Fully Malicious Authenticated PIR

Marian Dietz and Stefano Tessaro

Paul G. Allen School of Computer Science & Engineering
University of Washington
{mariand,tessaro}@cs.washington.edu

Abstract. Authenticated PIR enables a server to initially commit to a database of N items, for which a client can later privately obtain individual items with complexity sublinear in N, with the added guarantee that the retrieved item is consistent with the committed database. A crucial requirement is *privacy with abort*, i.e., the server should not learn anything about a query *even* if it learns whether the client aborts.

This problem was recently considered by Colombo et al. (USENIX '23), who proposed solutions secure under the assumption that the database is committed to *honestly*. Here, we close this gap, and present a solution that tolerates fully malicious servers that provide potentially malformed commitments. Our scheme has communication and client computational complexity $\mathcal{O}_\lambda(\sqrt{N})$, solely relies on the DDH assumption, and does not introduce heavy machinery (e.g., generic succinct proofs). Privacy with abort holds provided the server succeeds in correctly answering λ validation queries, which, from its perspective, are computationally indistinguishable from regular PIR queries. In fact, server side, our scheme is exactly the DDH-based scheme by Colombo et al.

1 Introduction

Private Information Retrieval (PIR) [14] allows a client to retrieve items from a database of N items held by a server, without revealing which items are being retrieved. Crucially, both the communication and the client's runtime are sublinear in N. In this paper, we are interested in the more challenging setting of *single server* PIR [32], for which constructions are known from a variety of assumptions (cf. e.g. [32,10,35,39,18,19,1,3,21,27,16]).

Maliciously secure PIR. The literature has by now surfaced countless applications of PIR, such as anonymous web search [26], anonymous messaging [4,13], private media delivery [25], certificate transparency logs [36,27], secure analytics [24], secure breach alerts [2] and more. In most use cases, a PIR service is queried by *multiple* clients, but PIR schemes usually assume a semi-honest server and do not ensure integrity. Hence, clients are not guaranteed consistent answers. This paper considers a setting, referred to as *authenticated PIR*, where the server succinctly *commits* to a database $\mathbf{x}$, and then can only provide answers consistent with this commitment. At first, it appears easy to solve the problem: The server publishes a succinct vector commitment d to $\mathbf{x}$ (e.g., the root of a Merkle Tree), and then runs a PIR scheme on a modified database $\mathbf{x}'$ where $\mathbf{x}'[i]$ consists of $\mathbf{x}[i]$ and a valid proof π_i of inclusion with respect to d.

While this solution indeed guarantees integrity, Kushilevitz and Ostrovsky [32] already observed that this approach is prone to so-called *selective failure attacks*, where the server intentionally includes an invalid proof π_i for a position $i \in [N]$, and learning about a failure reveals that the client's query is for this particular index. Such attacks are realistic, as failures are hard to hide (e.g., because an application will behave differently in an observable way), and they have been considered in a number of settings, such as encryption [8,37] and two-party computation [30,11,34,28].

Authenticated PIR. Only very recently, Colombo et al. [15] initiated the study of selective-failure attacks in the context of PIR. They propose in particular single-server PIR schemes based on the LWE and DDH assumptions, both with $O_\lambda(\sqrt{N})$ complexity, which, while less practical than non-authenticated counterparts, offer acceptable performance, integrity, and privacy under selective-failure attacks. However, their model *explicitly disallows* maliciously generated digests. Still, it is natural and prudent to assume that a data owner willing to deviate maliciously from a PIR protocol would also attempt to do so with help of a malicious digest. For this reason, they explicitly pose the question of strengthening their construction to this setting as an open problem.

Earlier proposals to make single-server PIR authentic [38,41] have not considered selective failures, and only provide authenticity guarantees that a query is consistent with *some* valid database, but not a specific one the server a-priori committed to. In contrast, multi-server verifiable PIR [40] considers a similar functionality, however without covering selective failures.

Our contributions. In this paper, we prove that under the DDH assumption, a variant of the scheme from [15] is in fact secure against a fully malicious server generating malicious digests, achieving both integrity and privacy against selective-failure attacks. To achieve the latter property, our solution requires the client to ask λ additional *validation queries* which are indistinguishable from regular PIR queries from the perspective of the server—privacy under selective-failure attacks holds provided these validation queries are successful. We give a version of our scheme that has communication complexity $O_\lambda(\sqrt{N})$ for PIR queries and validation, and the digest can be made constant size.

We give an overview of our scheme and our analysis in Section 1.1. The core of our approach is a new technique which turns an adversary that succeeds in the validation phase into an extraction algorithm able to answer *any* query to the PIR server. This shows that the information of whether a query would abort can be *simulated*, thus reducing selective-failure privacy to regular privacy.

Strawman solutions. To further appreciate our solution, it is helpful to discuss a few alternative solutions based on generic (and heavier) techniques added on top of a simple PIR scheme like the one from [15]. For instance, the server could include in each query response a succinct proof of knowledge of an opening of the digest d to which this response is consistent. This could be done using a SNARK (for which constructions under standard falsifiable assumptions hit well-known barriers [22]), or a succinct two-move proof (which are, as of now, not known to be easier to achieve than SNARKs).

The generic blueprint from Section 3 suggests an alternative where the validation phase is build from a succinct interactive proof of knowledge of an opening of the digest d to a *binary* database. (In essence, we could use the extractor of this proof to achieve answer awareness.) One could use Kilian's four-round protocol [29], which has also been shown to be a proof of knowledge [5]. Roughly, the first protocol round, which consists of a description of a collision-resistant hash function, would be included in the public parameters. Then, Kilian's second round, which is a succinct Merkle commitment to a suitable PCP, would be merged with the digest, whereas the remaining two rounds would constitute the actual validation stage verifying the (committed) PCP by opening a random subset of Merkle paths.

Crucially, not only this approach is more complex and less efficient than what we propose (it makes non-black box use the underlying group), but it is also impossible to see its validation phase as consisting of regular PIR queries, which is an important feature of our own construction.

Other related work. We note here that Ben David et al. [7] recently studied a different and unrelated notion of verifiable PIR which allows the server to prove properties about the database with sublinear complexity. A number of works [6,23,17,31,20] studied robustness of PIR against malicious servers, i.e., the goal is to guarantee answers even when some of the servers fail/deviate, but they rely on the availability of some honest server. There have also been impressive advances on single-server PIR with sublinear server complexity [33,9,12], but this is not considered in the context of this work.

Open problems. The most obvious open question is to develop a lattice analogue of our results, e.g., on top of the simple LWE-based scheme from [15]. We think this should be possible, in principle, but the fact that LWE-based schemes need to be robust to added noise in the server response increases the complexity of our approach in ways we could not manage. Another interesting question is to add updatability to the scheme.

1.1 Technical Overview

The CNCWF DDH-based scheme. Our starting point is the DDH-based scheme from [15] (henceforth referred to as CNCWF), which only provides security against selective-failure attacks when the digest was selected *honestly*. The scheme relies on a cyclic group $\mathbb{G}$ of order q in which the DDH assumption holds. We only discuss here the unbalanced version of the scheme, for which queries consist of N group elements, whereas responses are a single group element. (One can turn this into a scheme with $O(\sqrt{N})$ size queries and responses via standard re-balancing techniques which will preserve all claimed properties.)

In CNCWF, the server initially commits to an N-bit database $\mathbf{x} \in \{0,1\}^N$ via a digest $d = \prod_{i=1}^{N} \mathbf{h}[i]^{\mathbf{x}[i]}$, where $\mathbf{h}$ is a vector of N independent generators of $\mathbb{G}$, included in the public parameters of the scheme, along with an extra generator g. To retrieve item $i \in [N]$, the client samples $r \leftarrow \mathbb{Z}_q, \alpha \leftarrow \mathbb{Z}_q^*$, and computes a *query vector* $\tilde{\mathfrak{e}} \in \mathbb{G}^N$, where

$$\tilde{\mathfrak{e}}[j] = \begin{cases} \mathbf{h}[j]^r \cdot g^\alpha & \text{if } j = i, \\ \mathbf{h}[j]^r & \text{if } j \in [N] \setminus \{i\} \end{cases} .$$

We also write this succinctly as $\tilde{\mathfrak{e}} = \mathbf{h}^r \circ (g^{\mathbf{e}_i})^\alpha$, where $\mathbf{e}_i$ is the i-th unit vector, $\circ$ denotes component-wise vector multiplication, and $g^{\mathbf{v}} = (g^{\mathbf{v}[1]}, \ldots, g^{\mathbf{v}[N]})$. The server's response is $\tilde{e} = \prod_{j=1}^{N} \tilde{\mathfrak{e}}[j]^{\mathbf{x}[j]}$, and the client computes $e = (d^{-r}\tilde{e})^{\alpha^{-1}}$. It returns 1 if $e = g$, 0 if $e = 1$, and otherwise aborts if $e \notin \{1, g\}$.

Integrity, but no privacy with abort. Under the DDH assumption, we will show that this scheme already prevents an attacker from providing a malicious digest d along with *two* different valid answer answers e'_0, e'_1 to the same query. This is a stronger notion of integrity than what is shown in [15].

However, there is a simple selective failure attack, in which a malicious server can commit to an (invalid) database with $x_1 = 2$ and $x_2 = 1$. An honest client would abort on a query for $i = 1$, as $g^2 \notin \{1, g\}$, but succeed for a query with $i = 2$. Learning whether the operation aborts or not hence tells the adversary whether the query was for $i = 1$ or $i = 2$.

Digest validation. In an attempt to mitigate the above attack, we introduce an initial *digest validation* phase, which consists of a single round-trip of interaction between the client and the server. If successfully completed, we expect the scheme to be private even under selective-failure attacks, a notion we henceforth refer to as "privacy with abort."

Our validation approach leverages the fact that a client can actually learn $g^{\mathbf{x} \cdot \mathbf{s}}$ (where $\cdot$ denotes inner product) for *any vector* $\mathbf{s}$ via the query $\tilde{\mathfrak{s}} = (\mathbf{h})^r \circ (g^{\mathbf{s}})^\alpha$, with no modification to the server. Further, under the DDH assumption, the server cannot distinguish such a query from a regular query, which corresponds to the case $\mathbf{s} = \mathbf{e}_i$. Concretely, the client asks λ queries $\tilde{\mathfrak{s}}_1, \ldots, \tilde{\mathfrak{s}}_\lambda$ associated with $\mathbf{s}_1, \ldots, \mathbf{s}_\lambda \leftarrow\$ \{0,1\}^N$, and accepts if *all of them* return a value in the range $\{g^0, g^1, \ldots, g^N\}$.

Intuitively, this ensures that the server answers using a database with entries $\mathbf{x}[i]$ which are small enough (say, in $\{-N, \ldots, N\}$), hence preventing the above attack if we additionally modify processing a query answer to output 1 whenever $e \in \{g^{-N}, \ldots, g^N\} \setminus \{1\}$, instead of

$e = g$ as in the original CNCWF scheme. Of course, we need to prove that any other attack is prevented, and furthermore, we want to do so while relying solely on the DDH assumption.

Answer extractability. The stepping stone to proving privacy with abort, upon passing validation, is proving a notion we call *answer extractability.* To define it, we consider any two-stage adversary $(\mathcal{D}, \mathcal{V})$, where $\mathcal{D}$ initially outputs a digest d, along with a state $\mathsf{st}_\mathcal{D}$. Then, $\mathcal{V}$, on input validation queries $\mathfrak{s}_1, \ldots, \mathfrak{s}_\lambda$, as well as $\mathsf{st}_\mathcal{D}$, returns answers $\tilde{s}_1, \ldots, \tilde{s}_\lambda$. Validation passes if these answers reconstruct to values $s_1, \ldots, s_\lambda \in \{g^0, g^1, \ldots, g^N\}$.

Answer extractability asks for an extractor $\mathcal{E}_{\mathsf{ans}}$ which, on input $\mathsf{st}_\mathcal{D}$ and d, is able to answer each honest PIR query $\tilde{\mathbf{c}} = (\mathbf{h})^r \circ (g^{\mathbf{e}_i})^\alpha$ in a non-aborting way, i.e., $\mathcal{E}_{\mathsf{ans}}(\tilde{\mathbf{c}}, \mathsf{st}_\mathcal{D}, d)$ returns with overwhelming probability $\tilde{e} \in \mathbb{G}$ such that

$$(d^{-r}\tilde{e})^{1/\alpha} \in \{g^{-N}, \ldots, g^N\} .$$

This is too strong of a requirement, so we will only require $\mathcal{E}_{\mathsf{ans}}$ to succeed whenever the state $\mathsf{st}_\mathcal{D}$ is such that $\mathcal{V}$ is able to pass the validation stage with non-negligible probability ν given that state. We also do allow the running time of $\mathcal{E}_{\mathsf{ans}}$ to depend polynomially on $1/\nu$.

When combined with standard PIR privacy and integrity, we will show that answer extractability implies privacy with abort. Intuitively, the idea is to use the extractor to simulate the abort information in the standard privacy experiment. We omit details here, and refer to the body of the paper.

The extractor construction. The core of the security proof is our construction of an extractor $\mathcal{E}_{\mathsf{ans}}$ for answer extractability, along with its analysis. Given an honest query $\tilde{\mathbf{c}} = \mathbf{h}^r \circ (g^{\mathbf{e}_i})^\alpha$, along with d and $\mathsf{st}_\mathcal{D}$, a natural strategy for $\mathcal{E}_{\mathsf{ans}}$ to answer this query is to repeatedly invoke $\mathcal{V}$ as

$$(\tilde{s}_1, \ldots, \tilde{s}_\lambda) \leftarrow \mathcal{V}(\mathsf{st}_\mathcal{D}, (\tilde{\mathbf{c}}, \tilde{\mathfrak{s}}_2, \ldots, \tilde{\mathfrak{s}}_\lambda)) .$$

where $\tilde{\mathfrak{s}}_2, \ldots, \tilde{\mathfrak{s}}_\lambda$ are "fake" validation queries The hope is that the fact that $\mathcal{V}$ passes validation with non-negligible probability ν also implies that, with overwhelming probability, $\tilde{s}_1$ is a good answer for $\tilde{\mathbf{c}}$ after $\mathrm{poly}(1/\nu)$ attempts. There are however two obvious challenges:

(1) $\mathcal{V}$ is only guaranteed to provide accurate answers for validation queries, but it is not clear why this would imply that it provides correct answers for a regular PIR query.
(2) We need to be able to check when $\tilde{s}_1$ is correct without knowing the randomness r, α.

To overcome (2), we will show that the following strategy works: in each iteration we invoke $\mathcal{V}$ *twice*, the second call as

$$(\tilde{s}_1', \ldots, \tilde{s}_\lambda) \leftarrow \mathcal{V}(\mathsf{st}_\mathcal{D}, (\tilde{\mathbf{c}}', \tilde{\mathfrak{s}}_2', \ldots, \tilde{\mathfrak{s}}_\lambda')) , \tag{1}$$

where $\tilde{\mathbf{c}}'$ is obtained by randomizing $\tilde{\mathbf{c}}$, i.e., $\tilde{\mathbf{c}}' = \mathbf{h}^{r'} \circ \tilde{\mathbf{c}}^{\alpha'}$, and $\tilde{\mathfrak{s}}_2', \ldots, \tilde{\mathfrak{s}}_\lambda'$ are fresh validation queries. Then $\mathcal{E}_{\mathsf{ans}}$ outputs $\tilde{s}_1$ whenever for two such calls we have $(d^{-r'}\tilde{s}_1')^{1/\alpha'} = \tilde{s}_1$. If this never happens, it outputs $\perp$.

To achieve sufficient independence for this argument to go through, the extractor needs to in fact also re-randomize $\tilde{\mathbf{c}}$ itself for each iteration, and then remove this randomization from answers $\tilde{s}_1$ and $\tilde{s}_1'$. This has the added benefit of simplifying notation, as a statistical argument implies the fact that $\mathcal{E}_{\mathsf{ans}}(d, \mathsf{st}_\mathcal{D}, \tilde{\mathbf{c}})$ returning a value $\tilde{e}$ such that $(d^{-r}\tilde{e})^{1/\alpha} \in \{g^{-N}, \ldots, g^N\}$ is *equivalent* to

$$\mathcal{E}_{\mathsf{ans}}(d, \mathsf{st}_\mathcal{D}, g^{\mathbf{e}_i}) \in \{g^{-N}, \ldots, g^N\} ,$$

where the input $g^{\mathbf{e}_i}$ is not randomized.

Correctness. To prove that $\mathcal{E}_{\mathsf{ans}}$ works as desired, hence overcoming (1), a crucial property we establish is that it behaves *homomorphically*—with overwhelming probability, for any $\mathfrak{p}_1, \ldots, \mathfrak{p}_m \in \mathbb{G}^N$ and $t_1, \ldots, t_m \in \mathbb{Z}_q$, if we let $\mathfrak{p}_{m+1} = \prod_{i=1}^m \mathfrak{p}_i^{t_i}$ and compute $p_i \leftarrow \mathcal{E}_{\mathsf{ans}}(d, \mathsf{st}_{\mathcal{D}}, \mathfrak{p}_i)$ for $i \in [m+1]$, then

$$p_1, \ldots, p_{m+1} \neq \perp \Longrightarrow p_{m+1} = \prod_{i=1}^{m} p_i^{t_i} \ . \tag{2}$$

Then, the proof considers an extended experiment where we initially run $\mathcal{D}$ to sample $(d, \mathsf{st}_{\mathcal{D}})$, and subsequently:

- Compute $e_i \leftarrow \mathcal{E}_{\mathsf{ans}}(d, \mathsf{st}_{\mathcal{D}}, g^{\mathbf{e}_i})$ for all $i \in [N]$.
- Pick $\mathbf{s}_{j,k} \leftarrow\!\!\$\ \{0,1\}^N$ for $k \in [\lambda]$ and sufficiently many j's, then run $s_{j,k} \leftarrow \mathcal{E}_{\mathsf{ans}}(d, \mathsf{st}_{\mathcal{D}}, g^{\mathbf{s}_{j,k}})$ for each j and $k \in [\lambda]$.

In this experiment, we prove that, with overwhelming probability, the event that $(d, \mathsf{st}_{\mathcal{D}})$ is good implies the following two events:

(1) $e_i \neq \perp$ for all $i \in [N]$
(2) There exists j^* such that $s_{j^*,k} \in \{1, g, \ldots, g^N\}$ for all $k \in [\lambda]$

Both properties are non-obvious to prove, but they follow from the fact that $\mathcal{V}$ does well on a good $(d, \mathsf{st}_{\mathcal{D}})$. Then, Equation 2 yields

$$s_{j^*,k} = \prod_{i=1}^{N} e_i^{\mathbf{s}_{j^*,k}[i]} \in \{1, g, \ldots, g^N\} \text{ for all } k \in [\lambda] \ .$$

A simple statistical lemma shows that if $e_1, \ldots, e_N$ were *not* all in $\{g^{-N}, \ldots, g^N\}$, then due to $\mathbf{s}_{j,k}$ being uniformly sampled from $\{0,1\}^N$, the above was unlikely to have happened. Hence, all e_i's are in $\{g^{-N}, \ldots, g^N\}$, which implies that the extractor can answer any honest query.

2 Preliminaries

Basic notation and computational model. We let λ denote the security parameter, and use $\mathrm{poly}(\lambda)$ and $\mathsf{negl}(\lambda)$ as placeholders for a generic polynomial and negligible function in λ, respectively. All algorithms in this paper are randomized unless otherwise stated. We assume a standard model of computation, and refer to our adversaries as "ppt" to indicate they run in polynomial time. For convenience, we assume these algorithms are non-uniform (i.e., they can also use polynomial-time advice), although it is not hard to extend our treatment to the uniform setting. Throughout this paper, we use $\leftarrow$ to denote the random sampling of an output of a randomized algorithm, $\leftarrow\!\!\$$ to denote the uniform sampling of an element from a set, and $:=$ to denote assignment of a value to a variable. We say that two families of distributions $X = \{X_\lambda\}_{\lambda \in N}$ and $Y = \{Y_\lambda\}_{\lambda \in \mathbb{N}}$ are computationally indistinguishable, denoted $X \approx_{\mathsf{c}} Y$, if the size of X_λ, Y_λ is polynomial in λ, and, for all ppt $\mathcal{A}$, we have $\Pr[\mathcal{A}(X_\lambda) = 1] - \Pr[\mathcal{A}(Y_\lambda) = 1] = \mathsf{negl}(\lambda)$.

Group-theoretic preliminaries. We will work with cyclic groups throughout this paper. We will typically use a *group parameter generator* GG to sample parameters. Such an algorithm, on input 1^λ, outputs $(\mathbb{G}, g, q)$ consisting of the description of a group $\mathbb{G}$ with generator g and order $2^\lambda \leq q < 2^{\lambda+1}$. The group also allows for computing group operations in time $\mathrm{poly}(\lambda)$, and in particular its elements are also represented as polynomially-sized strings.

Decisional Diffie-Hellman. Our results rely on the Decisional Diffie-Hellman (DDH) assumption, which we repeat below for completeness.

Definition 1 (DDH). *The Decisional Diffie-Hellman assumption (DDH) holds for a group generator* GG *if, for* $\mathsf{pp} = (\mathbb{G}, q, g) \leftarrow \mathsf{GG}(1^\lambda)$,

$$\left\{(\mathsf{pp}, g^a, g^b, g^{ab}) \;\middle|\; a, b \leftarrow\!\!\$\, \mathbb{Z}_q\right\} \approx_c \left\{(\mathsf{pp}, g^a, g^b, g^c) \;\middle|\; a, b, c \leftarrow\!\!\$\, \mathbb{Z}_q\right\} .$$

We will also repeatedly need the following standard lemma, which we prove for completeness in Appendix A.

Lemma 1. *If the DDH assumption holds, then for all polynomials* $N(\lambda) = \mathrm{poly}(\lambda)$, *and* $\mathsf{pp}' = (\mathbb{G}, q, g) \leftarrow \mathsf{GG}(1^\lambda)$, *we have*

$$\left\{(\mathsf{pp}', \mathbf{h}^r) \;\middle|\; \mathbf{h} \leftarrow\!\!\$\, \mathbb{G}^N,\ r \leftarrow\!\!\$\, \mathbb{Z}_q\right\} \approx_c \left\{(\mathsf{pp}', \overline{\mathbf{h}}) \;\middle|\; \overline{\mathbf{h}} \leftarrow\!\!\$\, \mathbb{G}^N\right\} .$$

3 Authenticated PIR: Definitions and Basic Properties

3.1 Basic Definitions

PIR Syntax. In this section, we define our notion of authenticated PIR. In contrast to prior work, we allow for a short validation phase. Here, we do not impose any restrictions on how this phase is run, although our concrete scheme (Section 4) will guarantee that server is unable to tell validation queries from regular ones, which is a fundamental feature of such a scheme.

Definition 2 (Authenticated PIR). *An authenticated PIR scheme (APIR)* APIR *consists of the following eight algorithms:*

- *The* setup *algorithm, on input the security parameter and the database length, both in unary, outputs the public parameters* $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^N)$.
- *The* digest *algorithm, on input* pp *and database* $\mathbf{x} \in \{0,1\}^N$, *outputs a digest* $d \leftarrow \mathsf{Digest}(\mathsf{pp}, \mathbf{x})$.
- *The client's* validation query *algorithm only takes as input* pp, *and outputs* $(\mathsf{st}, v) \leftarrow \mathsf{VQuery}(\mathsf{pp})$, *where* st *is a state and* v *is the validation query.*
- *The server's* validation response *algorithm takes as input* pp, *the database* $\mathbf{x}$, *and a validation query* v, *and returns a response* $a_\mathsf{v} \leftarrow \mathsf{VAnswer}(\mathsf{pp}, \mathbf{x}, v)$.
- *The client's* validation check *algorithm takes as inputs* pp, *the state* st, *digest* d, *and a response* a_v, *and returns a decision bit* $\mathsf{VCheck}(\mathsf{pp}, \mathsf{st}, d, a_\mathsf{v}) \in \{0,1\}$.
- *The client's* query *algorithm takes as input* pp, *along with a query index* $i \in [N]$, *and outputs a pair* $(\mathsf{st}, q) \leftarrow \mathsf{Query}(\mathsf{pp}, i)$ *consisting of a state* st *and a query* q.
- *The server's* query response *algorithm takes as input* pp, *the database* $\mathbf{x}$, *and the query* q, *and returns an answer* $a \leftarrow \mathsf{Answer}(\mathsf{pp}, \mathbf{x}, q)$.
- *The client's* reconstruction *algorithm takes as inputs* pp, *the state* st, *digest* d, *and an answer* a, *and returns a value* $\mathsf{Rec}(\mathsf{pp}, \mathsf{st}, d, a) \in \{0, 1, \bot\}$.

For correctness, we require that for all $\lambda, N \in \mathbb{N}$ *and* $\mathsf{pp} \in \mathsf{Supp}(\mathsf{Setup}(1^\lambda, 1^N))$, *databases* $\mathbf{x} \in \{0,1\}^N$, *and indices* $i \in [N]$,

$$\Pr\left[\mathsf{VCheck}(\mathsf{pp}, \mathsf{st}, d, a_\mathsf{v}) = 1 \;\middle|\; \begin{array}{r} d \leftarrow \mathsf{Digest}(\mathsf{pp}, \mathbf{x}) \\ \mathsf{st}, v \leftarrow \mathsf{VQuery}(\mathsf{pp}) \\ a_\mathsf{v} \leftarrow \mathsf{VAnswer}(\mathsf{pp}, \mathbf{x}, v) \end{array}\right] = 1 ,$$

$$\Pr\left[\mathsf{Rec}(\mathsf{pp}, \mathsf{st}, d, a) = \mathbf{x}[i] \;\middle|\; \begin{array}{r} d \leftarrow \mathsf{Digest}(\mathsf{pp}, \mathbf{x}) \\ \mathsf{st}, q \leftarrow \mathsf{Query}(\mathsf{pp}, i) \\ a \leftarrow \mathsf{Answer}(\mathsf{pp}, \mathbf{x}, q) \end{array}\right] = 1 .$$

Process $\mathrm{Conf}^{\mathcal{D}}_{\mathsf{APIR}}(\lambda, N)$

$\mathsf{pp} \leftarrow \mathsf{Setup}(1^{\lambda}, 1^{N})$

$\mathsf{st}_{\mathcal{D}}, d \leftarrow \mathcal{D}(\mathsf{pp})$

return $c := (\mathsf{pp}, d, \mathsf{st}_{\mathcal{D}})$

Game $\mathrm{INTEGRITY}^{\mathcal{D},\mathcal{A}}_{\mathsf{APIR}}(\lambda, N, i)$

$c = (\mathsf{pp}, d, \mathsf{st}_{\mathcal{D}}) \leftarrow \mathrm{Conf}^{\mathcal{D}}(\lambda, N)$

$\mathsf{st}, q \leftarrow \mathsf{Query}(\mathsf{pp}, i)$

$a, a' \leftarrow \mathcal{A}(c, q)$

return $\mathbb{I}\begin{bmatrix} a \neq a' \wedge \mathsf{Rec}(\mathsf{st}, d, a) \neq \bot \\ \wedge\, \mathsf{Rec}(\mathsf{st}, d, a') \neq \bot \end{bmatrix}$

Fig. 1. Definition of the configuration generation process (left) and of the integrity definition game (right).

Remark 1. We can assume without loss of generality that VCheck and Rec are both deterministic, as their randomness can be chosen ahead of time by VQuery and Query, and included in the state. We will also often omit pp from the algorithms when they are understood from the context.

Digest generators. We introduce a number of security properties that are meant to hold against *fully malicious* adversaries. These are in particular allowed to choose the digest themselves. All of our adversaries will proceed in stages, and we denote by $\mathcal{D}$ the initial stage of the adversary (common to all of our security games) that generates the digest d and outputs a state $\mathsf{st}_{\mathcal{D}}$ that may be used by later stages of the adversary. We refer to such adversaries as *digest generators*. It will be convenient to introduce the following formalism that will be reused across definitions to capture the initial stage of the game.

Definition 3 (Configuration). *For an APIR scheme* APIR*, and adversarial digest generator* $\mathcal{D}$*, we define a configuration* $c = (\mathsf{pp}, d, \mathsf{st}_{\mathcal{D}})$ *as a triple consisting of* pp*,* d*, and* $\mathsf{st}_{\mathcal{D}}$*, where* $\mathsf{pp} \in \mathsf{Supp}(\mathsf{Setup}(1^{\lambda}, 1^{N}))$ *and* $(\mathsf{st}_{\mathcal{D}}, d) \in \mathsf{Supp}(\mathcal{D}(\mathsf{pp}))$*. We define* $\mathrm{Conf}^{\mathcal{D}}_{\mathsf{APIR}}(\lambda, N)$ *as the random process generating a configuration* c *as on the left-hand side of Figure 1.*

Integrity. We target a strong definition of integrity which prevents the adversary from coming up with two different non-aborting answers for the same query, even if they reconstruct to the same value.

Definition 4 (Integrity). *The PIR scheme* APIR *fulfills integrity if for all ppt adversaries* $\mathcal{A}^* = (\mathcal{D}, \mathcal{A})$*, database sizes* $N = N(\lambda) \leq \mathrm{poly}(\lambda)$*, and indices* $i = i(\lambda) \in [N(\lambda)]$*,*

$$\Pr\left[\mathrm{INTEGRITY}^{\mathcal{A}^*}_{\mathsf{APIR}}(\lambda, N, i) = 1\right] \leq \mathsf{negl}(\lambda)\ ,$$

where $\mathrm{INTEGRITY}^{\mathcal{A}^*}_{\mathsf{APIR}}(\lambda, N, i)$ *is defined on the right-hand side of Figure 1.*

Privacy. We consider two notions of privacy. The former is the standard definition of PIR privacy in a setting where the adversary can generate malicious digests, whereas the latter allows the attacker to additionally learn whether a response to a query aborts. The two notions are defined by the games described in Figure 2.

Definition 5 (Standard Privacy). *We say the PIR scheme* APIR *fulfills privacy if for all ppt adversaries* $\mathcal{A}^* = (\mathcal{D}, \mathcal{A})$*, database sizes* $N = N(\lambda) \leq \mathrm{poly}(\lambda)$*, and indices* $i_0 = i_0(\lambda), i_1 = i_1(\lambda) \in [N(\lambda)]$*,*

$$|\Pr[\mathrm{PRIV}^{\mathcal{A}^*}_{\mathsf{APIR}}(\lambda, N, i_0) = 1] - \Pr[\mathrm{PRIV}^{\mathcal{A}^*}_{\mathsf{APIR}}(\lambda, N, i_1) = 1]| \leq \mathsf{negl}(\lambda)\ ,$$

where $\mathrm{PRIV}^{\mathcal{A}^*}(\lambda, N, i)$ *is defined on the left of Figure 2.*

Game $\mathrm{PRIV}_{\mathsf{APIR}}^{\mathcal{D},\mathcal{A}}(\lambda, N, i)$

$c = (\mathsf{pp}, d, \mathsf{st}_{\mathcal{D}}) \leftarrow \mathrm{Conf}^{\mathcal{D}}(\lambda, N)$
$\mathsf{st}, q \leftarrow \mathsf{Query}(\mathsf{pp}, i)$
return $\mathcal{A}(c, q)$

Game $\mathrm{PRIV/A}_{\mathsf{APIR}}^{\mathcal{D},\mathcal{A}_1,\mathcal{A}_2,\mathcal{A}_3}(\lambda, N, i)$

$c = (\mathsf{pp}, d, \mathsf{st}_{\mathcal{D}}) \leftarrow \mathrm{Conf}^{\mathcal{D}}(\lambda, N)$
$\mathsf{st}_{\mathsf{v}}, v \leftarrow \mathsf{VQuery}(\mathsf{pp})$
$\mathsf{st}_{\mathcal{A}}, a_{\mathsf{v}} \leftarrow \mathcal{A}_1(c, v)$
if $\mathsf{VCheck}(\mathsf{st}_{\mathsf{v}}, d, a_{\mathsf{v}}) = 0$, **return** 0
$\mathsf{st}_{\mathsf{q}}, q \leftarrow \mathsf{Query}(\mathsf{pp}, i)$
$\mathsf{st}'_{\mathcal{A}}, a_{\mathsf{q}} \leftarrow \mathcal{A}_2(\mathsf{st}_{\mathcal{A}}, q)$
$\mathsf{abort} := \mathbb{I}\left[\mathsf{Rec}(\mathsf{st}_{\mathsf{q}}, d, a_{\mathsf{q}}) = \bot\right]$
return $\mathcal{A}_3(\mathsf{st}'_{\mathcal{A}}, \mathsf{abort})$

Fig. 2. Authenticated PIR Privacy Notions. The two above experiments define our standard privacy (left) and privacy with abort (right) security notions.

Definition 6 (Privacy with abort). *The PIR scheme fulfills privacy with abort if for all ppt adversaries $\mathcal{A}^* = (\mathcal{D}, \mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$, database sizes $N = N(\lambda) \leq \mathrm{poly}(\lambda)$, and indices $i_0 = i_0(\lambda), i_1 = i_1(\lambda) \in [N(\lambda)]$,*

$$|\Pr[\mathrm{PRIV/A}_{\mathsf{APIR}}^{\mathcal{A}^*}(\lambda, N, i_0) = 1] - \Pr[\mathrm{PRIV/A}_{\mathsf{APIR}}^{\mathcal{A}^*}(\lambda, N, i_1) = 1]| \leq \mathsf{negl}(\lambda) ,$$

where $\mathrm{PRIV}^{\mathcal{A}^*}(\lambda, N, i)$ is defined on the right of Figure 2.

One can also consider an alternative definition where validation and the PIR query happen concurrently, and the client learns if either of them aborts. It is not hard to show that this notion is implied by that in Definition 6. (We discuss this in Appendix B.)

Uniform definitions. Our definitions quantify over functions of the security parameter, e.g., $N(\lambda)$, $i_0(\lambda)$, $i_1(\lambda)$. Without further restrictions, this leads to non-uniform proofs of security. One could alternatively give uniform counterparts of all definitions by allowing adversarial choice of these values. Our results would easily extend, although with significant notational clutter we seek to avoid here.

3.2 Answer Extractability

Our approach to proving that a PIR scheme fulfills privacy with abort relies on an intermediate notion which we refer to as *answer extractability.* Informally, this notion means that ability to pass the validation phase implies ability to answer arbitrary PIR queries. The correct definition of this notion is rather tricky, along with showing that answer extractability, when combined with (regular) privacy and integrity, implies privacy with abort. Our definition relies on measuring the adversary's probability of passing validation, conditioned on a particular configuration c having been produced, which we now define.

Definition 7 (Validation success probability). *Let APIR be an APIR scheme, $(\mathcal{D}, \mathcal{V})$ be an adversary, where $\mathcal{D}$ generates a digest and $\mathcal{V}$ attempts to pass validation queries. Then, for any fixed $\lambda, N \in \mathbb{N}$ and configuration $c = (\mathsf{pp}, d, \mathsf{st}_{\mathcal{D}}) \in \mathsf{Supp}(\mathrm{Conf}_{\mathsf{APIR}}^{\mathcal{D}}(\lambda, N))$, the adversary's validation success probability is defined as*

$$\nu_{\mathcal{D},\mathcal{V}}^{\mathsf{APIR}}(c) := \Pr\left[\mathsf{VCheck}(\mathsf{st}, d, \mathcal{V}(c, v)) = 1 \mid \mathsf{st}, v \leftarrow \mathsf{VQuery}(\mathsf{pp})\right] .$$

For a pair $(\mathcal{D}, \mathcal{V})$, a corresponding *answer extractor* $\mathcal{E}_{\mathsf{ans}}$ is a ppt algorithm that takes a running time parameter 1^r, the configuration c output by $\mathrm{Conf}_{\mathsf{APIR}}^{\mathcal{D}}(\lambda, N)$, and an arbitrary query q. Its task is to give a non-aborting answer for q. The extractor construction will typically depend on $\mathcal{V}$ (this is the case for our construction in Section 4.4), and it is natural that its response time is inversely proportional to $\nu^{\mathsf{APIR}}_{\mathcal{D},\mathcal{V}}(c)$, i.e., when $\mathcal{V}$ has a low validation success probability for some configuration c, then this will also make it "harder" for $\mathcal{E}_{\mathsf{ans}}$ to extract the answer for any given query. For this reason, the actual definition of answer extractability given next sets the running time parameter 1^r so that the running time of $\mathcal{E}_{\mathsf{ans}}$ may grow with the inverse of $\nu^{\mathsf{APIR}}_{\mathcal{D},\mathcal{V}}(c)$.

Definition 8 (Answer extractability). *The validation step of an APIR scheme* APIR *fulfills answer extractability if for all ppt adversaries $(\mathcal{D},\mathcal{V})$, there exists an answer extractor $\mathcal{E}_{\mathsf{ans}}$ s.t. for all database sizes $N(\lambda) \le \mathrm{poly}(\lambda)$, indices $i(\lambda) \in [N]$, and polynomials $r(\lambda)$:*

$$\Pr\left[\begin{array}{l} \nu^{\mathsf{APIR}}_{\mathcal{D},\mathcal{V}}(c) \ge \frac{1}{r(\lambda)} \\ \Rightarrow \mathsf{Rec}(\mathsf{st}_q, d, \mathcal{E}_{\mathsf{ans}}(1^{r(\lambda)}, c, q)) \neq \bot \end{array} \middle| \begin{array}{r} c \leftarrow \mathrm{Conf}^{\mathcal{D}}_{\mathsf{APIR}}(\lambda, N) \\ \mathsf{st}_q, q \leftarrow \mathsf{Query}(\mathsf{pp}, i) \end{array}\right] \ \ge\ 1 - \mathsf{negl}(\lambda)\ . \quad (3)$$

Proving privacy with abort. Together with *integrity* and *standard privacy*, our notion of *answer extractability* implies *privacy with abort*: We can transform an adversary $\mathcal{A}^*$ for privacy with abort into an adversary $\mathcal{B}^*$ that breaks standard privacy. The idea is that $\mathcal{B}^*$ makes use of the extractor $\mathcal{E}_{\mathsf{ans}}$ guaranteed by extractability, to simulate the abort-bit by itself without knowing whether the answer given by $\mathcal{A}^*$ actually aborts. In particular, $\mathcal{B}^*$ simply asks $\mathcal{E}_{\mathsf{ans}}$ for the "correct" answer to the given query, and compare it with the actual answer to check whether there will be an abort.

Lemma 2. *If a PIR scheme* APIR *fulfills (1) integrity, (2) standard privacy, and (3) answer extractability, then it also fulfills privacy with abort.*

Proof. We start with some notation. For some adversary $\mathcal{A}^* = (\mathcal{D}, \mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$, we often consider games for a *fixed* configuration $c \in \mathsf{Supp}(\mathrm{Conf}^{\mathcal{D}}(\lambda, N))$: we define $\mathrm{PRIV/A}^{\mathcal{A}^*}(c, i)$ to be exactly the same as $\mathrm{PRIV/A}^{\mathcal{A}^*}(\lambda, N, i)$, except that c is fixed and not sampled in the first step anymore. Similarly, for adversary $\mathcal{B}^* = (\mathcal{D}, \mathcal{B})$, we denote by $\mathrm{PRIV}^{\mathcal{B}^*}(c, i)$ the privacy game in which c is already fixed. By abuse of notation, we use $\nu_{\mathcal{A}^*}(c)$ to denote the validation success probability $\nu_{\mathcal{D},\mathcal{A}'_1}(c)$, where we $\mathcal{A}'_1$ is identical to $\mathcal{A}_1$, except that it does not output $\mathsf{st}_{\mathcal{A}}$. If λ and N are clear from the context, we write $\Pr_c[\cdot]$ for a probability where c is implicitly sampled as $c \leftarrow \mathrm{Conf}^{\mathcal{D}}(\lambda, N)$.

Assume that privacy with abort does not hold, and let $\mathcal{A}^* = (\mathcal{D}, \mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$ be the adversary who wins the *privacy with abort* game for database sizes $N(\lambda)$ and indices $i_0(\lambda), i_1(\lambda) \in [N(\lambda)]$ with non-negligible probability. That is, there exists a polynomial $r(\lambda)$, s.t.

$$\left|\Pr_c[\mathrm{PRIV/A}^{\mathcal{A}^*}(c, i_0)] - \Pr_c[\mathrm{PRIV/A}^{\mathcal{A}^*}(c, i_1)]\right| \ge \frac{1}{r(\lambda)} \quad (4)$$

holds for infinitely many $\lambda \in \mathbb{N}$. Furthermore, let $\mathcal{E}_{\mathsf{ans}}$ be the answer extractor for $(\mathcal{D}, \mathcal{A}_1)$ that is guaranteed to exist by answer extractability.

We construct a ppt adversary $\mathcal{B}^* = (\mathcal{D}, \mathcal{B})$ for *standard privacy*, where $\mathcal{B}$ is defined as follows:

Adversary $\mathcal{B}(c = (\mathsf{pp}, d, \mathsf{st}_{\mathcal{D}}), q)$

$c = (\mathsf{pp}, d, \mathsf{st}_{\mathcal{D}}) \leftarrow \mathrm{Conf}^{\mathcal{D}}(\lambda, N)$

$\mathsf{st}_{\mathsf{v}}, v \leftarrow \mathsf{VQuery}()$

$\mathsf{st}_{\mathcal{A}}, a_{\mathsf{v}} \leftarrow \mathcal{A}_1(c, v)$

if $\mathsf{VCheck}(\mathsf{st}_{\mathsf{v}}, d, a_{\mathsf{v}}) = \bot$, **return** 0

$\mathsf{st}'_{\mathcal{A}}, a_{\mathsf{q}} \leftarrow \mathcal{A}_2(\mathsf{st}_{\mathcal{A}}, q)$

$\tilde{a}_{\mathsf{q}} \leftarrow \mathcal{E}_{\mathsf{ans}}(1^{4r(\lambda)}, c, q)$

$\mathsf{abort} := \mathbb{I}[a_{\mathsf{q}} \neq \tilde{a}_{\mathsf{q}}]$

return $\mathcal{A}_3(\mathsf{st}'_{\mathcal{A}}, \mathsf{abort})$

Note that $\mathrm{PRIV}^{\mathcal{B}^*}$ is exactly the same as $\mathrm{PRIV/A}^{\mathcal{A}^*}$, except that abort is computed as $\mathbb{I}\left[a_{\mathsf{q}} \neq \tilde{a}_{\mathsf{q}}\right]$ (with $\tilde{a}_{\mathsf{q}} \leftarrow \mathcal{E}_{\mathsf{ans}}(1^{4r(\lambda)}, c, q)$) instead of $\mathbb{I}\left[\mathsf{Rec}(\mathsf{st}_{\mathsf{q}}, d, a_{\mathsf{q}}) = \bot\right]$.

Towards proving that $\mathcal{B}^*$ breaks standard privacy, we distinguish between configurations c with $\nu_{\mathcal{A}^*}(c) < \frac{1}{4r(\lambda)}$, and those with $\nu_{\mathcal{A}^*}(c) \geq \frac{1}{4r(\lambda)}$.

Case (1): $\nu_{\mathcal{A}^*}(c) < \frac{1}{4r(\lambda)}$. In this case, we cannot be certain that the answer $\tilde{a}_{\mathsf{q}}$ given by $\mathcal{E}_{\mathsf{ans}}$ is correct. Therefore, we would like to bound the impact on $\mathcal{B}^*$'s advantage. Indeed, $\mathcal{B}$ returns 0 whenever the validation step with $\mathcal{A}_1$ does not succeed. Furthermore, by assumption, the validation success probability in this case is at most $\frac{1}{4r(\lambda)}$, which means that for both $b = 0, 1$:

$$\Pr_c\left[\begin{matrix}\mathrm{PRIV}^{\mathcal{B}^*}(c, i_b) = 1 \\ \wedge\ \nu_{\mathcal{A}^*}(c) < \frac{1}{4r(\lambda)}\end{matrix}\right] \leq \Pr_c\left[\begin{matrix}\mathsf{VCheck}(\mathsf{st}_{\mathsf{v}}, d, a_{\mathsf{v}}) \\ \wedge\ \nu_{\mathcal{A}^*}(c) < \frac{1}{4r(\lambda)}\end{matrix}\ \middle|\ \begin{matrix}\mathsf{st}_{\mathsf{v}}, v \leftarrow \mathsf{VQuery}() \\ \mathsf{st}_{\mathcal{A}}, a_{\mathsf{v}} \leftarrow \mathcal{A}_1(c, v)\end{matrix}\right]$$
$$= \sum_{\substack{c \in \mathsf{Supp}(\mathrm{Conf}^{\mathcal{D}}(\lambda, N)) \\ \nu_{\mathcal{A}^*}(c) < \frac{1}{4r(\lambda)}}} \Pr[c] \cdot \nu_{\mathcal{A}^*}(c) \quad \leq \frac{1}{4r(\lambda)}\ .$$

Therefore, in case $\nu_{\mathcal{A}^*}(c) < \frac{1}{4r(\lambda)}$, $\mathcal{B}^*$ has no significant (and therefore also no significant *negative*) advantage:

$$\left|\Pr_c\left[\begin{matrix}\mathrm{PRIV}^{\mathcal{B}^*}(c, i_0) = 1 \\ \wedge\ \nu_{\mathcal{A}^*}(c) < \frac{1}{4r(\lambda)}\end{matrix}\right] - \Pr_c\left[\begin{matrix}\mathrm{PRIV}^{\mathcal{B}^*}(c, i_1) = 1 \\ \wedge\ \nu_{\mathcal{A}^*}(c) < \frac{1}{4r(\lambda)}\end{matrix}\right]\right| \leq \frac{1}{4r(\lambda)}\ . \tag{5}$$

Case (2): $\nu_{\mathcal{A}^*}(c) \geq \frac{1}{4r(\lambda)}$. In this case, we are going to show that $\mathrm{PRIV}^{\mathcal{B}^*}$ and $\mathrm{PRIV/A}^{\mathcal{A}^*}$ are close, by showing that w.h.p. their respective ways of defining abort are identical. We use both answer extractability and integrity:

– By answer extractability, for both $b = 0, 1$:

$$\Pr_c\left[\begin{matrix}\nu_{\mathcal{A}^*}(c) \geq \frac{1}{4r(\lambda)} \Rightarrow \\ \mathsf{Rec}(\mathsf{st}_{\mathsf{q}}, d, \tilde{a}_{\mathsf{q}}) \neq \bot\end{matrix}\ \middle|\ \begin{matrix}\mathsf{st}_{\mathsf{q}}, q \leftarrow \mathsf{Query}(i_b) \\ \tilde{a}_{\mathsf{q}} \leftarrow \mathcal{E}_{\mathsf{ans}}(1^{4r(\lambda)}, c, q)\end{matrix}\right] \geq 1 - \mathsf{negl}(\lambda) \tag{6}$$

– By *integrity*, for both $b = 0, 1$:

$$\Pr_c\left[\begin{matrix}a_{\mathsf{q}} \neq \tilde{a}_{\mathsf{q}} \\ \wedge\ \mathsf{Rec}(\mathsf{st}_{\mathsf{q}}, d, a_{\mathsf{q}}) \neq \bot \\ \wedge\ \mathsf{Rec}(\mathsf{st}_{\mathsf{q}}, d, \tilde{a}_{\mathsf{q}}) \neq \bot\end{matrix}\ \middle|\ \begin{matrix}\mathsf{st}_{\mathsf{q}}, q \leftarrow \mathsf{Query}(i_b) \\ \mathsf{st}_{\mathsf{v}}, v \leftarrow \mathsf{VQuery}() \\ \mathsf{st}_{\mathcal{A}}, a_{\mathsf{v}} \leftarrow \mathcal{A}_1(c, v) \\ \mathsf{st}'_{\mathcal{A}}, a_{\mathsf{q}} \leftarrow \mathcal{A}_2(\mathsf{st}_{\mathcal{A}}, q) \\ \tilde{a}_{\mathsf{q}} \leftarrow \mathcal{E}_{\mathsf{ans}}(1^{4r(\lambda)}, c, q)\end{matrix}\right] \leq \mathsf{negl}(\lambda) \tag{7}$$

Note that if both of the events

$$\{\mathsf{Rec}(\mathsf{st}_{\mathsf{q}}, d, \tilde{a}_{\mathsf{q}}) \neq \bot\} \quad \text{and} \quad \left\{\begin{matrix}a_{\mathsf{q}} = \tilde{a}_{\mathsf{q}} \\ \vee\ \mathsf{Rec}(\mathsf{st}_{\mathsf{q}}, d, a_{\mathsf{q}}) = \bot \\ \vee\ \mathsf{Rec}(\mathsf{st}_{\mathsf{q}}, d, \tilde{a}_{\mathsf{q}}) = \bot\end{matrix}\right\}$$

are true, then the two ways $\mathbb{I}[a_{\mathsf{q}} \neq \tilde{a}_{\mathsf{q}}]$ and $\mathbb{I}[\mathsf{Rec}(\mathsf{st}_{\mathsf{q}}, d, a_{\mathsf{q}}) = \bot]$ of defining abort in $\mathrm{PRIV}^{\mathcal{B}^*}$ and $\mathrm{PRIV/A}^{\mathcal{A}^*}$ are identical:

– If $a_{\mathsf{q}} \neq \tilde{a}_{\mathsf{q}}$, then together with the left event $\mathsf{Rec}(\mathsf{st}_{\mathsf{q}}, d, \tilde{a}_{\mathsf{q}}) \neq \bot$, the right event proves that $\mathsf{Rec}(\mathsf{st}_{\mathsf{q}}, d, a_{\mathsf{q}}) = \bot$.
– If $a_{\mathsf{q}} = \tilde{a}_{\mathsf{q}}$, then (because Rec is deterministic), the left event proves that $\mathsf{Rec}(\mathsf{st}_{\mathsf{q}}, d, a_{\mathsf{q}}) \neq \bot$.

Thus, combining Equations 6 and 7 with a union bound yields

$$\Pr_c\left[\begin{array}{l} \nu_{\mathcal{A}^*}(c) \geq \frac{1}{4r(\lambda)} \Rightarrow \\ \mathbb{I}[a_{\mathsf{q}} \neq \tilde{a}_{\mathsf{q}}] = \mathbb{I}[\mathsf{Rec}(\mathsf{st}_{\mathsf{q}}, d, a_{\mathsf{q}}) = \bot] \end{array} \middle| \begin{array}{l} \mathsf{st}_{\mathsf{q}}, q \leftarrow \mathsf{Query}(i_b) \\ \mathsf{st}_{\mathsf{v}}, v \leftarrow \mathsf{VQuery}() \\ \mathsf{st}_{\mathcal{A}}, a_{\mathsf{v}} \leftarrow \mathcal{A}_1(c, v) \\ \mathsf{st}'_{\mathcal{A}}, a_{\mathsf{q}} \leftarrow \mathcal{A}_2(\mathsf{st}_{\mathcal{A}}, q) \\ \tilde{a}_{\mathsf{q}} \leftarrow \mathcal{E}_{\mathsf{ans}}(1^{4r(\lambda)}, c, q) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda)\ .$$

As a result, we can use the two definitions of abort interchangingly (when $\nu_{\mathcal{A}^*}(c) \geq \frac{1}{4r(\lambda)}$), with only negligible difference. As this is the only thing that makes $\text{PRIV}^{\mathcal{B}^*}$ different from $\text{PRIV/A}^{\mathcal{A}^*}$, we get for both $b = 0, 1$

$$\left|\Pr_c\left[\begin{array}{l}\text{PRIV}^{\mathcal{B}^*}(c, i_b) = 1 \\ \wedge\ \nu_{\mathcal{A}^*}(c) \geq \frac{1}{4r(\lambda)}\end{array}\right] - \Pr_c\left[\begin{array}{l}\text{PRIV/A}^{\mathcal{A}^*}(c, i_b) = 1 \\ \wedge\ \nu_{\mathcal{A}^*}(c) \geq \frac{1}{4r(\lambda)}\end{array}\right]\right| \leq \mathsf{negl}(\lambda)\ . \tag{8}$$

It remains to lower bound the advantage that $\text{PRIV/A}^{\mathcal{A}^*}$ has in case $\nu_{\mathcal{A}^*}(c) \geq \frac{1}{4r(\lambda)}$. Note that for any configuration $c \in \mathsf{Supp}(\text{Conf}^{\mathcal{D}}(\lambda, N))$ and index $i \in [N]$, we have $\Pr[\text{PRIV/A}^{\mathcal{A}^*}(c, i) = 1] \leq \nu_{\mathcal{A}^*}(c)$ (because $\text{PRIV/A}^{\mathcal{A}^*}$ can only output 1 whenever validation succeeds). Thus, we get the following bound (for infinitely many λ, by using Equation 4)

$$\begin{aligned} &\left|\Pr_c\left[\begin{array}{l}\text{PRIV/A}^{\mathcal{A}^*}(c, i_0) = 1 \\ \wedge\ \nu_{\mathcal{A}^*}(c) \geq \frac{1}{4r(\lambda)}\end{array}\right] - \Pr_c\left[\begin{array}{l}\text{PRIV/A}^{\mathcal{A}^*}(c, i_1) = 1 \\ \wedge\ \nu_{\mathcal{A}^*}(c) \geq \frac{1}{4r(\lambda)}\end{array}\right]\right| \\ \geq\ &\frac{1}{r(\lambda)} - \sum_{\substack{c \in \mathsf{Supp}(\text{Conf}^{\mathcal{D}}(\lambda, N)) \\ \nu_{\mathcal{A}^*}(c) < \frac{1}{4r(\lambda)}}} \Pr[c] \cdot \underbrace{|\Pr[\text{PRIV/A}^{\mathcal{A}^*}(c, i_0)] - \Pr[\text{PRIV/A}^{\mathcal{A}^*}(c, i_1)]|}_{\leq \nu_{\mathcal{A}^*}(c) < \frac{1}{4r(\lambda)}} \geq \frac{3}{4r(\lambda)} \end{aligned} \tag{9}$$

By combining Equations 5, 8, and 9, the advantage of $\text{PRIV}^{\mathcal{B}^*}$ is non-negligible:

$$\begin{aligned} &\left|\Pr[\text{PRIV}^{\mathcal{B}^*}(\lambda, N, i_0) = 1] - \Pr[\text{PRIV}^{\mathcal{B}^*}(\lambda, N, i_1) = 1]\right| \\ \geq\ &(9) - 2 \cdot (8) - (5) \geq \frac{1}{2r(\lambda)} - \mathsf{negl}(\lambda) \end{aligned}$$

for infinitely many λ. □

3.3 Rebalancing

We revisit the standard rebalancing trick, originally proposed in [32], in the context of Authenticated PIR. Given a PIR scheme APIR, we construct a *rebalanced* scheme $\overline{\mathsf{APIR}}$. It splits a database of size N into $\sqrt{N}$ chunks of size $\sqrt{N}$ (w.l.o.g. we may assume that N is a perfect square, as the database can always be padded with dummy values).

For database $\mathbf{x} \in \{0,1\}^N$, we denote by $\mathbf{x}_i$ the i-th chunk (which is a vector in $\{0,1\}^{\sqrt{N}}$), s.t. $\mathbf{x} = (\mathbf{x}_1^T, \ldots, \mathbf{x}_{\sqrt{N}}^T)$. Any index $i \in [N]$ can be split into two values $i_{\mathsf{row}} \in [\sqrt{N}]$ and $i_{\mathsf{col}} \in [\sqrt{N}]$, s.t. $\mathbf{x}[i] = \mathbf{x}_{i_{\mathsf{col}}}[i_{\mathsf{row}}]$ (i_{col} denotes the chunk that i is in, and i_{row} denotes the position within that chunk). The full scheme is described in Figure 3.

It is easy to see that if APIR fulfills *integrity*, then $\overline{\mathsf{APIR}}$ fulfills integrity as well, and if APIR fulfills *standard privacy*, then $\overline{\mathsf{APIR}}$ fulfills standard privacy. Furthermore, we can prove that if APIR fulfills *answer extractability*, then $\overline{\mathsf{APIR}}$ does so as well.

Lemma 3. *If* APIR *fulfills answer extractability, then the re-balanced scheme* $\overline{\mathsf{APIR}}$ *fulfills answer extractability as well.*

$\mathsf{Setup}(1^\lambda, 1^N)$	$\mathsf{Digest}(\mathbf{x})$
1. **return** $\mathsf{APIR.Setup}(1^\lambda, 1^{\sqrt{N}})$	1. $d_j \leftarrow \mathsf{APIR.Digest}(\mathbf{x}_j) \quad \forall j \in \sqrt{N}$ 2. **return** $d := (d_1, \ldots, d_{\sqrt{N}})$
Validation	**Query**
$\mathsf{VQuery}()$ 1. **return** $\mathsf{st}, v \leftarrow \mathsf{APIR.VQuery}()$	$\mathsf{Query}(i)$ 1. $\mathsf{st}, q \leftarrow \mathsf{APIR.Query}(i_{\mathsf{row}})$ 2. **return** $\overline{\mathsf{st}} := (\mathsf{st}, i_{\mathsf{col}}), q$
$\mathsf{VAnswer}(\mathbf{x}, v)$ 1. $a_j \leftarrow \mathsf{APIR.VAnswer}(\mathbf{x}_j, v) \quad \forall j \in [\sqrt{N}]$ 2. **return** $a := (a_1, \ldots, a_{\sqrt{N}})$	$\mathsf{Answer}(\mathbf{x}, q)$ 1. $a_j \leftarrow \mathsf{APIR.Answer}(\mathbf{x}_j, q) \quad \forall j \in [\sqrt{N}]$ 2. **return** $a := (a_1, \ldots, a_{\sqrt{N}})$
$\mathsf{VCheck}(\mathsf{st}, d = \{d_j\}, a = \{a_j\})$ 1. $t_j \leftarrow \mathsf{APIR.VCheck}(\mathsf{st}, d_j, a_j) \quad \forall j \in [\sqrt{N}]$ 2. **return** $\mathbb{I}\left[t_1 \wedge \cdots \wedge t_{\sqrt{N}}\right]$	$\mathsf{Rec}((\mathsf{st}, i_{\mathsf{col}}), d = \{d_j\}, a = \{a_j\})$ 1. $t_j \leftarrow \mathsf{APIR.Rec}(\mathsf{st}, d_j, a_j) \quad \forall j \in [\sqrt{N}]$ 2. **if** $\exists j \in [\sqrt{N}]$ with $t_j = \bot$, **return** $\bot$ 3. **else return** $t_{i_{\mathsf{col}}}$

Fig. 3. Rebalancing $\overline{\mathsf{APIR}}$ of an PIR scheme APIR. We assume that the public parameters pp output by Setup are available to all algorithms, but omit them for ease of readability.

Remark 2. As in [15], we can compress the digest by letting $\overline{\mathsf{APIR}}.\mathsf{Digest}$ return a short hash of $(d_1, \ldots, d_{\sqrt{N}})$ (any collision-resistant hash function chosen by $\overline{\mathsf{APIR}}.\mathsf{Setup}$ suffices). Then, $\overline{\mathsf{APIR}}.\mathsf{VAnswer}$ and $\overline{\mathsf{APIR}}.\mathsf{Answer}$ would both output the full digest $(d_1, \ldots, d_{\sqrt{N}})$ (in addition to the actual answers), and $\overline{\mathsf{APIR}}.\mathsf{VCheck}$ and $\overline{\mathsf{APIR}}.\mathsf{Rec}$ test whether it matches the digest hash before doing anything else.

Proof (of Lemma 3). Let $(\overline{\mathcal{D}}, \overline{\mathcal{V}})$ be a ppt adversary for $\overline{\mathsf{APIR}}$ as in Definition 8. We define an adversary $(\mathcal{D}, \mathcal{V})$ for the original scheme APIR as follows:

- $\mathcal{D}(\mathsf{pp})$ runs $\overline{\mathsf{st}_\mathcal{D}}, d \leftarrow \overline{\mathcal{D}}(\mathsf{pp})$, parses $d = (d_1, \ldots, d_{\sqrt{N}})$, and samples a random $i^*_{\mathsf{col}} \leftarrow\$ [\sqrt{N}]$. Then, it outputs $(\mathsf{st}_\mathcal{D}, d_{i^*_{\mathsf{col}}})$, where $\mathsf{st}_\mathcal{D} := (\overline{\mathsf{st}_\mathcal{D}}, d, i^*_{\mathsf{col}})$.
- $\mathcal{V}(c = (\mathsf{pp}, d_{i^*_{\mathsf{col}}}, (\overline{\mathsf{st}_\mathcal{D}}, d, i^*_{\mathsf{col}})), v)$ runs $(a_1, \ldots, a_{\sqrt{N}}) \leftarrow \overline{\mathcal{V}}((\mathsf{pp}, d, \overline{\mathsf{st}_\mathcal{D}}), v)$ and outputs $a_{i^*_{\mathsf{col}}}$.

For a configuration $\bar{c} = (\mathsf{pp}, d, \overline{\mathsf{st}_\mathcal{D}}) \in \mathsf{Supp}(\mathrm{Conf}^{\overline{\mathcal{D}}}_{\overline{\mathsf{APIR}}}(\lambda, N))$ of $\overline{\mathsf{APIR}}$ (with digest $d = (d_1, \ldots, d_{\sqrt{N}})$) and any $i^*_{\mathsf{col}} \in [\sqrt{N}]$, we define the configuration $c_{i^*_{\mathsf{col}}} = (\mathsf{pp}, d_{i^*_{\mathsf{col}}}, (\overline{\mathsf{st}_\mathcal{D}}, d, i^*_{\mathsf{col}})) \in \mathsf{Supp}(\mathrm{Conf}^{\mathcal{D}}_{\mathsf{APIR}}(\lambda, \sqrt{N}))$ of APIR.

By *answer extractability* for APIR, there exists an answer extractor $\mathcal{E}_{\mathsf{ans}}$, s.t. for all database sizes $N(\lambda) \leq \mathrm{poly}(\lambda)$, indices $i_{\mathsf{row}}(\lambda) \in [\sqrt{N}]$, and polynomials $r(\lambda)$:

$$\Pr\left[\begin{array}{l} \nu_{\mathcal{D},\mathcal{V}}(c_{i^*_{\mathsf{col}}}) \geq \frac{1}{r(\lambda)} \Rightarrow \\ \mathsf{APIR.Rec}(\mathsf{st}, d_{i^*_{\mathsf{col}}}, \mathcal{E}_{\mathsf{ans}}(1^{r(\lambda)}, c_{i^*_{\mathsf{col}}}, q)) \neq \bot \end{array} \middle| \begin{array}{r} i^*_{\mathsf{col}} \leftarrow\$ [\sqrt{N}] \\ \bar{c} \leftarrow \mathrm{Conf}^{\overline{\mathcal{D}}}_{\overline{\mathsf{APIR}}}(\lambda, N) \\ \mathsf{st}, q \leftarrow \mathsf{APIR.Query}(\mathsf{pp}, i_{\mathsf{row}}) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda) .$$

Note that for any i^*_{col}, we have $\nu_{\mathcal{D},\mathcal{V}}(c_{i^*_{\mathsf{col}}}) \geq \nu_{\overline{\mathcal{D}},\overline{\mathcal{V}}}(\bar{c})$, because $\overline{\mathcal{V}}$ succeeds with answering validation queries only whenever all of the $\sqrt{N}$ instances of APIR succeed. Thus, in the probability above, we may replace $\nu_{\mathcal{D},\mathcal{V}}(c_{i^*_{\mathsf{col}}})$ by $\nu_{\overline{\mathcal{D}},\overline{\mathcal{V}}}(\bar{c})$ without decreasing the probability.

$\underline{\mathsf{Setup}(1^\lambda, 1^N)}$	$\underline{\mathsf{Digest}(\mathbf{x})}$
1. $(\mathbb{G}, q, g) = \mathsf{pp}' \leftarrow \mathsf{GG}(1^N)$ 2. $\mathbf{h} \leftarrow\$\ \mathbb{G}^N$ 3. **return** $\mathsf{pp} := (\mathbb{G}, q, g, \mathbf{h})$	1. **return** $d := \prod_{i \in [N]} \mathbf{h}[i]^{\mathbf{x}[i]}$
Validation	**Query**
$\underline{\mathsf{VQuery}()}$ 1. $\mathbf{s}_j \leftarrow\$\ \{0,1\}^N \quad \forall j \in [\lambda]$ 2. $\mathsf{st}_j, \tilde{\mathfrak{s}}_j \leftarrow \mathsf{randomize}(g^{\mathbf{s}_j}) \quad \forall j \in [\lambda]$ 3. **return** $\mathsf{st} := (\mathsf{st}_j)_{j\in[\lambda]}, v := (\tilde{\mathfrak{s}}_j)_{j \in [\lambda]}$	$\underline{\mathsf{Query}(i)}$ 1. $\mathsf{st}, \tilde{\mathfrak{e}} \leftarrow \mathsf{randomize}(g^{\mathbf{e}_i})$ 2. **return** $\mathsf{st}, q := \tilde{\mathfrak{e}}$
$\underline{\mathsf{VAnswer}(\mathbf{x}, v = (\tilde{\mathfrak{s}}_j)_{j\in[\lambda]})}$ 1. $\tilde{s}_j := \prod_{i\in[N]} (\tilde{\mathfrak{s}}_j[i])^{\mathbf{x}[i]} \quad \forall j \in [\lambda]$ 2. **return** $a := (\tilde{s}_j)_{j\in[\lambda]}$	$\underline{\mathsf{Answer}(\mathbf{x}, q = \tilde{\mathfrak{e}})}$ 1. **return** $a := \tilde{e} = \prod_{i\in[N]} (\tilde{\mathfrak{e}}[i])^{\mathbf{x}[i]}$
$\underline{\mathsf{VCheck}(\mathsf{st} = (\mathsf{st}_j)_{j\in[\lambda]}, d, a = (\tilde{s}_j)_{j\in[\lambda]})}$ 1. $s_j \leftarrow \mathsf{recover}(\mathsf{st}_j, d, \tilde{s}_j) \quad \forall j \in [\lambda]$ 2. **return** $\mathbb{I}\left[s_1, \ldots, s_\lambda \in \{g^0, \ldots, g^N\}\right]$	$\underline{\mathsf{Rec}(\mathsf{st}, d, a = \tilde{e})}$ 1. $e \leftarrow \mathsf{recover}(\mathsf{st}, d, \tilde{e})$ 2. **if** $e \in \{g^{-N}, \ldots, g^N\} \setminus \{g^0\}$, **return** 1 3. **else if** $e = g^0$, **return** 0 4. **else return** $\perp$

Fig. 4. Our PIR scheme. We assume that the public parameters pp output by Setup are available to all algorithms, but omit them for ease of readability.

Furthermore, by taking a union bound over all fixed i^*_{col}, we get

$$\Pr\left[\begin{array}{l} \nu_{\overline{\mathcal{D}},\overline{\mathcal{V}}}(\overline{c}) \geq \frac{1}{r(\lambda)} \Rightarrow \\ \forall i^*_{\mathsf{col}}:\ \mathsf{APIR.Rec}(\mathsf{st}, d_{i^*_{\mathsf{col}}}, \mathcal{E}_{\mathsf{ans}}(1^{r(\lambda)}, c_{i^*_{\mathsf{col}}}, q)) \neq \perp \end{array} \middle| \begin{array}{c} \overline{c} \leftarrow \mathrm{Conf}^{\overline{\mathcal{D}}}_{\mathsf{APIR}}(\lambda, N) \\ \mathsf{st}, q \leftarrow \mathsf{APIR.Query}(\mathsf{pp}, i_{\mathsf{row}}) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda)\,. \tag{10}$$

Now we construct the answer extractor $\overline{\mathcal{E}_{\mathsf{ans}}}$ for $(\overline{\mathcal{D}}, \overline{\mathcal{V}})$ as follows:

- $\overline{\mathcal{E}_{\mathsf{ans}}}(1^r, \overline{c}, q)$ runs $a_{i^*_{\mathsf{col}}} \leftarrow \mathcal{E}_{\mathsf{ans}}(1^r, c_{i^*_{\mathsf{col}}}, q)$ for each $i^*_{\mathsf{col}} \in [\sqrt{N}]$, and outputs $a_1, \ldots, a_{\sqrt{N}}$.

Fix the database size $N(\lambda) \leq \mathrm{poly}(\lambda)$, indices $i(\lambda) \in [\sqrt{N}]$, and polynomials $r(\lambda)$. Note that for any $(\mathsf{st}, i_{\mathsf{col}}), q \leftarrow \overline{\mathsf{APIR}}.\mathsf{Query}(\mathsf{pp}, i)$, we have

$$\begin{aligned} &\overline{\mathsf{APIR}}.\mathsf{Rec}((\mathsf{st}, i_{\mathsf{col}}), d, \overline{\mathcal{E}_{\mathsf{ans}}}(1^{r(\lambda)}, \overline{c}, q)) \neq \perp \\ \text{iff}\quad &\forall i^*_{\mathsf{col}} \in [\sqrt{N}]:\ \mathsf{APIR.Rec}(\mathsf{st}, d_{i^*_{\mathsf{col}}}, \mathcal{E}_{\mathsf{ans}}(1^{r(\lambda)}, c_{i^*_{\mathsf{col}}}, q)) \neq \perp \end{aligned}$$

Therefore, the desired property

$$\Pr\left[\begin{array}{l} \nu_{\overline{\mathcal{D}},\overline{\mathcal{V}}}(\overline{c}) \geq \frac{1}{r(\lambda)} \Rightarrow \\ \overline{\mathsf{APIR}}.\mathsf{Rec}((\mathsf{st}, i_{\mathsf{col}}), d, \overline{\mathcal{E}_{\mathsf{ans}}}(1^{r(\lambda)}, \overline{c}, q)) \neq \perp \end{array} \middle| \begin{array}{c} \overline{c} \leftarrow \mathrm{Conf}^{\overline{\mathcal{D}}}_{\mathsf{APIR}}(\lambda, N) \\ (\mathsf{st}, i_{\mathsf{col}}), q \leftarrow \overline{\mathsf{APIR}}.\mathsf{Query}(\mathsf{pp}, i) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda)\,.$$

is implied by the Equation 10. □

4 The Authenticated PIR Scheme and its Security

4.1 Description and Security

This section describes our APIR scheme, and states the security theorems we then prove below. A description of the scheme is given in Figure 4.

Main ideas behind the scheme. We discuss first the unbalanced version of our scheme, which we refer to as APIR. We obtain a re-balanced version $\overline{\mathsf{APIR}}$ which achieve $O(\sqrt{N})$ complexity following the transformation of Section 3.3. The scheme is based on the DDH scheme from Colombo et al. [15]. The server is in fact *identical*: It initially publishes a Pedersen hash $d = \prod_{i\in[N]} \mathbf{h}[i]^{\mathbf{x}[i]}$ of the database as the digest, where $\mathbf{h} \in \mathbb{G}^N$ is a vector of independent random generators for $\mathbb{G}$. The server then answers any query vector $\mathfrak{p} \in \mathbb{G}^N$ by returning the product $\prod_{i\in[N]} \mathfrak{p}[i]^{\mathbf{x}[i]}$.

Item retrieval. To retrieve an item $i \in [N]$ privately, the client sends a randomized version of the vector $\mathfrak{e} = g^{\mathbf{e}_i}$, where $\mathbf{e}_i$ is the i-th unit vector. The easiest way to hide the query, under the DDH assumption, is to multiply $\mathfrak{e}$ component-wise with a blinding vector $\mathbf{h}^r$. Then, the client would receive the response $\tilde{e} = d^r \cdot g^{\mathbf{x}[i]}$, and $e = g^{\mathbf{x}[i]}$ is recovered by dividing off d^r. To achieve integrity, however, we will need to additionally randomize $\mathfrak{e}$ by exponentiating it with a non-zero $\alpha \leftarrow\$ \mathbb{Z}_q^*$, where q is the group order. Then, if the server answers honestly with a value $\tilde{e}$, we get the final answer as $e = (d^{-r}\tilde{e})^{1/\alpha}$. However, in contrast to [15], to accommodate the validation process we describe next, we will need to allow e to take values in $\{g^{-N}, \ldots, g^N\}$. We think of any value different than $1 = g^0$ as corresponding to the output 1, whereas $1 = g^0$ is mapped to 0.

In summary, we define a general query randomization mechanism (and the associated recovery operation), used for both validation and normal queries in Figure 4, as

$\mathsf{randomize}(\mathfrak{p})$

1 : $r \leftarrow\$ \mathbb{Z}_q$
2 : $\alpha \leftarrow\$ \mathbb{Z}_q^*$
3 : **return** $\mathsf{st} := (r, \alpha), \tilde{\mathfrak{p}} := \mathbf{h}^r \circ \mathfrak{p}^\alpha$

$\mathsf{recover}(\mathsf{st} = (r, \alpha), d, \tilde{p})$

1 : **return** $p := (d^{-r} \cdot \tilde{p})^{1/\alpha}$

Generalized queries. It will be convenient to think of queries in a more general sense, where the the client can ask for any *selection vector* $\mathbf{p} \in \mathbb{Z}_q^N$, turning it into a corresponding vector of group elements $\mathfrak{p} = g^{\mathbf{p}}$, which is then randomized as $\tilde{\mathfrak{p}}$. We usually denote the answer as $\tilde{p}$, and the de-randomized answer as p. This defines some notational conventions we follow throughout our proofs.

Digest validation. Our digest validation procedure consists of λ parallel generalized queries where the selection vectors are independent random binary vectors. The client accepts if all results of these queries are in the set $\{g^0, g^1, \ldots, g^N\}$. Note that if the server produces the digest and answers honestly, the check passes, but this check also accommodates for answers where some of the database components are larger. (Say one of them is N, all others are 0.) This is the main reason why we do relax the allowable range of values for e for PIR queries.

Security of APIR. Below, we prove the following three theorems in Sections 4.2, 4.3, and 4.4, respectively.

Theorem 1. *Assuming that the DDH assumption holds,* APIR *fulfills integrity.*

Theorem 2. *Assuming that the DDH assumption holds,* APIR *fulfills standard privacy.*

Theorem 3. *Assuming that the DDH assumption holds,* APIR *fulfills answer extractability.*

Combining these theorems with Lemmas 2 and 3 immediately yields the following corollary.

Corollary 1. *Assuming that the DDH assumption holds, both* APIR *and its rebalancing* $\overline{\mathsf{APIR}}$ *fulfill integrity, standard privacy, and privacy with abort.*

Efficiency. After rebalancing our scheme as described in Section 3.3, a single query and its answer both have size proportional to that of $\sqrt{N}$ group elements. A validation and a validation answer both have size proportional to that of $\sqrt{N} \cdot \lambda$ group elements.

Note that if we apply the transformation naïvely, $\overline{\mathsf{APIR}}.\mathsf{Rec}$ would run $\mathsf{APIR}.\mathsf{Rec}$ for $\sqrt{N}$ times, each of them re-computing the same $2\sqrt{N}+1$ powers of g during the call to $\mathsf{recover}$. This would result in computational complexity $O(N)$. Obviously, a client should compute these powers only *once*, store them in a suitable data structure (e.g., a hash table), and perform a single constant-time look up. While we did not perform own benchmark, it should be clear that the same performance profile as in [15] would emerge here.

Remark 3. In order to remove clutter, when clear from context (e.g., configuration $c = (\mathsf{pp}, d, \mathsf{st}_{\mathcal{D}})$ is fixed) we will typically omit the parameter d passed to $\mathsf{recover}(\mathsf{st}, d, \tilde{p})$ when proving security of APIR in the following sections.

In some places, an answer $\tilde{p} \in \mathbb{G} \cup \{\perp\}$ may be equal to $\perp$. Any operations involving $\tilde{p} = \perp$ will result in $\perp$, e.g. $\mathsf{recover}(\mathsf{st}, d, \perp) = \perp$.

4.2 Proof of Theorem 1 (Integrity)

Lemma 4 establishes a slightly stronger version of integrity that will be useful within proofs of other theorems below. Specifically:

1. We let $\mathcal{A}$ choose the query vector $\mathfrak{p} \in \mathbb{G}^N$ that will be randomized. For integrity, it suffices to consider $\mathfrak{p} = g^{\mathbf{e}_i}$.
2. We let $\mathcal{A}$ choose a *small* set S, along with its answers $\tilde{p}$, $\tilde{p}'$. The adversary wins whenever their two answers $\tilde{p}$ and $\tilde{p}'$ are distinct, and *the ratio between the two recovered answers* is in S.

We show that the probability of winning is small, and we refer to this as "Evasive ratio" lemma. Below, we show integrity easily follows.

Lemma 4 (Evasive Ratio). *Assuming that the DDH assumption holds, then for any ppt adversaries $\mathcal{D}, \mathcal{A}_1, \mathcal{A}_2$ (where $\mathcal{A}_1(c)$ outputs state $\mathsf{st}_{\mathcal{A}}$ and a query vector $\mathfrak{p} \in \mathbb{G}^N$, and $\mathcal{A}_2(\mathsf{st}_{\mathcal{A}}, \tilde{\mathfrak{p}})$ outputs two answers $\tilde{p}, \tilde{p}' \in \mathbb{G} \cup \{\perp\}$ and a set $S \subseteq \mathbb{G}$ of size $|S| \leq \mathrm{poly}(\lambda, N)$) and database sizes $N(\lambda) \leq \mathrm{poly}(\lambda)$:*

$$\Pr\left[\begin{array}{l} \tilde{p} \neq \tilde{p}' \wedge \\ \dfrac{\mathsf{recover}(\mathsf{st}, \tilde{p}')}{\mathsf{recover}(\mathsf{st}, \tilde{p})} \in S \end{array} \;\middle|\; \begin{array}{l} c = (\mathsf{pp}, d, \mathsf{st}_{\mathcal{D}}) \leftarrow \mathrm{Conf}^{\mathcal{D}}(\lambda, N) \\ \mathsf{st}_{\mathcal{A}}, \mathfrak{p} \leftarrow \mathcal{A}_1(c) \\ \mathsf{st}, \tilde{\mathfrak{p}} \leftarrow \mathsf{randomize}(\mathfrak{p}) \\ \tilde{p}, \tilde{p}', S \leftarrow \mathcal{A}_2(\mathsf{st}_{\mathcal{A}}, \tilde{\mathfrak{p}}) \end{array}\right] \leq \mathsf{negl}(\lambda) .$$

Proof. By unrolling the calls to $\mathsf{randomize}$ and $\mathsf{recover}$ in the probability above, we need to show

$$\Pr[\mathsf{Hyb}_0(\lambda, N) = 1] \leq \mathsf{negl}(\lambda) ,$$

where Hyb_0 is defined as follows.

$\mathsf{Hyb}_0(\lambda, N)$

1 : $c = (\mathsf{pp}, d, \mathsf{st}_{\mathcal{D}}) \leftarrow \mathrm{Conf}^{\mathcal{D}}(\lambda, N)$
2 : $\mathsf{st}_{\mathcal{A}}, \mathfrak{p} \leftarrow \mathcal{A}_1(c)$
3 : $(r, \alpha) \leftarrow\!\!\$\; \mathbb{Z}_q \times \mathbb{Z}_q^*$
4 : $\tilde{\mathfrak{p}} := \mathbf{h}^r \circ \mathfrak{p}^\alpha$
5 : $\tilde{p}, \tilde{p}', S \leftarrow \mathcal{A}_2(\mathsf{st}_{\mathcal{A}}, \tilde{\mathfrak{p}})$
6 : **return** $\mathbb{I}\left[\tilde{p} \neq \tilde{p}' \wedge \left(\dfrac{\tilde{p}'}{\tilde{p}}\right)^{1/\alpha} \in S\right]$

We do so by a hybrid argument.

Hyb_1 Note that in Hyb_0, the random $r \leftarrow\$ \mathbb{Z}_q$ is *only* used for computing $\mathbf{h}^r$. By DDH (Lemma 1), this means that $\mathbf{h}^r$ looks identical to a uniformly random vector in $\mathbb{G}^N$. Therefore, in Hyb_1, we replace the lines

$$\begin{array}{l}(r,\alpha) \leftarrow\$ \mathbb{Z}_q \times \mathbb{Z}_q^* \\ \tilde{\mathfrak{p}} := \mathbf{h}^r \circ \mathfrak{p}^\alpha\end{array} \quad \text{by} \quad \begin{array}{l}\alpha \leftarrow\$ \mathbb{Z}_q^* \\ \overline{\mathbf{h}} \leftarrow\$ \mathbb{G}^N \\ \tilde{\mathfrak{p}} := \overline{\mathbf{h}} \circ \mathfrak{p}^\alpha\end{array} .$$

We use the following ppt adversary $\mathcal{B}$ distinguishing between the two distributions in Lemma 1:

- $\mathcal{B}(\mathsf{pp}', \overline{\mathbf{h}})$: Run $(\mathsf{st}_\mathcal{D}, d) \leftarrow \mathcal{D}(\mathsf{pp})$ (with $\mathsf{pp} := (\mathsf{pp}', \overline{\mathbf{h}})$) to create the configuration $c := (\mathsf{pp}, d, \mathsf{st}_\mathcal{D})$, and run $\mathsf{st}_\mathcal{A}, \mathfrak{p} \leftarrow \mathcal{A}_1(c)$. Then, sample $\alpha \leftarrow\$ \mathbb{Z}_q^*$ and compute $\tilde{\mathfrak{p}} := \overline{\mathbf{h}} \circ \mathfrak{p}^\alpha$. Run $\tilde{p}, \tilde{p}', S \leftarrow \mathcal{A}_2(\mathsf{st}_\mathcal{A}, \tilde{\mathfrak{p}})$, and return $\mathbb{I}[(\frac{\tilde{p}'}{\tilde{p}})^{1/\alpha} \in S \setminus \{1\}]$.

Note that running $\mathcal{B}$ on left-hand side distribution in Lemma 1 is identical to Hyb_0 (i.e., $\overline{\mathbf{h}} := \mathbf{h}^r$ for $r \leftarrow\$ \mathbb{Z}_q$), and that running $\mathcal{B}$ on the right-hand side distribution is identical to Hyb_1 (i.e., $\overline{\mathbf{h}} \leftarrow\$ \mathbb{G}^N$). Thus, by Lemma 1, we get

$$|\Pr[\mathsf{Hyb}_1(\lambda, N) = 1] - \Pr[\mathsf{Hyb}_0(\lambda, N) = 1]| \le \mathsf{negl}(\lambda) .$$

Hyb_2 Note that $\tilde{\mathfrak{p}} := \overline{\mathbf{h}} \circ \mathfrak{p}^\alpha$ for random $\overline{\mathbf{h}} \leftarrow\$ \mathbb{G}^N$ is identically distributed as a uniformly random vector in $\mathbb{G}^N$. Thus, in Hyb_2, we replace the lines

$$\begin{array}{l}\overline{\mathbf{h}} \leftarrow\$ \mathbb{G}^N \\ \tilde{\mathfrak{p}} := \overline{\mathbf{h}} \circ \mathfrak{p}^\alpha\end{array} \quad \text{by} \quad \tilde{\mathfrak{p}} \leftarrow\$ \mathbb{G}^N \quad .$$

We have $\mathsf{Hyb}_2 \equiv \mathsf{Hyb}_1$.

Note that in Hyb_2, $\alpha \leftarrow\$ \mathbb{Z}_q^*$ is only used in the very last line, which returns $\mathbb{I}[\tilde{p} \neq \tilde{p}' \wedge (\frac{\tilde{p}'}{\tilde{p}})^{1/\alpha} \in S]$. Furthermore, if $\tilde{p} \neq \tilde{p}'$ (i.e., $\frac{\tilde{p}'}{\tilde{p}} \neq 1$), then $(\frac{\tilde{p}'}{\tilde{p}})^{1/\alpha}$ will be $\perp$ or a uniformly random element in $\mathbb{G} \setminus \{1\}$. Thus, the probability of returning 1 is at most $\frac{|S|}{q-1}$. Because of $|S| \le \mathrm{poly}(\lambda, N)$, we get

$$\Pr[\mathsf{Hyb}_2(\lambda, N) = 1] \le \mathsf{negl}(\lambda) ,$$

which concludes the proof by our hybrid argument. □

Now we prove integrity: let $\mathcal{D}, \mathcal{A}$ be an adversary for the integrity game, $N(\lambda) \le \mathrm{poly}(\lambda)$ the database size, and $i(\lambda) \in [N]$ an index. We need to show

$$\Pr\left[\begin{array}{l}\tilde{p} \neq \tilde{p}' \wedge \\ \mathsf{Rec}(\mathsf{st}, d, \tilde{p}) \neq \perp \wedge \\ \mathsf{Rec}(\mathsf{st}, d, \tilde{p}') \neq \perp\end{array} \middle| \begin{array}{l} c = (\mathsf{pp}, d, \mathsf{st}_\mathcal{D}) \leftarrow \mathrm{Conf}^\mathcal{D}(\lambda, N) \\ \mathsf{st}, \tilde{\mathfrak{p}} \leftarrow \mathsf{randomize}(g^{\mathbf{e}_i}) \\ \tilde{p}, \tilde{p}' \leftarrow \mathcal{A}(c, \tilde{\mathfrak{p}})\end{array}\right] \le \mathsf{negl}(\lambda) .$$

Note that the event $\mathsf{Rec}(\mathsf{st}, d, \tilde{p}) \neq \perp$ holds iff $\mathsf{recover}(\mathsf{st}, \tilde{p}) \in \{g^{-N}, \ldots, g^N\}$ (and analogously for $\tilde{p}'$). Therefore,

$$\begin{array}{l}\tilde{p} \neq \tilde{p}' \wedge \\ \mathsf{Rec}(\mathsf{st}, d, \tilde{p}) \neq \perp \wedge \\ \mathsf{Rec}(\mathsf{st}, d, \tilde{p}') \neq \perp\end{array} \quad \Rightarrow \quad \begin{array}{l}\tilde{p} \neq \tilde{p}' \wedge \\ \dfrac{\mathsf{recover}(\mathsf{st}, \tilde{p}')}{\mathsf{recover}(\mathsf{st}, \tilde{p})} \in \{g^{-2N}, \ldots, g^{2N}\}\end{array} ,$$

and it suffices to prove that the probability of the event on the right-hand side is negligible. This follows immediately by applying Lemma 4 to the following adversary $\mathcal{D}, \mathcal{B}_1, \mathcal{B}_2$.

- $\mathcal{B}_1(c)$ outputs $\mathsf{st}_\mathcal{B} := c$, $\mathfrak{p} := g^{\mathbf{e}_i}$.
- $\mathcal{B}_2(\mathsf{st}_\mathcal{B}, \tilde{\mathfrak{p}})$ runs $\tilde{p}, \tilde{p}' \leftarrow \mathcal{A}(c, \tilde{\mathfrak{p}})$, chooses $S := \{g^{-2N}, \ldots, g^{2N}\}$, and outputs $\tilde{p}, \tilde{p}', S$.

4.3 Proof of Theorem 2 (Standard Privacy)

We prove a stronger notion of privacy, called *chosen-vector indistinguishability*, which will be useful elsewhere, and whose security game is as follows:

$$\begin{array}{ll}
\multicolumn{2}{l}{\text{Game } \mathrm{CVA}^{\mathcal{A}^*}(\lambda, N, b)} \\
\hline
1: & c = (\mathsf{pp}, d, \mathsf{st}_{\mathcal{D}}) \leftarrow \mathrm{Conf}^{\mathcal{D}}(\lambda, N) \\
2: & \mathsf{st}_{\mathcal{A}}, \mathfrak{v}_0, \mathfrak{v}_1 \leftarrow \mathcal{A}_1(c); r \leftarrow\!\!\$\ \mathbb{Z}_q \\
3: & \textbf{return } \mathcal{A}_2(\mathsf{st}_{\mathcal{A}}, \mathbf{h}^r \circ \mathfrak{v}_b)
\end{array}$$

Lemma 5 (Chosen Vector Indistinguishability). *Assuming that the DDH assumption holds, then for any ppt adversaries $\mathcal{A}^* = (\mathcal{D}, \mathcal{A}_1, \mathcal{A}_2)$ and database sizes $N(\lambda) \leq \mathrm{poly}(\lambda)$,*

$$|\Pr[\mathrm{CVA}^{\mathcal{A}^*}(\lambda, N, 0) = 1] - \Pr[\mathrm{CVA}^{\mathcal{A}^*}(\lambda, N, 1) = 1]| \leq \mathsf{negl}(\lambda) .$$

Remark 4. In order to simplify reductions involving $\mathrm{CVA}^{\mathcal{A}^*}$, we defined this experiment to sample a configuration $c = (\mathsf{pp}, d, \mathsf{st}_{\mathcal{D}}) \leftarrow \mathrm{Conf}^{\mathcal{D}}(\lambda, N)$. Lemma 5 would still hold if $\mathrm{CVA}^{\mathcal{A}^*}$ does not run the digest generator $\mathsf{st}_{\mathcal{D}}, d \leftarrow \mathcal{D}(\mathsf{pp})$, and instead only provides $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^N)$ as input to $\mathcal{A}_1$.

Proof. We show that both $\mathrm{CVA}^{\mathcal{A}^*}(\lambda, N, 0)$ and $\mathrm{CVA}^{\mathcal{A}^*}(\lambda, N, 1)$ are close to Hyb defined below.

$$\begin{array}{ll}
\multicolumn{2}{l}{\mathsf{Hyb}(\lambda, N)} \\
\hline
1: & c = (\mathsf{pp}, d, \mathsf{st}_{\mathcal{D}}) \leftarrow \mathrm{Conf}^{\mathcal{D}}(\lambda, N) \\
2: & \mathsf{st}_{\mathcal{A}}, \mathfrak{v}_0, \mathfrak{v}_1 \leftarrow \mathcal{A}_1(c) \\
3: & \tilde{\mathfrak{v}} \leftarrow\!\!\$\ \mathbb{G}^N \\
4: & \textbf{return } \mathcal{A}_2(c, \tilde{\mathfrak{v}})
\end{array}$$

By a hybrid argument, it suffices to show

$$|\Pr[\mathrm{CVA}^{\mathcal{A}^*}(\lambda, N, b) = 1] - \Pr[\mathsf{Hyb}(\lambda, N) = 1]| \leq \mathsf{negl}(\lambda), \tag{11}$$

for both $b = 0, 1$.

To show that Equation 11 holds, we construct an adversary $\mathcal{B}$ that distinguishes between the two distributions stated in Lemma 1 in the following way:

- $\mathcal{B}(\mathsf{pp}', \overline{\mathbf{h}})$: Run $(\mathsf{st}_{\mathcal{D}}, d) \leftarrow \mathcal{D}(\mathsf{pp})$ (with $\mathsf{pp} := (\mathsf{pp}', \overline{\mathbf{h}})$) to create the configuration $c := (\mathsf{pp}, d, \mathsf{st}_{\mathcal{D}})$. Then, compute $\mathsf{st}_{\mathcal{A}}, \mathfrak{v}_0, \mathfrak{v}_1 \leftarrow \mathcal{A}_1(c)$. Run and output $\mathcal{A}_2(\mathsf{st}_{\mathcal{A}}, \overline{\mathbf{h}} \circ \mathfrak{v}_b)$.

Note that running $\mathcal{B}$ on the left-hand side distribution of Lemma 1 is identical to $\mathrm{CVA}^{\mathcal{A}^*}(\lambda, N, b)$. Furthermore, running $\mathcal{B}$ on the right-hand side distribution of Lemma 1 is identical to $\mathsf{Hyb}(\lambda, N)$. Equation 11 follows. □

Now we prove standard privacy. Let $\mathcal{D}, \mathcal{A}$ be an adversary for the privacy game, $N(\lambda) \leq \mathrm{poly}(\lambda)$ the database size, and $i_0(\lambda), i_1(\lambda) \in [N]$ indices.

We plug in the following adversary $\mathcal{B}^* := (\mathcal{D}, \mathcal{B}_1, \mathcal{B}_2)$ into Lemma 5:

- $\mathcal{B}_1(c)$: sample $\alpha \leftarrow\!\!\$\ \mathbb{Z}_q^*$ and output $\mathsf{st}_{\mathcal{A}} := c$, $\mathfrak{v}_0 := (g^{\mathbf{e}_{i_0}})^\alpha$, $\mathfrak{v}_1 := (g^{\mathbf{e}_{i_1}})^\alpha$.
- $\mathcal{B}_2(c, \tilde{\mathfrak{v}})$: run and output $\mathcal{A}(c, \tilde{\mathfrak{v}})$.

Privacy follows from Lemma 5 because of $\mathrm{CVA}^{\mathcal{B}^*}(\lambda, N, b) \equiv \mathrm{PRIV}^{\mathcal{D},\mathcal{A}}(\lambda, N, i_b)$.

$$
\begin{array}{l}
\text{Procedure } \mathcal{E}_{\mathsf{ans}}(1^r, c = (\mathsf{pp}, d, \mathsf{st}_{\mathcal{D}}), \mathfrak{p}) \\
\hline
\textbf{repeat } \lambda \cdot r^3 \textbf{ times} \\
\quad \mathsf{st}, \tilde{\mathfrak{p}} \leftarrow \mathsf{randomize}(\mathfrak{p}) \\
\quad \tilde{p} \leftarrow \overline{\mathcal{V}}(c, \tilde{\mathfrak{p}}) \\
\quad \mathsf{st}', \tilde{\mathfrak{p}}' \leftarrow \mathsf{randomize}(\tilde{\mathfrak{p}}) \\
\quad \tilde{p}' \leftarrow \mathsf{recover}(\mathsf{st}', \overline{\mathcal{V}}(c, \tilde{\mathfrak{p}}')) \\
\quad \textbf{if } \tilde{p} = \tilde{p}', \textbf{return } \mathsf{recover}(\mathsf{st}, \tilde{p}) \\
\textbf{return } \bot
\end{array}
\qquad
\begin{array}{l}
\text{Procedure } \overline{\mathcal{V}}(c, \tilde{\mathfrak{p}}) \\
\hline
\mathbf{s}_j \leftarrow\!\!\$\ \{0,1\}^N \quad \forall j \in [\lambda] \setminus \{1\} \\
\_, \tilde{\mathfrak{s}}_j \leftarrow \mathsf{randomize}(g^{\mathbf{s}_j}) \quad \forall j \in [\lambda] \setminus \{1\} \\
(\tilde{p}, \_, \ldots, \_) \leftarrow \mathcal{V}(c, (\tilde{\mathfrak{p}}, \tilde{\mathfrak{s}}_2, \ldots, \tilde{\mathfrak{s}}_\lambda)) \\
\textbf{return } \tilde{p}
\end{array}
$$

Fig. 5. Description of the extractor $\mathcal{E}_{\mathsf{ans}}$ (left-hand side) used in the proof of Theorem 3. The right-hand side describes a sub-procedure used within $\mathcal{E}_{\mathsf{ans}}$.

4.4 Proof of Theorem 3 (Answer extractability)

Let $(\mathcal{D}, \mathcal{V})$ be an adversary against answer extractability for our PIR scheme, where $\mathcal{D}$ is the digest generator, and $\mathcal{V}$ answers validation queries. Our answer extractor $\mathcal{E}_{\mathsf{ans}}(1^r, c, \mathfrak{p})$ is described in Figure 5. The core idea is to repeatedly invoke $\mathcal{V}$ to get answers for $\mathfrak{p}$. Because $\mathcal{V}$ expects a list $v = (\tilde{\mathfrak{s}}_1, \ldots, \tilde{\mathfrak{s}}_\lambda)$ of λ validation queries, we use a wrapper procedure $\overline{\mathcal{V}}(c, \tilde{\mathfrak{p}})$ which embeds $\tilde{\mathfrak{p}}$ as the first component of a vector consisting of otherwise honestly generated $\lambda - 1$ validation queries.

Throughout λr^3 iterations, the extractor $\mathcal{E}_{\mathsf{ans}}$ queries first $\overline{\mathcal{V}}(c, \tilde{\mathfrak{p}})$ on a fresh randomization $\tilde{\mathfrak{p}}$ of $\mathfrak{p}$, and obtains answer $\tilde{p}$. Then, $\mathcal{E}_{\mathsf{ans}}$ generates a fresh randomization $\tilde{\mathfrak{p}}'$ of $\tilde{\mathfrak{p}}$, and runs $\overline{\mathcal{V}}(c, \tilde{\mathfrak{p}}')$. Intuitively, we use the fact that the recovered answer $\tilde{p}'$ matches $\tilde{p}$ as an indication that $\overline{\mathcal{V}}$ has answered correctly. Therefore, in this case, $\mathcal{E}_{\mathsf{ans}}$ returns $\mathsf{recover}(\mathsf{st}, \tilde{p})$. However, if no iteration is successful, then $\mathcal{E}_{\mathsf{ans}}$ returns $\bot$.

Roadmap. We now give an informal description of why $\mathcal{E}_{\mathsf{ans}}$ indeed works. We refer to several lemmas that we will state formally afterwards, followed by the full proof of answer extractability using these lemmas.

First, Lemma 9, establishes what we refer to as the *answer guarantee* property, i.e., the fact that $\mathcal{E}_{\mathsf{ans}}$ never answers with $\bot$ on any (even adversarially chosen) query, as long as $\nu_{\mathcal{D},\mathcal{V}}(c)$ is large enough. The proof will use crucially the indistinguishability of any two randomized queries.

The bulk of the proof considers the scenario in which we query $\mathcal{E}_{\mathsf{ans}}$ on $g^{\mathbf{e}_i}$ for all $i \in [N]$, obtaining answers

$$e_i \leftarrow \mathcal{E}_{\mathsf{ans}}(1^{r(\lambda)}, c, g^{\mathbf{e}_i}) \quad \forall i \in [N] \; .$$

The queries $g^{\mathbf{e}_i}$ are not randomized, which is going to simplify our analysis–Lemma 6 (*randomization independence*) shows that this is equivalent to randomizing $g^{\mathbf{e}_i}$ first, and later recovering $\mathcal{E}_{\mathsf{ans}}$'s answer. Therefore, to infer answer extractability, we simply need to show that $e_i \in \{g^{-N}, \ldots, g^N\}$ for each $i \in [N]$.

Also, assume that we already know that $\mathcal{E}_{\mathsf{ans}}$ answers *validations* successfully, i.e., for λ random vectors $\mathbf{s}_1, \ldots, \mathbf{s}_\lambda \in \{0,1\}^N$ the answers $s'_k \leftarrow \mathcal{E}_{\mathsf{ans}}(1^{r(\lambda)}, c, g^{\mathbf{s}_k})$ will fulfill $s'_1, \ldots, s'_\lambda \in \{g^0, \ldots, g^N\}$. To draw a connection to the individual answers e_i, we need a *homomorphism* property (Lemma 8), which states that

$$s'_k = \prod_{i \in [N]} e_i^{\mathbf{s}_k[i]} \quad \forall k \in [\lambda] \; .$$

Because vectors $\mathbf{s}_k$ are random, a simple argument (Lemma 10, *random subset testing*) shows that, due to the small range of all s'_k, with probability at least $1 - \frac{1}{2^\lambda}$, all the individual answers are also in a small range: $e_i \in \{g^{-N}, \ldots, g^N\}$.

It remains to ensure that there is a good chance of $\mathcal{E}_{\mathsf{ans}}$ answering all validation queries $\mathbf{s}_1, \ldots, \mathbf{s}_\lambda$ correctly. Indeed, whenever $\overline{\mathcal{V}}(c, \tilde{\mathfrak{p}})$ manages to reply to the randomization $\tilde{\mathfrak{p}}$ of a query $g^{\mathbf{s}_k}$ with a value $s_k \in \{g^0, \ldots, g^N\}$, then w.h.p., $\mathcal{E}_{\mathsf{ans}}$ will output the exact same value $s'_k = s_k$ (Lemma 7, $\mathcal{E}_{\mathsf{ans}}$ *agrees with small answers*). Therefore, whenever $\nu_{\mathcal{D},\mathcal{V}}(c)$ is large enough, we can be sure that (after testing sufficiently many different validations), $\mathcal{E}_{\mathsf{ans}}$'s responses fulfill $s'_1, \ldots, s'_\lambda \in \{g^0, \ldots, g^N\}$, thereby implying $e_1, \ldots, e_N \in \{g^{-N}, \ldots, g^N\}$.

Basic extractor properties. We state four essential properties of our $\mathcal{E}_{\mathsf{ans}}$ construction that suffice to prove answer extractability. We defer their proofs to Sections 4.5, 4.6, 4.7, and 4.8.

Lemma 6 (Randomization Independence). *For all $\lambda, N \in \mathbb{N}$ and configurations $c = (\mathsf{pp}, d, \mathsf{st}_{\mathcal{D}}) \in \mathsf{Supp}(\mathrm{Conf}^{\mathcal{D}}(\lambda, N))$, queries $\mathfrak{p} \in \mathbb{G}^N$, randomizations $(\mathsf{st}, \tilde{\mathfrak{p}}) \in \mathsf{Supp}(\mathsf{randomize}(\mathfrak{p}))$ of $\mathfrak{p}$, and runtime parameters $r \in \mathbb{N}$, the following two distributions are exactly the same:*

$$\mathcal{E}_{\mathsf{ans}}(1^r, c, \mathfrak{p}) \quad \text{and} \quad \mathsf{recover}(\mathsf{st}, \mathcal{E}_{\mathsf{ans}}(1^r, c, \tilde{\mathfrak{p}}))$$

Lemma 7 (Agrees with Small Answers). *Assuming the DDH assumption holds, then for all ppt adversaries $\mathcal{A}_1, \mathcal{A}_2$ (where $\mathcal{A}_1(c)$ outputs a state $\mathsf{st}_{\mathcal{A}}$ and a query vector $\mathfrak{p} \in \mathbb{G}^N$, and $\mathcal{A}_2(\mathsf{st}_{\mathcal{A}}, \tilde{\mathfrak{p}}^*)$ outputs an answer $p^* \in \mathbb{G} \cup \{\perp\}$), database sizes $N(\lambda) \leq \mathrm{poly}(\lambda)$, and polynomials $r(\lambda)$:*

$$\Pr\left[\begin{array}{l} p^* \in \{g^0, \ldots, g^N\} \\ \ \Rightarrow p \in \{p^*, \perp\} \end{array} \middle| \begin{array}{l} c = (\mathsf{pp}, d, \mathsf{st}_{\mathcal{D}}) \leftarrow \mathrm{Conf}^{\mathcal{D}}(\lambda, N) \\ \mathsf{st}_{\mathcal{A}}, \mathfrak{p} \leftarrow \mathcal{A}_1(c) \\ \mathsf{st}^*, \tilde{\mathfrak{p}}^* \leftarrow \mathsf{randomize}(\mathfrak{p}) \\ p^* \leftarrow \mathsf{recover}(\mathsf{st}^*, \mathcal{A}_2(\mathsf{st}_{\mathcal{A}}, \tilde{\mathfrak{p}}^*)) \\ p \leftarrow \mathcal{E}_{\mathsf{ans}}(1^{r(\lambda)}, c, \mathfrak{p}) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda) .$$

Lemma 8 (Homomorphism). *Assuming the DDH assumption holds, then for all ppt adversaries $\mathcal{A}$ (where $\mathcal{A}(c)$ outputs m query vectors $\mathfrak{p}_1, \ldots, \mathfrak{p}_m \in \mathbb{G}^N$ and m exponents $t_1, \ldots, t_m \in \mathbb{Z}_q$), and $N(\lambda), r(\lambda) \leq \mathrm{poly}(\lambda)$,*

$$\Pr\left[\begin{array}{l} p, p_1, \ldots, p_m \neq \perp \\ \ \Rightarrow p = \displaystyle\prod_{i \in [m]} p_i^{t_i} \end{array} \middle| \begin{array}{l} c = (\mathsf{pp}, d, \mathsf{st}_{\mathcal{D}}) \leftarrow \mathrm{Conf}^{\mathcal{D}}(\lambda, N) \\ \mathfrak{p}_1, \ldots, \mathfrak{p}_m, t_1, \ldots, t_m \leftarrow \mathcal{A}(c) \\ p \leftarrow \mathcal{E}_{\mathsf{ans}}(1^{r(\lambda)}, c, \prod_{i \in [m]} \mathfrak{p}_i^{t_i}) \\ p_i \leftarrow \mathcal{E}_{\mathsf{ans}}(1^{r(\lambda)}, c, \mathfrak{p}_i) \quad \forall i \in [m] \end{array}\right] \geq 1 - \mathsf{negl}(\lambda) .$$

Lemma 9 (Answer Guarantee). *Assuming the DDH assumption holds, then for all ppt adversaries $\mathcal{A}$ (where $\mathcal{A}(c)$ outputs a query vector $\mathfrak{p} \in \mathbb{G}^N$), database sizes $N(\lambda) \leq \mathrm{poly}(\lambda)$, and polynomials $r(\lambda)$,*

$$\Pr\left[\begin{array}{l} \nu_{\mathcal{D},\mathcal{V}}(c) \geq \frac{1}{r(\lambda)} \\ \Rightarrow \mathcal{E}_{\mathsf{ans}}(1^{r(\lambda)}, c, \mathfrak{p}) \neq \perp \end{array} \middle| \begin{array}{c} c = (\mathsf{pp}, d, \mathsf{st}_{\mathcal{D}}) \leftarrow \mathrm{Conf}^{\mathcal{D}}(\lambda, N) \\ \mathfrak{p} \leftarrow \mathcal{A}(c) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda) .$$

Random Subset Testing. We will also make use of the following lemma, which is proved in Section 4.9.

Lemma 10 (Random Subset Testing). *Let $N \in \mathbb{N}$, let $\mathbb{G}$ be a cyclic group with generator $g \in \mathbb{G}$, and let $e_1, \ldots, e_N \in \mathbb{G}$ be arbitrary group elements. Further, assume that for at least one $i^* \in [N]$, we have $e_{i^*} \notin \{g^{-N}, \ldots, g^N\}$. Then, for $\mathbf{s}_1, \ldots, \mathbf{s}_\lambda \leftarrow\$ \{0,1\}^N$,*

$$\Pr\left[\forall k \in [\lambda] : \prod_{i \in [N]} e_i^{\mathbf{s}_k[i]} \in \{g^0, \ldots, g^N\}\right] \leq \frac{1}{2^\lambda} .$$

THE ACTUAL PROOF. We are now ready to combine all of the above elements and prove answer extractability (Equation 3), i.e., for all database sizes $N(\lambda) \leq \text{poly}(\lambda)$, indices $i(\lambda) \in [N(\lambda)]$, polynomials $r(\lambda) \leq \text{poly}(\lambda)$,

$$\Pr\left[\begin{array}{l} \nu_{\mathcal{D},\mathcal{V}}(c) \geq \frac{1}{r(\lambda)} \Rightarrow \\ \mathsf{Rec}(\mathsf{st}, d, \tilde{e}) \neq \bot \end{array}\middle| \begin{array}{r} c = (\mathsf{pp}, d, \mathsf{st}_{\mathcal{D}}) \leftarrow \mathrm{Conf}^{\mathcal{D}}(\lambda, N) \\ \mathsf{st}, \tilde{\mathfrak{e}} \leftarrow \mathsf{Query}(i) \\ \tilde{e} \leftarrow \mathcal{E}_{\mathsf{ans}}(1^{r(\lambda)}, c, \tilde{\mathfrak{e}}) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda) .$$

Using Lemma 6, this simplifies to

$$\Pr\left[\begin{array}{l} \nu_{\mathcal{D},\mathcal{V}}(c) \geq \frac{1}{r(\lambda)} \Rightarrow \\ e \in \{g^{-N}, \ldots, g^{N}\} \end{array}\middle| \begin{array}{r} c = (\mathsf{pp}, d, \mathsf{st}_{\mathcal{D}}) \leftarrow \mathrm{Conf}^{\mathcal{D}}(\lambda, N) \\ e \leftarrow \mathcal{E}_{\mathsf{ans}}(1^{r(\lambda)}, c, g^{\mathbf{e}_i}) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda) .$$

In fact, we will prove something even stronger, which is that for all database sizes $N(\lambda) \leq \text{poly}(\lambda)$ and polynomials $r(\lambda)$, the answers of $\mathcal{E}_{\mathsf{ans}}$ on the whole database (not just a single index i) is in $\{g^{-N}, \ldots, g^{N}\}$:

$$\Pr\left[\begin{array}{l} \nu_{\mathcal{D},\mathcal{V}}(c) \geq \frac{1}{r(\lambda)} \Rightarrow \\ \forall i \in [N] : e_i \in \{g^{-N}, \ldots, g^{N}\} \end{array}\middle| \begin{array}{l} c = (\mathsf{pp}, d, \mathsf{st}_{\mathcal{D}}) \leftarrow \mathrm{Conf}^{\mathcal{D}}(\lambda, N) \\ e_i \leftarrow \mathcal{E}_{\mathsf{ans}}(1^{r(\lambda)}, c, g^{\mathbf{e}_i}) \quad \forall i \in [N] \end{array}\right] \geq 1 - \mathsf{negl}(\lambda) . \quad (12)$$

To prove this, it is helpful to consider the following augmented experiment, which additionally introduces several validation queries, and both $\mathcal{V}$'s and $\mathcal{E}_{\mathsf{ans}}$'s answers to those:

1. First, sample configuration $c = (\mathsf{pp}, d, \mathsf{st}_{\mathcal{D}}) \leftarrow \mathrm{Conf}^{\mathcal{D}}(\lambda, N)$.
2. Second, query $\mathcal{E}_{\mathsf{ans}}$ on the whole database:

$$e_i \leftarrow \mathcal{E}_{\mathsf{ans}}(1^{r(\lambda)}, c, g^{\mathbf{e}_i}) \qquad \forall i \in [N]$$

3. Third, sample $\lambda \cdot r(\lambda)$ random validations, and run $\mathcal{V}$ on them:

$$\begin{array}{ll} \mathbf{s}_{j,k} \leftarrow\!\!\$\ \{0,1\}^N & \forall j \in [\lambda \cdot r(\lambda)], k \in [\lambda] \\ \mathsf{st}_{j,k}, \tilde{\mathfrak{s}}_{j,k} \leftarrow \mathsf{randomize}(g^{\mathbf{s}_{j,k}}) & \forall j \in [\lambda \cdot r(\lambda)], k \in [\lambda] \\ \tilde{s}_{j,1}, \ldots, \tilde{s}_{j,\lambda} \leftarrow \mathcal{V}(c, (\tilde{\mathfrak{s}}_{j,1}, \ldots, \tilde{\mathfrak{s}}_{j,\lambda})) & \forall j \in [\lambda \cdot r(\lambda)] \\ s_{j,k} \leftarrow \mathsf{recover}(\mathsf{st}_{j,k}, \tilde{s}_{j,k}) & \forall j \in [\lambda \cdot r(\lambda)], k \in [\lambda] \end{array}$$

4. And finally, we run $\mathcal{E}_{\mathsf{ans}}$ on the same validation queries:

$$s'_{j,k} \leftarrow \mathcal{E}_{\mathsf{ans}}(1^{r(\lambda)}, c, g^{\mathbf{s}_{j,k}}) \qquad \forall j \in [\lambda \cdot r(\lambda)], k \in [\lambda]$$

We are now going to apply Lemmas 7, 8, and 9. In each of the following probabilities, we refer to the distribution of all c, e_i, $\mathbf{s}_{j,k}$, $s_{j,k}$, and $s'_{j,k}$ as described above.

– Note that Step 3 performs $\lambda \cdot r(\lambda)$ independently sampled validations identically to $\mathsf{VQuery}()$. Thus, under the assumption that configuration c has a high success probability ($\nu_{\mathcal{D},\mathcal{V}}(c) \geq \frac{1}{r(\lambda)}$), there will be at least one validation that succeeds: The probability that for no $j^* \in [\lambda \cdot r(\lambda)]$, the j^*-th validation succeeds, is at most

$$\left(1 - \frac{1}{r(\lambda)}\right)^{\lambda \cdot r(\lambda)} \leq e^{-\frac{1}{r(\lambda)} \cdot \lambda \cdot r(\lambda)} = e^{-\lambda} \leq \mathsf{negl}(\lambda) .$$

  A successful validation (as defined by $\mathsf{VCheck}(\{\mathsf{st}_{j^*,k}\}, d, \{\tilde{s}_{j^*,k}\})$) means that $s_{j^*,1}, \ldots, s_{j^*,\lambda} \in \{g^0, \ldots, g^N\}$, and therefore:

$$\Pr\left[\begin{array}{l} \nu_{\mathcal{D},\mathcal{V}}(c) \geq \frac{1}{r(\lambda)} \Rightarrow \\ \exists j^* \in [\lambda \cdot r(\lambda)] \text{ with } s_{j^*,1}, \ldots, s_{j^*,\lambda} \in \{g^0, \ldots, g^N\} \end{array}\right] \geq 1 - \mathsf{negl}(\lambda) \qquad (13)$$

- For all $j \in [\lambda \cdot r(\lambda)]$ and $k \in [\lambda]$, we apply Lemma 7 ($\mathcal{E}_{\mathsf{ans}}$ *agrees with small answers*) to the following adversary $\mathcal{A}$:
  - $\mathcal{A}_1(c)$ samples $\mathbf{s}_{j,k} \leftarrow\$ \{0,1\}^N$ and returns $\mathsf{st}_{\mathcal{A}} := (c, \mathbf{s}_{j,k})$ and $g^{\mathbf{s}_{j,k}}$.
  - $\mathcal{A}_2(\mathsf{st}_{\mathcal{A}}, \tilde{\mathfrak{s}}_{j,k})$, for all $k' \in [\lambda] \backslash \{k\}$, samples $\mathbf{s}_{j,k'} \leftarrow\$ \{0,1\}^N$ and randomizes $\mathsf{st}_{j,k'}, \tilde{\mathfrak{s}}_{j,k'} \leftarrow \mathsf{randomize}(g^{\mathbf{s}_{j,k'}})$, then runs $\tilde{s}_{j,1}, \ldots, \tilde{s}_{j,\lambda} \leftarrow \mathcal{V}(c, (\tilde{\mathfrak{s}}_{j,1}, \ldots, \tilde{\mathfrak{s}}_{j,\lambda}))$ and outputs $\tilde{s}_{j,k}$.

  Then, by Lemma 7, for $s_{j,k}$ produced by $\mathcal{V}$ that is within the set $\{g^0, \ldots, g^N\}$, we can be certain that $\mathcal{E}_{\mathsf{ans}}$'s answer $s'_{j,k}$ is equal to $s_{j,k}$ or $\perp$:

$$\Pr\begin{bmatrix} s_{j,k} \in \{g^0, \ldots, g^N\} \Rightarrow \\ s'_{j,k} \in \{s_{j,k}, \perp\} \end{bmatrix} \geq 1 - \mathsf{negl}(\lambda) \quad \forall j \in [\lambda \cdot r(\lambda)], k \in [\lambda] \tag{14}$$

- For all $j \in [\lambda \cdot r(\lambda)]$ and $k \in [\lambda]$, we apply Lemma 8 (*homomorphism*) to the following adversary $\mathcal{A}$:
  - $\mathcal{A}(c)$ samples $\mathbf{s}_{j,k} \leftarrow\$ \{0,1\}^N$, and outputs the N unit queries $g^{\mathbf{e}_1}, \ldots, g^{\mathbf{e}_N}$, and exponents $\mathbf{s}_{j,k}[1], \ldots, \mathbf{s}_{j,k}[N]$.

  Then, by Lemma 8, we can be certain that $\mathcal{E}_{\mathsf{ans}}$'s answer $s'_{j,k}$ to validation query $g^{\mathbf{s}_{j,k}}$ is equal to the product $\prod_{i\in[N]} e_i^{\mathbf{s}_{j,k}[i]}$ of answers e_i (as long as none of these answers is $\perp$):

$$\Pr\begin{bmatrix} s'_{j,k}, e_1, \ldots, e_N \neq \perp \Rightarrow \\ s'_{j,k} = \prod_{i\in[N]} e_i^{\mathbf{s}_{j,k}[i]} \end{bmatrix} \geq 1 - \mathsf{negl}(\lambda) \quad \forall j \in [\lambda \cdot r(\lambda)], k \in [\lambda] \tag{15}$$

- For all $i \in [N]$, we apply Lemma 9 (*guaranteed to answer*) to the following adversary $\mathcal{A}$:
  - $\mathcal{A}(c)$ returns $g^{\mathbf{e}_i}$.

  Then, by Lemma 9, whenever c has high validation success probability (i.e., $\nu_{\mathcal{D},\mathcal{V}}(c) \geq \frac{1}{r(\lambda)}$), then w.h.p. $\mathcal{E}_{\mathsf{ans}}$ answered all queries without ever returning $\perp$:

$$\Pr\begin{bmatrix} \nu_{\mathcal{D},\mathcal{V}}(c) \geq \frac{1}{r(\lambda)} \Rightarrow \\ e_i \neq \perp \end{bmatrix} \geq 1 - \mathsf{negl}(\lambda) \quad \forall i \in [N] \tag{16}$$

  Similarly, Lemma 9 also shows that none of the answers $s'_{j,k}$ of $\mathcal{E}_{\mathsf{ans}}$ is $\perp$:

$$\Pr\begin{bmatrix} \nu_{\mathcal{D},\mathcal{V}}(c) \geq \frac{1}{r(\lambda)} \Rightarrow \\ s'_{j,k} \neq \perp \end{bmatrix} \geq 1 - \mathsf{negl}(\lambda) \quad \forall j \in [\lambda \cdot r(\lambda)], k \in [\lambda] \tag{17}$$

We can now take the union bound of the probabilities in Equations 13 through 17. If all of the given events hold, and furthermore $\nu_{\mathcal{D},\mathcal{V}}(c) \geq \frac{1}{r(\lambda)}$ for the sampled configuration c, then:

- By Equation 13, there is a $j^* \in [\lambda \cdot r(\lambda)]$, s.t. $\mathcal{V}$'s answers to the j^*-th validation are good: $s_{j^*,1}, \ldots, s_{j^*,\lambda} \in \{g^0, \ldots, g^N\}$.
- By Equation 14, this in turn means that $\mathcal{E}_{\mathsf{ans}}$ answers to the same queries are good or $\perp$: $s'_{j^*,1}, \ldots, s'_{j^*,\lambda} \in \{\perp, g^0, \ldots, g^N\}$.
- By Equation 17, we know that none of $s'_{j^*,k}$ is $\perp$, and therefore we can infer: $s'_{j^*,1}, \ldots, s'_{j^*,\lambda} \in \{g^0, \ldots, g^N\}$.
- By Equations 16 and 15, we can infer that not only $s'_{j^*,k}$'s themselves, but also the answers to $g^{\mathbf{s}_{j,k}}$ computed as combinations of e_i's are all good. That is, for all $k \in [\lambda]$: $\prod_{i\in[N]} e_i^{\mathbf{s}_{j^*,k}[i]} \in \{g^0, \ldots, g^N\}$.

In summary, this yields:

$$\Pr\begin{bmatrix} \nu_{\mathcal{D},\mathcal{V}}(c) \geq \frac{1}{r(\lambda)} \Rightarrow \\ \exists j^* \text{ s.t. } \forall k \in [\lambda] : \prod_{i\in[N]} e_i^{\mathbf{s}_{j^*,k}[i]} \in \{g^0, \ldots, g^N\} \end{bmatrix} \geq 1 - \mathsf{negl}(\lambda) \tag{18}$$

Now we use Lemma 10, which says that for any set of fixed e_i's, if at least one of them is not within $\{g^{-N}, \ldots, g^N\}$, then w.h.p. at least one of the products $\prod_{i \in [N]} e_i^{\mathbf{s}_{j^*,k}[i]}$ will be outside of $\{g^0, \ldots, g^N\}$. More precisely:

$$\Pr\left[\begin{array}{l}\forall i \in [N] : e_i \in \{g^{-N}, \ldots, g^N\} \ \vee \\ \exists k \in [\lambda] : \prod_{i \in [N]} e_i^{\mathbf{s}_{j,k}[i]} \notin \{g^0, \ldots, g^N\}\end{array}\right] \geq 1 - \mathsf{negl}(\lambda) \qquad \forall j \in [\lambda \cdot r(\lambda)] \tag{19}$$

Taking the union bound of Equations 18 and 19, we get that w.h.p., whenever $\nu_{\mathcal{D},\mathcal{V}}(c) \geq \frac{1}{r(\lambda)}$, then $e_i \in \{g^{-N}, \ldots, g^N\}$ for all $i \in [N]$ (Equation 12). This concludes the proof of Theorem 3.

4.5 Proof of Lemma 6 (Randomization Independence)

The *randomization independence* property of $\mathcal{E}_{\mathsf{ans}}$ follows from the simple statistical observation that randomizing a query vector multiple times is identical to randomizing it a single time. Any new randomization "absorbs" the previous randomization. We capture this in the following lemma, which will be useful not only for randomization independence, but also later on.

Lemma 11 (Double Randomization). *For all $\lambda, N \in \mathbb{N}$, configurations $c = (\mathsf{pp}, d, \mathsf{st}_{\mathcal{D}}) \in \mathsf{Supp}(\mathrm{Conf}^{\mathcal{D}}(\lambda, N))$, queries $\mathfrak{p} \in \mathbb{G}^N$ and its randomizations $(\mathsf{st}^*, \mathfrak{p}^*) \in \mathsf{Supp}(\mathsf{randomize}(\mathfrak{p}))$, and unbounded adversaries $\mathcal{A}$ (with output in $\mathbb{G} \cup \{\perp\}$), the following two distributions are identical:*

$$\begin{aligned} &\{\mathsf{recover}(\mathsf{st}, \mathcal{A}(\tilde{\mathfrak{p}})) \mid (\mathsf{st}, \tilde{\mathfrak{p}}) \leftarrow \mathsf{randomize}(\mathfrak{p})\} \\ \equiv &\{\mathsf{recover}(\mathsf{st}^*, \mathsf{recover}(\mathsf{st}, \mathcal{A}(\tilde{\mathfrak{p}}^*))) \mid (\mathsf{st}, \tilde{\mathfrak{p}}^*) \leftarrow \mathsf{randomize}(\mathfrak{p}^*)\}\end{aligned}$$

Proof. Let $\mathsf{st}^* = (r^*, \alpha^*)$ and $\mathfrak{p}^* = \mathbf{h}^{r^*} \circ \mathfrak{p}^{\alpha^*}$. By unrolling $\mathsf{randomize}$ and $\mathsf{recover}$, the two distributions are equal to

$$\begin{aligned} &\left\{\left(d^{-r} \cdot \mathcal{A}(\mathbf{h}^r \circ \mathfrak{p}^\alpha)\right)^{1/\alpha} \,\middle|\, (r, \alpha) \leftarrow\!\!\$\, \mathbb{Z}_q \times \mathbb{Z}_q^*\right\} \\ \text{and} \quad &\left\{\left(d^{-r-r^*\alpha} \cdot \mathcal{A}(\mathbf{h}^{r+r^*\alpha} \circ \mathfrak{p}^{\alpha^*\alpha})\right)^{1/(\alpha^*\alpha)} \,\middle|\, (r, \alpha) \leftarrow\!\!\$\, \mathbb{Z}_q \times \mathbb{Z}_q^*\right\} .\end{aligned}$$

When substituting r by $r + r^*\alpha$, and α by $\alpha^*\alpha$ in the first distribution, the result is exactly the second one. This substitution does not have any impact on distribution, because for any fixed $r^* \in \mathbb{Z}_q$ and $\alpha^* \in \mathbb{Z}_q^*$, the following identity holds.

$$\{(r, \alpha) \mid (r, \alpha) \leftarrow\!\!\$\, \mathbb{Z}_q \times \mathbb{Z}_q^*\} \equiv \{(\underbrace{r + r^*\alpha}_{\text{over } \mathbb{Z}_q}, \underbrace{\alpha^*\alpha}_{\text{over } \mathbb{Z}_q^*}) \mid (r, \alpha) \leftarrow\!\!\$\, \mathbb{Z}_q \times \mathbb{Z}_q^*\}$$

This concludes the proof. □

Now we can prove *randomization independence* of $\mathcal{E}_{\mathsf{ans}}$, meaning that for any query $\mathfrak{p} \in \mathbb{G}^N$, and its randomization $(\mathsf{st}^*, \mathfrak{p}^*) \in \mathsf{Supp}(\mathsf{randomize}(\mathfrak{p}))$, applying $\mathcal{E}_{\mathsf{ans}}$ to $\mathfrak{p}$ yields the same outcome as applying $\mathcal{E}_{\mathsf{ans}}$ to $\tilde{\mathfrak{p}}$, and then recovering:

$$\mathcal{E}_{\mathsf{ans}}(1^r, c, \mathfrak{p}) \quad \equiv \quad \mathsf{recover}(\mathsf{st}^*, \mathcal{E}_{\mathsf{ans}}(1^r, c, \mathfrak{p}^*))$$

To see that this holds, consider the k-th iteration (where $k \in [\lambda \cdot r^3]$) of $\mathcal{E}_{\mathsf{ans}}$, and apply Lemma 11 with the following adversary.

- $\mathcal{A}(\tilde{\mathfrak{p}})$ simulates lines 3-5 of $\mathcal{E}_{\mathsf{ans}}$ by running $\tilde{p} \leftarrow \overline{\mathcal{V}}(c, \tilde{\mathfrak{p}})$, $\mathsf{st}', \tilde{\mathfrak{p}}' \leftarrow \mathsf{randomize}(\tilde{\mathfrak{p}})$, and $\tilde{p}' \leftarrow \mathsf{recover}(\mathsf{st}', \overline{\mathcal{V}}(c, \tilde{\mathfrak{p}}'))$. Then, if $\tilde{p} = \tilde{p}'$, output $\tilde{p}$. Otherwise, output $\perp$.

The k-th iteration of $\mathcal{E}_{\mathsf{ans}}$ can be rewritten as

$$\begin{aligned}&\mathsf{st}, \tilde{\mathfrak{p}} \leftarrow \mathsf{randomize}(\mathfrak{p})\\ &p \leftarrow \mathsf{recover}(\mathsf{st}, \mathcal{A}(\tilde{\mathfrak{p}}))\\ &\textbf{if } p \neq \bot, \textbf{return } p\end{aligned}$$

By Lemma 11, the behavior stays the same when replacing $\mathfrak{p}$ by $\mathfrak{p}^*$ and returning $\mathsf{recover}(\mathsf{st}^*, p)$ instead of p (since $p \neq \bot$ is equivalent to $\mathsf{recover}(\mathsf{st}^*, p) \neq \bot$). The result is exactly $\mathsf{recover}(\mathsf{st}^*, \mathcal{E}_{\mathsf{ans}}(1^r, c, \mathfrak{p}^*))$, thereby concluding the proof of $\mathcal{E}_{\mathsf{ans}}$'s *randomization independence.*

Remark 5. Note that in both *randomization independence* of $\mathcal{E}_{\mathsf{ans}}$, and *double randomization* for arbitrary adversary $\mathcal{A}$, the first randomization $(\mathsf{st}^*, \mathfrak{p}^*) \in \mathsf{Supp}(\mathsf{randomize}(\mathfrak{p}))$ does not need to be sampled by actually calling $\mathsf{randomize}(\mathfrak{p})$. There simply need to exist $r^* \in \mathbb{Z}_q$ and $\alpha^* \in \mathbb{Z}_q^*$ with $\mathsf{st}^* = (r^*, \alpha^*)$ and $\mathfrak{p}^* = \mathbf{h}^{r^*} \circ \mathfrak{p}^{\alpha^*}$.

4.6 Proof of Lemma 7 (Agrees with Small Answers)

In this section, we prove Lemma 7, i.e., that if an adversary $\mathcal{A}$ is capable of answering a query $\mathfrak{p}$ with $p^* \in \{g^0, \ldots, g^N\}$, then the value p returned by $\mathcal{E}_{\mathsf{ans}}$ on $\mathfrak{p}$ is equal to p^* (if it is not $\bot$).

To do so, we open up the definition of $\mathcal{E}_{\mathsf{ans}}$: let $\tilde{p}$ and $\tilde{p}'$ be the two answers to $\tilde{\mathfrak{p}}$ as defined in Figure 5. Furthermore, let $p := \mathsf{recover}(\mathsf{st}, \tilde{p})$ and $p' := \mathsf{recover}(\mathsf{st}, \tilde{p}')$ be the two recovered answers. Note that $\mathcal{E}_{\mathsf{ans}}$ returns p whenever $\tilde{p} = \tilde{p}'$ (or equivalently, $p = p'$, due to the bijectivity of $\mathsf{recover}(\mathsf{st}, \cdot)$).

Therefore, by a union bound over all $\lambda \cdot r(\lambda)$ iterations of $\mathcal{E}_{\mathsf{ans}}$, we just need to prove the following (which states that it is unlikely that $\mathcal{E}_{\mathsf{ans}}$ terminates in the current iteration if the answer does not match p^*):

$$\Pr\left[\begin{array}{l|l} p^* \in \{g^0, \ldots, g^N\} & c = (\mathsf{pp}, d, \mathsf{st}_{\mathcal{D}}) \leftarrow \mathrm{Conf}^{\mathcal{D}}(\lambda, N) \\ \wedge\ p = p' & \mathsf{st}_{\mathcal{A}}, \mathfrak{p} \leftarrow \mathcal{A}_1(c) \\ \wedge\ p \neq p^* & (p, p', p^*) \leftarrow \mathrm{Answers}(\mathsf{st}_{\mathcal{A}}, \mathfrak{p}) \end{array}\right] \leq \mathsf{negl}(\lambda)\ . \tag{20}$$

To remove clutter in our calculations, the three answers p, p', p^* are generated by the following procedure $\mathrm{Answers}(\lambda, N)$, which computes p and p' exactly as an iteration of $\mathcal{E}_{\mathsf{ans}}$ would, and p^* as the answer computed by $\mathcal{A}_2$.

Procedure $\mathrm{Answers}(\mathsf{st}_{\mathcal{A}}, \mathfrak{p})$

1: $p^* \leftarrow \mathsf{recover}(\mathsf{st}^*, \mathcal{A}_2(\mathsf{st}_{\mathcal{A}}, \tilde{\mathfrak{p}}^*))$, where $\mathsf{st}^*, \tilde{\mathfrak{p}}^* \leftarrow \mathsf{randomize}(\mathfrak{p})$

2: $p \leftarrow \mathsf{recover}(\mathsf{st}, \overline{\mathcal{V}}(c, \tilde{\mathfrak{p}}))$, where $\mathsf{st}, \tilde{\mathfrak{p}} \leftarrow \mathsf{randomize}(\mathfrak{p})$

3: $p' \leftarrow \mathsf{recover}(\mathsf{st}, \mathsf{recover}(\mathsf{st}', \overline{\mathcal{V}}(c, \tilde{\mathfrak{p}}')))$, where $\mathsf{st}', \tilde{\mathfrak{p}}' \leftarrow \mathsf{randomize}(\tilde{\mathfrak{p}})$

4: **return** (p, p', p^*)

To provide some intuition for the remainder of this proof, note that our generalized notion of integrity (*evasive ratio*, Lemma 4) shows that for any two *different* ways of answering a query, it is impossible to predict their ratio. However, if the probability above were not negligible, then there is (1) the option of answering with p^*, given by $\mathcal{A}_2$, and (2) the option of answering with p', given by $\overline{\mathcal{V}}$. Furthermore, it would be *easy* to predict the ratio $\frac{p'}{p^*}$: we just need to query $\overline{\mathcal{V}}$ again to get another answer p. Then, with non-negligible probability, the ratio would be in the small set $\{\frac{p}{g^0}, \ldots, \frac{p}{g^N}\}$.

To formalize this idea, note that the probability in Equation 20 is upper bounded by

$$\Pr\left[\begin{array}{l} p' \neq p^* \\ \wedge \frac{p'}{p^*} \in \left\{\frac{p}{g^0}, \ldots, \frac{p}{g^N}\right\} \end{array} \middle| \begin{array}{l} c = (\mathsf{pp}, d, \mathsf{st}_{\mathcal{D}}) \leftarrow \mathrm{Conf}^{\mathcal{D}}(\lambda, N) \\ \mathsf{st}_{\mathcal{A}}, \mathfrak{p} \leftarrow \mathcal{A}_1(c) \\ (p, p', p^*) \leftarrow \mathrm{Answers}(\mathsf{st}_{\mathcal{A}}, \mathfrak{p}) \end{array}\right] . \tag{21}$$

We face one technical difficulty preventing us from applying Lemma 4: we would need an adversary that computes two answers $\tilde{p}'$ and $\tilde{p}^*$ from the *same* randomization $\mathsf{st}^*, \tilde{\mathfrak{p}}^* \leftarrow \mathsf{randomize}(\mathfrak{p})$ of query $\mathfrak{p}$ (without knowing st^*). While $\tilde{p}^* = \mathcal{A}_2(\mathsf{st}_{\mathcal{A}}, \tilde{\mathfrak{p}}^*)$ is an answer to $\tilde{\mathfrak{p}}^*$, the value $\tilde{p}' \leftarrow \mathsf{recover}(\mathsf{st}', \overline{\mathcal{V}}(c, \tilde{\mathfrak{p}}'))$ is an answer to $\tilde{\mathfrak{p}}$, a separate randomization of $\mathfrak{p}$. We fix this issue using *double randomization* (Lemma 11) twice with adversary $\overline{\mathcal{V}}(c, \cdot)$: we may replace line 3 of $\mathrm{Answers}(\mathsf{st}_{\mathcal{A}}, \mathfrak{p})$ by

$$p' \leftarrow \mathsf{recover}(\mathsf{st}^*, \mathsf{recover}(\mathsf{st}', \overline{\mathcal{V}}(c, \tilde{\mathfrak{p}}'))), \text{ where } \mathsf{st}', \tilde{\mathfrak{p}}' \leftarrow \mathsf{randomize}(\tilde{\mathfrak{p}}^*) ,$$

to obtain the procedure $\mathrm{Answers}'(\mathsf{st}_{\mathcal{A}}, \mathfrak{p})$ which behaves identical to $\mathrm{Answers}(\mathsf{st}_{\mathcal{A}}, \mathfrak{p})$. Now, we use Lemma 4 (*evasive ratio*) with the following adversary $\mathcal{B}_1, \mathcal{B}_2$.

- $\mathcal{B}_1(c)$: run $\mathsf{st}_{\mathcal{A}}, \mathfrak{p} \leftarrow \mathcal{A}_1(c)$, and output $\mathsf{st}_{\mathcal{B}} := (\mathsf{st}_{\mathcal{A}}, c)$ and $\mathfrak{p}$.
- $\mathcal{B}_2(\mathsf{st}_{\mathcal{B}}, \tilde{\mathfrak{p}}^*)$: Compute the first answer as $\tilde{p}^* \leftarrow \mathcal{A}_2(\mathsf{st}_{\mathcal{A}}, \tilde{\mathfrak{p}}^*)$, and the second answer by running $\mathsf{st}', \tilde{\mathfrak{p}}' \leftarrow \mathsf{randomize}(\tilde{\mathfrak{p}}^*)$ and $\tilde{p}' \leftarrow \mathsf{recover}(\mathsf{st}', \overline{\mathcal{V}}(c, \tilde{\mathfrak{p}}'))$. Choose the set $S := \left\{\frac{p}{g^0}, \ldots, \frac{p}{g^N}\right\}$ by running $p \leftarrow \mathsf{recover}(\mathsf{st}, \overline{\mathcal{V}}(c, \tilde{\mathfrak{p}}))$ with $\mathsf{st}, \tilde{\mathfrak{p}} \leftarrow \mathsf{randomize}(\tilde{\mathfrak{p}})$. Output $\tilde{p}^*$, $\tilde{p}'$, S.

Lemma 4, applied to adversary $\mathcal{B}_1, \mathcal{B}_2$ shows that

$$\Pr\left[\begin{array}{l} \tilde{p}' \neq \tilde{p}^* \wedge \\ \frac{\mathsf{recover}(\mathsf{st}^*, \tilde{p}')}{\mathsf{recover}(\mathsf{st}^*, \tilde{p}^*)} \in \left\{\frac{p}{g^0}, \ldots, \frac{p}{g^N}\right\} \end{array} \middle| \begin{array}{l} c = (\mathsf{pp}, d, \mathsf{st}_{\mathcal{D}}) \leftarrow \mathrm{Conf}^{\mathcal{D}}(\lambda, N) \\ \mathsf{st}_{\mathcal{A}}, \mathfrak{p} \leftarrow \mathcal{A}_1(c) \\ \mathsf{st}^*, \tilde{\mathfrak{p}}^* \leftarrow \mathsf{randomize}(\mathfrak{p}) \\ \tilde{p}^* \leftarrow \mathcal{A}_2(\mathsf{st}_{\mathcal{A}}, \tilde{\mathfrak{p}}^*) \\ \mathsf{st}', \tilde{\mathfrak{p}}' \leftarrow \mathsf{randomize}(\tilde{\mathfrak{p}}^*) \\ \tilde{p}' \leftarrow \mathsf{recover}(\mathsf{st}', \overline{\mathcal{V}}(c, \tilde{\mathfrak{p}}')) \\ \mathsf{st}, \tilde{\mathfrak{p}} \leftarrow \mathsf{randomize}(\mathfrak{p}) \\ p \leftarrow \mathsf{recover}(\mathsf{st}, \overline{\mathcal{V}}(c, \tilde{\mathfrak{p}})) \end{array}\right] \leq \mathsf{negl}(\lambda) . \tag{22}$$

Note that for $p' := \mathsf{recover}(\mathsf{st}^*, \tilde{p}')$ and $p^* := \mathsf{recover}(\mathsf{st}^*, \tilde{p}^*)$, the equivalence

$$\tilde{p}' \neq \tilde{p}^* \quad \Leftrightarrow \quad p' \neq p^*$$

holds due to the bijectivity of $\mathsf{recover}(\mathsf{st}^*, \cdot)$. Therefore, we can observe that the probability in Equation 22 is identical to the one in Equation 21 (after replacing $\mathrm{Answers}(\mathsf{st}_{\mathcal{A}}, \mathfrak{p})$ by the equivalent $\mathrm{Answers}'(\mathsf{st}_{\mathcal{A}}, \mathfrak{p})$). This concludes the proof of Lemma 7.

4.7 Proof of Lemma 8 (Homomorphism)

In this section, we prove the *homomorphism* property of $\mathcal{E}_{\mathsf{ans}}$. Let $\mathcal{A}$ be a ppt adversary, and let $N(\lambda)$ and $r(\lambda)$ be polynomials. W.l.o.g., we may assume that $t_{i^*} \neq 0$ holds for each t_{i^*} chosen by $\mathcal{A}$ (otherwise, the adversary would only *increase* its success probability by *not* returning $\mathfrak{p}_{i^*}$ and t_{i^*} in the first place, because neither $p = \mathcal{E}_{\mathsf{ans}}(1^{r(\lambda)}, c, \prod_{i \in [m]} \mathfrak{p}_i^{t_i})$ nor $\prod_{i \in [m]} p_i^{t_i}$ would be affected by doing so.)

Consider the following hybrid experiment $\mathsf{Hyb}_{i^*}(\lambda, N)$ (for any $i^* \in \{0, \ldots, m\}$) that corresponds to the statement of Lemma 8, except that the first i^* queries have been changed from $\mathfrak{p}_i$ to $\mathbb{1}$ (where $\mathbb{1}$ is the vector consisting of N times the neutral group element 1).

$\mathsf{Hyb}_{i^*}(\lambda, N)$

$$
\begin{array}{l}
c = (\mathsf{pp}, d, \mathsf{st}_{\mathcal{D}}) \leftarrow \mathrm{Conf}^{\mathcal{D}}(\lambda, N) \\
\mathfrak{p}_1, \ldots, \mathfrak{p}_m, t_1, \ldots, t_m \leftarrow \mathcal{A}(c) \\
p \leftarrow \mathcal{E}_{\mathsf{ans}}(1^{r(\lambda)}, c, \prod_{i \in \{i^*+1, \ldots, m\}} \mathfrak{p}_i^{t_i}) \\
p_i \leftarrow \mathcal{E}_{\mathsf{ans}}(1^{r(\lambda)}, c, \mathbb{1}) \quad \forall i \in \{1, \ldots, i^*\} \\
p_i \leftarrow \mathcal{E}_{\mathsf{ans}}(1^{r(\lambda)}, c, \mathfrak{p}_i) \quad \forall i \in \{i^*+1, \ldots, m\} \\
\textbf{return } \mathbb{I}\left[p, p_1, \ldots, p_m \neq \bot \Rightarrow p = \prod_{i \in [m]} p_i^{t_i}\right]
\end{array}
$$

Note that the statement of Lemma 8 is identical to

$$\Pr[\mathsf{Hyb}_0(\lambda, N) = 1] \geq 1 - \mathsf{negl}(\lambda) \ ,$$

which is what we will show with a hybrid argument. That is, we need

$$\left|\Pr[\mathsf{Hyb}_{i^*-1}(\lambda, N) = 1] - \Pr[\mathsf{Hyb}_{i^*}(\lambda, N) = 1]\right| \leq \mathsf{negl}(\lambda) \quad \forall i \in [w] \ , \tag{23}$$

and

$$\Pr[\mathsf{Hyb}_m(\lambda, N) = 1] \geq 1 - \mathsf{negl}(\lambda) \ . \tag{24}$$

We start with proving Equation 23, by applying Lemma 5 (*chosen vector indistinguishability*) to the following ppt adversary $\mathcal{B}_1, \mathcal{B}_2$:

- $\mathcal{B}_1(c)$: Run $\mathfrak{p}_1, \ldots, \mathfrak{p}_m, t_1, \ldots, t_m \leftarrow \mathcal{A}(c)$. Output state $\mathsf{st}_{\mathcal{B}} := (c, \{\mathfrak{p}_i\}, \{t_i\})$ and vectors $\mathfrak{v}_0 := \mathfrak{p}_{i^*}^{t_{i^*}}$, $\mathfrak{v}_1 := \mathbb{1}$.
- $\mathcal{B}_2(\mathsf{st}_{\mathcal{B}}, \tilde{\mathfrak{v}})$: Compute the two answers $\tilde{p} \leftarrow \mathcal{E}_{\mathsf{ans}}(1^{r(\lambda)}, c, \tilde{\mathfrak{v}} \cdot \prod_{i \in \{i^*+1, \ldots, m\}} \mathfrak{p}_i^{t_i})$ and $\tilde{p}_{i^*} \leftarrow \mathcal{E}_{\mathsf{ans}}(1^{r(\lambda)}, c, \tilde{\mathfrak{v}})$. Further, compute answers $p_i \leftarrow \mathcal{E}_{\mathsf{ans}}(1^{r(\lambda)}, c, \mathbb{1})$ for each $i \in [i^* - 1]$ and $p_i \leftarrow \mathcal{E}_{\mathsf{ans}}(1^{r(\lambda)}, c, \mathfrak{p}_i)$ for each $i \in \{i^*+1, \ldots, m\}$. Then output

$$\mathbb{I}\left[\tilde{p}, p_1, \ldots, p_{i^*-1}, \tilde{p}_{i^*}, p_{i^*+1}, \ldots, p_m \neq \bot \ \Rightarrow \ \tilde{p} = \tilde{p}_{i^*} \cdot \prod_{i \in [m] \setminus \{i^*\}} p_i^{t_i}\right] \ .$$

Note that the chosen-vector experiment $\mathrm{CVA}^{\mathcal{D}, \mathcal{B}_1, \mathcal{B}_2}(\lambda, N, 0)$ for bit $b = 0$ (as defined in Lemma 5) is exactly the following:

Game $\mathrm{CVA}^{\mathcal{D}, \mathcal{B}_1, \mathcal{B}_2}(\lambda, N, 0)$

$$
\begin{array}{l}
c = (\mathsf{pp}, d, \mathsf{st}_{\mathcal{D}}) \leftarrow \mathrm{Conf}^{\mathcal{D}}(\lambda, N) \\
\mathfrak{p}_1, \ldots, \mathfrak{p}_m, t_1, \ldots, t_m \leftarrow \mathcal{A}(c) \\
r \leftarrow\!\!\$\ \mathbb{Z}_q \\
\tilde{p} \leftarrow \mathcal{E}_{\mathsf{ans}}(1^{r(\lambda)}, c, \mathbf{h}^r \circ \prod_{i \in \{i^*, \ldots, m\}} \mathfrak{p}_i^{t_i}) \\
p_i \leftarrow \mathcal{E}_{\mathsf{ans}}(1^{r(\lambda)}, c, \mathbb{1}) \quad \forall i \in \{1, \ldots, i^* - 1\} \\
\tilde{p}_{i^*} \leftarrow \mathcal{E}_{\mathsf{ans}}(1^{r(\lambda)}, c, \mathbf{h}^r \circ \mathfrak{p}_{i^*}^{t_{i^*}}) \\
p_i \leftarrow \mathcal{E}_{\mathsf{ans}}(1^{r(\lambda)}, c, \mathfrak{p}_i) \quad \forall i \in \{i^*+1, \ldots, m\} \\
\textbf{return } \mathbb{I}\left[\tilde{p}, p_1, \ldots, p_{i^*-1}, \tilde{p}_{i^*}, p_{i^*+1}, \ldots, p_m \neq \bot \ \Rightarrow \ \tilde{p} = \tilde{p}_{i^*} \cdot \prod_{i \in [m] \setminus \{i^*\}} p_i^{t_i}\right]
\end{array}
$$

By *randomization independence* (Lemma 6) (which we can apply due to $t_{i^*} \neq 0$), the following two equivalences hold:

$$\tilde{p} = \mathcal{E}_{\mathsf{ans}}(1^{r(\lambda)}, c, \mathbf{h}^r \circ \prod_{i \in [m']} \mathfrak{p}_i^{t_i}) \ \equiv \ d^r \cdot \mathcal{E}_{\mathsf{ans}}(1^{r(\lambda)}, c, \prod_{i \in [m']} \mathfrak{p}_i^{t_i}) \ ,$$
$$\tilde{p}_{i^*} = \mathcal{E}_{\mathsf{ans}}(1^{r(\lambda)}, c, \mathbf{h}^r \circ \mathfrak{p}_{i^*}^{t_{i^*}}) \ \equiv \ d^r \cdot (\mathcal{E}_{\mathsf{ans}}(1^{r(\lambda)}, c, \mathfrak{p}_{i^*}))^{t_{i^*}} \ .$$

Thus, without affecting the distribution, in $\mathrm{CVA}^{\mathcal{D},\mathcal{B}_1,\mathcal{B}_2}(\lambda, N, 0)$ we can replace line 4 by $p \leftarrow \mathcal{E}_{\mathsf{ans}}(1^{r(\lambda)}, c, \prod_{i\in\{i^*,\ldots,m\}} \mathfrak{p}_i^{t_i})$, line 6 by $p_{i^*} \leftarrow \mathcal{E}_{\mathsf{ans}}(1^{r(\lambda)}, c, \mathfrak{p}_{i^*})$, and the return value by

$$\mathbb{I}\left[d^r \cdot p, p_1, \ldots, p_{i^*-1}, d^r \cdot p_{i^*}^{t_{i^*}}, p_{i^*+1}, \ldots, p_m \neq \bot \;\Rightarrow\; d^r \cdot p = d^r \cdot \textstyle\prod_{i\in[m]} p_i^{t_i}\right] .$$

We note that $d^r \cdot p = \bot \Leftrightarrow p = \bot$ and $d^r \cdot p_{i^*}^{t_{i^*}} \Leftrightarrow p_{i^*} = \bot$, and that the factor d^r cancels out on both sides in the equality test. The result is identical to $\mathsf{Hyb}_{i^*-1}(\lambda, N)$, i.e., $\mathrm{CVA}^{\mathcal{D},\mathcal{B}_1,\mathcal{B}_2}(\lambda, N, 0) \equiv \mathsf{Hyb}_{i^*-1}(\lambda, N)$. Analogously, we also get $\mathrm{CVA}^{\mathcal{D},\mathcal{B}_1,\mathcal{B}_2}(\lambda, N, 1) \equiv \mathsf{Hyb}_{i^*}(\lambda, N)$. Therefore, Equation 23 follows from Lemma 5.

It remains to prove Equation 24. Note that in $\mathsf{Hyb}_m(\lambda, N)$, $\mathcal{E}_{\mathsf{ans}}$ is only ever called on query $\mathbb{1}$. If all those calls to $\mathcal{E}_{\mathsf{ans}}$ return $\bot$ or 1, then $\mathsf{Hyb}_m(\lambda, N)$ will output 1. Indeed, we do have such a property, captured by the following lemma. Applying it to all $m+1$ calls to $\mathcal{E}_{\mathsf{ans}}(1^{r(\lambda)}, c, \mathbb{1})$ in $\mathsf{Hyb}_m(\lambda, N)$ concludes the proof of *homomorphism*.

Lemma 12. *For any $\lambda \in \mathbb{N}$ and polynomials $N(\lambda) \leq \mathrm{poly}(\lambda)$ and $r(\lambda) \leq \mathrm{poly}(\lambda)$:*

$$\Pr\left[\mathcal{E}_{\mathsf{ans}}(1^{r(\lambda)}, c, \mathbb{1}) \in \{1, \bot\} \mid c \leftarrow \mathrm{Conf}^{\mathcal{D}}(\lambda, N)\right] \geq 1 - \mathsf{negl}(\lambda)$$

Proof. We show that in each of the $\lceil \lambda \cdot (r(\lambda))^3 \rceil$ iterations of $\mathcal{E}_{\mathsf{ans}}(1^{r(\lambda)}, c, \mathbb{1})$, the probability of returning anything $\neq 1$ is negligible. The full lemma then follows by a union bound.

By unrolling $\mathsf{randomize}$ and $\mathsf{recover}$, we can rewrite a single iteration of $\mathcal{E}_{\mathsf{ans}}$ as on the left-hand side in the following equation.

$$\begin{array}{l} r, r' \leftarrow\!\!\$\; \mathbb{Z}_q \text{ and } \alpha, \alpha' \leftarrow\!\!\$\; \mathbb{Z}_q^* \\ \tilde{p} \leftarrow \overline{\mathcal{V}}(c, \mathbf{h}^r) \\ \tilde{p}' \leftarrow (d^{-r'} \cdot \overline{\mathcal{V}}(c, \mathbf{h}^{r'+r\alpha'}))^{1/\alpha'} \\ \textbf{if } \tilde{p} = \tilde{p}', \textbf{return } (d^{-r} \cdot \tilde{p})^{1/\alpha} \end{array} \quad \equiv \quad \begin{array}{l} r, r' \leftarrow\!\!\$\; \mathbb{Z}_q \text{ and } \alpha, \alpha' \leftarrow\!\!\$\; \mathbb{Z}_q^* \\ \tilde{p} \leftarrow \overline{\mathcal{V}}(c, \mathbf{h}^r) \\ \tilde{p}' \leftarrow \overline{\mathcal{V}}(c, \mathbf{h}^{r'}) \\ \textbf{if } d^{-r} \cdot \tilde{p} = (d^{-r'} \cdot \tilde{p}')^{1/\alpha'}, \textbf{return } (d^{-r} \cdot \tilde{p})^{1/\alpha} \end{array}$$

By replacing r' with $(r' - r\alpha') \bmod q$ (which are identically distributed for $r' \leftarrow\!\!\$\; \mathbb{Z}_q$), we obtain the right-hand side. Note that α' is *only* used for checking the the final **if**-condition. For any $x, y \in \mathbb{G}$ with $x \neq 1$, it is easy to see that

$$\Pr\left[x = y^{1/\alpha'} \;\middle|\; \alpha' \leftarrow\!\!\$\; \mathbb{Z}_q^*\right] \leq \frac{1}{q-1} \leq \mathsf{negl}(\lambda) .$$

Thus, plugging in $x = d^{-r} \cdot \tilde{p}$ and $y = d^{-r'} \cdot \tilde{p}'$ shows that the probability of $\mathcal{E}_{\mathsf{ans}}$ returning a value $\neq 1$ is negligible. □

4.8 Proof of Lemma 9 (Answer Guarantee)

In this section, we prove that $\mathcal{E}_{\mathsf{ans}}$ is guaranteed to answer $\neq \bot$ to any query, as long as the runtime parameter 1^r is large enough:

$$\Pr\left[\begin{array}{l} \nu_{\mathcal{D},\mathcal{V}}(c) \geq \frac{1}{r(\lambda)} \\ \Rightarrow \mathcal{E}_{\mathsf{ans}}(1^{r(\lambda)}, c, \mathfrak{p}) \neq \bot \end{array} \;\middle|\; \begin{array}{c} c = (\mathsf{pp}, d, \mathsf{st}_{\mathcal{D}}) \leftarrow \mathrm{Conf}^{\mathcal{D}}(\lambda, N) \\ \mathfrak{p} \leftarrow \mathcal{A}(c) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda) . \tag{25}$$

In our proof, we want to apply *chosen vector indistinguishability* (Lemma 5) in order to switch between an honest validation query $g^{\mathbf{s}_1}$ and the query $\mathfrak{p}$ that $\mathcal{E}_{\mathsf{ans}}$ wants to answer. However, we have no way of computing $\nu_{\mathcal{D},\mathcal{V}}(c)$ exactly. Therefore, we will only *approximate* this value with the following Lemma, which is going to be sufficient.

Lemma 13. *There exists a ppt estimator $\mathcal{C}$ of $\nu_{\mathcal{D},\mathcal{V}}$ and a negligible function $\mathsf{negl}(\lambda)$, s.t. for any λ, N, and configurations $c \in \mathsf{Supp}(\mathrm{Conf}^{\mathcal{D}}(\lambda, N))$:*

$$\Pr\left[|\mathcal{C}(1^{r(\lambda)}, c) - \nu_{\mathcal{D},\mathcal{V}}(c)| > \frac{1}{r(\lambda)}\right] < \mathsf{negl}(\lambda) \ .$$

Proof (of Lemma 13). Our $\mathcal{C}(1^{r(\lambda)}, c)$ will compute an estimation $\tilde{\nu}$ of $\nu_{\mathcal{D},\mathcal{V}}(c)$ by repeatedly querying $\mathcal{V}$ on random validations for $q(\lambda)$ times, and averaging the number of successful runs ($q(\lambda)$ is a polynomial to be specified later):

- **Input**: $1^r, c = (\mathsf{pp}, d, \mathsf{st}_{\mathcal{D}})$
- For each $j \in [q(\lambda)]$, check if $\mathcal{V}$ would pass a random validation:
  - $\mathsf{st}, v \leftarrow \mathsf{VQuery}()$
  - $a_{\mathsf{v}} \leftarrow \mathcal{V}(c, v)$
  - $b_j \leftarrow \mathbb{I}\{\mathsf{VCheck}(\mathsf{st}, d, a_{\mathsf{v}}) = \top\}$
- Output $\tilde{\nu} := \frac{b_1 + \cdots + b_{q(\lambda)}}{q(\lambda)}$

For any fixed c, we now estimate the probability of $\tilde{\nu}$ (as generated by $\mathcal{C}$) being far from the real $\nu_{\mathcal{D},\mathcal{V}}(c)$. Using Hoeffding's inequality, we receive for any $\varepsilon > 0$:

$$\Pr\left[|\tilde{\nu} - \nu_{\mathcal{D},\mathcal{V}}(c)| > \varepsilon\right] \leq 2e^{-2(\varepsilon \cdot q(\lambda))^2} \ ,$$

and therefore with $\varepsilon := \frac{1}{r(\lambda)}$ and $q(\lambda) := \lambda \cdot r(\lambda)$:

$$\Pr\left[|\tilde{\nu} - \nu_{\mathcal{D},\mathcal{V}}(c)| > \frac{1}{r(\lambda)}\right] \leq \mathsf{negl}(\lambda) \ .$$

$\square$

Now we are ready to prove Lemma 9. Let $\mathcal{A}$ be an adversary (who generates a query $\mathfrak{p}$), and let $N(\lambda)$ and $r(\lambda)$ be polynomials.

Our starting point is the following distribution $\mathsf{Hyb}_0(\lambda, N)$, for which we can easily prove that it returns 1 with overwhelming probability. We will transform $\mathsf{Hyb}_0(\lambda, N)$ into hybrid $\mathsf{Hyb}_6(\lambda, N)$ that resembles Equation 25, which then holds by the hybrid argument.

$\mathsf{Hyb}_0(\lambda, N)$

1 : $c \leftarrow \mathrm{Conf}^{\mathcal{D}}(\lambda, N)$
2 : **if** $\nu_{\mathcal{D},\mathcal{A}}(c) < \frac{1}{2r(\lambda)}$, **return** 1
3 : **repeat** $\lambda \cdot (r(\lambda))^3$ **times**
4 : $\quad \mathbf{s}_1 \leftarrow\$ \{0,1\}^N$
5 : $\quad \mathsf{st}, \tilde{\mathfrak{p}} \leftarrow \mathsf{randomize}(g^{\mathbf{s}_1})$
6 : $\quad p \leftarrow \mathsf{recover}(\mathsf{st}, \overline{\mathcal{V}}(c, \tilde{\mathfrak{p}}))$
7 : $\quad \mathsf{st}', \tilde{\mathfrak{p}}' \leftarrow \mathsf{randomize}(g^{\mathbf{s}_1})$
8 : $\quad p' \leftarrow \mathsf{recover}(\mathsf{st}', \overline{\mathcal{V}}(c, \tilde{\mathfrak{p}}'))$
9 : $\quad$ **if** $p, p' \in \{g^0, \ldots, g^N\}$, **return** 1
10 : **return** 0

- Unrolling the definition of $\overline{\mathcal{V}}$, note that lines 4-8 of $\mathsf{Hyb}_0(\lambda, N)$ run $\mathcal{V}$ on two independently generated validations, *except that they use the same selection vector* $\mathbf{s}_1$. We return 1 whenever both validations succeed. (In fact, the condition $p, p' \in \{g^0, \ldots, g^N\}$ is even more

relaxed, because only the *first out of* λ *queries* need to be within the set $\{g^0, \dots, g^N\}$. The "real" VCheck would require this for *all* λ queries.)
Let $\gamma(c, \mathbf{s}_1)$ be the probability that $\mathcal{V}$ passes a single validation with fixed configuration c and fixed first selection vector $\mathbf{s}_1$. Let c be any configuration with $\nu_{\mathcal{D},\mathcal{A}}(c) \geq \frac{1}{2r(\lambda)}$. Then, it is clear that

$$\Pr\left[\gamma(c, \mathbf{s}_1) \geq \frac{1}{4r(\lambda)} \,\middle|\, \mathbf{s}_1 \leftarrow\!\$\ \{0,1\}^N\right] \geq \frac{1}{4r(\lambda)} \ ,$$

i.e., with probability $\frac{1}{4r(\lambda)}$, the choice of $\mathbf{s}_1$ was "good". Note that whenever $\gamma(c, \mathbf{s}_1) \geq \frac{1}{4r(\lambda)}$, then the probability of Hyb_0 returning 1 in this iteration is at least $\frac{1}{16(r(\lambda))^2}$. Thus, in each iteration, the probability of returning 1 is at least $\frac{1}{64(r(\lambda))^3}$.
In conclusion, for a c with $\nu_{\mathcal{D},\mathcal{A}}(c) \geq \frac{1}{2r(\lambda)}$, 0 is only returned with probability

$$\leq \left(1 - \frac{1}{64(r(\lambda))^3}\right)^{\lambda \cdot (r(\lambda))^3} \leq e^{-\frac{1}{64(r(\lambda))^3} \cdot \lambda \cdot (r(\lambda))^3} = e^{-\lambda/64} \leq \mathsf{negl}(\lambda) \ .$$

Furthermore, for any c with $\nu_{\mathcal{D},\mathcal{A}}(c) < \frac{1}{2r(\lambda)}$, 0 is returned with probability 0, and therefore

$$\Pr[\mathsf{Hyb}_0(\lambda, N) = 1] \geq 1 - \mathsf{negl}(\lambda) \ .$$

– In $\mathsf{Hyb}_0(\lambda, N)$, replace the lines

$$\begin{array}{lcl} \mathsf{st}', \tilde{\mathfrak{p}}' \leftarrow \mathsf{randomize}(g^{\mathbf{s}_1}) & & \mathsf{st}', \tilde{\mathfrak{p}}' \leftarrow \mathsf{randomize}(\tilde{\mathfrak{p}}) \\ p' \leftarrow \mathsf{recover}(\mathsf{st}', \overline{\mathcal{V}}(c, \tilde{\mathfrak{p}}')) & \text{by} & p' \leftarrow \mathsf{recover}(\mathsf{st}, \mathsf{recover}(\mathsf{st}', \overline{\mathcal{V}}(c, \tilde{\mathfrak{p}}'))) \end{array}$$

to obtain Hyb_1. By Lemma 11 (*double randomization*) with adversary $\overline{\mathcal{V}}(c, \cdot)$, we have $\mathsf{Hyb}_1 \equiv \mathsf{Hyb}_0$.
– In $\mathsf{Hyb}_1(\lambda, N)$, replace the line

$$\textbf{if } p, p' \in \{g^0, \dots, g^N\}, \textbf{return } 1 \quad \text{by} \quad \textbf{if } \frac{p'}{p} \in \{g^{-N}, \dots, g^N\}, \textbf{return } 1 \ ,$$

and call the new distribution $\mathsf{Hyb}_2(\lambda, N)$.
Note that this modification relaxes the condition for returning 1. As $p, p' \in \{g^0, \dots, g^N\}$ implies $\frac{p'}{p} \in \{g^{-N}, \dots, g^N\}$, we get

$$\Pr[\mathsf{Hyb}_2(\lambda, N) = 1] \geq \Pr[\mathsf{Hyb}_1(\lambda, N) = 1] \ .$$

– Now, we are going to use Lemma 4 (*evasive ratio*) to show that the success condition $\frac{p'}{p} \in \{g^{-N}, \dots, g^N\}$ can be simplified to $\tilde{p}' = \tilde{p}$.
Let $\mathsf{Hyb}_3(\lambda, N)$ be the same as $\mathsf{Hyb}_2(\lambda, N)$, except that we replace

$$\begin{array}{rcl} p \leftarrow \mathsf{recover}(\mathsf{st}, \overline{\mathcal{V}}(c, \tilde{\mathfrak{p}})) & \text{by} & \tilde{p} \leftarrow \overline{\mathcal{V}}(c, \tilde{\mathfrak{p}}) \ , \\ p' \leftarrow \mathsf{recover}(\mathsf{st}, \mathsf{recover}(\mathsf{st}', \overline{\mathcal{V}}(c, \tilde{\mathfrak{p}}'))) & \text{by} & \tilde{p}' \leftarrow \mathsf{recover}(\mathsf{st}', \overline{\mathcal{V}}(c, \tilde{\mathfrak{p}}')) \ , \text{ and} \\ \textbf{if } \frac{p'}{p} \in \{g^{-N}, \dots, g^N\}, \textbf{return } 1 & \text{by} & \textbf{if } \tilde{p} = \tilde{p}', \textbf{return } 1 \ . \end{array}$$

We can construct an adversary $\mathcal{B}_1, \mathcal{B}_2$ for Lemma 4 in the following way:
- $\mathcal{B}_1(c)$ samples $\mathbf{s}_1 \leftarrow\!\$\ \{0,1\}^N$ and outputs state $\mathsf{st}_{\mathcal{B}} := c$ and $\mathfrak{p} := g^{\mathbf{s}_1}$.

- $\mathcal{B}_2(\mathsf{st}_\mathcal{B}, \tilde{\mathfrak{p}})$ computes the first answer as $\tilde{p} \leftarrow \overline{\mathcal{V}}(c, \tilde{\mathfrak{p}})$ and the second answer as $\tilde{p}' \leftarrow \mathsf{recover}(\mathsf{st}', \overline{\mathcal{V}}(c, \tilde{\mathfrak{p}}'))$ with $\mathsf{st}', \tilde{\mathfrak{p}}' \leftarrow \mathsf{randomize}(\tilde{\mathfrak{p}})$, chooses $S := \{g^{-N}, \dots, g^N\}$, and outputs $(\tilde{a}, \tilde{a}', S)$.

Applying Lemma 4 with the adversary above to the k-th iteration (for each $k \in [\lambda \cdot (r(\lambda))^3]$), shows that

$$\Pr\left[\tilde{p} \neq \tilde{p}' \wedge \frac{p'}{p} \in \{g^{-N}, \dots, g^N\}\right] \leq \mathsf{negl}(\lambda),$$

where this probability is taken over the distribution $\mathsf{Hyb}_3(\lambda, N)$, and $\tilde{p}, \tilde{p}'$ denote the values computed in the k-th iteration, and $p = \mathsf{recover}(\mathsf{st}, \tilde{p})$, $p' = \mathsf{recover}(\mathsf{st}, \tilde{p}')$. Therefore:

$$\Pr[\mathsf{Hyb}_3(\lambda, N) = 1] \geq \Pr[\mathsf{Hyb}_2(\lambda, N) = 1] - \mathsf{negl}(\lambda) \; .$$

– Now we switch from $\nu_{\mathcal{D},\mathcal{A}}(c)$ to its estimation. That is, in $\mathsf{Hyb}_3(\lambda, N)$, replace

$$\textbf{if } \nu_{\mathcal{D},\mathcal{A}}(c) < \frac{1}{2r(\lambda)}, \textbf{return } 1 \quad \text{by} \quad \textbf{if } \mathcal{C}(1^{4r(\lambda)}, c) < \frac{3}{4r(\lambda)}, \textbf{return } 1 \; ,$$

and call the new distribution $\mathsf{Hyb}_4(\lambda, N)$. Note that by Lemma 13, for any configuration c with $\nu_{\mathcal{D},\mathcal{A}}(c) < \frac{1}{2r(\lambda)}$, the probability of $\mathcal{C}(1^{4r(\lambda)}, c) \geq \frac{3}{4r(\lambda)}$ is negligible, i.e., $\mathsf{Hyb}_4(\lambda, N)$ is likely to return 1. For any other c, the probability of returning 1 may only increase by making this change. Thus,

$$\Pr[\mathsf{Hyb}_4(\lambda, N) = 1] \geq \Pr[\mathsf{Hyb}_3(\lambda, N) = 1] - \mathsf{negl}(\lambda) \; .$$

– Now, we replace the random query g^{s_1} by a randomization of the actual query $\mathfrak{p}$ that $\mathcal{E}_{\mathsf{ans}}$ wants to answer.
Let $\mathsf{Hyb}_{5,k}(\lambda, N)$ be the same as $\mathsf{Hyb}_4(\lambda, N)$, except that we add the line

$$\mathfrak{p} \leftarrow \mathcal{A}(c)$$

right after generating c, and in the first k iterations of the loop we replace

$$\begin{array}{l} \mathsf{s}_1 \leftarrow\!\!\$ \{0,1\}^N \\ \mathsf{st}, \tilde{\mathfrak{p}} \leftarrow \mathsf{randomize}(g^{\mathsf{s}_1}) \end{array} \quad \text{by} \quad \mathsf{st}, \tilde{\mathfrak{p}} \leftarrow \mathsf{randomize}(\mathfrak{p}) \; .$$

We apply *chosen vector indistinguishability* (Lemma 5) to the following adversary $\mathcal{B}_1, \mathcal{B}_2$.
- $\mathcal{B}_1(c)$ runs $\mathfrak{p} \leftarrow \mathcal{A}(c)$, samples $\alpha \leftarrow\!\!\$ \mathbb{Z}_q^*$, and returns $\mathsf{st}_\mathcal{B} := c$, $\mathfrak{p}^\alpha$.
- $\mathcal{B}_2(\mathsf{st}_\mathcal{B}, \tilde{\mathfrak{p}})$ behaves like $\mathsf{Hyb}_{4,k-1}$, except that it does not execute the first two steps that sample c and $\mathfrak{p}$, and in the k-th iteration, it uses $\tilde{\mathfrak{p}}$ directly instead of computing it as a randomization of g^{s_1}.

Note that the distribution $\mathrm{CVA}^{\mathcal{D},\mathcal{B}_1,\mathcal{B}_2}(\lambda, N, 0)$ is identical to $\mathsf{Hyb}_{5,k-1}$, and the distribution $\mathrm{CVA}^{\mathcal{D},\mathcal{B}_1,\mathcal{B}_2}(\lambda, N, 1)$ is identical to $\mathsf{Hyb}_{5,k}$. Thus, by Lemma 5, the following holds for all $k \geq 1$:

$$\left|\Pr[\mathsf{Hyb}_{5,k}(\lambda, N) = 1] - \Pr[\mathsf{Hyb}_{5,k-1}(\lambda, N) = 1]\right| \leq \mathsf{negl}(\lambda)$$

Let $\mathsf{Hyb}_5(\lambda, N) = \mathsf{Hyb}_{5,\lambda\cdot(r(\lambda))^3}(\lambda, N)$ denote the hybrid in which *all* loop iterations have been modified. Then,

$$|\Pr[\mathsf{Hyb}_5(\lambda, N) = 1] - \Pr[\mathsf{Hyb}_4(\lambda, N) = 1]| \leq \mathsf{negl}(\lambda) \; .$$

– In $\mathsf{Hyb}_5(\lambda, N)$, replace the line

$$\textbf{if } \mathcal{C}(1^{4r(\lambda)}, c) < \frac{3}{4r(\lambda)}, \textbf{return } 1 \quad \text{by} \quad \textbf{if } \nu_{\mathcal{D},\mathcal{A}}(c) < \frac{1}{r(\lambda)}, \textbf{return } 1 \ ,$$

and call the new distribution $\mathsf{Hyb}_6(\lambda, N)$. We use Lemma 13 again, which shows that

$$\Pr[\mathsf{Hyb}_6(\lambda, N) = 1] \geq \Pr[\mathsf{Hyb}_5(\lambda, N) = 1] - \mathsf{negl}(\lambda) \ .$$

Finally, by the hybrid argument, we have shown that

$$\Pr[\mathsf{Hyb}_6(\lambda, N) = 1] \geq 1 - \mathsf{negl}(\lambda) \ .$$

Furthermore, note that

$$\Pr[\mathsf{Hyb}_6(\lambda, N) = 1] = \Pr\left[\begin{matrix} \nu_{\mathcal{D},\mathcal{A}}(c) \geq \frac{1}{r(\lambda)} \\ \Rightarrow \mathcal{E}_{\mathsf{ans}}(1^{r(\lambda)}, c, \mathfrak{p}) \neq \bot \end{matrix} \middle| \begin{matrix} c = (\mathsf{pp}, d, \mathsf{st}_{\mathcal{D}}) \leftarrow \mathrm{Conf}^{\mathcal{D}}(\lambda, N) \\ \mathfrak{p} \leftarrow \mathcal{A}(c) \end{matrix}\right] \ ,$$

which concludes the proof of Lemma 9.

4.9 Proof of Lemma 10 (Random Subset Testing)

Note that, since all $\mathbf{s}_k$ are sampled independently of each other, it suffices to show the following:

$$\Pr_{\mathbf{s} \leftarrow \$ \{0,1\}^N}\left[\prod_{i\in[N]} e_i^{\mathbf{s}[i]} \in \{g^0, \ldots, g^N\}\right] \leq \frac{1}{2} \ .$$

Fix $\mathbf{s}[i] \in \{0, 1\}$ for all $i \in [N] \setminus \{i^*\}$. It suffices to show that

$$\Pr_{\mathbf{s}[i^*] \leftarrow \$ \{0,1\}}\left[e_{i^*}^{\mathbf{s}[i^*]} \cdot \prod_{i\in[N]\setminus\{i^*\}} e_i^{\mathbf{s}[i]} \in \{g^0, \ldots, g^N\}\right] \leq \frac{1}{2} \ ,$$

i.e., at most one of

$$\prod_{i\in[N]\setminus\{i^*\}} e_i^{\mathbf{s}[i]} \quad \text{and} \quad e_{i^*} \cdot \prod_{i\in[N]\setminus\{i^*\}} e_i^{\mathbf{s}[i]}$$

is in $\{g^0, \ldots, g^N\}$. Note that if this is the case for the left term, then the right term will be equal to $e_{i^*} \cdot g^m$ for some $m \in \{0, \ldots, N\}$.

Now assume that, additionally, $e_{i^*} \cdot g^m \in \{g^0, \ldots, g^N\}$. This implies that $e_{i^*} = g^{m'}$ for some $m' \in \{-N, \ldots, N\}$, which contradicts this lemma's assumptions. □

Acknowledgments

We thank Chenzhi Zhu for his involvement in the initial stage of this project. This research was partially supported by NSF grants CNS-2026774, CNS-2154174, a JP Morgan Faculty Award, a CISCO Faculty Award, and a gift from Microsoft.

References

1. Aguilar Melchor, C., Barrier, J., Fousse, L., Killijian, M.O.: XPIR: Private information retrieval for everyone. PoPETs **2016**(2), 155–174 (Apr 2016)
2. Ali, A., Lepoint, T., Patel, S., Raykova, M., Schoppmann, P., Seth, K., Yeo, K.: Communication-computation trade-offs in PIR. In: Bailey, M., Greenstadt, R. (eds.) USENIX Security 2021. pp. 1811–1828. USENIX Association (Aug 2021)
3. Angel, S., Chen, H., Laine, K., Setty, S.T.V.: PIR with compressed queries and amortized query processing. In: 2018 IEEE Symposium on Security and Privacy. pp. 962–979. IEEE Computer Society Press (May 2018). https://doi.org/10.1109/SP.2018.00062
4. Angel, S., Setty, S.T.V.: Unobservable communication over fully untrusted infrastructure. In: Keeton, K., Roscoe, T. (eds.) 12th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2016, Savannah, GA, USA, November 2-4, 2016. pp. 551–569. USENIX Association (2016), https://www.usenix.org/conference/osdi16/technical-sessions/presentation/angel
5. Barak, B., Goldreich, O.: Universal arguments and their applications. SIAM J. Comput. **38**(5), 1661–1694 (2008). https://doi.org/10.1137/070709244, https://doi.org/10.1137/070709244
6. Beimel, A., Stahl, Y.: Robust information-theoretic private information retrieval. Journal of Cryptology **20**(3), 295–321 (Jul 2007). https://doi.org/10.1007/s00145-007-0424-2
7. Ben-David, S., Kalai, Y.T., Paneth, O.: Verifiable private information retrieval. In: Kiltz, E., Vaikuntanathan, V. (eds.) TCC 2022, Part III. LNCS, vol. 13749, pp. 3–32. Springer, Heidelberg (Nov 2022). https://doi.org/10.1007/978-3-031-22368-6_1
8. Bleichenbacher, D.: Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1. In: Krawczyk, H. (ed.) CRYPTO'98. LNCS, vol. 1462, pp. 1–12. Springer, Heidelberg (Aug 1998). https://doi.org/10.1007/BFb0055716
9. Boyle, E., Ishai, Y., Pass, R., Wootters, M.: Can we access a database both locally and privately? In: Kalai, Y., Reyzin, L. (eds.) TCC 2017, Part II. LNCS, vol. 10678, pp. 662–693. Springer, Heidelberg (Nov 2017). https://doi.org/10.1007/978-3-319-70503-3_22
10. Cachin, C., Micali, S., Stadler, M.: Computationally private information retrieval with polylogarithmic communication. In: Stern, J. (ed.) EUROCRYPT'99. LNCS, vol. 1592, pp. 402–414. Springer, Heidelberg (May 1999). https://doi.org/10.1007/3-540-48910-X_28
11. Camenisch, J., Neven, G., shelat, a.: Simulatable adaptive oblivious transfer. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 573–590. Springer, Heidelberg (May 2007). https://doi.org/10.1007/978-3-540-72540-4_33
12. Canetti, R., Holmgren, J., Richelson, S.: Towards doubly efficient private information retrieval. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017, Part II. LNCS, vol. 10678, pp. 694–726. Springer, Heidelberg (Nov 2017). https://doi.org/10.1007/978-3-319-70503-3_23
13. Cheng, R., Scott, W., Masserova, E., Zhang, I., Goyal, V., Anderson, T.E., Krishnamurthy, A., Parno, B.: Talek: Private group messaging with hidden access patterns. In: ACSAC '20: Annual Computer Security Applications Conference, Virtual Event / Austin, TX, USA, 7-11 December, 2020. pp. 84–99. ACM (2020). https://doi.org/10.1145/3427228.3427231, https://doi.org/10.1145/3427228.3427231
14. Chor, B., Goldreich, O., Kushilevitz, E., Sudan, M.: Private information retrieval. In: 36th FOCS. pp. 41–50. IEEE Computer Society Press (Oct 1995). https://doi.org/10.1109/SFCS.1995.492461
15. Colombo, S., Nikitin, K., Corrigan-Gibbs, H., Wu, D.J., Ford, B.: Authenticated private information retrieval. In: Calandrino, J.A., Troncoso, C. (eds.) 32nd USENIX Security Symposium, USENIX Security 2023, Anaheim, CA, USA, August 9-11, 2023. USENIX Association (2023), https://www.usenix.org/conference/usenixsecurity23/presentation/colombo
16. Davidson, A., Pestana, G., Celi, S.: FrodoPIR: Simple, scalable, single-server private information retrieval. PoPETs **2023**(1), 365–383 (Jan 2023). https://doi.org/10.56553/popets-2023-0022
17. Devet, C., Goldberg, I., Heninger, N.: Optimally robust private information retrieval. In: Kohno, T. (ed.) USENIX Security 2012. pp. 269–283. USENIX Association (Aug 2012)
18. Dong, C., Chen, L.: A fast single server private information retrieval protocol with low communication cost. In: Kutylowski, M., Vaidya, J. (eds.) ESORICS 2014, Part I. LNCS, vol. 8712, pp. 380–399. Springer, Heidelberg (Sep 2014). https://doi.org/10.1007/978-3-319-11203-9_22
19. Doröz, Y., Sunar, B., Hammouri, G.: Bandwidth efficient PIR from NTRU. In: Böhme, R., Brenner, M., Moore, T., Smith, M. (eds.) FC 2014 Workshops. LNCS, vol. 8438, pp. 195–207. Springer, Heidelberg (Mar 2014). https://doi.org/10.1007/978-3-662-44774-1_16
20. Eriguchi, R., Kurosawa, K., Nuida, K.: Multi-server PIR with full error detection and limited error correction. Cryptology ePrint Archive, Report 2022/500 (2022), https://eprint.iacr.org/2022/500
21. Gentry, C., Halevi, S.: Compressible FHE with applications to PIR. In: Hofheinz, D., Rosen, A. (eds.) TCC 2019, Part II. LNCS, vol. 11892, pp. 438–464. Springer, Heidelberg (Dec 2019). https://doi.org/10.1007/978-3-030-36033-7_17

22. Gentry, C., Wichs, D.: Separating succinct non-interactive arguments from all falsifiable assumptions. In: Fortnow, L., Vadhan, S.P. (eds.) 43rd ACM STOC. pp. 99–108. ACM Press (Jun 2011). https://doi.org/10.1145/1993636.1993651
23. Goldberg, I.: Improving the robustness of private information retrieval. In: 2007 IEEE Symposium on Security and Privacy. pp. 131–148. IEEE Computer Society Press (May 2007). https://doi.org/10.1109/SP.2007.23
24. Green, M., Ladd, W., Miers, I.: A protocol for privately reporting ad impressions at scale. In: Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi, S. (eds.) ACM CCS 2016. pp. 1591–1601. ACM Press (Oct 2016). https://doi.org/10.1145/2976749.2978407
25. Gupta, T., Crooks, N., Mulhern, W., Setty, S.T.V., Alvisi, L., Walfish, M.: Scalable and private media consumption with popcorn. In: Argyraki, K.J., Isaacs, R. (eds.) 13th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2016, Santa Clara, CA, USA, March 16-18, 2016. pp. 91–107. USENIX Association (2016), https://www.usenix.org/conference/nsdi16/technical-sessions/presentation/gupta-trinabh
26. Henzinger, A., Dauterman, E., Corrigan-Gibbs, H., Zeldovich, N.: Private web search with Tiptoe. In: van Renesse, R., Zeldovich, N. (eds.) SOSP '23: ACM SIGOPS 29th Symposium on Operating Systems Principles. ACM (2023)
27. Henzinger, A., Hong, M.M., Corrigan-Gibbs, H., Meiklejohn, S., Vaikuntanathan, V.: One server for the price of two: Simple and fast single-server private information retrieval. In: Calandrino, J.A., Troncoso, C. (eds.) 32nd USENIX Security Symposium, USENIX Security 2023, Anaheim, CA, USA, August 9-11, 2023. USENIX Association (2023), https://www.usenix.org/conference/usenixsecurity23/presentation/henzinger
28. Huang, Y., Katz, J., Evans, D.: Efficient secure two-party computation using symmetric cut-and-choose. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 18–35. Springer, Heidelberg (Aug 2013). https://doi.org/10.1007/978-3-642-40084-1_2
29. Kilian, J.: On the complexity of bounded-interaction and noninteractive zero-knowledge proofs. In: 35th FOCS. pp. 466–477. IEEE Computer Society Press (Nov 1994). https://doi.org/10.1109/SFCS.1994.365744
30. Kiraz, M.S., Schoenmakers, B.: A protocol issue for the malicious case of Yao's garbled circuit construction. pp. 283—-290 (2006)
31. Kurosawa, K.: How to correct errors in multi-server PIR. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019, Part II. LNCS, vol. 11922, pp. 564–574. Springer, Heidelberg (Dec 2019). https://doi.org/10.1007/978-3-030-34621-8_20
32. Kushilevitz, E., Ostrovsky, R.: Replication is NOT needed: SINGLE database, computationally-private information retrieval. In: 38th FOCS. pp. 364–373. IEEE Computer Society Press (Oct 1997). https://doi.org/10.1109/SFCS.1997.646125
33. Lin, W.K., Mook, E., Wichs, D.: Doubly efficient private information retrieval and fully homomorphic RAM computation from ring LWE. Cryptology ePrint Archive, Report 2022/1703 (2022), https://eprint.iacr.org/2022/1703
34. Lindell, Y., Pinkas, B.: Secure two-party computation via cut-and-choose oblivious transfer. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 329–346. Springer, Heidelberg (Mar 2011). https://doi.org/10.1007/978-3-642-19571-6_20
35. Lipmaa, H.: An oblivious transfer protocol with log-squared communication. In: Zhou, J., Lopez, J., Deng, R.H., Bao, F. (eds.) ISC 2005. LNCS, vol. 3650, pp. 314–328. Springer, Heidelberg (Sep 2005)
36. Ryan, M.D.: Enhanced certificate transparency and end-to-end encrypted mail. In: NDSS 2014. The Internet Society (Feb 2014)
37. Vaudenay, S.: Security flaws induced by CBC padding - applications to SSL, IPSEC, WTLS... In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 534–546. Springer, Heidelberg (Apr / May 2002). https://doi.org/10.1007/3-540-46035-7_35
38. Wang, X., Zhao, L.: Verifiable single-server private information retrieval. In: Naccache, D., Xu, S., Qing, S., Samarati, P., Blanc, G., Lu, R., Zhang, Z., Meddahi, A. (eds.) ICICS 18. LNCS, vol. 11149, pp. 478–493. Springer, Heidelberg (Oct 2018). https://doi.org/10.1007/978-3-030-01950-1_28
39. Yi, X., Kaosar, M.G., Paulet, R., Bertino, E.: Single-database private information retrieval from fully homomorphic encryption. IEEE Trans. Knowl. Data Eng. **25**(5), 1125–1134 (2013). https://doi.org/10.1109/TKDE.2012.90, https://doi.org/10.1109/TKDE.2012.90
40. Zhang, L.F., Safavi-Naini, R.: Verifiable multi-server private information retrieval. In: Boureanu, I., Owesarski, P., Vaudenay, S. (eds.) ACNS 14. LNCS, vol. 8479, pp. 62–79. Springer, Heidelberg (Jun 2014). https://doi.org/10.1007/978-3-319-07536-5_5
41. Zhao, L., Wang, X., Huang, X.: Verifiable single-server private information retrieval from LWE with binary errors. Inf. Sci. **546**, 897–923 (2021). https://doi.org/10.1016/j.ins.2020.08.071, https://doi.org/10.1016/j.ins.2020.08.071

A Proof of Lemma 1 (DDH Lemma)

Define hybrid Hyb_i as the following distribution:

$$\left\{ \begin{array}{l} (\mathbb{G}, q, g, (g^{a_1}, \ldots, g^{a_N}), \\ (g^{c_1}, \ldots, g^{c_i}, g^{a_{i+1} \cdot r}, \ldots, g^{a_N \cdot r})) \end{array} \middle| \begin{array}{r} (\mathbb{G}, q, g) \leftarrow \mathsf{GG}(1^\lambda) \\ a_1, \ldots, a_N \leftarrow\$ \mathbb{Z}_q \\ c_1, \ldots, c_i \leftarrow\$ \mathbb{Z}_q \\ r \leftarrow\$ \mathbb{Z}_q \end{array} \right\}$$

Note that Hyb_0 is identical to the left-hand side of the Lemma's statement, and Hyb_N is identical to the right-hand side (because a random vector $\mathbf{h} \leftarrow\$ \mathbb{G}^N$ is the same as $(g^{c_1}, \ldots, g^{c_N})$ for random $c_1, \ldots, c_N \leftarrow\$ \mathbb{Z}_q$). Thus, it just remains to prove $\mathsf{Hyb}_i \approx_{\mathsf{c}} \mathsf{Hyb}_{i-1}$ for $i \in [N]$.

Let $\mathcal{A}$ be an adversary that distinguishes Hyb_i and Hyb_{i-1}, i.e.,

$$\left|\Pr[\mathcal{A}(I) = 1 \mid I \leftarrow \mathsf{Hyb}_{i-1}(\lambda)] - \Pr[\mathcal{A}(I) = 1 \mid I \leftarrow \mathsf{Hyb}_i(\lambda)]\right| \leq \mathsf{negl}(\lambda) . \quad (26)$$

Then we construct adversary $\mathcal{B}(\mathbb{G}, q, g, x, y, z)$ for DDH as follows:

- Sample random $a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_N \leftarrow\$ \mathbb{Z}_q$.
- Sample random $c_1, \ldots, c_{i-1} \leftarrow\$ \mathbb{Z}_q$.
- Output $\mathcal{A}(\mathbb{G}, q, g, (g^{a_1}, \ldots, g^{a_{i-1}}, y, g^{a_{i+1}}, \ldots, g^{a_N}), (g^{c_1}, \ldots, g^{c_{i-1}}, z, x^{a_{i+1}}, \ldots, x^{a_N}))$.

There are two cases:

1. If the input for $\mathcal{B}$ is generated as $(\mathbb{G}, q, g) \leftarrow \mathsf{GG}(1^\lambda)$ and $x = g^r$, $y = g^a$, $z = g^{a \cdot r}$ with $r, a \leftarrow\$ \mathbb{Z}_q$, then $\mathcal{B}(\mathbb{G}, q, g, x, y, z)$ is identical to running $\mathcal{A}(I)$ for $I \leftarrow \mathsf{Hyb}_{i-1}(\lambda)$.
2. If the input for $\mathcal{B}$ is generated as $(\mathbb{G}, q, g) \leftarrow \mathsf{GG}(1^\lambda)$ and $x = g^r$, $y = g^a$, $z = g^c$ with $r, a, c \leftarrow\$ \mathbb{Z}_q$, then $\mathcal{B}(\mathbb{G}, q, g, x, y, z)$ is identical to running $\mathcal{A}(I)$ for $I \leftarrow \mathsf{Hyb}_i(\lambda)$.

Note that case (1) means that $\mathcal{B}$ was run with an input generated using the left-hand side distribution of DDH, and case (2) means that $\mathcal{B}$ was run with an input generated using the right-hand side distribution of DDH. By the DDH assumption, this implies Equation (26).

□

B Combining Validations with Queries

In this section, we prove that it is possible to *jointly* run the validation and query procedures. Compared to the setting considered in the main body of this paper, this approach will save the validation round-trip that a client needs to perform before making its first query.

Consider any APIR scheme that fulfills privacy with abort. We construct a scheme APIR′ identically to APIR, except that every query has a built-in validation. If the validation fails, then Rec aborts.

- $\mathsf{APIR}'.\mathsf{Query}(i)$ runs $(\mathsf{st_v}, v) \leftarrow \mathsf{APIR.VQuery}()$ and $(\mathsf{st_q}, q) \leftarrow \mathsf{APIR.Query}(i)$. It outputs state $(\mathsf{st_v}, \mathsf{st_q})$ and query (v, q).
- $\mathsf{APIR}'.\mathsf{Answer}(\mathbf{x}, (v, q))$ returns $(\mathsf{APIR.VAnswer}(\mathbf{x}, v), \mathsf{APIR.Answer}(\mathbf{x}, q))$.
- $\mathsf{APIR}'.\mathsf{Rec}((\mathsf{st_v}, \mathsf{st_q}), d, (a_\mathsf{v}, a_\mathsf{q}))$ runs $t \leftarrow \mathsf{APIR.VCheck}(\mathsf{st_v}, d, a_\mathsf{v})$ and $b \leftarrow \mathsf{Rec}(\mathsf{st_q}, d, a_\mathsf{q})$. If $t = 0$, it outputs $\perp$. Otherwise, it outputs b.

The scheme APIR′ does not contain any validation algorithms VQuery, VAnswer, and VCheck. Thus, while integrity of APIR′ follows immediately from integrity of APIR, our original notion of privacy with abort (Definition 6) does not even match the syntax of APIR′ anymore. Instead, we modify the PRIV/A game (Figure 2) by removing steps 2-4, and we say that APIR′ fulfills privacy with abort if the output of the new game PRIV/A′ is indistinguishable for any $i_0(\lambda)$ and $i_1(\lambda)$.

Lemma 14. *If* APIR *fulfills privacy with abort, then* APIR' *fulfills the modified notion of privacy with abort.*

Proof sketch. Let $\mathcal{A}^* := (\mathcal{D}, \mathcal{A}_1, \mathcal{A}_2)$ be an adversary that breaks the modified notion of privacy with abort in APIR'. That is, $\mathcal{A}_1(c, (v, q))$ outputs a state $\mathsf{st}_\mathcal{A}$ and answers $a_\mathsf{v}, a_\mathsf{q}$, and $\mathcal{A}_2(\mathsf{st}_\mathcal{A}, \mathsf{abort}')$ outputs a decision bit.

We construct an adversary $\mathcal{B}^* := (\mathcal{D}, \mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3)$ against privacy with abort in scheme APIR as follows:

- $\mathcal{B}_1(c, v)$ attempts to pass validation v by generating an arbitrary query, say $\mathsf{st}^*_\mathsf{q}, q^* \leftarrow \mathsf{APIR.Query}(1)$, and then running $\mathsf{st}^*_\mathcal{A}, a_\mathsf{v}, a^*_\mathsf{q} \leftarrow \mathcal{A}_1(c, (v, q^*))$. It outputs state $\mathsf{st}_\mathcal{B} := c$ and validation answer a_v. *(Note that* $\mathsf{st}^*_\mathcal{A}, a^*_\mathsf{q}$ *will not be used at all, because* $\mathcal{B}_1$ *just tries to get past the validation requirement of* PRIV/A*.)*
- $\mathcal{B}_2(\mathsf{st}_\mathcal{B}, q)$ generates a validation $\mathsf{st}^*_\mathsf{v}, v^* \leftarrow \mathsf{APIR.VQuery}()$ and runs $\mathsf{st}_\mathcal{A}, a^*_\mathsf{v}, a_\mathsf{q} \leftarrow \mathcal{A}_1(c, (v^*, q))$. It checks $t \leftarrow \mathsf{APIR.VCheck}(\mathsf{st}^*_\mathsf{v}, d, a^*_\mathsf{v})$, and then outputs state $\mathsf{st}'_\mathcal{B} := (\mathsf{st}_\mathcal{A}, t)$ and answer a_q.
- $\mathcal{B}_3(\mathsf{st}'_\mathcal{B}, \mathsf{abort})$ computes the modified abort bit $\mathsf{abort}' := \mathbb{I}[t = 0 \vee \mathsf{abort}]$, and then runs and outputs $\mathcal{A}_2(\mathsf{st}_\mathcal{A}, \mathsf{abort}')$.

If we were to remove line 4 from the definition of PRIV/A (which returns 0 whenever $\mathcal{B}_1$ does not respond successfully to validation v), then the distribution of $\mathrm{PRIV/A}^{\mathcal{B}^*}(\lambda, N, i)$ would be identical to that of $\mathrm{PRIV/A}'^{\mathcal{A}^*}(\lambda, N, i)$. Furthermore, $\mathcal{B}_1$ simply answers the initial validation v (which is completely independent of v^*) by querying $\mathcal{A}_1$ together with a query generated for index 1. Therefore, for any $i \in [N]$ and fixed configuration $c \in \mathsf{Supp}(\mathrm{Conf}^\mathcal{D}(\lambda, N))$, we get

$$\Pr[\mathrm{PRIV/A}^{\mathcal{B}^*}_{\mathsf{APIR}}(c, i) = 1] = \Pr[\mathrm{PRIV/A}^{\mathcal{A}^*}_{\mathsf{APIR}}(c, i) = 1] \cdot \mu_{\mathcal{A}^*}(c, 1) ,$$

where the experiments above use the fixed configuration c instead of generating it newly, and $\mu_{\mathcal{A}^*}(c, j)$ is the probability of $\mathcal{A}^*$ successfully passing the validation component of a query $\mathsf{APIR}'.\mathsf{Query}(j)$.

Furthermore, by standard privacy, for any $i_b \in [N]$ we have

$$\left|\mathop{\mathbb{E}}_c[\mu_{\mathcal{A}^*}(c, 1)] - \mathop{\mathbb{E}}_c[\mu_{\mathcal{A}^*}(c, i_b)]\right| \le \mathsf{negl}(\lambda) ,$$

where both expectations are taken over the random choice of $c \leftarrow \mathrm{Conf}^\mathcal{D}(\lambda, N)$.

However, because $\mathcal{A}^*$ breaks the modified notion of privacy with abort, we know that $\mathbb{E}_c[\mu_{\mathcal{A}^*}(i_b)]$ (for at least one of $b = 0, 1$) is not negligible. Combining this with the previous two equations shows that $\mathcal{B}^*$ breaks privacy with abort of APIR. □