Verifiable Private Information Retrieval

Shany Ben-David[1], Yael Tauman Kalai[2], and Omer Paneth[1]

[1] Tel Aviv University
[2] Microsoft Research and MIT

Abstract. A computational PIR scheme allows a client to privately query a database hosted on a single server without downloading the entire database. We introduce the notion of verifiable PIR (vPIR) where the server can convince the client that the database satisfies certain properties without additional rounds and while keeping the communication sub-linear. For example, the server can prove that the number of rows in the database that satisfy a predicate P is exactly n.
We define security by modeling vPIR as an ideal functionality and following the real-ideal paradigm. Starting from a standard PIR scheme, we construct a vPIR scheme for any database property that can be verified by a machine that reads the database once and maintains a bounded size state between rows. We also construct vPIR with public verification based on LWE or on DLIN. The main technical hurdle is to demonstrate a simulator that extracts a long input from an adversary that sends a single short message.
Our vPIR constructions are based on the notion of batch argument for NP. As contribution of independent interest, we show that batch arguments are equivalent to quasi-arguments—a relaxation of SNARKs which is known to imply succinct argument for various sub-classes of NP.

1 Introduction

A single-server computational private information retrieval (PIR) scheme [13] allows clients to query a database privately, without revealing to the server any information about their query. Such a PIR scheme is non-trivial if the server's message is shorter than the database. In the standard notion of PIR, the server is free to use any database to answer the clients query. In this work, we introduce a variant of PIR that we call *verifiable PIR* (vPIR). In vPIR, the server can prove to the client that the database it is using to answer the query satisfies some property, for example: the database entries are sorted, or the database does not contain some value X more than n times. As before, in a non-trivial scheme the server's answer *including the proof* should be shorter than the database.

In this work we focus on *one round* vPIR schemes: the client sends a query and the server responds with a single message that includes both the PIR answer and a proof that the database satisfies the required property. Indeed, one-round is the standard when constructing PIR schemes, and it is required for many applications.[3]

[3] We expand on the advantages of one-round schemes and discuss solutions with more rounds at the end of this introduction.

vPIR via secure computation. In plain PIR the only standard security requirement is that the client's message hides its query. However, in the setting of vPIR, defining security against malicious servers that may not use a valid database requires more care. One natural approach is to define vPIR as a special case of secure two-party computation for the vPIR functionality. The vPIR functionality takes a query i from the client and a database D from the server, and returns $D[i]$ to the client if D satisfies the required property, or $\perp$ otherwise.

The problem with this approach is that non-trivial one-round vPIR schemes require secure computation with very strong properties. Specifically, we need a one-round secure computation protocol (also known as non-interactive secure computation [8]) with security against a malicious server where the server's communication is sub-linear in its input. Currently, such protocols are only known in the CRS model based on succinct non-interactive arguments of knowledge (SNARKs) for NP.[4] In all known constructions of SNARKs for NP, soundness is either heuristic, or based on so-called non-falsifiable knowledge assumptions. SNARK constructions with an explicit knowledge extractor are not known and are subject to strong barriers [3]. We note that even if we relax security and allow for super-polynomial, or even unbounded simulation, we do not know of any solutions that do not use SNARKs. In light of this barrier, in this work we propose alternative formulations of vPIR and realize them under standard assumptions.

1.1 Our Contribution

Our first contribution is proposing two definitions of vPIR that relax the security definition based on secure computation: A game-based definition for *local* database properties that only depends on a small number of entries, and a simulation-based definition for *global* properties that depends the entire database. We explore possible applications and discuss the limitations of each definition. Then, we show how to construct vPIR schemes that satisfy our definitions for a rich class of properties based on various standard assumptions.[5] Finally, we show that the notion of vPIR is closely connected to the notions of quasi-arguments and batch arguments for NP [4, 10] that were recently proposed in relation to delegating computation [6, 11]. Based on these connections we derive new results and simplify existing results in the area of delegation. In what follows, we elaborate on our contributions.

[4] Since for vPIR we do not require any security against a corrupted client, the client can send the CRS as part of its message.

[5] We mention that succinct non-interactive arguments (SNARGs) for certain subclasses of NP, including the class of properties supported by our constructions, are known under standard assumptions [1, 6, 11, 16]. Such SNARGs, however, do not give succinct non-interactive secure computation for the vPIR functionality for any non-trivial property since they lack an efficient knowledge extractor. We elaborate on this below where we state our result in more detail.

Definitions of vPIR. Before presenting our security definitions, we elaborate on the syntax of a vPIR scheme. In a standard PIR scheme the client generates, together with its query, a secret key dk used to decode the answer. In a vPIR scheme the client also generates a verification key vk. The server answers the query using a database that satisfies a property P and its answer includes a proof of this fact. The client verifies the server's answer using vk, and if the answer is accepted, the client decodes it using dk.

In the setting where multiple queries are answered with the same database we may need to verify, not only that each query was answered using some database that satisfies P, but also that all queries where answered using the same database. To this end, we first verify each answer with its own verification key, and then verify the consistency of all the answers together. We require that verifying the consistency of the answers can be done without any verification key. This feature is particularly useful in settings where the queries are generated by multiple clients since it does not require clients to share their verification key.

We consider two flavors of vPIR schemes: publicly verifiable vPIR where vk can be made public, and designated verifier vPIR where vk must be kept secret. In the designated verifier setting, a cheating server that learns vk may be able to break soundness. However, we require that the client's query remains hidden even if vk becomes public. This, in particular, guarantees that by revealing its decision to accept or reject, the client may compromise its verification key and lose soundness, but it does not compromise the secrecy of its query.

We emphasize that even if verification requires only a public key, decoding the server's answer always requires the secret key dk. Nonetheless, public verification has several advantages: First, it allows any user (an not just the client) to audit the server. Furthermore, it allows to reuse the same vPIR query many times without compromising soundness, even if the verification results are made public. Finally, in what follows, we use public verifiability for composition and to derive new results on publicly verifiable delegation.

Next we describe two definitions of vPIR security, for local and for global properties. Each of these security requirements is made in combination with the standard PIR privacy requirement that the client's query completely hides the database location queried.

vPIR for local properties. Our first security notion for vPIR deals with local database properties that depend only on a small number of entries. In more detail, we model an ℓ-local property as an efficient program P that takes as input ℓ locations $q_1, \ldots, q_\ell$ and the corresponding database entries $a_1, \ldots, a_\ell$ and produces a binary output. A server holding a database D can prove that it satisfies the ℓ-local property P if the property is satisfied by any ℓ locations. That is, for every $q_1, \ldots, q_\ell$, the property P accepts $q_1, \ldots, q_\ell$ and $a_1, \ldots, a_\ell$ where $\mathbf{a}_i = D[\mathbf{q}_i]$.

Intuitively, our security requirement states that if we query the server on any ℓ locations and the server's answers pass verification then the decoded entries must satisfy the ℓ-local property P. In more detail, we consider the following game between a challenger and an adversary playing the role of the server.

The challenger generates ℓ queries for locations $q_1, \ldots, q_\ell$, sends them to the server and obtains ℓ answers. The challenger then checks that each answer passes the verification with its own verification key, and that all ℓ answers together are consistent (this check does not require the verification keys). Finally, the challenger decodes each answer using its own decoding key and obtains the entries $a_1, \ldots, a_\ell$. The adversary wins the game if all the answers are found to be valid and consistent, but the property P rejects $q_1, \ldots, q_\ell$ and $a_1, \ldots, a_\ell$. The requirement is that for any ℓ locations, the adversary wins with negligible probability.

Use cases. We discuss some examples of using the above definition. Starting with the simple case of 1-local properties, consider a server that offers its clients private access to a database of articles. The server claims that all the articles in its database have been fact-checked and digitally signed by a trusted third party. To prove this claim, we can use a vPIR for the 1-local property checking that a given entry contains a valid digital signature. By the vPIR security the client is guaranteed that if the answer passes verification, the decoded entry will contain a valid signature.

Note that instead of verifying the answer, the client can simply decode it and check if it contains a valid signature. However, in this case, simply revealing the fact that the server is cheating may compromise the secrecy of the clients query. To see that, consider a database that contains a single unsigned article. If the client detects cheating then its query location is completely revealed. In contrast, using vPIR, if the server's database contains even one unsigned article, then, since the client's query is hiding, and since this hiding holds even given the verification key, the server's answer will be rejected with all but negligible probability, regardless of the client's query location. In particular, the verification result does not compromise the client's privacy.

Another important use-case of vPIR for local properties is in the setting of multiple clients. Consider, for example a cheating server that claims to use the same database of articles when interacting with all of its clients, but, in reality, clients asking for the same article may get different content (or no content) based on their identifying information. As in the previous example, clients accusing the server of cheating may be exposing their secret query location. Moreover, in this example, even detecting that the server answers are inconsistent requires the clients to reveal their secret query location to *each other*. Using vPIR two or more clients can guarantee the consistency of their answers, that is, they can verify that if they made the same query they also received the same answer, while keeping their query completely private, even from each other.

Going beyond consistency checking, clients can use vPIR to verify more complicated relations between their answers. For example, if the query locations correspond to the nodes of a graph and the database contains a valid coloring of the graph, then clients can check that their answers verify the coloring constraints (again, while keeping their query location completely hidden, even from each other) by verifying the following 2-local property: given locations q_1, q_2 and colors a_1, a_2 the property rejects if and only if (q_1, q_2) is an edge but $a_1 = a_2$.

vPIR for global properties. Next we consider setting where the server would like to prove to its clients something that be expressed as a local property of the client's queries and server's answers. For example, consider again our server that is holding a database of fact-checked and digitally signed articles. Now, suppose that the articles in the server's database update frequently and, as a result, at any given moment a small fraction of the articles, say 10%, have yet to be checked and signed by the third party. Since the database always contains some unsigned articles, the server can no longer prove that the database satisfies the 1-local property. Instead, the server would like to prove that the database satisfies a global property: at least 90% of the entries contain a valid signature.

With this motivation in mind, we propose another security notion for vPIR dealing with global properties that may depend on the entire database. We model a global property P as an efficient program that takes as input the entire database D and produces a binary output. When defining security for global properties, we observe that simply requiring that an accepting answer is consistent with *some* database that satisfies P is insufficient. For example, if the property P asserts that 90% of the entries contain a valid signature, then any answer, whether it is signed or not, is always consistent with some database that satisfies P. Intuitively, the issue with this naive definition is that it does not guarantee that server uses a database that is independent of the query location. To resolve this issue, we propose a simulation based security definition for vPIR with global properties. Our definition is a relaxation of secure computation for the vPIR functionality (where the adversary corrupts the server). Looking ahead, this relaxation will be crucial for our analysis. Nonetheless, we argue that our definition still provides meaningful security in the setting of vPIR and demonstrate its applications.

The definition follows the real-ideal paradigm. In the real experiment, we send the adversary, playing the role of the server, a query for location q and obtain an answer. The output is either the decoded entry or $\perp$ if the answer fails to verify. In the ideal experiment, the simulator submits an entire database to the trusted party that computes the output: either the value at location q or $\perp$ if the database does not satisfy the property P. We require that for any adversarial server there exists an efficient simulator such that for every location q, the output of the two experiments are indistinguishable. We will also consider a variant of this definition where the simulator is allowed to run in super-polynomial time.

vPIR vs. secure computation. The above definition is similar to the definition of secure computation for the vPIR functionality with one important relaxation: In the definition of secure computation, the output of the real and ideal experiments contain both the output of the honest client and the output of the adversary/simulator. In our definition, however, only the client's output is included in the output of the experiments. Recall that in addition to the above security requirement, we also require explicitly that client's query is hiding. Therefore, we can always simulate the view of the adversary by generating a query for an arbitrary location. However, while we can simulate the output of the client and

the server individually, we may not be able to simulate the joint distribution of the outputs as required for secure computation.

Intuitively, since we simulate the client's output, our definition guarantees that a malicious server skew the client's output distribution. However, since we do not simulate the adversary's view together with the client's output, it may be possible for the adversary to learn something that be simulated in the ideal experiment (as long as the query location remains hidden). For example, going back to our server proving that 90% of the entries in its database are signed, if the client queries a random location, then the ideal experiment's output is either $\perp$, or it contains a signed entry with probability at least 0.9 (this holds even if the simulator is computationally unbounded). Therefore, the same is guaranteed also in the real experiment. However, we cannot guarantee that a malicious server cannot learn, for example, whether the client's answer is signed or not.

Constructions of vPIR. We next provide an overview of our vPIR constructions both for local and for global properties under standard assumptions.

vPIR for local properties. Our first result is a designated verifier vPIR for all ℓ-local properties for any constant ℓ under the minimal assumption that PIR schemes exist.

Theorem 1 (Informal). *Assuming poly-logarithmic PIR scheme exists, for any constant ℓ there exists a designated verifier poly-logarithmic vPIR scheme for all ℓ-local properties decidable in polynomial-time. For security parameter κ and a database with m rows, each of length w, the communication complexity and verification time are $w \cdot \mathrm{poly}(\kappa, \log m)$.*
For $\ell > 1$ the vPIR scheme requires public parameters that contain the description of a collision-resistant hash.

Theorem 1 is stated based on poly-logarithmic PIR where the communication complexity is $\mathrm{poly}(\kappa, \log m)$ for a databases of m bits. More generally, assuming PIR with communication complexity $C(m, \kappa)$ we get vPIR with communication complexity $C \cdot w \cdot \mathrm{poly}(\kappa, \log m)$.

In the publicly verifiable setting, we construct vPIR for the same properties under LWE or under bilinear paring.

Theorem 2 (Informal). *Under the LWE assumption, for any constant ℓ there exists a publicly verifiable vPIR scheme for all ℓ-local properties decidable in polynomial-time with communication complexity and verification time $w \cdot \mathrm{poly}(\kappa, \log m)$.*

Theorem 3 (Informal). *Under the DLIN assumption in a prime-order pairing group, for any constants ℓ, ϵ there exists a publicly verifiable vPIR scheme for all ℓ-local properties decidable in polynomial-time T with communication complexity and verification time $(w + T^{\epsilon}) \cdot 2^{\sqrt{O(\log \kappa \log m)}}$.*

The main tool we use in our construction of vPIR for local properties is a batch argument for NP [4]. We give a generic construction based on any batch argument that satisfies certain properties [6] (see the technical overview for more details). Theorems 2 and 3 are based the batch arguments from [6] and [16] respectively. Theorem 1 is based by the batch argument constructed in [4] that we modify to meet our requirements.

Beyond constant locality. The vPIR schemes in the theorems above can only handle properties with constant locality ℓ. To justify this limitation observe that when ℓ is super-constant, a vPIR scheme with an efficient client and server for all ℓ-local properties is unlikely. This is the case since deciding if a given database D satisfies an ℓ-local property P (for every ℓ locations) may require time exponential in ℓ. In contrast, if we had a vPIR scheme for P with an efficient client and server we could also decide if D satisfies P efficiently, by running the honest vPIR server on ℓ queries (for arbitrary locations) and checking if the answers pass verification. If D satisfies P the answers should be accepted. However, if there exist ℓ locations that do not satisfy P then no efficient algorithm queried on these locations should be able to produce answers that pass verification with non-negligible probability. Therefore, since the queries are hiding (even given the verification key) verification must also fail when the queries are for arbitrary locations.

vPIR for global properties. Before stating our results on vPIR with global properties, we start by mentioning some barriers. We observe that vPIR for *a*ll efficiently testable global properties implies SNARKs and, therefore, we do not expect to realize such vPIR under standard assumptions. In more detail, a designated verifier vPIR scheme implies designated verifier SNARKs, while publicly verifiable vPIR implies full-fledged SNARKs. Moreover, vPIR with inefficient simulation implies succinct non-interactive arguments (SNARGs) for NP. None of these notions are known based on any falsifiable assumption, and such constructions are subject to barriers [7, 3]. We can transform a vPIR scheme into a SNARK as follows: the CRS contains a query for an arbitrary location. To prove some NP statement, the prover views the witness as a database of bits and proves that it satisfies the global property testing the witness validity. The verifier accepts if the vPIR answer passes verification. If the vPIR scheme is non-trivial then the proof is succinct. To argue knowledge soundness, consider an adversary that convinces the verifier with non-negligible probability. It follows that in the ideal experiment the simulator must submit a database that contains a valid witness to the trusted party and, therefore, it can be used as an extractor.

In light of this barrier, we focus on constructing vPIR for a restricted, yet useful class of global properties. Specifically, we consider the class of properties decidable by a Turing machine that reads the database once and maintains a state of bounded length between rows. In more detail, each property in the class can be tested by a machine P that maintains a state of length S (shorter than the database). For every row in the database, the machine P updates its state

by applying an arbitrary efficient function to the current state and database row. (We emphasize the length of the database row or the space of the update function may not be bound by S.) For example, for any efficiently computable predicate Q, we can verify that the number of database rows that satisfy Q is exactly n using a state of size $S = \log(n)$. As another example, verifying that the database is sorted can be done with a state of size $S = w + 1$, where w is the length of each row.

We are ready to state our results on vPIR for global properties, starting with the designated verifier setting.

Theorem 4 (Informal). *Let $\Lambda(\kappa)$ be a function of the security parameter. Assuming Λ-secure poly-logarithmic PIR scheme exists, there exists a designated verifier vPIR scheme for any global property decidable by a polynomial-time read-once Turing machine P with description and state of length at most $\log \Lambda$. The communication complexity and verification time are as in Theorem 1. For every constant $c \in \mathbb{N}$ there exists a simulator running in time $\Lambda^{O(c)}$ such that every $\mathrm{poly}(\Lambda)$-size distinguisher has advantage at most $\Lambda^{-O(c)}$.*

In the publicly verifiable setting, we construct vPIR for the same class of global properties and with the same simulation time under the Λ-hardness of LWE or DLIN. The communication complexity and verification time are as in Theorem 2 and Theorem 3 respectively.

Note that in Theorem 4 we allow the description size of the machine P to grow with the security parameter as long as $|P| \leq \log \Lambda$. If we restrict attention to the non-adaptive setting where the machine P is fixed before the query then we can prove a stronger result where the description size of P is not bounded. We emphasize that even in the non-adaptive setting, the adversary may still choose its database adaptively.

On the simulation time. For a bound $\Lambda = \mathrm{poly}(\kappa)$, our result gives a vPIR scheme with polynomial time simulation (with inverse polynomial accuracy). For a super-polynomial Λ, the result captures a larger set of properties, however, with simulation that runs in super-polynomial time (and based on super-polynomial hardness assumptions). However, as argued above, vPIR with super-polynomial, or even unbounded, simulation still provides meaningful security. (Intuitively, it still guarantees that adversary's database is independent of the query location.) Theorem 4 leaves open the possibility of constructing a vPIR scheme where the running time of the simulation does not grow exponentially with the length of the machine's state and description. We discuss some barriers towards such improvements.

As argued above, vPIR for arbitrary global properties implies a SNARKs for NP. We show that this implication can be extended also to vPIR that only supports properties decidable by a read-once Turing machine as long as the simulation time is sub-exponential in the length of the machine's state (and assuming sufficiently strong collision-resistant hashing). As before, we construct a SNARK by letting the prover encode its NP witness in the database. However now, to allow for verification by a read-once machine with a succinct state, we

first encode the witness so that each row contains the next bit required by the NP verification procedure. If the same witness bit is used multiple times it will appear in multiple rows. To verify that all occurrences of the same witness bit are consistent we compute a hash tree over the entire witness, adding the root to the machine's state and adding to each row the authentication path for the next witness bit. To argue knowledge soundness, consider an adversary that convinces the verifier with non-negligible probability. It follows that in the ideal experiment the simulator must submit a database that contains either a valid witness or a hash collision. If the simulation time is sub-exponential in the length of the machine's state (or, the length of the hash root) and assuming the hash is sufficiently strong, then the simulator cannot find collisions and, therefore, it can be used as an extractor.

We can further extend this argument to vPIR where the simulation time is sub-exponential in the length of the machine's description. The idea is to hard-code the root into the machine's description instead of adding it to the machine's state. Note that this requires vPIR with adaptive security. Indeed, in the non-adaptive setting we do get vPIR where the simulation time does not grow with the machine's description under standard assumptions.

On vPIR from SNARKs for sub-classes of NP. As mentioned above, a vPIR scheme can be constructed based on SNARKs for NP by compiling a semi-honest protocol based on any standard PIR scheme. Therefore, a natural approach to proving Theorem 4 is to construct a vPIR based on SNARKs for a sub-class of NP. Specifically, observe that vPIR for any global property decidable by a read-once machine with state of length S can be constructed by combining a standard PIR scheme where the honest server is implemented by a read-once machine with state of length $\text{poly}(\kappa)$ (such PIR schemes are known under various standard assumptions) with a SNARK for NP languages decidable by a read-once machine with state of length $S' = S + \text{poly}(\kappa)$. Currently, however, SNARKs for languages decidable by a read-once machine with super-logarithmic state are not known under any falsifiable assumption. The work of [1] constructed designated verifier *SNARGs* for the same class under $2^{S'}$-hardness assumption,[6] In particular however, we do not know how to compile a standard PIR scheme into a vPIR scheme without relying on the SNARK's efficient knowledge extractor. While the SNARGs of [1] do not directly imply vPIR, the simulation strategy we use to prove Theorem 4 is based on techniques from the analysis of [1].

While the above construction instantiated with the SNARGs of [1] may not satisfy our notion of vPIR for global properties, we can, nonetheless, use it to get a vPIR for 1-local properties where simulation is not required. This is based on the fact that verifying that a databases satisfies a 1-local property can be done by a read-once machine with state of length 1. The advantage of the vPIR

[6] The work of [1] constructed a SNARG for a slightly more restricted class: languages decidable by read-one non-deterministic Turing machines with bounded space. However, based on standard techniques, their result can be extended to the case where only the length of the state between rows is bounded.

scheme given in Theorem 1 is that it can be based on polynomial PIR rather than sub-exponential.

vPIR and delegation. As our final contribution we demonstrate connections between the notion of vPIR and other notions recently introduced in the context of delegating computation. Based on these connections we reinterpret and simplify recent results and prove new results in this area.

Our first connection is between vPIR and the notion of batch arguments for NP [4]. Batch arguments allows us to verify many NP statements with the same complexity as verifying a single statement. They have several applications including SNARGs for P [4–6, 11, 16]. Specifically, we observe that a vPIR for 1-local properties is equivalent to a stronger variant of batch arguments introduced by [6] known as batch arguments for the index language satisfying the somewhere argument of knowledge property. This equivalence holds in both the designated verifier and the publicly verifiable settings.

We also show a connection between vPIR and quasi-arguments for NP [14, 10]. Quasi-arguments are succinct non interactive arguments for NP satisfying a weak soundness property known as local non-signaling knowledge extraction. They are known to imply SNARGs for P, batch NP, and several other subclasses of NP [9, 4, 1, 5]. We show that vPIR for 3-local properties is equivalent to quasi-arguments satisfying a natural property that we call PIR-friendliness (satisfied by all known constructions). This equivalence as well holds in both the designated verifier and the publicly verifiable settings.

As part of the proof of Theorem 1 we give a transformation from vPIR for 1-local properties to vPIR for ℓ-local properties for any constant ℓ. As a corollary we get a new equivalence between the notions of batch arguments for the index language and PIR friendly quasi-arguments. Based on the recent batch arguments from [6, 16] we get new constructions of quasi-arguments under LWE or DLIN. We also get the first quasi-argument with a sublinear CRS. Moreover, since quasi-arguments are known to imply SNARGs for P, we get an alternative construction of SNARGs for P based on batch arguments [6] that is simple and modular. We can also rederive the applications in [11] as direct applications of quasi-arguments and without relying on non-signaling MIPs or sub-exponential security.

We also use the above connections to give a new bootstrapping theorem for quasi-argument. The work of [10] constructs quasi-arguments with a long CRS. Then they used a variant of the bootstrapping technique [13, 15, 2] to construct SNARGs for P with a sub-linear CRS. However, there were not able to use their technique to bootstrap their quasi-arguments directly. We give a new bootstrapping theorem for vPIR for 1-local properties, similar to bootstrapping theorems proved in previous work [10, 16]. Using the connection between vPIR and quasi argument we also get a bootstrapping theorem for quasi-arguments.

1.2 On the Round Complexity of vPIR

In this work we construct vPIR schemes which consist of one round, which is the standard when constructing PIR schemes. Compared to solutions with multiple rounds, one round schemes have several advantages that make them appealing in applications. First, they allow to reuse the same query to produce multiple answers. Another advantage of one-round schemes is that the server can fix its database right before producing its answer. This may be useful, for example, in settings where the client and server are communicating asynchronously (e.g., they are not online simultaneously) and there might be a significant delay between the query and answer. This also allows the server to fix its database as a function of any incoming communication from the client that might be transmitted with the query. Finally, we mention that some techniques such PIR composition [13] require one-round schemes.

We mention that a vPIR scheme with multiple rounds can be constructed based on any succinct interactive argument of knowledge [12]. A scheme with 3 messages can be constructed based on known results on delegating RAM computations under various standard assumptions [9, 4, 10, 6, 16]. In this scheme the server's first message is a short commitment to its database. Therefore, when receiving the clients query the server can no longer change its database.

1.3 Open Questions

Our results on vPIR for global properties leaves open several important questions. One natural question is to identify other interesting class of global properties can be proven based on standard assumptions. Another important direction is to construct vPIR that satisfies the full secure computation definition where we simulate the view of both the client and the server together. A related question is to identify other non-trivial functionalities (beyond vPIR) that can be securely realized based on standard assumptions in a single round and with communication complexity that is sub-linear in the input length of the malicious party (perhaps settling for our relaxed notion of simulation).

Our last question deals with simulation security for multiple queries. While our simulation definition for vPIR naturally extends to multiple queries, the proof techniques behind Theorem 4 do not seem to go beyond a single query. In fact we do not know if simulating the answers to multiple queries is possible even for the plain PIR functionality where there are no restrictions on the database used.

2 Technical Overview

In this section we overview our vPIR constructions for local and for global properties.

2.1 vPIR for Local Properties

We describe our construction of vPIR for ℓ-local properties for any constant ℓ in both the designated verifier and the publicly verifiable settings. Our construction proceeds in two steps: first we construct a vPIR scheme for one local properties and then we transform it into a vPIR scheme for properties with constant locality.

Batch arguments. The main tool we use are batch arguments for the index language [6]. We start from either designated verifier or publicly verifiable batch arguments and get vPIR of the same type. Batch arguments are one message arguments relative to a public key (in the designated verifier setting, the public key is generated together with a secret verification key). In batch arguments for NP the prover and verifier share m NP statements $x_1, \ldots, x_m$ and the prover should convince the verifier that all statements are true. The communication complexity and verification time may grow polynomially with the security parameter and with k, the length of a single witness, but they should be independent of the number of statements m. In batch argument for the index language, the statements $x_1, \ldots, x_m$ are replaced by a single polynomial-time machine M. The prover should convince the verifier that for every $i \in [m]$ there exists a witness $w_i \in \{0,1\}^k$ such that $M(i, w_i)$ accepts.

In the adaptive setting, the prover may choose the statement M as a function of the public key. Batch arguments with full-fledged adaptive soundness are known to imply SNARKs for NP [4]. Therefore, we focus on a weaker notion of adaptive soundness known as somewhere argument of knowledge [6]. The soundness requirement is that for every index $i \in [m]$, we can generate a "programmed" public key for i together with a secret extraction key such that the programmed public key is indistinguishable from an honestly generated key (in particular it hides i). Moreover, given any accepting proof we can use the extraction key to recover a witness w_i such that $M(i, w_i)$ accepts with overwhelming probability.

vPIR for 1-local properties. Our vPIR scheme for a 1-local property P is as follows. The client's query for location $q \in [m]$ contains a public key for the batch argument programmed with the index q. The vPIR verification key is the verification key of the batch argument and the vPIR decryption key is the secret extraction key. Given a database D that satisfies P (that is, $P(i, D[i])$ accepts for every $i \in [m]$), the server computes a batch argument proof for the machine P using the witness $D[1], \ldots, D[m]$ and sends it as the vPIR answer. To verify the proof, the client simply verifies the batch argument. To decrypt the answer the client uses the extraction key to recover the witness $D[q]$. The resulting vPIR is non-trivial because the communication complexity of the batch argument does not grow with the database size m. The vPIR query hides the location q since the programmed public key is indistinguishable from an honestly generated key. The security of the vPIR follows directly from the somewhere argument of knowledge property of the batch argument.

In the publicly verifiable setting we can instantiate the construction under LWE or DLIN based on the batch arguments from [6, 16]. In the designated

verifier setting our starting point is the batch arguments for NP based on PIR from [4]. Their work did not construct batch argument for the index language and security was only proved in the non-adaptive setting. We modify their construction to work for the index language. In a nutshell, to get batch argument for the index language we need to show that the batch NP statement given by a machine M can be described as a 3CNF formula that has a short algebraic representation independent of the number of statements m.

Better complexity via composition. We show that the communication complexity and verification time of the a vPIR scheme for 1-local properties can be made sublinear (or even independent) in the running time of the property P by composing it with any SNARG for P. The idea is to add to each database row a SNARG proving that the row satisfies the property P and to prove that the database satisfies the 1-local property P' that each row contains an accepting SNARG. A similar construction was suggested in [6] in the context of batch arguments.

We also show how to transform any publicly verifiable vPIR scheme for 1-local properties with a long query (linear in the database size m]) into a vPIR scheme with sub-linear query length by composing it with itself. The idea follows the standard bootstrapping technique for plain PIR [13]. We divide the database into blocks. We use the vPIR to retrieve one entry from each block and then use the vPIR again to retrieve the PIR answer from one of the blocks. In each invocation of the inner vPIR we prove that each entry in the block satisfies the 1-local property P. In the outer vPIR we prove that the inner vPIR answer of each block satisfies the 1-local property P' that verifies the inner vPIR answer. A similar construction was suggested by [10] in the context of quasi-arguments and by [16] in the context of batch arguments.[7]

vPIR for properties with constant locality. Given a vPIR scheme for 1-local properties we construct a vPIR scheme for ℓ-local properties for any constant ℓ. For simplicity, in this overview we set $\ell = 2$. Our construction requires public parameters that contain a collision-resistant hash key. The server computes a hash tree over the database with root rt. Then, it constructs the database D' with m^2 rows such that row (i, j) contains the pair (r_i, r_j) together with the authentication paths from r_i and from r_j to rt. To query location i of D, the client uses the vPIR scheme for 1-local properties to generate a query q for the location $(i, 1)$ in D'. For the security proof to go through, we need to add to q another dummy query q^* for the row $(1, 1)$. The server generates the vPIR answer a, a^* for q, q^* using the database D' proving that it satisfies the following 1-local property P'. The property P' has the root rt hard-coded in it and given a location (i, j) and a database row containing a pair (r_i, r_j) and two authentication paths,

[7] The construction in [16] composes the batch argument with itself a constant number of times reducing the public key size to m^ϵ for any constant $\epsilon > 0$. We show how to get communication complexity and query generation time that are sub-polynomial in m (under super-polynomial hardness assumptions) by composing the vPIR with itself a super-constant number of times as in [10].

P' checks that the authentication paths are valid with respect to the root rt, and that the 2-local property P accepts the locations (i, j) and rows (r_i, r_j). The server's answer to the query (q, q^*) contains the root rt and the two vPIR answers (a, a^*). To verify the answer (rt, a, a^*), the client verifies both answers a, a^* of the underlying vPIR scheme. To decode the answer, the client decodes the underlying vPIR answer a and retrieves the row r_i. To verify that two answers $(\mathsf{rt}_1, a_1, a_1^*)$ and $(\mathsf{rt}_2, a_2, a_2^*)$ are consistent with the same database we check that $\mathsf{rt}_1 = \mathsf{rt}_2$.

Analysis. Since the queries of the underlying vPIR are hiding, so are the queries of the resulting vPIR. To argue security, consider two sets of queries (q_1, q_1^*) and (q_2, q_2^*) where the queries q_1, q_2 are for locations $(i_1, 1), (i_2, 1)$ respectively, and the dummy queries q_1^*, q_2^* are both for location $(1, 1)$. The adversary, playing the role of the server produces two answers $(\mathsf{rt}_1, a_1, a_1^*)$ and $(\mathsf{rt}_2, a_2, a_2^*)$. We need to show that if the adversary's answers are consistent with each other and each answer passes verification, then the decrypted rows r_{i_1}, r_{i_2} satisfy P. To argue this, we first move to a hybrid experiment where the dummy query q_1^* is for location (i_1, i_2) instead of $(1, 1)$. Since the query q_1^* is generated independently of the queries q_1, q_2, it must remain hiding even given the decryption keys for q_1 and q_2. Moreover, q_1^* is hiding even given the verification keys for all queries. Therefore, it follows that the decrypted rows r_{i_1}, r_{i_2} satisfy P with roughly the same probability in the original security game and in the hybrid experiment. Let $(r_{i_1}^*, r_{i_2}^*)$ be the rows decrypted from the answer a_1^* in the hybrid experiment. If the two answers are consistent with each other then $\mathsf{rt}_1 = \mathsf{rt}_2$. Since each answer passes verification, it follows from the security of the underlying vPIR that the decoded answers a_1, a_2 and a_1^* all satisfy the 1-local property P'. Therefore, it must be that $(r_{i_1}^*, r_{i_2}^*) = (r_{i_1}, r_{i_2})$ or we found a hash collision. Moreover, Since the decoded answer a_1^* satisfies P', the pair $(r_{i_1}^*, r_{i_2}^*)$ satisfies P as required.

2.2 vPIR for Global Properties

Next, we describe our construction of vPIR for any global property decidable by a read-once machine with a bounded length state.

Main challenge. Before describing our construction, we discuss the main technical challenge in proving simulation security for vPIR. Given an adversary playing the role of the server we need to design a simulator that generates a full database and submits it to the trusted party such that the output of the real and ideal experiments are indistinguishable. If we assume, for simplicity, that the adversary returns an accepting answer with probability 1, then the database submitted by the simulator must always satisfy the global property P. Additionally, for every location i, the distribution of the i-th row in the submitted database must be indistinguishable from the distribution of the adversary's decoded answer given a random query for location i.

In the context of secure computation, the simulator typically extracts the adversary's input (in our case, the database) by using the adversary as a black

box. In the case of a vPIR, this approach is problematic: the simulator can execute the adversary with some query, and obtain an answer. However, since the answer is shorter than the database, the simulator cannot possibly extract the entire database from a single answer. It may try to rewind the adversary, execute it with multiple queries and obtain multiple answers. The simulator can extract a small piece of the database from each answer and then try to put these pieces together to create a full database. The problem is that the adversary may use a database that is chosen adaptively, as a function of the query. In particular, the adversary may produce each answer based on a completely different database. In this case, the database reconstructed by the simulator may be different than any database the adversary would ever use and, in particular, it may not satisfy the required property. While we do not know how to extract the "real" database used by the adversary, we show how to carefully put together the extracted pieces and reconstruct some database that makes the output of the real and ideal experiments indistinguishable.

To demonstrate this idea, we start with a warm up: We show that any PIR scheme is also a vPIR scheme for the trivial property that accepts any database. In the real experiment, the client queries the adversary on some location j and decodes the answer r_j. In the ideal experiment, our simulator first queries the adversary on every location $i \in [m]$ and decrypts the answer $\tilde{r}_i$. The simulator puts all the answers together in a single database that contains $\tilde{r}_i$ in location i and submits it to the trusted party. Since the property is always satisfied, the trusted party always outputs $\tilde{r}_j$. The simulation is valid since the outputs r_j and $\tilde{r}_j$ are identically distributed.

We highlight that even in this simple warm up, we are already relying on fact that, following our definition of vPIR security, the output of the real experiment includes only the client's output and not the view of the adversary. In the real experiment, the adversary's view contains a random query q_j for location j and the database it uses (in particular, the row r_j) may depend on q_j. The simulator, however, does not know j and therefore it is not clear how it could simulate a query $\tilde{q}$ such that (q_j, r_j) is indistinguishable from $(\tilde{q}, \tilde{r}_j)$.

Simulation for non-trivial properties. The argument above clearly fails for non-trivial properties. Indeed, not every PIR scheme is also a vPIR scheme for non-trivial properties and, therefore, we propose a different construction. The main tool we use to construct vPIR for global properties is a vPIR scheme for 2-local properties. We start from either designated verifier or publicly verifiable vPIR scheme and construct a vPIR of the same type.

Fix a property that can be decided by a polynomial-time read-once Turing machine P with state of length S. We assume that our underlying vPIR for 2-local properties is Λ-secure where $|P|, S \leq \log \Lambda$. To query the database at location i, the client uses the underlying vPIR scheme for 2-local properties to generate two queries: a query q for location i and another dummy query q^* for location 1. Given a database $D = (r_1, \ldots, r_m)$ the server emulates the execution of the global property P on D and obtains all the intermediate states of P as it reads D. Let c_0 be the initial state of P and let c_j be the state after reading

the first j rows. Then, the server constructs a new database D' with m rows where the j-th row contains c_{j-1} and r_j. The server generates the vPIR answer a, a^* for q, q^* using the database D' proving that it satisfies the following 2-local property P'. Roughly speaking, the property P' checks that each two rows in D' are consistent with a valid execution of P. In more details, given two locations j, j' and two rows $(c, r), (c', r')$ the property P' checks that:

- If $j = 1$ then c is the initial state c_0 of P.
- If $j = m$ then P with state c accepts after reading the row r .
- If $j = j' + 1$ then P with state c' transitions to state c after reading the row r'.

The server's answer to the query (q, q^*) contains the two vPIR answers (a, a^*). To verify the answer, the client verifies that the answers a, a^* of the underlying vPIR scheme are consistent with each other and that each answer passes verification. To decode the answer, the client decodes the underlying vPIR answer a and retrieves the row r_i.

Analysis. Since the queries of the underlying vPIR are hiding, so are the queries of the resulting vPIR. To argue security we demonstrate a simulator. Our simulation strategy is inspired by the analysis of the SNARGs of [1] for NP languages decidable by a read-once machine with state of bounded length. Translating their techniques to our setting, we can show that if the server's answers are accepting then with overwhelming probability there *exists* a database that satisfies the global property P. The additional challenge facing our simulation is to produce a database that, in addition, has the correct distribution.

We start with a high-level description of our simulation strategy. For simplicity, in this overview we assume that the property P is fixed non-adaptively and that the adversary's answers always pass verification. For every location $i < m$, we define a distribution D_i as follows: we generate a pair of queries (q, q^*) of the underlying vPIR scheme for locations i and $i + 1$ respectively. We send the query (q, q^*) to the prover and get back the answer (a, a^*). If the answer passes verification we decode both a and a^* and obtain the decoded answers $(c_{i-1}, r_i), (c_i, r_{i+1})$. The sample consists of (c_{i-1}, r_i, c_i). For $i = m$ we define D_i similarly, except that we set $c_{m+1} = \perp$.

By the security of the underlying vPIR we have that with overwhelming probability, the decoded answers satisfy the 2-local property P' and, therefore, P with state c_{i-1} transitions to state c_i after reading the row r_i. Thus, we can think of a sample (c_{i-1}, r_i, c_i) from D_i as one step of P. Given a step from D_i and a step from D_{i+1} we say that these steps connect if the value of c_i in both steps is the same. Given one step from each D_i, if each two consecutive steps connect then we can reconstruct a full accepting execution of P on some database. Note that if we sample a step from each D_i independently, these samples may not necessarily connect to a full execution of P. Nonetheless, we show how to sample one step from each D_i *in a correlated way* such that the steps connect to each other with high probability, without changing the marginal distribution of each individual step.

In more detail, using the fact that the vPIR queries hide the locations, we show that the marginal distributions of c_i in D_i and in D_{i+1} are close. Therefore, we can *couple* together the distributions $D_1, \ldots, D_m$ and get a joint distribution D over all m steps such that the marginals of D are exactly the distributions $D_1, \ldots, D_m$, and where steps i and $i+1$ connect with high probability for every i. It follows that a sample from D contains, with high probability, a full accepting execution of P on some database $(r_1, \ldots, r_m)$. Since the marginal distribution of the i-th step is exactly D_i it follows that r_i is indistinguishable from output of the client in the real experiment with location i. While it may be hard to sample from D, we show how to sample from a distribution $\tilde{D}$ that is close to it in time $\text{poly}(\Lambda)$. Our simulator samples from $\tilde{D}$, extracts the database $(r_1, \ldots, r_m)$ and submits it to the trusted party.

We proceed to describe the security proof in more detail. For every $i \in [m]$, let (C'_{i-1}, R_i, C_i) denote a random sample from D_i. By the security of the underlying vPIR, we have that except with probability $\Lambda^{-\omega(1)}$:

- C'_0 is the initial state of P.
- P with state C'_{m-1} accepts after reading the row R_m .
- P with state C'_{i-1} transitions to state C_i after reading the row R_i.

We argue that C_i and C'_i are Λ-indistinguishable. In fact, since the length of the state is at most $\log \Lambda$, C_i and C'_i are also $\Lambda^{-\omega(1)}$-close in statistical distance. We argue this in a sequence of hybrid experiments where in each hybrid we change the way the queries q, q^* are generated and the way we decode the state c_i from the answers a, a^*:

- In the first hybrid, (q, q^*) are for locations $(i, i+1)$ respectively, and we decode c_i from a^*. We have that c_i is distributed exactly like C_i.
- In the second hybrid, (q, q^*) are for locations $(i+1, i+1)$ respectively, and we decode c_i from a^*. Since the underlying vPIR is Λ-hiding, we have that c_i is Λ-indistinguishable from the previous hybrid.
- In the third hybrid, (q, q^*) are for locations $(i+1, i+1)$ respectively, and we decode c_i from a. By the security of the underlying vPIR, since we check that a and a^* are consistent we have that c_i is Λ-indistinguishable from the previous hybrid.
- In the forth hybrid, (q, q^*) are for locations $(i+1, i+2)$ respectively, and we decode c_i from a. Since the underlying vPIR is Λ-hiding, we have that c_i is Λ-indistinguishable from the previous hybrid and it is distributed exactly like C'_i.

The simulator works as follows: For every i the simulator first obtains L independent samples from D_i for some sufficiently large L. Let $\tilde{D}_i$ be the empirical distribution that picks one of these L samples at random and let $(\tilde{C}'_{i-1}, \tilde{R}_i, \tilde{C}_i)$ denote a random sample from $\tilde{D}_i$. Let $\epsilon = \Lambda^{-O(1)}$ be the required simulation accuracy and let $\epsilon' = \epsilon/m^2$. Since the length of the states is bounded by $\log \Lambda$, by setting $L = \text{poly}(\Lambda, 1/\epsilon)$ we can guarantee that (C'_{i-1}, C_i) and

$(\tilde{C}'_{i-1}, \tilde{C}_i)$ are ϵ'-close, except with probability at most $\mathrm{negl}(\Lambda)$ (over the L samples from D_i). Since we have that C_i and C'_i are $\Lambda^{-\omega(1)}$-close, it follows that $\tilde{C}_i$ and $\tilde{C}'_i$ must be $O(\epsilon')$-close. Therefore, there exists a joint distribution $\tilde{D}$ over $\tilde{D}_1 \times \cdots \times \tilde{D}_m$ whose marginals are just $\tilde{D}_1, \ldots, \tilde{D}_m$ such that for every i, a sample $(\tilde{C}'_{i-1}, \tilde{R}_i, \tilde{C}_i)_{i\in[m]}$ from $\tilde{D}$ satisfies $\tilde{C}_i = \tilde{C}'_i$ with probability at least $1 - O(\epsilon')$. By the union bound, we have that the database $(\tilde{R}_1, ..., \tilde{R}_m)$ satisfies P with probability at least $1 - O(m \cdot \epsilon')$. Moreover, since each distribution $\tilde{D}_i$ is just uniform over a list of length $L = \mathrm{poly}(\Lambda)$, we can also samples from $\tilde{D}$ in time $\mathrm{poly}(\Lambda)$. Finally, the simulator samples $(\tilde{C}'_{i-1}, \tilde{R}_i, \tilde{C}_i)_{i\in[m]}$ from $\tilde{D}$ and submits the database $(\tilde{R}_1, ..., \tilde{R}_m)$ to the trusted party.

It remains to show that for any location j, the output of the real and ideal experiments are indistinguishable. In the real experiment, the queries (q, q^*) are for locations $(j, 1)$ respectively. The client obtains the answers (a, a^*), decodes (c_{j-1}, r_j) from a and outputs r_j. Since the underlying vPIR is Λ-hiding, r_j is Λ-indistinguishable from R_j.

In the ideal experiment the simulator submits to the trusted party the database $(\tilde{R}_1, ..., \tilde{R}_m)$ where $\tilde{R}_j$ is distributed like $\tilde{D}_j$. Since each sample from $\tilde{D}_j$ is itself distributed like D_j we have that $\tilde{R}_j$ and R_j are identically distributed. The output of the ideal experiment is either $\tilde{R}_j$ or $\perp$ in case the database $(\tilde{R}_1, ..., \tilde{R}_m)$ does not satisfy P. Since this happens with probability at most $O(m \cdot \epsilon') \leq \epsilon/2$ it follows that any $\mathrm{poly}(\Lambda)$-size distinguish has advantage at most ϵ in distinguishing the output of the real and ideal experiment.

3 Preliminaries

Parts of this section are taken verbatim from [10] and [6].

Vectors. For a set U, vector $\mathbf{v} = (v_1, ..., v_n) \in U^n$ and index $t \in [n]$ let $\mathbf{v}_t$ denote the element v_t. For a vector of indexes $\mathbf{t} \in [n]^\ell$ denote $\mathbf{v}[\mathbf{t}]$ be the vector $(\mathbf{v}_{\mathbf{t}_1}, ..., \mathbf{v}_{\mathbf{t}_\ell})$.

The universal language. Let $\mathcal{L}_U$ be the language of all triplets (Γ, x, y, T) such that Γ is a description of a Turing machine that on input x outputs y in T steps. We write $(\Gamma, x, T) \in \mathcal{L}_U$ as a shorthand for $(\Gamma, x, 1, T) \in \mathcal{L}_U$, i.e., Γ accepts x in T steps.

3.1 Private Information Retrival

In this section we define private information retrieval (PIR) schemes.

A private information retrieval scheme consists of the algorithms:

$$(\mathsf{PIR.Q}, \mathsf{PIR.A}, \mathsf{PIR.D}) \ ,$$

with the following syntax:

Query: The randomized query algorithm $\mathsf{PIR.Q}$ takes as input the security parameter κ, the number of database rows m, the row size w, and an index $t \in [m]$. It outputs a decryption key dk and a query q.

Answer: The deterministic answer algorithm $\mathsf{PIR.A}$ takes as input a database $D \in \{0,1\}^{m \times w}$ and a query q. It outputs an answer a.

Decryption: The deterministic decryption algorithm $\mathsf{PIR.D}$ takes as input the decryption key dk and an answer a. It outputs a row $r \in \{0,1\}^w$.

Definition 1. *A Λ-secure poly-logarithmic private information retrieval scheme satisfies the following requirements:*

Completeness. *For every $\kappa \in \mathbb{N}$, $m, w \leq 2^\kappa$, database $D \in \{0,1\}^{m \times w}$, and query $t \in [m]$:*

$$\Pr\left[\mathsf{PIR.D}(\mathsf{dk}, a) = D[t] \;\middle|\; \begin{array}{l} (\mathsf{dk}, q) \leftarrow \mathsf{PIR.Q}(\kappa, m, w, t) \\ a \leftarrow \mathsf{PIR.A}(D, q) \end{array}\right] = 1 \ .$$

Efficiency. *In the completeness experiment above:*

- *The query algorithm runs in time* $\mathrm{poly}(\kappa)$.
- *The answer algorithm runs in time* $\mathrm{poly}(\kappa, m, w)$.
- *The decryption algorithm runs in time* $w \cdot \mathrm{poly}(\kappa, \log(m))$.

Λ-Privacy. *For every* $\mathrm{poly}(\Lambda)$*-size adversary* Adv *and functions $m, w \leq \Lambda$ there exist a negligible function μ such that for every $\kappa \in \mathbb{N}$ and $t_0, t_1 \in [m]$:*

$$\Pr\left[b' = b \;\middle|\; \begin{array}{l} b \leftarrow \{0,1\} \\ (\mathsf{dk}_0, q_0) = \mathsf{PIR.Q}(\kappa, m, w, t_0) \\ (\mathsf{dk}_1, q_1) = \mathsf{PIR.Q}(\kappa, m, w, t_1) \\ b' \leftarrow \mathsf{Adv}(\mathsf{pp}, q_b) \end{array}\right] \leq \frac{1}{2} + \mu(\Lambda(\kappa)) \ .$$

3.2 Batch Arguments

In this section we define non-interactive batch arguments for the index language. The definition is adapted from [6] (see discussion following Definition 2). In a batch argument for the index language, the statement is given by a Turing machine M and the number of instances m. The prover convinces the verifier that for every $i \in [m]$ there exists a witness w_i such that $M(i, w_i)$ accepts. We fix the parameter m as well as the running time, description size, and witness length of M in setup time.

A non-interactive batch argument for the index language consist of algorithms:

$$(\mathsf{BA.S}, \mathsf{BA.T}, \mathsf{BA.P}, \mathsf{BA.V}, \mathsf{BA.E}) \ ,$$

with the following syntax:

Setup: The randomized setup algorithm $\mathsf{BA.S}$ takes as input the security parameter κ, the running time T and size N of a Turing machine, the number of instances m, and the witness length l. It outputs a prover key pk, and a verifier key vk.

Trapdoor Setup: The randomized trapdoor setup algorithm $\mathsf{BA.T}$ takes as input the security parameter κ, the running time T and size N of a Turing machine, the number of instances m, the witness length l and an index $i \in [m]$. It outputs prover key pk, verifier key vk, and a decryption key dk.

Prover: The deterministic prover algorithm $\mathsf{BA.P}$ takes as input the prover key pk, a Turing machine M and witnesses $w_1, \ldots, w_m \in \{0,1\}^l$. It outputs a proof Π.

Verifier: The deterministic verifier algorithm $\mathsf{BA.V}$ takes as input the verifier key vk, a Turing machine M, and a proof Π. It outputs an acceptance bit.

Extractor: The deterministic extraction algorithm $\mathsf{BA.E}$ takes as input the decryption key dk and a proof Π. It outputs a witness $w \in \{0,1\}^l$.

Definition 2. *A Λ-secure non-interactive batch argument for the index language satisfies the following requirements:*

Completeness. *For every functions $m, l, T, N \leq 2^\kappa$ there exists a negligible function μ such that for every $\kappa \in \mathbb{N}$, witnesses $w_1, \ldots, w_m \in \{0,1\}^l$, index $t \in [m]$, and Turing machine $M \in \{0,1\}^N$ such that $\forall i \in [m]$: $(M, (i, w_i), T) \in \mathcal{L}_U$:*

$$\Pr\left[\begin{matrix}\mathsf{BA.E}(\mathsf{dk}, \Pi) = w_t \\ \mathsf{BA.V}(\mathsf{vk}, M, \Pi) = 1\end{matrix} \;\middle|\; \begin{matrix}(\mathsf{pk}, \mathsf{vk}, \mathsf{dk}) \leftarrow \mathsf{BA.T}(\kappa, T, N, m, l, t) \\ \Pi \leftarrow \mathsf{BA.P}(\mathsf{pk}, M, (w_1, \ldots, w_m))\end{matrix}\right] = 1 \ .$$

Efficiency. *In the completeness experiment above:*

- *The prover algorithm runs in time* $\mathrm{poly}(\kappa, T, N, m, l)$.
- *The extraction algorithm runs in polynomial time in its input length.*
- *The verifier and decryption keys are of size* $\mathrm{poly}(\kappa, l)$.

We define the following efficiency measures of the scheme:

- *Let $T_{\mathsf{S}}(\kappa, T, N, m, l)$ be the running time of the setup and trapdoor setup algorithms.*
- *Let $T_{\mathsf{V}}(\kappa, T, N, m, l)$ be the running time of the verification algorithm.*
- *Let $L_{\mathsf{pk}}(\kappa, T, N, m, l)$ be the size of the prover key.*
- *Let $L_\Pi(\kappa, T, N, m, l)$ be the size of the proof.*

Λ-Key Indistinguishability. *For every* $\mathrm{poly}(\Lambda(\kappa))$*-size adversary* Adv *and functions $T, N, m, l \leq \Lambda$ there exists a negligible function μ such that for every $\kappa \in \mathbb{N}$ and index $i \in [m]$:*

$$\left|\begin{matrix}\Pr\left[\mathsf{Adv}(\mathsf{pk}, \mathsf{vk}) = 1 \mid (\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{BA.S}(\kappa, T, N, m, l)\right] \\ -\Pr\left[\mathsf{Adv}(\mathsf{pk}, \mathsf{vk}) = 1 \mid (\mathsf{pk}, \mathsf{vk}, \mathsf{dk}) \leftarrow \mathsf{BA.T}(\kappa, T, N, m, l, i)\right]\end{matrix}\right| \leq \mu(\Lambda(\kappa)) \ .$$

Λ-Somewhere Argument of Knowledge. *For every* $\mathrm{poly}(\Lambda(\kappa))$*-size adversary* Adv *and functions $T, N, m, l \leq \Lambda$ there exists a negligible function μ such that for every $\kappa \in \mathbb{N}$ and index $i \in [m]$:*

$$\Pr\left[\begin{matrix}\mathsf{BA.V}(\mathsf{vk}, M, \Pi) = 1 \\ (M, (i, w), T) \notin \mathcal{L}_U\end{matrix} \;\middle|\; \begin{matrix}(\mathsf{pk}, \mathsf{vk}, \mathsf{dk}) \leftarrow \mathsf{BA.T}(\kappa, T, N, m, l, i) \\ (M, \Pi) \leftarrow \mathsf{Adv}(\mathsf{pk}) \\ w \leftarrow \mathsf{BA.E}(\mathsf{dk}, \Pi)\end{matrix}\right] \leq \mu(\Lambda(\kappa)) \ .$$

If the somewhere argument of knowledge holds even when Adv *is given the verifier key* vk *then we say that the batch argument is publicly verifiable.*

We discuss some differences between Definition 2 and the definition in [6]:

- Our completeness requirement is stronger than the one in [6]. In addition to the verifier accepting, we also require that the extraction algorithm outputs the same witness used to generate the proof.
- The work of [6] only defines publicly verifiable batch arguments while we also consider a designated verifier version. We emphasize that in both the publicly verifiable and the designated verifier setting, key indistinguishability is required to hold even given the verification key.

We construct batch arguments for the index language based on PIR. This extends the result of [4] that obtain the weaker notion of batch arguments for NP (and not for the index language).

Theorem 5 (Batch Argument from PIR). *If there exists a Λ-secure poly-logarithmic PIR, then there exists a Λ-secure non-interactive batch argument for the index language with the following efficiency:*

$$\begin{aligned} T_{\mathsf{S}} &= l \cdot \mathrm{poly}(\kappa, N) \\ T_{\mathsf{V}} &= (N + l) \cdot \mathrm{poly}(\kappa) \\ L_{\mathsf{pk}} &= (N + l) \cdot \mathrm{poly}(\kappa) \\ L_{\Pi} &= (N + l) \cdot \mathrm{poly}(\kappa) \end{aligned}$$

The proof of this theorem appears in the full version of this work.

Theorem 6 (Publicly Verifiable Batch Argument from LWE [6]). *If the Learning with Errors problem is Λ-hard, then there exists a Λ-secure publicly verifiable non-interactive batch argument for the index language with the following efficiency:*

$$\begin{aligned} T_{\mathsf{S}} &= \mathrm{poly}(\kappa, l) \\ T_{\mathsf{V}} &= N \cdot \mathrm{poly}(\kappa, l) \\ L_{\mathsf{pk}} &= \mathrm{poly}(\kappa, l) \\ L_{\Pi} &= \mathrm{poly}(\kappa, l) \end{aligned}$$

Theorem 7 (Publicly Verifiable Batch Argument from Paring [16]). *If Λ-hardness of the k-Lin assumption (for any $k > 1$) holds in a prime-order pairing group, then for every $\epsilon > 0$ there exists a Λ-secure publicly verifiable non-interactive batch argument for the index language with the following efficiency:*

$$\begin{aligned} T_{\mathsf{S}} &= \mathrm{poly}(\kappa, m) \\ T_{\mathsf{V}} &= \mathrm{poly}(\kappa, T, N, l) \\ L_{\mathsf{pk}} &= m^{\epsilon} \cdot \mathrm{poly}(\kappa) \\ L_{\Pi} &= \mathrm{poly}(\kappa, T, N, l) \end{aligned}$$

We remark that the notion of batch arguments defined and constructed in [16] is slightly different than the notion in Definition 2. We explain how to modify the [16] construction to obtain our notion:

- In [16], the prover and verifier are given a circuit implementing the NP verification procedure, while in our notion, the NP verification procedure is given by a Turing machine. As discussed in [6, Section 6], the construction of [16] can be modified to work with a Turing machine verification by composing it with a RAM delegation scheme such as the one constructed in [16] under the k-Lin assumption.
- The construction in [16] has a negligible completeness error. The extraction algorithm may fail to extract the witness with negligible probability over the choice of the public key. However, if we slightly modify the trapdoor setup algorithm to only sample "good" public keys that do not lead to extraction failure we get a construction with perfect completeness.

4 Verifiable PIR

In this section we define the notion of verifiable PIR. We consider two security definition: simulation security and a game-based definition that we call local security.

Let κ be the security parameter. In what follows we consider a database with $m(\kappa)$ rows, where each row is a bit string of length $w(\kappa)$. Let $\mathcal{U} = \mathcal{U}_\kappa$ be a set of constraint-checking Turing machines. Each $U \in \mathcal{U}$ takes as input a constraint description Γ and a database D and outputs 1 if and only if D satisfies the constraint Γ.

A verifiable PIR (vPIR) scheme for $\mathcal{U}$ is given by algorithms:

$$(\mathsf{vPIR.S}, \mathsf{vPIR.Q}, \mathsf{vPIR.A}, \mathsf{vPIR.D}, \mathsf{vPIR.V}) \ ,$$

with the following syntax:

Setup: The randomized setup algorithm $\mathsf{vPIR.S}$ takes as input a security parameter κ. It outputs public parameters pp. If pp are always empty we say that the protocol has no setup.

Query: The randomized query algorithm $\mathsf{vPIR.Q}$ takes as input the security parameter κ, the public parameters pp, a machine $U \in \mathcal{U}$, database dimensions m and w, and an index $t \in [m]$. It outputs a decryption key dk, a verifier key vk, and a query q.

Answer: The deterministic answer algorithm $\mathsf{vPIR.A}$ takes as input the public parameters pp, a database $D \in \{0,1\}^{m \times w}$, a constraint Γ, and a query q. It outputs an answer a.

Decryption: The deterministic decryption algorithm $\mathsf{vPIR.D}$ takes as input the decryption key dk and an answer a. It outputs a row $r \in \{0,1\}^w$.

Verification: The deterministic verification algorithm $\mathsf{vPIR.V}$ takes as input a constraint Γ, a verifier key vk and an answer a. It outputs a bit indicating if it accepts or rejects.

Multiple queries. We also define a version of the verification algorithm $\mathsf{vPIR.V}$ for multiple answers. In this setting we allow multiple users to jointly verify the answers to all their queries without sharing their verification keys. To this end, each user invokes the verification algorithm with all the answers but only its own verification key. The verification algorithm is also given the index of the answer that matches the input verification key.

Verification (multiple queries): The deterministic verification algorithm $\mathsf{vPIR.V}$ takes as input a constraint Γ, a verifier key vk, a vector $\mathbf{a}$ of d answers and an index $i \in [d]$. It outputs a bit indicating if it accepts or rejects.

We also use the following shorthand for working with $d > 1$ queries:

- $\mathsf{vPIR.VQ}(\kappa, \mathsf{pp}, U, m, w, \mathbf{t})$ stands for $(\mathbf{dk}, \mathbf{vk}, \mathbf{q})$ such that $\forall i \in [d] : (\mathbf{dk}_i, \mathbf{vk}_i, \mathbf{q}_i) = \mathsf{vPIR.Q}(\kappa, \mathsf{pp}, U, m, w, \mathbf{t}_i)$.
- $\mathsf{vPIR.VA}(\mathsf{pp}, \Gamma, D, \mathbf{q})$ stands for $\mathbf{a}$ such that $\forall i \in [d] : \mathbf{a}_i = \mathsf{vPIR.A}(\mathsf{pp}, \Gamma, D, \mathbf{q}_i)$.
- $\mathsf{vPIR.VD}(\mathbf{dk}, \mathbf{a})$ stands for $\mathbf{r}$ such that $\forall i \in [d] : \mathbf{r}_i = \mathsf{vPIR.D}(\mathbf{dk}_i, \mathbf{a}_i)$.
- $\mathsf{vPIR.VV}(\Gamma, \mathbf{vk}, \mathbf{a})$ stands for $\bigwedge_{i \in [d]} \mathsf{vPIR.V}(\Gamma, \mathbf{vk}_i, \mathbf{a}, i)$.

Next we define the properties required by a vPIR scheme. In Definition 3 we give the completeness, efficiency and privacy properties. We consider two different definitions for security against malicious server: a simulation based definition with a single query (Definition 4) and a game based definition for multiple queries (Definition 5).

Definition 3. *Let $\mathcal{U}$ be a set of Turing machines. A vPIR scheme for $\mathcal{U}$ satisfies the following requirements:*

Completeness. *For every $m, w, d \leq 2^\kappa$ there exists a negligible function μ such that for every $\kappa \in \mathbb{N}$, machine $U \in \mathcal{U}$, database $D \in \{0,1\}^{m \times w}$, vector of queries $\mathbf{t} \in [m]^d$, and constraint Γ such that $U(\Gamma, D) = 1$:*

$$\Pr\left[\begin{array}{l|l} \mathsf{vPIR.VD}(\mathbf{dk}, \mathbf{a}) = D[\mathbf{t}] & \mathsf{pp} \leftarrow \mathsf{vPIR.S}(\kappa) \\ \mathsf{vPIR.VV}(\Gamma, \mathbf{vk}, \mathbf{a}) = 1 & (\mathbf{dk}, \mathbf{vk}, \mathbf{q}) \leftarrow \mathsf{vPIR.VQ}(\kappa, \mathsf{pp}, U, m, w, \mathbf{t}) \\ & \mathbf{a} \leftarrow \mathsf{vPIR.VA}(\mathsf{pp}, \Gamma, D, \mathbf{q}) \end{array}\right] = 1 \ .$$

Efficiency. *In the completeness experiment above:*

- *The setup algorithm runs in time* $\mathrm{poly}(\kappa)$.
- *The decryption algorithm runs in polynomial time in its input length.*
- *The answer algorithm runs time* $\mathrm{poly}(\kappa, m, w, T)$ *where T is the running time of $U(\Gamma, D)$.*

We define the following efficiency measures of the scheme:

- *Let $T_{\mathsf{Q}}(\kappa, U, m, w)$ be the running time of the query algorithm.*
- *Let $T_{\mathsf{V}}(\kappa, U, m, w)$ be the running time of the verification algorithm.*
- *Let $L_q(\kappa, U, m, w)$ be the size of the query.*

- *Let* $L_{\mathsf{dk}}(\kappa, U, m, w)$ *be the size of the decryption key.*
- *Let* $L_{\mathsf{vk}}(\kappa, U, m, w)$ *be the size of the verifier key.*
- *Let* $L_a(\kappa, U, m, w)$ *be the size of the answer.*

Λ-Privacy. *For every* $\mathrm{poly}(\Lambda)$*-size adversary* Adv *and functions* $m, w \leq \Lambda$ *there exists a negligible function* μ *such that for every* $\kappa \in \mathbb{N}$*, machine* $U \in \mathcal{U}$*, and* $t_1, t_2 \in [m]$*:*

$$\Pr\left[i' = i \;\middle|\; \begin{array}{l} i \leftarrow [2] \\ \mathsf{pp} \leftarrow \mathsf{vPIR.S}(\kappa) \\ (\mathbf{dk}, \mathbf{vk}, \mathbf{q}) = \mathsf{vPIR.VQ}(\kappa, \mathsf{pp}, U, m, w, (t_1, t_2)) \\ i' \leftarrow \mathsf{Adv}(\mathsf{pp}, \mathbf{vk}_i, \mathbf{q}_i) \end{array} \right] \leq \frac{1}{2} + \mu(\Lambda(\kappa)) \ .$$

4.1 Simulation Security

We give a simulation based security definition for vPIR. In this work we only consider simulation security for a single query. However, the definition can be extended to multiple queries.

Definition 4 (vPIR Security). *Let* $\mathcal{U}$ *be a set of Turing machines, let* $U = \{U_\kappa \in \mathcal{U}\}_{\kappa \in \mathbb{N}}$ *and let* $\Lambda(\kappa), m(\kappa), w(\kappa)$ *be functions. A vPIR scheme for* $\mathcal{U}$ *satisfying Definition 3 is* Λ*-secure with respect to* (U, m, w) *if for every* $\mathrm{poly}(\Lambda)$*-size adversary* Adv*, and polynomial* P *there exists a* $\mathrm{poly}(\Lambda)$*-size simulator* Sim *such that for every* $\mathrm{poly}(\Lambda)$*-size distinguisher* D*,* $\kappa \in \mathbb{N}$*, and* $t \in [m]$*:*

$$|\Pr[\mathsf{D}(\mathrm{Real}_{\mathsf{Adv}}(\kappa, t) = 1] - \Pr[\mathsf{D}(\mathrm{Ideal}_{\mathsf{Sim}}(\kappa, t)) = 1]| \leq \frac{1}{P(\Lambda(\kappa))} \ .$$

where the experiments $\mathrm{Real}_{\mathsf{Adv}}(\kappa, t)$ *and* $\mathrm{Ideal}_{\mathsf{Sim}}(\kappa, t)$ *are defined as follows:*

$\mathrm{Real}_{\mathsf{Adv}}(\kappa, t)$**:**

- *Sample parameters* $\mathsf{pp} \leftarrow \mathsf{vPIR.S}(\kappa)$*.*
- *Generate a query* $(\mathsf{dk}, \mathsf{vk}, q) \leftarrow \mathsf{vPIR.Q}(\kappa, \mathsf{pp}, U_\kappa, m, w, t)$*.*
- *Run the adversary and obtain* $(\Gamma, a) \leftarrow \mathsf{Adv}(\mathsf{pp}, q)$*.*
- *If* $\mathsf{vPIR.V}(\Gamma, \mathsf{vk}, a) = 0$ *output* $(\Gamma, \perp)$*. Otherwise output* $(\Gamma, \mathsf{vPIR.D}(\mathsf{dk}, a))$*.*

$\mathrm{Ideal}_{\mathsf{Sim}}(\kappa, t)$**:**

- *Run the simulator and obtain* $(\Gamma, D) \leftarrow \mathsf{Sim}(\kappa)$*.*
- *If* $D = \perp$ *or* $U_\kappa(\Gamma, D) = 0$ *output* $(\Gamma, \perp)$*. Otherwise output* $(\Gamma, D[t])$*.*

If the above holds even when Adv *is given the verifier key* vk *then we say that the vPIR is publicly verifiable.*

4.2 Local Security

In the multi-query setting, we introduce the notion of ℓ-local vPIR that satisfies a game-based security definition. In local vPIR, instead of verifying a global constraint on the database, we verify that each set of ℓ database rows satisfy some local constraint. In more detail, we say that the a set of constraint-checking Turing machines $\mathcal{U}$ is ℓ-local if for every $U \in \mathcal{U}$ there exists a machine denoted by $\tilde{U}$, such that $U(\Gamma, D) = 1$ if and only if $\tilde{U}(\Gamma, (\mathbf{t}, D[\mathbf{t}])) = 1$ for every $\mathbf{t} \in [m]^\ell$.

Definition 5 (ℓ-local vPIR security). *Let $\mathcal{U}$ be an ℓ-local set of Turing machines. Let $U = \{U_\kappa \in \mathcal{U}\}_{\kappa\in\mathbb{N}}$ and let $\Lambda(\kappa), m(\kappa), w(\kappa)$ be functions. An ℓ-local vPIR scheme for $\mathcal{U}$ satisfying Definition 3 is Λ-secure for (U, m, w) if for every* $\mathrm{poly}(\Lambda)$*-size adversary* Adv *there exist a negligible function μ such that for every $\kappa \in \mathbb{N}$ and for every $\mathbf{t} \in [m]^\ell$:*

$$\Pr\left[\begin{array}{l|l} \mathsf{vPIR.VV}(\Gamma, \mathbf{vk}, \mathbf{a}) = 1 & \mathsf{pp} \leftarrow \mathsf{vPIR.S}(\kappa) \\ \tilde{U}_\kappa(\Gamma, (\mathbf{t}, \mathbf{r})) = 0 & (\mathbf{dk}, \mathbf{vk}, \mathbf{q}) \leftarrow \mathsf{vPIR.VQ}(\kappa, \mathsf{pp}, U_\kappa, m, w, \mathbf{t}) \\ & (\Gamma, \mathbf{a}) \leftarrow \mathsf{Adv}(\mathsf{pp}, \mathbf{q}) \\ & \mathbf{r} \leftarrow \mathsf{vPIR.VD}(\mathbf{dk}, \mathbf{a}) \end{array}\right] \leq \mu(\Lambda(\kappa)) \ .$$

If the above holds even when Adv *is given the verifier keys* $\mathbf{vk}$ *then we say that the vPIR is publicly verifiable.*

In what follows we consider the set $\mathcal{U}_\kappa^\ell = \{U_{T,N}\}_{T,N\leq 2^\kappa}$ where $\tilde{U}_{T,N}(\Gamma, (\mathbf{t}, D[\mathbf{t}])) = 1$ if and only if Γ is a description of a Turing machine of length N and $(\Gamma, (\mathbf{t}, D[\mathbf{t}]), T) \in \mathcal{L}_U$ for every $\mathbf{t} \in [m]^\ell$. We say that an ℓ-local vPIR for $\mathcal{U}^\ell$ is Λ-secure if for every functions $T, N, m, w \leq \Lambda$ the vPIR is Λ-secure and Λ-private for $(U_{T,N}, m, w)$.

5 From Batch Arguments to 1-Local vPIR

In this section we give a construction of a publicly verifiable 1-local vPIR from batch arguments.

Theorem 8. *If there exists a Λ-secure non-interactive batch argument for the index language with efficiency $(T_{\mathsf{S}}^{\mathsf{BA}}, T_{\mathsf{V}}^{\mathsf{BA}}, L_{\mathsf{pk}}^{\mathsf{BA}}, L_{\Pi}^{\mathsf{BA}})$ then there exists a Λ-secure 1-local vPIR for $\mathcal{U}^1$ with no setup and with the following efficiency:*

$$\begin{aligned} T_{\mathsf{Q}}(\kappa, U_{T,N}, m, w) &= T_{\mathsf{S}}^{\mathsf{BA}}(\kappa, T, N, m, w) \\ T_{\mathsf{V}}(\kappa, U_{T,N}, m, w) &= T_{\mathsf{V}}^{\mathsf{BA}}(\kappa, T, N, m, w) \\ L_q(\kappa, U_{T,N}, m, w) &= L_{\mathsf{pk}}^{\mathsf{BA}}(\kappa, T, N, m, w) \\ L_{\mathsf{dk}}(\kappa, U_{T,N}, m, w) &= \mathrm{poly}(\kappa, w) \\ L_{\mathsf{vk}}(\kappa, U_{T,N}, m, w) &= \mathrm{poly}(\kappa, w) \\ L_a(\kappa, U_{T,N}, m, w) &= L_{\Pi}^{\mathsf{BA}}(\kappa, T, N, m, w) \end{aligned}$$

Moreover, if the non-interactive batch argument is publicly verifiable then the 1-local vPIR is publicly verifiable.

In the full version of this work we show how to transform a publicly verifiable 1-local vPIR into one with shorter query. Combined with Theorems 7 and 8 we get the following theorem.

Theorem 9. *If Λ-hardness of the k-Lin assumption (for any $k > 1$) holds in a prime-order pairing group, then for every $\epsilon > 0$ there exists a Λ-secure 1-local publicly verifiable vPIR for $\mathcal{U}^1$ with no setup and with the following efficiency:*

$$\begin{aligned}
T_{\mathsf{Q}}(\kappa, U_{T,N}, m, w) &= \mathrm{poly}(T, N, w) \cdot 2^{\sqrt{O(\log \kappa \log m)}} \\
T_{\mathsf{V}}(\kappa, U_{T,N}, m, w) &= \mathrm{poly}(N, w) \cdot 2^{\sqrt{O(\log \kappa \log m)}} \\
L_q(\kappa, U_{T,N}, m, w) &= (\mathrm{poly}(N, w) + T^{\epsilon}) \cdot 2^{\sqrt{O(\log \kappa \log m)}} \\
L_{\mathsf{dk}}(\kappa, U_{T,N}, m, w) &= \mathrm{poly}(N, w) \cdot 2^{\sqrt{O(\log \kappa \log m)}} \\
L_{\mathsf{vk}}(\kappa, U_{T,N}, m, w) &= \mathrm{poly}(N, w) \cdot 2^{\sqrt{O(\log \kappa \log m)}} \\
L_a(\kappa, U_{T,N}, m, w) &= \mathrm{poly}(N, w) \cdot 2^{\sqrt{O(\log \kappa \log m)}}
\end{aligned}$$

The proof of Theorems 8 and Theorem 9 appears in the full version of this work.

6 From 1-Local vPIR to vPIR with Constant Locality

In this section we transform a 1-local vPIR into an ℓ-local vPIR for any constant ℓ.

Theorem 10. *If there exists a Λ-secure family of collision resistant hash functions and a Λ-secure 1-local vPIR for $\mathcal{U}^1$ with efficiency $(T'_{\mathsf{Q}}, T'_{\mathsf{V}}, L'_q, L'_{\mathsf{dk}}, L'_{\mathsf{vk}}, L'_a)$, then for every $\ell \in \mathbb{N}$ there exists a Λ-secure ℓ-local vPIR for $\mathcal{U}^\ell$ with the following efficiency:*

$$\begin{aligned}
T_{\mathsf{Q}}(\kappa, U_{T,N}, m, w) &= 2 \cdot T'_{\mathsf{Q}}(\kappa, U_{T',N'}, m', w') + \mathrm{poly}(\kappa) \\
T_{\mathsf{V}}(\kappa, U_{T,N}, m, w) &= 2 \cdot T'_{\mathsf{V}}(\kappa, U_{T',N'}, m', w') + w \cdot \mathrm{poly}(\kappa) \\
L_q(\kappa, U_{T,N}, m, w) &= 2 \cdot L'_q(\kappa, U_{T',N'}, m', w') \\
L_{\mathsf{dk}}(\kappa, U_{T,N}, m, w) &= L'_{\mathsf{dk}}(\kappa, U_{T',N'}, m', w') \\
L_{\mathsf{vk}}(\kappa, U_{T,N}, m, w) &= 2 \cdot L'_{\mathsf{vk}}(\kappa, U_{T',N'}, m', w') + \mathrm{poly}(\kappa) \\
L_a(\kappa, U_{T,N}, m, w) &= 2 \cdot L'_a(\kappa, U_{T',N'}, m', w') + w \cdot \mathrm{poly}(\kappa)
\end{aligned}$$

where:

$$T' = T + \ell \cdot w \cdot \mathrm{poly}(\kappa)\ ,\ N' = N + w \cdot \mathrm{poly}(\kappa)\ ,\ m' = m^{\ell}\ ,\ w' = \ell \cdot w \cdot \mathrm{poly}(\kappa)\ .$$

Moreover, if the 1-local vPIR is publicly verifiable then the ℓ-local vPIR is publicly verifiable.

The proof of this theorem appears in the full version of this work.

7 From Local vPIR to Simulation Secure vPIR

In this section we give a vPIR scheme for the set $\mathcal{U}$ of global constraints that read the database once and maintain a bounded size state between rows. In more detail, let $\mathcal{U}_\kappa = \{U_{T,N,S}\}_{T,N,S\leq 2^\kappa}$ where $U_{T,N,S}(\Gamma, D = (r_i)_{i\in[m]}) = 1$ if and only if Γ is a description of a Turing machine of length N and for every $i \in [m]$ there exists $c_i \in \{0,1\}^S$ such that $(\Gamma, (c_{i-1}, r_i), c_i, T) \in \mathcal{L}_U$ where c_0 and c_m are some fixed starting and accepting configuration, respectively.

Theorem 11. *If there exists a Λ-secure 2-local vPIR for $\mathcal{U}^2$ with efficiency $(T'_{\mathsf{Q}}, T'_{\mathsf{V}}, L'_q, L'_{\mathsf{vk}}, L'_{\mathsf{dk}}, L'_a)$, then for every $N, S = O(\log \Lambda)$ and $m, w, T \leq \Lambda$ there exists a vPIR scheme for $\mathcal{U}$ that is Λ-secure and Λ-private with respect to $(U_{T,N,S}, m, w)$ with the following efficiency:*

$$\begin{aligned}
T_{\mathsf{Q}}(\kappa, U_{T,N,S}, m, w) &= 2 \cdot T'_{\mathsf{Q}}(\kappa, U_{N',T'}, m, w') \\
T_{\mathsf{V}}(\kappa, U_{T,N,S}, m, w) &= 2 \cdot T'_{\mathsf{V}}(\kappa, U_{N',T'}, m, w') \\
L_q(\kappa, U_{T,N,S}, m, w) &= 2 \cdot L'_q(\kappa, U_{N',T'}, m, w') \\
L_{\mathsf{dk}}(\kappa, U_{T,N,S}, m, w) &= L'_{\mathsf{dk}}(\kappa, U_{N',T'}, m, w') \\
L_{\mathsf{vk}}(\kappa, U_{T,N,S}, m, w) &= 2 \cdot L'_{\mathsf{vk}}(\kappa, U_{N',T'}, m, w') \\
L_a(\kappa, U_{T,N,S}, m, w) &= 2 \cdot L'_a(\kappa, U_{N',T'}, m, w')
\end{aligned}$$

where:

$$N' = N + O(1) \ , \quad T' = T + O(\log(m) + S + w) \ , \ \ w' = S + w.$$

Moreover, if the 2-local vPIR is publicly verifiable then the vPIR is publicly verifiable.

The proof of this theorem appears in the full version of this work.

The following is corollary of Theorems 5,8,10 and 11.

Corollary 1. *If there exists a Λ-secure PIR, then for every $N, S = O(\log \Lambda)$ and $m, w, T \leq \Lambda$ there exists a vPIR for $\mathcal{U}$ that is Λ-secure and Λ-private and Λ-private with respect to $(U_{T,N,S}, m, w)$ with the following efficiency:*

$$\begin{aligned}
T_{\mathsf{Q}}(\kappa, U_{T,N,S}, m, w) &= \mathrm{poly}(\kappa, w) \\
T_{\mathsf{V}}(\kappa, U_{T,N,S}, m, w) &= w \cdot \mathrm{poly}(\kappa) \\
L_q(\kappa, U_{T,N,S}, m, w) &= w \cdot \mathrm{poly}(\kappa) \\
L_{\mathsf{dk}}(\kappa, U_{T,N,S}, m, w) &= w \cdot \mathrm{poly}(\kappa) \\
L_{\mathsf{vk}}(\kappa, U_{T,N,S}, m, w) &= w \cdot \mathrm{poly}(\kappa) \\
L_a(\kappa, U_{T,N,S}, m, w) &= w \cdot \mathrm{poly}(\kappa)
\end{aligned}$$

The following is corollary of Theorems 6,8,10 and 11.

Corollary 2. *If Learning with Errors is Λ-hard, then for every $N, S = O(\log \Lambda)$ and $m, w, T \leq \Lambda$ there exists a publicly verifiable* vPIR *for $\mathcal{U}$ that is Λ-secure and Λ-private with respect to $(U_{T,N,S}, m, w)$ with the following efficiency:*

$$\begin{aligned}
T_{\mathsf{Q}}(\kappa, U_{T,N,S}, m, w) &= \mathrm{poly}(\kappa, w) \\
T_{\mathsf{V}}(\kappa, U_{T,N,S}, m, w) &= \mathrm{poly}(\kappa, w) \\
L_q(\kappa, U_{T,N,S}, m, w) &= \mathrm{poly}(\kappa, w) \\
L_{\mathsf{dk}}(\kappa, U_{T,N,S}, m, w) &= \mathrm{poly}(\kappa, w) \\
L_{\mathsf{vk}}(\kappa, U_{T,N,S}, m, w) &= \mathrm{poly}(\kappa, w) \\
L_a(\kappa, U_{T,N,S}, m, w) &= \mathrm{poly}(\kappa, w)
\end{aligned}$$

The following is corollary of Theorems 7,9,10 and 11.

Corollary 3. *If Λ-hardness of the k-Lin assumption (for any $k > 1$) holds in a prime-order pairing group, then for every $\epsilon > 0$, and for every $N, S = O(\log \Lambda)$ and $m, w, T \leq \Lambda$ there exists a publicly verifiable* vPIR *for $\mathcal{U}$ that is Λ-secure and Λ-private with respect to $(U_{T,N,S}, m, w)$ with the following efficiency:*

$$\begin{aligned}
T_{\mathsf{Q}}(\kappa, U_{T,N,S}, m, w) &= \mathrm{poly}\,(\kappa, T, w) \cdot 2^{\sqrt{O(\log \kappa \log m)}} \\
T_{\mathsf{V}}(\kappa, U_{T,N,S}, m, w) &= \mathrm{poly}\,(\kappa, w) \cdot 2^{\sqrt{O(\log \kappa \log m)}} \\
L_q(\kappa, U_{T,N,S}, m, w) &= (\mathrm{poly}(\kappa, w) + (T + w \cdot \mathrm{poly}(\kappa))^{\epsilon}) \cdot 2^{\sqrt{O(\log \kappa \log m)}} \\
L_{\mathsf{dk}}(\kappa, U_{T,N,S}, m, w) &= \mathrm{poly}\,(\kappa, w) \cdot 2^{\sqrt{O(\log \kappa \log m)}} \\
L_{\mathsf{vk}}(\kappa, U_{T,N,S}, m, w) &= \mathrm{poly}\,(\kappa, w) \cdot 2^{\sqrt{O(\log \kappa \log m)}} \\
L_a(\kappa, U_{T,N,S}, m, w) &= \mathrm{poly}\,(\kappa, w) \cdot 2^{\sqrt{O(\log \kappa \log m)}}
\end{aligned}$$

Acknowledgements. Yael Tauman Kalai's work is supported by DARPA under Agreement No. HR00112020023. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Government or DARPA.

Omer Paneth is member of the checkpoint institute of information security and is supported by an Azrieli Faculty Fellowship, Len Blavatnik and the Blavatnik Foundation, the Blavatnik Interdisciplinary Cyber Research Center at Tel Aviv University, and ISF grant 1789/19.

References

1. Badrinarayanan, S., Kalai, Y.T., Khurana, D., Sahai, A., Wichs, D.: Succinct delegation for low-space non-deterministic computation. In: Diakonikolas, I., Kempe, D., Henzinger, M. (eds.) Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018. pp. 709–721. ACM (2018). https://doi.org/10.1145/3188745.3188924, `https://doi.org/10.1145/3188745.3188924`

2. Bitansky, N., Canetti, R., Chiesa, A., Tromer, E.: Recursive composition and bootstrapping for SNARKS and proof-carrying data. In: Boneh, D., Roughgarden, T., Feigenbaum, J. (eds.) Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013. pp. 111–120. ACM (2013). https://doi.org/10.1145/2488608.2488623, https://doi.org/10.1145/2488608.2488623
3. Bitansky, N., Canetti, R., Paneth, O., Rosen, A.: On the existence of extractable one-way functions. SIAM J. Comput. **45**(5), 1910–1952 (2016). https://doi.org/10.1137/140975048, https://doi.org/10.1137/140975048
4. Brakerski, Z., Holmgren, J., Kalai, Y.T.: Non-interactive delegation and batch NP verification from standard computational assumptions. In: Hatami, H., McKenzie, P., King, V. (eds.) Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017. pp. 474–482. ACM (2017). https://doi.org/10.1145/3055399.3055497, https://doi.org/10.1145/3055399.3055497
5. Brakerski, Z., Kalai, Y.: Witness indistinguishability for any single-round argument with applications to access control. In: Kiayias, A., Kohlweiss, M., Wallden, P., Zikas, V. (eds.) Public-Key Cryptography - PKC 2020 - 23rd IACR International Conference on Practice and Theory of Public-Key Cryptography, Edinburgh, UK, May 4-7, 2020, Proceedings, Part II. Lecture Notes in Computer Science, vol. 12111, pp. 97–123. Springer (2020). https://doi.org/10.1007/978-3-030-45388-6_4, https://doi.org/10.1007/978-3-030-45388-6_4
6. Choudhuri, A.R., Jain, A., Jin, Z.: Snargs for P from LWE. IACR Cryptol. ePrint Arch. p. 808 (2021), https://eprint.iacr.org/2021/808
7. Gentry, C., Wichs, D.: Separating succinct non-interactive arguments from all falsifiable assumptions. In: Fortnow, L., Vadhan, S.P. (eds.) Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011. pp. 99–108. ACM (2011). https://doi.org/10.1145/1993636.1993651, https://doi.org/10.1145/1993636.1993651
8. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Prabhakaran, M., Sahai, A.: Efficient non-interactive secure computation. In: Paterson, K.G. (ed.) Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings. Lecture Notes in Computer Science, vol. 6632, pp. 406–425. Springer (2011). https://doi.org/10.1007/978-3-642-20465-4_23, https://doi.org/10.1007/978-3-642-20465-4_23
9. Kalai, Y.T., Paneth, O.: Delegating RAM computations. In: Hirt, M., Smith, A.D. (eds.) Theory of Cryptography - 14th International Conference, TCC 2016-B, Beijing, China, October 31 - November 3, 2016, Proceedings, Part II. Lecture Notes in Computer Science, vol. 9986, pp. 91–118 (2016). https://doi.org/10.1007/978-3-662-53644-5_4, https://doi.org/10.1007/978-3-662-53644-5_4
10. Kalai, Y.T., Paneth, O., Yang, L.: How to delegate computations publicly. In: Charikar, M., Cohen, E. (eds.) Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019. pp. 1115–1124. ACM (2019). https://doi.org/10.1145/3313276.3316411, https://doi.org/10.1145/3313276.3316411
11. Kalai, Y.T., Vaikuntanathan, V., Zhang, R.Y.: Somewhere statistical soundness, post-quantum security, and snargs. In: Nissim, K., Waters, B. (eds.) Theory of Cryptography - 19th International Conference, TCC 2021, Raleigh, NC, USA, November 8-11, 2021, Proceedings, Part I. Lecture Notes in Computer Science,

vol. 13042, pp. 330–368. Springer (2021). https://doi.org/10.1007/978-3-030-90459-3_12, https://doi.org/10.1007/978-3-030-90459-3_12

12. Kilian, J.: A note on efficient zero-knowledge proofs and arguments (extended abstract). In: Kosaraju, S.R., Fellows, M., Wigderson, A., Ellis, J.A. (eds.) Proceedings of the 24th Annual ACM Symposium on Theory of Computing, May 4-6, 1992, Victoria, British Columbia, Canada. pp. 723–732. ACM (1992). https://doi.org/10.1145/129712.129782, https://doi.org/10.1145/129712.129782
13. Kushilevitz, E., Ostrovsky, R.: Replication is NOT needed: SINGLE database, computationally-private information retrieval. In: 38th Annual Symposium on Foundations of Computer Science, FOCS '97, Miami Beach, Florida, USA, October 19-22, 1997. pp. 364–373. IEEE Computer Society (1997). https://doi.org/10.1109/SFCS.1997.646125, https://doi.org/10.1109/SFCS.1997.646125
14. Paneth, O., Rothblum, G.N.: On zero-testable homomorphic encryption and publicly verifiable non-interactive arguments. In: Kalai, Y., Reyzin, L. (eds.) Theory of Cryptography - 15th International Conference, TCC 2017, Baltimore, MD, USA, November 12-15, 2017, Proceedings, Part II. Lecture Notes in Computer Science, vol. 10678, pp. 283–315. Springer (2017). https://doi.org/10.1007/978-3-319-70503-3_9, https://doi.org/10.1007/978-3-319-70503-3_9
15. Valiant, P.: Incrementally verifiable computation or proofs of knowledge imply time/space efficiency. In: Canetti, R. (ed.) Theory of Cryptography, Fifth Theory of Cryptography Conference, TCC 2008, New York, USA, March 19-21, 2008. Lecture Notes in Computer Science, vol. 4948, pp. 1–18. Springer (2008). https://doi.org/10.1007/978-3-540-78524-8_1, https://doi.org/10.1007/978-3-540-78524-8_1
16. Waters, B., Wu, D.J.: Batch arguments for NP and more from standard bilinear group assumptions. IACR Cryptol. ePrint Arch. p. 336 (2022), https://eprint.iacr.org/2022/336