$ 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 $\mathsf{pf}_{a,j} = \sum_{i\in[d]} \mathsf{p}_{i,j}\alpha_a^{i-1}$ for all $j \in [d]$. Then, since $\mathsf{PC}_{\mathsf{sqrt}}.\mathsf{Check}(\mathsf{pp}_{\mathsf{PC}}, \mathsf{cm}, \alpha_a, \beta_a, \mathsf{pf}_a) = 1$ (and thus $\beta_a = \sum_{j\in[d]} \mathsf{pf}_{a,j}\alpha_a^{d(j-1)}$) for all $a \in [\mathsf{L}]$, we have that for all $a \in [\mathsf{L}]$,

$$\begin{aligned} \beta_a &= \sum_{j\in[d]} \mathsf{pf}_{a,j}\alpha_a^{d(j-1)} \\ &= \sum_{j\in[d]} (\sum_{i\in[d]} \mathsf{p}_{i,j}\alpha_a^{i-1})\alpha_a^{d(j-1)} \\ &= \sum_{i,j\in[d]} \mathsf{p}_{i,j}\alpha_a^{(i-1)+d(j-1)} \ , \end{aligned}$$

which contradicts the assumption that $\deg(\mathsf{Lagrange}(\{((\alpha_a, \beta_a))_{a\in[\mathsf{L}]}\})) > \mathsf{D}$.

Finally, A_{DLogRel} outputs a non-trivial discrete logarithm relation $(\mathsf{pf}_{a^*,j} - \sum_{i\in[d]} \mathsf{p}_{i,j}\alpha_{a^*}^{i-1})_{j\in[d]}$ in Step 5d. To show this, we first note that the i-row $(\mathsf{p}_{i,j})_{j\in[d]}$ is a pre-image of the commitment cm_i, i.e., $\mathsf{cm}_i = \sum_{j\in[d]} \mathsf{p}_{i,j}G_j$ for all $i \in [d]$. More precisely, since $\mathsf{PC}_{\mathsf{sqrt}}.\mathsf{Check}(\mathsf{pp}_{\mathsf{PC}}, \mathsf{cm}, \alpha_a, \beta_a, \mathsf{pf}_a) = 1$ (and thus

$\sum_{i\in[d]} \alpha_a^{i-1}\mathsf{cm}_i = \sum_{j\in[d]} \mathsf{pf}_{a,j}G_j$) for all $a \in [\mathsf{L}]$, we have that for all $l \in [d]$:

$$\begin{aligned}\sum_{i\in[d]} \alpha_{a_l}^{i-1}\mathsf{cm}_i &= \sum_{j\in[d]} \mathsf{pf}_{a_l,j}G_j \\ &= \sum_{j\in[d]} (\sum_{i\in[d]} \mathsf{p}_{i,j}\alpha_{a_l}^{i-1})G_j \\ &= \sum_{i\in[d]} \alpha_{a_l}^{i-1} \sum_{j\in[d]} \mathsf{p}_{i,j}G_j \ ,\end{aligned}$$

and thus

$$\begin{bmatrix} 1 & \dots & \alpha_{a_1}^{d-1} \\ \vdots & \ddots & \vdots \\ 1 & \dots & \alpha_{a_d}^{d-1} \end{bmatrix} \begin{bmatrix} \mathsf{cm}_1 - \sum_{j\in[d]} \mathsf{p}_{1,j}G_j \\ \vdots \\ \mathsf{cm}_d - \sum_{j\in[d]} \mathsf{p}_{d,j}G_j \end{bmatrix} = \mathbf{0} \ .$$

Since the Vandermonde matrix is invertible, it must be that $\mathsf{cm}_i = \sum_{j\in[d]} \mathsf{p}_{i,j}G_j$ for all $i \in [d]$. This immediately implies that A_{DLogRel}'s output in Step 5d is a non-trivial discrete logarithm relation, since

$$\begin{aligned}\sum_{j\in[d]} (\mathsf{pf}_{a^*,j} - \sum_{i\in[d]} \mathsf{p}_{i,j}\alpha_{a^*}^{i-1})G_j &= \sum_{j\in[d]} \mathsf{pf}_{a^*,j}G_j - \sum_{i,j\in[d]} \mathsf{p}_{i,j}\alpha_{a^*}^{i-1}G_j \\ &= \sum_{i\in[d]} \alpha_{a^*}^{i-1}\mathsf{cm}_i - \sum_{i,j\in[d]} \mathsf{p}_{i,j}\alpha_{a^*}^{i-1}G_j \\ &= 0 \ .\end{aligned}$$

□

C.2 Bulletproofs-style polynomial commitment scheme

We show that a Bulletproofs-style [BCCGP16] interactive polynomial commitment scheme $\mathsf{PC}_{\mathsf{BP}}$ with logarithmically many rounds satisfies (state-restoration) function binding under the discrete logarithm assumption.

Lemma C.6 (State-restoration function binding of $\mathsf{PC}_{\mathsf{BP}}$)**.** *Assume the expected-time discrete logarithm relation assumption holds with error $\epsilon^\star_{\mathsf{DLogRel}} = \epsilon^\star_{\mathsf{DLogRel}}(\lambda, \mathsf{D}, t^\star_{\mathsf{DLogRel}})$ (Definition C.8), $\mathsf{PC}_{\mathsf{BP}}$ (Construction C.7) has state-restoration function binding error*

$$\epsilon^{\mathsf{SR}}_{\mathsf{FC}}(\lambda, \mathsf{D}, \mathsf{L}, s_{\mathsf{FC}}, \mathsf{m}_{\mathsf{FC}}, t_{\mathsf{FC}}) \leq \mathsf{L} \cdot (\mathsf{m}_{\mathsf{FC}} + 1) \cdot \frac{1 + 2\log(\mathsf{D}+1)}{2^\lambda} + \epsilon^\star_{\mathsf{DLogRel}}(\lambda, \mathsf{D}+2, t^\star_{\mathsf{DLogRel}}) \ ,$$

where $t^\star_{\mathsf{DLogRel}} = O\left(\mathsf{L} \cdot (\mathsf{m}_{\mathsf{FC}} + 1) \cdot (2(\mathsf{D}+1)^{\log 3})(t_{\mathsf{FC}} + \mathsf{poly}(\mathsf{D})) + \mathsf{L}^2 + \mathsf{L} \cdot (\mathsf{D}+1)^{\log 3}\right)$.

In Appendix C.2.1, we present an interactive polynomial commitment scheme $\mathsf{PC}_{\mathsf{BP}}$ inspired by the Bulletproofs protocol [BCCGP16] (we closely follow the description of Bünz et al. [BCMS20, Appendix A]). We show that $\mathsf{PC}_{\mathsf{BP}}$ 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 $\mathsf{D}+1$ is a power of 2. We write $\vec{\mathsf{p}} = (\mathsf{p}_0, \ldots, \mathsf{p}_{\mathsf{D}})$ for the coefficients of a polynomial $\mathsf{p} \in \mathbb{F}[X]^{\leq \mathsf{D}}$. We denote by $\mathsf{L}((v_1, \ldots, v_{2k})) \coloneqq (v_1, \ldots, v_k)$ the operation which returns the left half of a vector, and similarly $\mathsf{R}((v_1, \ldots, v_{2k}))$ for the right half.

Construction C.7. Let $\mathbb{G}$ be a group of prime order $p(\lambda) \geq 2^\lambda$.

- $\mathsf{PC}_{\mathsf{BP}}.\mathsf{Gen}(1^\lambda, \mathsf{D})$:
  1. Sample random generators $\vec{G} \leftarrow \mathbb{G}^{\mathsf{D}+1}$, $H \leftarrow \mathbb{G}$.
  2. Output $\mathsf{pp}_{\mathsf{FC}} = (\vec{G}, H)$.
- $\mathsf{PC}_{\mathsf{BP}}.\mathsf{Commit}(\mathsf{pp}_{\mathsf{FC}}, \mathsf{p})$: Outputs $\mathsf{cm} \coloneqq \langle \vec{\mathsf{p}}, \vec{G} \rangle$.
- $\mathsf{PC}_{\mathsf{BP}}.\mathsf{Open}, \mathsf{PC}_{\mathsf{BP}}.\mathsf{FC}.\mathsf{Check}$: The interactive protocol between $\mathcal{P}_{\mathsf{FC}}$ and $\mathcal{V}_{\mathsf{FC}}$ with common reference string $\mathsf{pp}_{\mathsf{FC}}$, instance $(\mathsf{cm}, \alpha, \beta)$ and witness p proceeds as follows:
  1. $\mathcal{V}_{\mathsf{FC}}$ samples $\xi_0 \leftarrow \mathbb{F} \setminus \{0\}$ uniformly at random, and sends it to $\mathcal{P}_{\mathsf{FC}}$.
  2. Both $\mathcal{P}_{\mathsf{FC}}$ and $\mathcal{V}_{\mathsf{FC}}$ set
     - $H' \coloneqq \xi_0 H$,
     - $\vec{\alpha}^{(0)} \coloneqq (1, \alpha, \ldots, \alpha^{\mathsf{D}})$, and
     - $\vec{G}^{(0)} \coloneqq \vec{G}$.
  3. $\mathcal{P}_{\mathsf{FC}}$ sets $\vec{\mathsf{p}}^{(0)} \coloneqq \vec{\mathsf{p}}$.
  4. $\mathcal{V}_{\mathsf{FC}}$ sets $C^{(0)} \coloneqq \mathsf{cm} + \beta \cdot H'$.
  5. For $i \in \{1, \ldots, \log(\mathsf{D}+1)\}$:
     (a) $\mathcal{P}_{\mathsf{FC}}$ sends $L^{(i)}$ and $R^{(i)}$ to the verifier, where
     $$L^{(i)} \coloneqq \langle \mathsf{R}(\vec{\mathsf{p}}^{(i-1)}), \mathsf{L}(\vec{G}^{(i-1)}) \rangle + \langle \mathsf{R}(\vec{\mathsf{p}}^{(i-1)}), \mathsf{L}(\vec{\alpha}^{(i-1)}) \rangle H'$$
     $$R^{(i)} \coloneqq \langle \mathsf{L}(\vec{\mathsf{p}}^{(i-1)}), \mathsf{R}(\vec{G}^{(i-1)}) \rangle + \langle \mathsf{L}(\vec{\mathsf{p}}^{(i-1)}), \mathsf{R}(\vec{\alpha}^{(i-1)}) \rangle H'$$
     (b) $\mathcal{V}_{\mathsf{FC}}$ samples $\xi_i \leftarrow \mathbb{F} \setminus \{0\}$, computes $C^{(i)} \coloneqq \xi_i^{-1} L^{(i)} + C^{(i-1)} + \xi_i R^{(i)}$, and sends ξ_i to $\mathcal{P}_{\mathsf{FC}}$.
     (c) $\mathcal{P}_{\mathsf{FC}}$ computes the inputs for the next round as follows:
     - $\vec{\mathsf{p}}^{(i)} \coloneqq \mathsf{L}(\vec{\mathsf{p}}^{(i-1)}) + \xi_i^{-1}\, \mathsf{R}(\vec{\mathsf{p}}^{(i-1)})$,
     - $\vec{\alpha}^{(i)} \coloneqq \mathsf{L}(\vec{\alpha}^{(i-1)}) + \xi_i\, \mathsf{R}(\vec{\alpha}^{(i-1)})$, and
     - $\vec{G}^{(i)} \coloneqq \mathsf{L}(\vec{G}^{(i-1)}) + \xi_i\, \mathsf{R}(\vec{G}^{(i-1)})$.
  6. $\mathcal{P}_{\mathsf{FC}}$ sends the constant polynomial $u \coloneqq \vec{\mathsf{p}}^{(\log(\mathsf{D}+1))} \in \mathbb{F}$ to $\mathcal{V}_{\mathsf{FC}}$.
  7. $\mathcal{V}_{\mathsf{FC}}$ defines $h(X) \coloneqq \prod_{i=0}^{\log(\mathsf{D}+1)-1} (1 + \xi_{\log(\mathsf{D}+1)-i} X^{2^i})$ (with coefficient vector $\vec{h}$) and checks that $C^{(\log(\mathsf{D}+1))} = u\langle \vec{h}, \vec{G} \rangle + u \cdot h(\alpha) H'$.

C.2.2 Function binding

We show Lemma C.6 by showing that $\mathsf{PC}_{\mathsf{BP}}$ is special function binding under the expected-time discrete logarithm relation assumption, and applying Lemma A.2.

Finally, we show that $\mathsf{PC}_{\mathsf{BP}}$ 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 $\epsilon^\star_{\mathsf{DLogRel}}$ if for every security parameter λ, length parameter $n \in \mathbb{N}$, expected $t^\star_{\mathsf{DLogRel}}$-time adversary A_{DLogRel} and group $\mathbb{G}$ of prime order $p(\lambda) \geq 2^\lambda$,*

$$\Pr\left[\exists a_i \neq 0 \wedge \textstyle\sum_{i\in[n]} a_i G_i = 0 \;\middle|\; \begin{array}{l} G_1,\ldots,G_n \leftarrow \mathbb{G} \\ a_1,\ldots,a_n \in \mathbb{Z}_p \leftarrow A_{\mathsf{DLogRel}}(\mathbb{G}, G_1,\ldots,G_n) \end{array} \right] \leq \epsilon^\star_{\mathsf{DLogRel}}(\lambda, n, t^\star_{\mathsf{DLogRel}}) \ .$$

Definition C.9 (Expected-time discrete logarithm assumption)**.** *The expected-time discrete logarithm assumption holds with error $\epsilon^\star_{\mathsf{DL}}$ if for every security parameter λ, expected $t^\star_{\mathsf{DL}}$-time adversary A_{DL}, and group $\mathbb{G}$ of prime order $p(\lambda) \geq 2^\lambda$ with generator G,*

$$\Pr\left[xG = H \;\middle|\; \begin{array}{l} H \leftarrow \mathbb{G} \\ x \leftarrow A_{\mathsf{DL}}(\mathbb{G}, p, G, H) \end{array} \right] \leq \epsilon^\star_{\mathsf{DL}}(\lambda, t^\star_{\mathsf{DL}}) \ .$$

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 ⇒ 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 $\lambda \in \mathbb{N}$, $t^\star_{\mathsf{DLogRel}} \in \mathbb{N}$, we have*

$$\epsilon^\star_{\mathsf{DLogRel}}(\lambda, n, t^\star_{\mathsf{DLogRel}}) \leq \epsilon^\star_{\mathsf{DL}}(\lambda, t^\star_{\mathsf{DL}}) + \frac{1}{2^\lambda} \ ,$$

where $t^\star_{\mathsf{DL}} = O(t^\star_{\mathsf{DLogRel}} + n)$.

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 $\mathsf{PC}_{\mathsf{BP}}$)**.** $\mathsf{PC}_{\mathsf{BP}}$ *has $(2,3,\ldots,3)$-special function binding error*

$$\epsilon_{\mathsf{SFB}}(\lambda, \mathsf{D}+1, \mathsf{L}, t^\star_{\mathsf{SFB}}) \leq \epsilon^\star_{\mathsf{DLogRel}}(\lambda, \mathsf{D}+2, t^\star_{\mathsf{DLogRel}}) \ ,$$

where $t^\star_{\mathsf{DLogRel}} = O(t^\star_{\mathsf{SFB}} + \mathsf{L}^2 + \mathsf{L}\cdot(\mathsf{D}+1)^{\log 3})$.

Proof. We define the following adversary A_{SFB}, which, when run on the output of a succesful special function binding reductor $\mathfrak{R}$, outputs a non-trivial discrete logarithm relation for the generators $G_0,\ldots,G_d,H$. In the following, let $d := \log(\mathsf{D}+1)$ denote the number of rounds in the opening protocol.

$A_{\mathsf{SFB}}(\mathsf{pp}_{\mathsf{FC}}, (\mathsf{cm}, ((\alpha_i, \beta_i, \mathsf{T}_i))_{i\in[\mathsf{L}]}))$:

1. For $i \in [\mathsf{L}]$:
   (a) Parse T_i as prover messages $((L_{i,r}^{(j_0,\ldots,j_{r-1})}, R_{i,r}^{(j_0,\ldots,j_{r-1})}))_{r\in[d],j_0\in[2],j_1,\ldots,j_{r-1}\in[3]}, (u_i^{(j_0,\ldots,j_d)})_{j_0\in[2],j_1,\ldots,j_d\in[3]}$ and verifier challenges $(\xi_{i,r}^{(j_0,\ldots,j_r)})_{r\in[d],j_0\in[2],j_1,\ldots,j_r\in[3]}$.
   (b) For $(j_0,\ldots,j_d) \in \{1,2\}\times\{1,2,3\}^d$:
      i. Set $J_i^{(j_0,\ldots,j_d)} := \sum_{s\in[d]} (\xi_{i,s}^{(j_0,\ldots,j_s)})^{-1} L_{i,s}^{(j_0,\ldots,j_{s-1})} + \mathsf{cm} + \beta_i \xi_{i,0}^{(j_0)} H + \sum_{s\in[d]} \xi_{i,s}^{(j_0,\ldots,j_s)} R_{i,s}^{(j_0,\ldots j_{s-1})}$.
      ii. Set $h_i^{(j_0,\ldots,j_d)}(X) := \prod_{s=0}^{\log(\mathsf{D}+1)-1} (1 + \xi_{i,\log(\mathsf{D}+1)-s}^{(j_0,\ldots,j_{\log(\mathsf{D}+1)-s})} X^{2^s})$
      iii. Set $\mathsf{p}_i^{(j_0,\ldots,j_d)}(X) := u_i^{(j_0,\ldots,j_d)} \cdot h_i^{(j_0,\ldots,j_d)}(X)$, with coefficient vector $\vec{\mathsf{p}}_i^{(j_0,\ldots,j_d)}$.

iv. Check that

$$J_i^{(j_0,\ldots,j_d)} = \langle \vec{\mathsf{p}}_i^{(j_0,\ldots,j_d)}, \vec{G} \rangle + \mathsf{p}_i^{(j_0,\ldots,j_d)}(\alpha_i) \cdot \xi_i^{(j_0)} H \ ;$$

abort otherwise.

(c) For $r = d$ to 1: For $(j_0, \ldots, j_{r-1}) \in \{1,2\} \times \{1,2,3\}^{r-1}$:

i. Check that $\xi_{i,r}^{(j_0,\ldots,j_{r-1},1)}, \xi_{i,r}^{(j_0,\ldots,j_{r-1},2)}, \xi_{i,r}^{(j_0,\ldots,j_{r-1},3)}$ are distinct; abort otherwise.

ii. Compute $(\nu_j)_{j\in[3]}$ such that

$$\begin{bmatrix} (\xi_{i,r}^{(j_0,\ldots,j_{r-1},1)})^{-1} & (\xi_{i,r}^{(j_0,\ldots,j_{r-1},2)})^{-1} & (\xi_{i,r}^{(j_0,\ldots,j_{r-1},3)})^{-1} \\ 1 & 1 & 1 \\ \xi_{i,r}^{(j_0,\ldots,j_{r-1},1)} & \xi_{i,r}^{(j_0,\ldots,j_{r-1},2)} & \xi_{i,r}^{(j_0,\ldots,j_{r-1},3)} \end{bmatrix} \begin{bmatrix} \nu_1 \\ \nu_2 \\ \nu_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \ .$$

iii. Set $J_i^{(j_0,\ldots,j_{r-1})} \coloneqq \sum_{j\in[3]} \nu_j J_i^{(j_0,\ldots,j_{r-1},j)}$

$$= \sum_{s\in[r-1]} (\xi_{i,s}^{(j_0,\ldots,j_s)})^{-1} L_{i,s}^{(j_0,\ldots,j_{s-1})} + \mathsf{cm} + \beta_i \xi_{i,0}^{(j_0)} H + \sum_{s\in[r-1]} \xi_{i,s}^{(j_0,\ldots,j_s)} R_{i,s}^{(j_0,\ldots j_{s-1})} \ .$$

iv. Set $\mathsf{p}_i^{(j_0,\ldots,j_{r-1})} \coloneqq \sum_{j\in[3]} \nu_j \mathsf{p}_i^{(j_0,\ldots,j_{r-1},j)}$.

(At this point, each $\mathsf{p}_i^{(j_0)}$ satisfies $\mathsf{cm} = \langle \vec{\mathsf{p}}_i^{(j_0)}, \vec{G} \rangle + (\mathsf{p}_i^{(j_0)}(\alpha_i) - \beta_i)\xi_{i,0}^{(j_0)} H$.)

2. If $\mathsf{p}_i^{(j_0)} \neq \mathsf{p}_j^{(j_0')}$ for some $i, j \in [\mathsf{L}], j_0, j_0' \in [2]$ (i.e., $\langle \vec{\mathsf{p}}_i^{(j_0)}, \vec{G} \rangle + (\mathsf{p}_i^{(j_0)}(\alpha_i) - \beta_i)\xi_{i,0}^{(j_0)} H = \langle \vec{\mathsf{p}}_j^{(j_0')}, \vec{G} \rangle + (\mathsf{p}_j^{(j_0')}(\alpha_{j,m}) - \beta_j)\xi_{j,0}^{(j_0')} H$):

(a) Set $\mathsf{p} \coloneqq \mathsf{p}_i^{(j_0)} - \mathsf{p}_j^{(j_0')}$.

(b) Output $\vec{\mathsf{p}}, (\mathsf{p}_i^{(j_0)}(\alpha_i) - \beta_i)\xi_{i,0}^{(j_0)} - (\mathsf{p}_j^{(j_0')}(\alpha_j) - \beta_j)\xi_{j,0}^{(j_0')}$.

3. Otherwise:

(a) Set $\mathsf{p} \coloneqq \mathsf{p}_1^{(1)} = \ldots = \mathsf{p}_{\mathsf{L}}^{(2)}$.

(b) If $\mathsf{p}(\alpha_i) \neq \beta_i$ for some $i \in [\mathsf{L}]$ (i.e., $(\mathsf{p}(\alpha_i) - \beta_i)\xi_{i,0}^{(1)} H = (\mathsf{p}(\alpha_i) - \beta_i)\xi_{i,0}^{(2)} H$):

Output $(0)_{i\in[\mathsf{D}+1]}, (\mathsf{p}(\alpha_i) - \beta_i)(\xi_{i,0}^{(1)} - \xi_{i,0}^{(2)})$.

(c) Otherwise: abort.

Running time. Each tree of transcripts has $2 \cdot 3^{\log(\mathsf{D}+1)} = 2 \cdot (\mathsf{D}+1)^{\log 3}$ nodes. In Step 1, A_{SFB} performs at most $\mathsf{L}(2 \cdot \log(\mathsf{D}+1)^{\log 3} + \sum_{r\in[\log(\mathsf{D}+1)]} 2 \cdot 3^r)$ operations. In Step 2, A_{SFB} searches for disagreeing answers for the same query among $2\mathsf{L}$ query-answer pairs, which requires $O(\mathsf{L}^2)$ operations. Finally, Step 3 only requires a linear scan over L values.

Success probability. Let $((\alpha_i, \beta_i, \mathsf{T}_i))_{i\in[\mathsf{L}]}$ 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 $\mathsf{SRTreeFinder}$ 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 $\mathsf{p}_i^{(j_0)} - \mathsf{p}_{j,m}^{(j_0')}$ is non-zero, and A_{SFB} outputs a valid discrete logarithm relation. Finally, note that the function binding condition

$$(\exists i, j \in [\mathsf{L}] : \alpha_i = \alpha_j \wedge \beta_i \neq \beta_j) \veebar \deg(\mathsf{Lagrange}(\{(\alpha_i, \beta_i)\})) > \mathsf{D}$$

is equivalent to the condition $\exists i \in [\mathsf{L}] : \mathsf{p}(\alpha_i) \neq \beta_i$ tested in Step 3b, which means that R does not abort in Step 3. In this case, $(\mathsf{p}(\alpha_i) - \beta_i)(\xi_{i,0}^{(1)} - \xi_{i,0}^{(2)})$ is non-zero, and A_{SFB} also outputs a valid discrete logarithm relation. □

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) $\deg(\mathsf{p}) \leq \mathsf{D}$, (ii) cm is a commitment to p, and (iii) $\mathsf{p}(\alpha) = \beta$ [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 $\mathsf{L} = 1$, 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 $(2, 3, \ldots, 3)$. 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 $\mathsf{PC}_{\mathsf{BP}}$ is state-restoration soundness. For $\mathsf{PC}_{\mathsf{BP}}$, we are not aware of any proof that specifically targets state-restoration soundness, or even soundness; however, note that for $\mathsf{L} = 1$, 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.

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.

[AGLMS22] Arasu Arun, Chaya Ganesh, Satya V. Lokam, Tushar Mopuri, and Sriram Sridhar. *Dew: Transparent Constant-sized zkSNARKs*. Cryptology ePrint Archive, Report 2022/419. 2022.

[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*. Cryptology ePrint Archive, Paper 2025/329. 2025. URL: https://eprint.iacr.org/2025/329.

[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.

[CDDGS25] Alessandro Chiesa, Marcel Dall'Agnol, Zijing Di, Ziyi Guan, and Nicholas Spooner. *Quantum Rewinding for IOP-Based Succinct Arguments*. Cryptology ePrint Archive, Report 2025/947. 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.

[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*. Cryptology ePrint Archive, Report 2025/118. 2025.

[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.

[LPS24a] 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.

[LPS24b] Helger Lipmaa, Roberto Parisella, and Janno Siim. *On Knowledge-Soundness of Plonk in ROM from Falsifiable Assumptions*. Cryptology ePrint Archive, Report 2024/994. 2024.

[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: 10.1561/3300000030. 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.