Aggelos Kiayias, Nikos Leonardos, Helger Lipmaa*, Kateryna Pavlyk, and Qiang Tang

Optimal Rate Private Information Retrieval from Homomorphic Encryption

Abstract: We consider the problem of minimizing the communication in single-database private information retrieval protocols in the case where the length of the data to be transmitted is large. We present first rate-optimal protocols for 1-out-of-n computationally-private information retrieval (CPIR), oblivious transfer (OT), and strong conditional oblivious transfer (SCOT). These protocols are based on a new optimal-rate leveled homomorphic encryption scheme for *large-output* polynomial-size branching programs, that might be of independent interest. The analysis of the new scheme is intricate: the optimal rate is achieved if a certain parameter s is set equal to the only positive root of a degree-$(m + 1)$ polynomial, where m is the length of the branching program. We show, by using Galois theory, that even when $m = 4$, this polynomial cannot be solved in radicals. We employ the Newton-Puiseux algorithm to find a Puiseux series for s, and based on this, propose a $\Theta(\log m)$-time algorithm to find an integer approximation to s.

Keywords: Branching programs, CPIR, Galois theory, homomorphic encryption, OT, Puiseux series, SCOT

DOI 10.1515/popets-2015-0016
Received 2015-02-15; revised 2015-05-15; accepted 2015-05-15.

1 Introduction

In a computational private information retrieval (CPIR) protocol [22] a client receives one ℓ-bit database element from an n-element database maintained by a server. It is required that the server should not gain any knowledge of which element was transferred to the client, and that the communication of the protocol should be smaller than that of the trivial protocol where the server just sends the whole database to the client. There exists a long line of works that improve on communication efficiency aspects of CPIR protocols [3, 18, 19, 25, 26, 37].

Usually, one strives to improve the communication-efficiency as a function of the database size n. In this paper, we focus on optimizing the communication of a CPIR protocol in the case where ℓ is especially large, and the communication overhead plays a dominant role in the usability of the protocol. We are motivated by an example application where the client has paid the server for downloading a movie (assuming all movies cost the same) and does not want the server to know which movie she is going to download. In such an application, one can easily have $\ell \gg 10^9$ while n is not more than say 10^5.

The communication complexity of non-private information retrieval is $\log_2 n + \ell$. Hence, the *communication rate* of a CPIR protocol is $(\log_2 n + \ell)/L$, where L is the communication of the CPIR protocol. Simply put, the rate of a CPIR protocol measures the communication-efficiency loss that the protocol suffers due to the added privacy requirement.

The CPIR protocol of Gentry and Ramzan [18] achieves rate $\frac{1}{4}$. The best previous rate, $\frac{1}{2}$, CPIR protocol was proposed by Lipmaa [26]. In practice, given a database consisting of very large database elements, even a rate-$\frac{1}{2}$ CPIR protocol can be prohibitively wasteful. On the other hand, optimizing the rate is very interesting from a theoretical viewpoint. In particular, optimizing the rate and constructing rate-optimal protocols is one of the focus points in areas like coding theory, where one's goal is often to obtain encoding rate that is optimal, or near-optimal (e.g., different from the optimal by a sub-constant factor; in what follows we will not distinguish between optimal and near-optimal rate).

We thus ask the following fundamental question:

Main question: *Is it possible to construct a CPIR protocol with a rate $1 - o(1)$, i.e., with a rate that is close to the rate of a non-private retrieval protocol?*

No such protocol is known or easy to derive from the existing solutions.

Given a CPIR protocol with rate r, it is also natural to ask whether one can construct other, related, protocols with rate that is close to r. Probably the most natu-

Aggelos Kiayias: National and Kapodistrian University of Athens, Greece
Nikos Leonardos: Université Paris Diderot – Paris 7, France
***Corresponding Author: Helger Lipmaa:** University of Tartu, Estonia
Kateryna Pavlyk: University of Tartu, Estonia
Qiang Tang: University of Connecticut, USA

ral such protocol to consider is *oblivious transfer* (OT). Informally, an OT protocol is just a CPIR protocol with an additional security guarantee that the client will obtain information only about one database element.

For example, in the motivating scenario the server would not like the client to obtain two movies for the price of one. (Intuitively, a rate-1 CPIR protocol cannot reveal more than one database element. However, say in the case of a database of movies, it can reveal the best bits — e.g., no advertisements or closing credits — of both. This can be undesirable, and thus one would still require an OT protocol.) While various round-preserving CPIR-to-OT transformations have been proposed [1, 23, 29], no previous work compared them from the viewpoint of rate preservation.

Finally, in a *strong conditional oblivious transfer* (SCOT, [2]) protocol, the client obtains $f_{Q(x,y)}$, where Q is a public predicate and x, y are private inputs of the client and the server respectively. This generalizes CPIR and OT in the case where the selection strategy of the client is not described as 1-out-of-n but as a more complex relation. Constructing a rate-1 SCOT is an equally interesting and wide open problem.

We are not aware of *any* computation-efficient generic CPIR-to-SCOT transformations that preserve the rate. In fact, a SCOT protocol intuitively requires the server to execute a secure computation of Q, without getting to know the result. Hence, a rate-optimal SCOT primitive for a certain class C of languages is intuitively equivalent to a rate-optimal public-key homomorphic encryption scheme for C.

The *rate* of a regular public-key encryption scheme, defined as $|x|/|\mathsf{Enc}(x)|$, as a function of $\ell = |x|$ is typically of no great concern in cryptography, because there exists a trivial construction achieving rate close to one via hybrid encryption. However, the latter does not preserve any homomorphic property that $\mathsf{Enc}(\cdot)$ may have. In fact, fully homomorphic encryption schemes (introduced in [17]) have very low and typically sub-constant rate, see [8] for a recent analysis of the parameters.

The only currently known rate-1 homomorphic cryptosystems due to Damgård and Jurik [11, 12] allow only an additive homomorphic property, i.e., homomorphic evaluation of arithmetic circuits with only addition gates. No rate-optimal homomorphic cryptosystems are known for more expressive language classes.

1.1 Our Contributions

To construct rate-optimal CPIR (and SCOT) protocols, we initiate the study of good rate homomorphic encryption schemes for non-trivial classes of languages. We propose an optimal-rate homomorphic encryption scheme for the class of functions (not necessarily predicates) that can be computed by polynomial-size branching programs. The new construction is a variation of older constructions [20, 25, 26]; however, the concrete variation and the accompanied performance analysis needed to optimize various parameters are novel. We then show how our construction can be used to optimize the communication of cryptographic tasks such as CPIR assuming that ℓ is large enough.

More precisely, we consider the class **LBP** of functions f that can be implemented by polynomial-size (large-output) branching programs. In this case, the client has an input x, the server has a branching program P_f that computes f, and the client will obtain $f(x)$ while the server obtains no information about x. Since we are mainly interested in concrete applications like CPIR and SCOT, we make some assumptions that are natural in such applications. Namely, we assume that m, (a sufficiently close upper bound on) the length of P_f, is known to the client before she sends an encryption $\mathsf{Enc}(x)$ to the server. Due to this assumption, we call the new homomorphic encryption scheme *leveled*. As usual in homomorphic encryption schemes, the server then applies an evaluation function Eval to P_f and $\mathsf{Enc}(x)$, and then returns the result $\mathsf{Eval}(P_f, \mathsf{Enc}(x))$ to the client.

Motivated by the applications, we say that the communication of the homomorphic encryption scheme is equal to the length of the useful information divided by the total communication, i.e.,

$$\frac{|x| + |f(x)|}{|\mathsf{Enc}(x)| + |\mathsf{Eval}(P_f, \mathsf{Enc}(x))|} \ .$$

The main reason the knowledge of m is public is that this allows both the client and the server to choose precise parameters to optimize the rate.

We construct an **LBP**-homomorphic PKE scheme that evaluates efficiently any (leveled) branching program P_f for a function $f : \{0,1\}^\chi \to \{0,1\}^\ell$ where we assume that P_f has length (= the maximum number of levels) m. Our construction has rate

$$1 - 2\sqrt{(w-1)\chi m k} \cdot \ell^{-1/2} + O_\ell(\ell^{-1}) \ , \qquad (1)$$

where w is the arity of the branching program and k is the security parameter. Our construction follows the paradigm of [22] as applied in [20] to branching program evaluation, with an array of crucial optimizations that are tailored to the goal of achieving the optimal rate.

In [20, 25, 26], one recursively applies a basic $(w, 1)$-CPIR, for a small w, that is based on the cryptosystem of [11]. The basic CPIR has linear communication (unavoidable by the lower bound of [32]) but importantly, it has a short answer from the server. Every recursion level i defines a length parameter s_i (intuitively, this means that the plaintexts on this level belong to N^{s_i}, where N is an RSA modulus); in all the aforementioned constructions the values s_i are strictly increasing.

Via our analysis using technique from multivariable calculus we show this setting is sub-optimal and, in fact, the optimal rate is achievable if the parameters $s_1, \ldots, s_m$ are all equal. In more detail, we show that the optimal communication results from choosing s as the unique positive root of the degree-$(m+1)$ polynomial $f_m(\cdot, \sigma)$, where $f_m(x, y) := yx^{m+1} - (x+1)^{m-1}$, and $\sigma = (w-1)k\chi/(\ell m)$ for some integer $w \geq 2$ (in the case of usual branching programs, $w = 2$).

Finding the root is impossible analytically when $m > 3$. In Sect. 6, we use basic Galois theory to show that, for example, one cannot solve $f_4(x, 1) = 0$ in radicals. Instead, we use the Newton-Puiseux algorithm [39] to compute the Puiseux series $\sum_{i=0}^{\infty} c_i \sigma^{(i-1)/2}$ of the optimal s. We then construct a simple algorithm that, given the first two partial sums of the Puiseux series, computes an integer approximation to the optimal s in $\log_2 m$ steps. This analysis is very intricate and we consider it to be one of our main contributions.

The whole approach seems to be novel in the context of the design and analysis of cryptographic protocols. In particular, we are not aware of any previous use of Galois-theory based impossibility results or of the Newton-Puiseux algorithm in cryptography.

To sum it up, the main difference of the new construction, as compared to [20, 25, 26] is the use of a single well-chosen parameter $s = s_1 = \cdots = s_m$. It is rather surprising that a simple modification like ours allows to achieve optimal rate. That optimal rate is achievable becomes clear only after extensive analysis of the parameters as explained above.

1.2 Applications

Based on the new rate-optimal LHE scheme, we show how to construct rate-optimal CPIR, OT, and SCOT protocols. First, we construct an $(n, 1)$-CPIR protocol with communication $\ell + 1.72 \log_2 n \cdot \sqrt{k\ell} + O(1)$ and an optimal rate $1 - 1.72 \log_2 n \cdot \sqrt{k/\ell} + O(\ell^{-1})$. This protocol just applies our new LHE scheme by using a complete 5-ary decision tree. The choice of 5-ary branching programs is somewhat unexpected, but follows from analysis. Hence, we answer positively to the previous "main question" of the current paper.

We propose concrete parameter choices for the new CPIR protocol, demonstrating that in an application where a client wants to privately retrieve a 2.56GB movie from the server's database of $5^7 = 78125$ movies, one can obtain rate 0.99.

We construct an optimal-rate semisimulatable OT protocol based on the CPIR-to-OT transformation by Naor and Pinkas [29]. The resulting OT protocol is only computationally secure for the server; we leave the construction of an information-theoretically server-private optimal-rate OT protocol as an interesting open problem. Alternatively, we note that one can use zero knowledge proofs to obtain an optimal-rate 2-message simulatable OT protocol in the random oracle model.

We outline how to construct an optimal-rate SCOT protocol for any predicate that can be implemented by a (large-output) polynomial-size branching program.

We note that recently, [21] applied the rate-optimal SCOT protocol from the current paper to construct a rate-1 asymmetric fingerprinting scheme.

1.3 Computation

Another important aspect of a CPIR protocol is the server's computation. While this is not a focus of the current work, we remark that the new protocol fares better than the CPIR protocols of [25, 26] also in this aspect. This is since instead of encrypting at least ℓ-bit strings, we encrypt in suitably small segments. Since encryption takes superquadratic time, we thus save significantly in computation. We make this claim concrete in Sect. 8 where we show that (for parameters interesting to us, i.e., for large ℓ) the new CPIR protocol is significantly (an off-hand calculation results in a factor of 10^5 times) faster than the CPIR protocol of [25].

We describe one possible strategy that allows us to further significantly optimize the server's computation, while decreasing its rate only marginally. Given the same parameter settings as mentioned in Sect. 1.2, decreasing the rate *less than* two times (to 0.52, which is still better than the rate in any previous CPIR protocol) results in 2^{12} times better server's computation. Such a dramatic optimization is possible due to the precise construction of the new CPIR protocol.

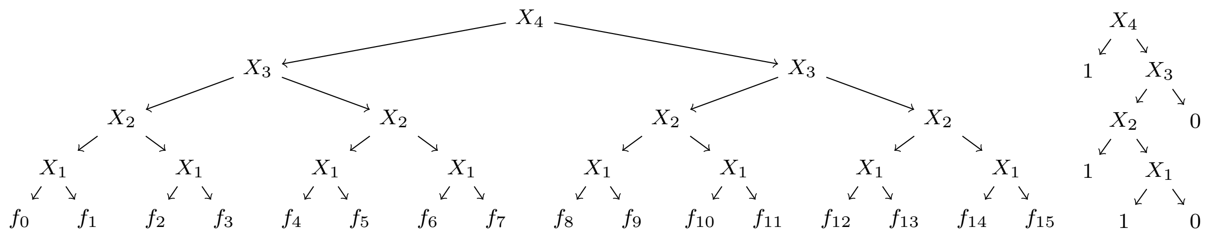

Fig. 1. The complete binary decision tree that returns f_x (left), and a branching program that returns 1 iff $x \leq 10$ (right). In both cases, $\mathsf{len}(P) = 4$

We stress that the above results still require high computation on the server side. Even the optimized CPIR protocol is computationally too inefficient (in Sect. 8, we talk about 2^{14} operations per bit of database data for the honest server).

Note that multi-server PIR protocols offer a weaker privacy guarantee compared to the CPIR setting since privacy only holds if at least one server remains honest. It is a very interesting open question to define and achieve communication optimality under such more relaxed privacy guarantees while at the same time achieving the computational complexity benefits that usually accompany this setting.

Another interesting open question is to finding other, perhaps application-dependent, trade-offs between the rate and server's computation. See [14, 15, 31] for some relevant work.

The major (computational) bottleneck in our construction is the use of the Damgård-Jurik cryptosystem. The major open question posed by the current paper is to construct a computationally more efficient rate-1 additively homomorphic cryptosystem.

1.4 Roadmap

In Sect. 2, we describe necessary preliminaries like branching programs, public-key encryption, and CPIR. In Sect. 3, we define the security and efficiency of a LHE scheme. In Sect. 4, we give the new parameterized construction of the new LHE scheme. In Sect. 5, we outline our strategy in optimizing the parameters, and state the communication complexity and the rate given the optimal parameters; we also describe an efficient algorithm for finding an integer approximation to the optimal parameters. In Sect. 7, we describe how to construct rate-optimal CPIR, OT, and SCOT protocols. Finally, in Sect. 8, we comment on the computational complexity of the CPIR protocol. Most of the mathematical details are given in the appendix.

2 Preliminaries

Since we are often interested in the growth of a function in several variables, we write the relevant variable as a subscript in Landau notation, like in $o_\ell(\ell \log n)$. Let k be the security parameter, i.e., we assume that adversaries work in probabilistic polynomial-time w.r.t. k. The current recommendation is to take $k \geq 2048$. If not specified, all logarithms take basis 2; we denote the natural logarithm of x by $\ln x$. If A is a probabilistic algorithm, then we write $a \leftarrow A(x; r)$, where r is the randomizer. If r is omitted, then it is chosen uniformly at random. For a predicate $P(x)$, let $[P(x)] = 1$ if $P(x)$ is true, and $= 0$, otherwise. Denote $[a] = \{1, \dots, a\}$.

2.1 Branching Programs

A *w-ary branching program* [40] is a fanout-w directed acyclic graph (V, E), where the non-terminal (that is, non-sink) nodes are indexed by variables from some variable set $\{X_1, \dots, X_\chi\}$, the sinks are labeled by ℓ-bit strings and the w outgoing edges of every internal node are indexed by values from 0 to $w - 1$. We denote by $\mathfrak{L}_v$ the label of the node v, by $\mathsf{ind}(v)$ the index of the (internal) node v. Since the new leveled LHE scheme also computes labels of internal nodes, in our case the difference between indexes and labels is strict. Usually, it is assumed that a branching program has 1-bit sink labels; then it can be assumed to have two terminal nodes. A large-output branching program (with longer sink labels) is thus sometimes called *multi-terminal.* A binary branching program is commonly known as a binary decision diagram or BDD.

A branching program with 1 source computes a function $f : \{0,1\}^\chi \to \{0,1\}^\ell$. Every assignment of the variables selects one path from the source to a sink as follows. The path starts from the source. If the current path does not end at a sink, test the variable $X_{\mathsf{ind}(v)}$ at the endpoint v of the path. Select one of the outgoing edges depending on the value of this variable, and append this edge and its endpoint to the path. (For the sake of concreteness, we assume that the leftmost edge is chosen iff the variable is 0.) If the path ends at a sink, return the label of this sink as the label of the source. The branching program's value is then equal to the source label. See Fig. 1 for some examples.

For a branching program P, let $\mathsf{len}(P)$ be its length (that is, the length of its longest path), $\mathsf{size}(P)$ be its size (that is, the number of non-terminal nodes). Let $\mathsf{BP}(f)$ be the minimal size of any branching program computing f. A Boolean function f has a polynomial-size branching program iff f belongs to $\mathbf{L/poly}$ [9], the complexity class of logarithmic space machines with a polynomial amount of advice. Branching program for functions $f : \{0,1\}^\chi \to \{0,1\}^\ell$ with non-Boolean output can be constructed in a natural way.

P is a *decision tree* if the underlying graph is a tree. P is *leveled* if its set of nodes can be divided into disjoint sets V_d such that every edge from a node in set V_d ends in a node in set V_{d-1}. An *oblivious* branching program is a leveled branching program in which all nodes of the same level are indexed by the same variable X_i. Each branching program can be efficiently transformed into a leveled branching program of the same length and quadratic size [36]; similarly, there exists an efficient transformation that makes a leveled branching program oblivious. In our applications, we start with a branching program that is oblivious and thus requires no additional transformation. We assume that the source is the only member of the set V_m. Let $\mathsf{size}(P, d)$ be the number of nodes P has on level d, thus $\mathsf{size}(P, m) = 1$.

The class $\mathbf{LBP}$ contains all functions $f : \{0, \dots, w-1\}^\chi \to \{0,1\}^\ell$ for which we have a large-output branching program with size that is polynomial on both parameters χ and $|f(\{0, \dots, w-1\}^\chi)|$.

Throughout the paper, P_f is a fixed leveled w-ary branching program that implements $f : \{0,1\}^\chi \to \{0,1\}^\ell$. For any node v, let $\mathsf{len}(v)$ be its *length*, i.e., $\mathsf{len}(v) = \mathsf{len}(P_f) - \bar{d}$, where $\bar{d}$ is the distance from the source to v. Thus $v \in V_{\mathsf{len}(v)}$, and the source has length $\mathsf{len}(P_f)$. (E.g., on Fig. 1 (left), all sinks have length 0 and the source has length 4.)

For a non-sink v, let $\mathsf{child}(v, i)$ for $i \in [0, w-1]$ be its ith leftmost child, and $X_{\mathsf{ind}(v)}$ be the index of v (that is, $\mathsf{ind}(v) = i$ if v is indexed by X_i). Thus, $\mathsf{len}(\mathsf{child}(v, i)) = \mathsf{len}(v) - 1$. Assume that the nodes of P_f are ordered from 1 to $\mathsf{size}(P_f)$ so that if there exists an edge $u \to v$ then $v < u$. Assume that P_f has n sinks. Hence, the first n nodes $v \le n$ are the sinks and the last node $v = \mathsf{size}(P_f)$ is the source of P_f. Recall that the description of P_f also contains the labels $\mathfrak{L}_v$ of the sinks of P_f.

2.2 Public-Key Cryptosystem

A public-key cryptosystem $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ consists of three polynomial-time algorithms, a probabilistic key generating algorithm $(\mathsf{pk}, \mathsf{sk}) \leftarrow_r \mathsf{Gen}(1^k)$, a probabilistic encryption algorithm $c \leftarrow \mathsf{Enc}_{\mathsf{pk}}(x; r)$, and a deterministic decryption algorithm $x \leftarrow \mathsf{Dec}_{\mathsf{sk}}(c)$. It is required that if $(\mathsf{pk}, \mathsf{sk}) \leftarrow_r \mathsf{Gen}(1^k)$, then for any x and r from corresponding domains, $\mathsf{Dec}_{\mathsf{sk}}(\mathsf{Enc}_{\mathsf{pk}}(x; r)) = x$.

The *rate* of Π is the length ℓ of a plaintext divided by the length of its ciphertext. The rate can be a function of ℓ. A cryptosystem is *CPA-secure* if for any x_0 and x_1 (possibly chosen by the adversary) of the same length, given pk and an encryption $\mathsf{Enc}_{\mathsf{pk}}(x_\beta; r)$ for randomly chosen $\beta \in \{0,1\}$ and r, no probabilistic polynomial-time adversary can guess β with probability $\frac{1}{2} + \varepsilon$, where ε is non-negligible in k.

2.3 Damgård-Jurik Cryptosystem

Assume ℓ is the length of the plaintexts in bits. The Damgård-Jurik cryptosystem [11] allows to encrypt plaintexts for arbitrary $\ell \ge 1$, so that the ciphertext length is not more than $\ell + 2k$. Therefore, it has rate $1 - o_\ell(1)$. The cryptosystem is defined as follows.

To generate the public and secret keys, one lets $N = pq$ to be a k-bit RSA modulus for two randomly generated $k/2$-bit primes p and q. The value N is the public key pk, and the factorization (p, q) of N is a part (together with some additional information that makes decrypting more efficient) of the secret key sk.

To encrypt an ℓ-bit string x, one chooses *a length parameter* s such that $\ell = s \cdot k$ (or $s = \lceil \ell/k \rceil$ if $k \nmid \ell$), chooses a randomizer $r \leftarrow_r \mathbb{Z}_N^*$, and then outputs

$$c = \mathsf{Enc}_N^s(x; r) \leftarrow (1+N)^x r^{N^s} \mod N^{s+1} \enspace .$$

Decryption can be done efficiently, see [11].

Clearly, the plaintext belongs to $\mathbb{Z}_{N^s}$ while the ciphertext belongs to $\mathbb{Z}_{N^{s+1}}$, that is, has the bitlength $\le \lceil \log_2 N^{s+1} \rceil \le (s+1)k$ bits. Due to the choice of s, the bitlength of the plaintext is at least $(s-1)k$. The *rate* of Damgård-Jurik is $|x|/|c| \ge (s-1)/(s+1)$. If $\ell \to \infty$, then $\ell = |x| \approx s \cdot k$, and the rate is $\approx 1 - 1/s$.

This cryptosystem is additively homomorphic, since

$$\mathsf{Enc}_N^s(x_0; r_0) \cdot \mathsf{Enc}_N^s(x_1; r_1) = \mathsf{Enc}_N^s(x_0 + x_1; r_0 r_1) \enspace .$$

Moreover, for $c \in \mathbb{Z}_N$,

$$\mathsf{Enc}^s_N(x;r)^c = \mathsf{Enc}^s_N(cx;r^c) \enspace .$$

Recall that arithmetic in the first (resp., second) parameter of Enc is done modulo N^s (resp., N).

The CPA-security of the Damgård-Jurik cryptosystem is based on the Decisional Composite Residuosity (DCR) assumption of Paillier [33].

2.4 CPIR

In an $(n, 1)$-CPIR (computationally-private information retrieval, [22]) protocol for ℓ-bit strings, the server has a database of n elements, $\vec{f} = (f_0, \dots, f_{n-1})$, each f_i being ℓ bits long, and the client has an input $x \in \{0, \dots, n-1\}$. The client needs to obtain f_x, while no probabilistic polynomial-time server should obtain any information about x. It is also required that the total communication complexity of the CPIR protocol is smaller than in the trivial case where the server just sends the whole database to the server.

A *two-message* CPIR protocol consists of the following three steps.

1. First, the client generates a secret/public key pair $(\mathsf{sk}, \mathsf{pk}) \leftarrow_r \mathsf{KGCPIR}(1^k)$, and sends to the server pk and a query $c = \mathsf{Que}_{\mathsf{pk}}(n, \ell, x; r)$; the latter depends on the security parameter k, the size of the database n, the length of the database elements ℓ, the client's input x, and some random coins r.
2. Second, the server replies with $C \leftarrow \mathsf{Rep}_{\mathsf{pk}}(\vec{f}, c; \hat{r})$ that depends on the server's input $\vec{f}$, the query c, and another randomizer $\hat{r}$.
3. Third, the client recovers $f_x \leftarrow \mathsf{Ans}_{\mathsf{sk}}(n, \ell, C)$, given access to C, n, ℓ, and the secret key sk.

It is required that for any valid key pair $(\mathsf{sk}, \mathsf{pk}) \leftarrow_r \mathsf{KGCPIR}(1^k)$ and any valid inputs $(\vec{f}, x)$ and randomizers $(r, \hat{r})$,

$$\mathsf{Ans}_{\mathsf{sk}}(n, \ell, \mathsf{Rep}_{\mathsf{pk}}(\vec{f}, \mathsf{Que}_{\mathsf{pk}}(n, \ell, x; r); \hat{r})) = f_x \enspace .$$

The (CPA-)security notion is similar to the one of cryptosystems, see, e.g., [25].

The *rate* $\mathsf{rate}(\Gamma)$ of a two-message CPIR protocol Γ is the number of "useful bits" (that is, $\log_2 n + \ell$) divided by the total communication $|\mathsf{Que}| + |\mathsf{Rep}|$ of the protocol. We do not include pk to the communication, since the same pk can (and will) be reused in many instantiation of CPIR protocols. Even if pk not reused, its length is minimal (e.g., $|\mathsf{pk}| = k = o_\ell(1)$ bits in the next example) and does hence not influence the rate significantly.

We remark that multi-server PIR protocols [7] do not rely on computational assumptions and are usually computationally more efficient than single-server CPIR protocols. However, such protocols have strongly subconstant rate and moreover, are not secure unless some of the servers are honest, an assumption that is not realistic in many scenarios.

2.5 Basic CPIR

The new LHE scheme of Sect. 4 is based on a careful recursion built on top of Lipmaa's two-message $(w, 1)$-CPIR protocol [25] for small w. The two main properties that we use is that (i) it is based on additively homomorphic encryption without any recursion (and thus must have linear-in-n communication, [32]), and (ii) it's *server communication* has rate-$(1 - o(1))$. Our construction in Sect. 4 is an efficient reduction of such a basic CPIR protocol to one that has rate $1 - o(1)$.

In this $(w, 1)$-CPIR protocol, the client generates secret and public key $(\mathsf{sk}, \mathsf{pk})$, with $\mathsf{pk} = N$, for the Damgård-Jurik cryptosystem. She sends pk together with a vector $\vec{c} = (c_1, \dots, c_{w-1})$ of $w - 1$ ciphertexts

$$\vec{c} \leftarrow \mathsf{Que}_{\mathsf{pk}}(w, \ell, x; \vec{r}) := (\mathsf{Enc}^s_N([i = x]; r_i))_{i=1}^{w-1}$$

to the server, where $[i = x] \in \{0, 1\}$ is equal to 1 iff $i = x$, $s = \lceil \ell / k \rceil$, $r_i \leftarrow_r \mathbb{Z}_N^*$, and $\vec{r} = (r_1, \dots, r_{w-1})$. Note that $|c_i| = (s+1)k$ and thus $|\vec{c}| = (w-1)(s+1)k$. Let $\vec{f} = (f_0, \dots, f_{w-1})$ be the server's database. The server answers, for random $\hat{r} \leftarrow_r \mathbb{Z}_N^*$, with

$$C \leftarrow \mathsf{Rep}_{\mathsf{pk}}(\vec{f}, \vec{c}; \hat{r}) := \mathsf{Enc}^s_N(f_0; \hat{r}) \cdot \prod_{i=1}^{w-1} c_i^{f_i - f_0} \enspace ,$$

that is,

$$C = \mathsf{Enc}^s_N(f_x; \prod_{i=1}^{w-1} r_i^{f_i - f_0} \cdot \hat{r}) \enspace .$$

Since $\hat{r}$ is random, then C is a random encryption of f_x. The client obtains f_x by decrypting C,

$$f_x \leftarrow \mathsf{Ans}_{\mathsf{sk}}(w, \ell, C) := \mathsf{Dec}^s_{\mathsf{sk}}(C) \enspace .$$

Clearly, the server's answer is a random encryption of f_x. Since the server only sees encrypted messages, the CPA-security of $(w, 1)$-Lipmaa's CPIR protocol immediately follows from the CPA-security of the Damgård-Jurik cryptosystem, and thus, from the DCR assumption. Here, for $s = \lceil \ell / k \rceil$ (and ignoring pk),

$$|\mathsf{Que}| = (w-1)(s+1)k \quad \text{and} \quad |\mathsf{Rep}| = (s+1)k \enspace , \quad (2)$$

and thus it has rate

$$\frac{\log w + \ell}{|\mathsf{Que}| + |\mathsf{Rep}|} \approx \frac{1}{w} \ .$$

However, its server-rate is $(\log w + \ell)/|\mathsf{Rep}| = 1 - o_\ell(1)$.

Due to the construction of the Damgård-Jurik cryptosystem, x and f_x must be encrypted by using the same length parameter s: if x was encrypted by using a parameter $z < s$, then the server's answer would encrypt $f_x \mod N^z$ and thus the server would not recover the whole value f_x. More discussion on this issue is provided in [26]. There it was also shown that there exists a function $\mathsf{Compress} = \mathsf{Compress}_N^s$ that takes as an input $\mathsf{Enc}_N^{s+1}(x; r)$ and outputs $\mathsf{Enc}_N^s(x \mod N^s; r^*)$, where $s \geq 1$, and r^* is a randomizer that depends on N, r and s. In fact, $\mathsf{Compress}_N^s(c) = c \mod N^{s+1}$.

3 LHE: Definitions

We introduce *leveled* homomorphic encryption for **LBP**, following the terminology of Gentry [16]. However, the definition will be somewhat different. The differences are motivated by both the applications (where such a definition makes sense, see Sect. 7) and by the construction (that achieves this definition, see Sect. 4). We first recall the following definition (Def. 2.1.5 from [16]).

We say that a family of homomorphic encryption schemes $\{\Pi^{(m)} : m \in \mathbb{Z}^+\}$ is *leveled fully homomorphic* if, for all $m \in \mathbb{Z}^+$, they all use the same decryption circuit, $\Pi^{(m)}$ compactly evaluates all circuits of depth at most m (that use some specified set of gates), and the computational complexity of $\Pi^{(m)}$'s algorithms is polynomial in k, m, and (in the case of the evaluation algorithm) the size of the circuit.

In practice, this definition means that each $\Pi^{(m)}$ can have a different private/public key pair $(\mathsf{sk}^{(m)}, \mathsf{pk}^{(m)})$. This is since the public moduli, used when encrypting, (and thus also the public key) depend on m.

The new (slightly stronger) definition requires the existence of a single key pair $(\mathsf{sk}, \mathsf{pk})$ usable for any m. Since the public key does not fix m, it has to be the client — who sends her message first — who picks the value m while encrypting the messages. The value m gives an *upper bound* on the length of the large-output branching program that the server can evaluate on these ciphertexts. Optimal rate is achieved if m is equal to the actual length of the evaluated large-output branching program. For this reason, in the definition we will concentrate on the case of level m branching programs. Since the rate in our case will be defined as the total length of the client's and the server's messages, it is natural that the client has to choose the parameter m, based on her knowledge of the server's input, to optimize the rate. Similar problem exists in leveled FHE.

The following definition formalizes this intuition.

Definition 1. *A* (single-key) leveled **LBP**-homomorphic encryption (LHE) scheme *is a four-tuple of efficient algorithms* $(\mathsf{KG}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$*, such that*

(i) the randomized key generation algorithm $\mathsf{KG}(1^k)$ *creates a secret key and public key pair* $(\mathsf{sk}, \mathsf{pk})$*,*

(ii) given a message x, the branching program length m, and a randomizer r, the randomized encryption algorithm $\mathsf{Enc}_{\mathsf{pk}}^m(x; r)$ *returns a ciphertext c,*

(iii) given a leveled branching program P_f of length m, a fresh ciphertext c (equal to $\mathsf{Enc}_{\mathsf{pk}}^m(x; r)$ *for some plaintext x and randomizer r) and a randomizer $\hat{r}$, the* randomized evaluation algorithm $\mathsf{Eval}_{\mathsf{pk}}(P_f, c; \hat{r})$ *returns an evaluated ciphertext C,*

(iv) given an evaluated ciphertext C and the branching program length m, the deterministic decryption algorithm $\mathsf{Dec}_{\mathsf{sk}}^m(C)$ *returns a plaintext x.*

The computational complexity of these four algorithms must be polynomial in k, m, and (in the case of Eval*) the size of the branching program.*

It is required that for any valid key pair $(\mathsf{sk}, \mathsf{pk})$, message x, randomizers r and $\hat{r}$, and a polynomial-size branching program P_f of length m,

$$\mathsf{Dec}_{\mathsf{sk}}^m(\mathsf{Eval}_{\mathsf{pk}}(P_f, \mathsf{Enc}_{\mathsf{pk}}^m(x; r); \hat{r})) = P_f(x) \ .$$

A leveled LHE scheme must satisfy two security requirements, *CPA-security* and *branching program privacy* (similar to circuit-privacy, [16, 17]). The first one is defined similarly to the case of arbitrary public-key cryptosystems, though one has to take into account the presence of Eval, see [16, 17], for a formal definition. However, to achieve optimal rate we allow the outputs of Enc^m and Eval to come from different distributions; we just require that the output of Eval does not reveal any unnecessary information about the evaluated branching program except its length.

Definition 2. (Perfect) branching program privacy *guarantees that for any* $(\mathsf{sk}, \mathsf{pk})$*, any m, any valid c produced by* $\mathsf{Enc}_{\mathsf{pk}}^m$*, and any two equal-length branching programs P_0 and P_1 such that* $P_0(\mathsf{Dec}_{\mathsf{sk}}^m(c)) = P_1(\mathsf{Dec}_{\mathsf{sk}}^m(c))$*, it holds that* $\mathsf{Eval}_{\mathsf{pk}}(P_0, c)$ *and* $\mathsf{Eval}_{\mathsf{pk}}(P_1, c)$ *have the same distribution.*

We require that the LHE scheme Π be communication-efficient in the sense that its *rate*

$$\mathsf{rate}(\Pi) := \frac{|x| + |P_f(x)|}{|\mathsf{Enc}^m_{\mathsf{pk}}(x;r)| + |\mathsf{Eval}_{\mathsf{pk}}(P_f, \mathsf{Enc}^m_{\mathsf{pk}}(x;r);\hat{r})|}$$

is as large as possible. Informally, Π is *optimal-rate*, if the rate is $1 - o_\ell(1)$ as a function of ℓ.

The rate takes into account the value $|\mathsf{Enc}|$, since it is possible to choose parameters so that $|\mathsf{Eval}|$ is very small while $|\mathsf{Enc}|$ is very large. It is also a natural measurement of the rate in many applications like $(n,1)$-CPIR. Similarly, the *communication complexity* of a leveled LHE scheme is equal to $|\mathsf{Enc}| + |\mathsf{Eval}|$.

4 Construction

In what follows, we propose a leveled LHE scheme that securely computes the values of any function $f : \{0,\dots,w-1\}^\chi \to \{0,1\}^\ell$, $f \in \mathbf{LBP}$, with the rate as in Eq. (1). Here, m is the length of a polynomial-size w-ary leveled branching program P_f that implements f. Since in the intended applications, $\ell \gg \chi mk$, the rate will approach 1 when ℓ increases. In the current section, the leveled LHE scheme will be parameterized; the claimed communication complexity and the rate will be achieved in Sect. 5 where we propose the optimal values of the parameters. For related notations about **LBP**, and the Lipmaa's CPIR protocol, we refer to Sect. 2.

4.1 High-Level Strategy

Following [20], the general strategy of our construction is as follows. Recall that in a w-ary branching program P_f, the label (that we denote by $\mathfrak{L}_v$) of a node v, indexed by $X_{\mathsf{ind}(v)}$, is equal to $\mathfrak{L}_{\mathsf{child}(v,X_{\mathsf{ind}(v)})}$. If privacy has to be preserved, this operation can be executed by applying the Rep function of a basic $(w,1)$-CPIR protocol to a query $\mathsf{Que}(\dots)$ corresponding to the value of $X_{\mathsf{ind}(v)}$, and to the database $\vec{f}_{(v)} = (\mathfrak{L}_{\mathsf{child}(v,0)}, \dots, \mathfrak{L}_{\mathsf{child}(v,w-1)})$. Instead of sending $\mathfrak{L}_v$ to the client, the server stores $\mathfrak{L}_v$ so that it can be later be used recursively to compute the label of v's parent. Finally, the server returns to the client only the value of $\mathfrak{L}_{\mathsf{size}(P_f)}$ and the client uses the function Ans recursively to obtain f_x from this.

This general strategy was introduced in [22], albeit with an inefficient basic $(w,1)$-CPIR protocol (that resulted in super-polylogarithmic communication for the recursive construction), and it has been used in many subsequent works. At every node in the branching program P_f to be evaluated, we use an efficient $(w,1)$-CPIR protocol, preceded and succeeded with well-chosen *non-cryptographic* operations (more precisely, splitting and concatenating bit-strings), to obtain optimal rate.

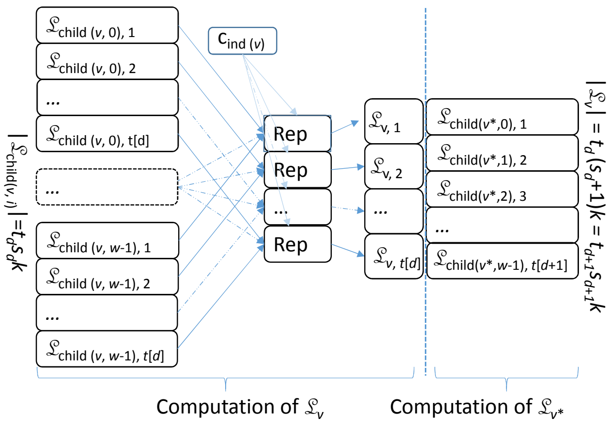
$\mathfrak{L}_{\text{child }(v,0),1}$
$\mathfrak{L}_{\text{child }(v,0),2}$
...
$\mathfrak{L}_{\text{child }(v,0),t[d]}$
...
$\mathfrak{L}_{\text{child }(v,w-1),1}$
$\mathfrak{L}_{\text{child }(v,w-1),2}$
...
$\mathfrak{L}_{\text{child }(v,w-1),t[d]}$
$|\mathfrak{L}_{\text{child}(v,i)}| = t_d s_d k$
$C_{\text{ind }(v)}$
Rep
Rep
...
Rep
$\mathfrak{L}_{v,1}$
$\mathfrak{L}_{v,2}$
...
$\mathfrak{L}_{v,t[d]}$
$\mathfrak{L}_{\text{child}(v^*,0),1}$
$\mathfrak{L}_{\text{child}(v^*,1),2}$
$\mathfrak{L}_{\text{child}(v^*,2),3}$
...
$\mathfrak{L}_{\text{child}(v^*,w-1),t[d+1]}$
$|\mathfrak{L}_v| = t_d(s_d+1)k = t_{d+1}s_{d+1}k$
Computation of $\mathfrak{L}_v$
Computation of $\mathfrak{L}_{v^*}$

Fig. 2. Local computation of $\vec{\mathfrak{L}}_v$

4.2 Detailed Description

We utilize a two-message $(w,1)$-CPIR protocol with a short reply $|\mathsf{Rep}|$. More precisely, we use Lipmaa's $(w,1)$-CPIR protocol $\Gamma = (\mathsf{KGCPIR}, \mathsf{Que}, \mathsf{Rep}, \mathsf{Ans})$ (see Sect. 2). We recall that then, $|\mathsf{Que}|$ and $|\mathsf{Rep}|$ are as in Eq. (2), where $s = \lceil \ell/k \rceil$. We also need the existence of the $\mathsf{Compress}$ function (see Sect. 2). We will summarize in Fig. 3 the required properties of this CPIR protocol.

Assume that all parties know w (the arity of the branching program) and $m = \mathsf{len}(P_f)$. For every level $d \in [m]$, let s_d be a level-specific length parameter that is used in the basic $(w,1)$-CPIR protocol of that level. Optimal parameters s_d, for $d \in [m]$, will be fixed in Sect. 5. Recall k is the security parameter. For some t_i, every node v has a label $\vec{\mathfrak{L}}_v$ of bitlength

$$|\vec{\mathfrak{L}}_v| = t_{\mathsf{len}(v)} s_{\mathsf{len}(v)} k \quad . \qquad (3)$$

Let $s_i^{\max}$ be the maximum of s_d, where d ranges among levels that have a node indexed by X_i. Sometimes, but not always, one can assume that the encrypter knows the values $s_i^{\max}$. If he does not, we set $s_i^{\max} := \max_{d\in[m]}\{s_d\}$. In the optimal case, see Sect. 5, all values s_d are equal, and thus it does not matter whether the encrypter knows $s_i^{\max}$. However, we first have to establish the optimality.

In the new LHE scheme (see Fig. 3 for a full description), on input x (plus pk and public parameters like m and $s_i^{\max}$), the encrypter writes $x = \sum x_i w^i$

```
System parameters: Let $f : \{0,\dots,w-1\}^\chi \to \{0,1\}^\ell$ be a function in LBP. Let $P_f$ be a
    polynomial-size leveled $w$-ary branching program, $w \in \mathbb{Z}^+$, that implements $f$, and let $m = \mathsf{len}(P_f)$.
    Let $\Gamma = (\mathsf{KGCPIR}, \mathsf{Que}, \mathsf{Rep}, \mathsf{Ans})$ be a $(w,1)$-CPIR protocol that has the Compress function,
    plaintext size $\ell = s \cdot k$, $|\mathsf{Que}(w,\ell,\dots)| = (w-1)(s+1)k$ and $|\mathsf{Rep}| = (s+1)k$, where $k$ is the security parameter.
Key generation $\mathsf{KG}(1^k)$: generate a key pair $(\mathsf{sk},\mathsf{pk})$ via $\mathsf{KGCPIR}(1^k)$.
Encryption $\mathsf{Enc}^m_{\mathsf{pk}}(x;\cdot)$:
    For $i = 1$ to $\chi$:
      – let $\vec{c}_i \leftarrow \mathsf{Que}_{\mathsf{pk}}(w, s_i^{\max} \cdot k, x_i; \vec{r}_i)$ for random $\vec{r}_i$.
    Return $c \leftarrow (\vec{c}_1,\dots,\vec{c}_\chi)$.
Evaluation $\mathsf{Eval}_{\mathsf{pk}}(P_f, c; \hat{r})$:
    Parse $c$ as $c = (\vec{c}_1,\dots,\vec{c}_\chi)$.
    For $v = n+1$ to $\mathsf{size}(P_f)$:
      – $d \leftarrow \mathsf{len}(v)$;
      – For $z \leftarrow 1$ to $t_d$:
          – Pick random $\hat{r}_{v,z}$;
          – Let $\vec{f}_{(v,z)} = (\mathfrak{L}_{\mathsf{child}(v,0),z},\dots,\mathfrak{L}_{\mathsf{child}(v,w-1),z})$;
          – $\mathfrak{L}_{v,z} \leftarrow \mathsf{Rep}_{\mathsf{pk}}(\vec{f}_{(v,z)}; \vec{c}_{\mathsf{ind}(v)}; \hat{r}_{v,z})$;
      – $\vec{\mathfrak{L}}_v \leftarrow (\mathfrak{L}_{v,1},\dots,\mathfrak{L}_{v,t_d})$;
    Return $\vec{C} \leftarrow \vec{\mathfrak{L}}_{\mathsf{size}(P_f)}$;
Decryption $\mathsf{Dec}^m_{\mathsf{sk}}(\vec{C})$:
    Parse $\vec{C} \leftarrow (C_1,\dots,C_{t_d})$;
    For $z = 1$ to $t_d$: $x_z \leftarrow \mathsf{Ans}_{\mathsf{sk}}(w, s_m k, C_z)$;
    Write $\vec{x} = (x_1,\dots,x_{t_d})$;
    If $d = 0$ then return $\vec{x}$ else return $\mathsf{Dec}^{m-1}_{\mathsf{sk}}(\vec{x})$;
```

Fig. 3. The new leveled LHE scheme $\Pi = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$.

with $x_i < w$, and then for each x_i computes $\vec{c}_i \leftarrow \mathsf{Que}_{pk}(w, s_i^{\max} \cdot k, x_i; \cdot)$ by using a fresh randomizer. The vector of those queries is the LHE encryption of x that is sent to the server. Note that x_i corresponds to an assignment to the formal variable X_i.

The server evaluates, by using function Eval, a branching program P_f on encrypted x. $\mathsf{Eval}_{\mathsf{pk}}(P_f, c; \hat{r})$ inputs a w-ary leveled branching program P_f and the queries $\vec{c}_i$ corresponding to assignments to all X_i. Recall that the choice of P_f fixes $\vec{\mathfrak{L}}_v$ for all sinks $v \le n$, where n is the total number of sink nodes in P_f. Eval recursively computes $\vec{\mathfrak{L}}_v$ for all non-sink nodes whose children already have assigned labels $\vec{\mathfrak{L}}_{\mathsf{child}(v,i)}$. Finally, Eval returns the label $\vec{\mathfrak{L}}_{\mathsf{size}(P_f)}$ of the source.

Up to now, the construction is not very different from those in [20, 26]. The crux of the new construction is in how exactly $\vec{\mathfrak{L}}_v$ is evaluated. Namely, at every non-sink node v, Eval does the following. (See Fig. 2; we will soon define all notation used there.) Let $d = \mathsf{len}(v)$; since v is a non-sink node, $d \in \{1,\dots,m\}$. Recall from Eq. (3) that $\vec{\mathfrak{L}}_v$ has bit-length $t_d s_d k$ and thus $\vec{\mathfrak{L}}_{\mathsf{child}(v,i)}$ has bit-length $t_{d-1}(s_{d-1}+1)k$. For the evaluation to succeed, we set recursively

$$\begin{aligned} t_0 s_0 k &= \ell \ , \quad \text{and} \\ t_d s_d k &= t_{d-1}(s_{d-1}+1)k \quad \text{for } d \in [1,m] \ . \end{aligned} \tag{4}$$

For the sake of simplicity, in our theoretical analysis we allow t_i to be non-integers. In practice, one must use appropriate integer ceiling functions. If ℓ is large enough, the latter causes a very small change; we will give a numerical example in Sect. 7.1.

Eval writes

$$\vec{\mathfrak{L}}_{\mathsf{child}(v,i)} = (\mathfrak{L}_{\mathsf{child}(v,i),1},\dots,\mathfrak{L}_{\mathsf{child}(v,i),t_d})$$

and

$$\vec{\mathfrak{L}}_v = (\mathfrak{L}_{v,1},\dots,\mathfrak{L}_{v,t_d}) \ ,$$

where $|\mathfrak{L}_{\mathsf{child}(v,i),z}| = s_d k$ and due to the properties of the underlying $(w,1)$-CPIR protocol, $|\mathfrak{L}_{v,z}| = |\mathsf{Rep}_{\mathsf{pk}}()| = (s_d+1)k$. (We note that later, when v will play the role of a child of some other node v^*, $\vec{\mathfrak{L}}_v$ will be divided into t_{d+1} parts, see Fig. 2.)

For each z, Eval then computes $\mathfrak{L}_{v,z}$ by applying the $\mathsf{Rep}_{\mathsf{pk}}(\cdot)$ algorithm on the intermediate database

$$\vec{f}_{(v,z)} := (\mathfrak{L}_{\mathsf{child}(v,0),z},\dots,\mathfrak{L}_{\mathsf{child}(v,w-1),z})$$

and $\vec{c}_{\mathsf{ind}(v)}$, see Fig. 3. After that, $\vec{\mathfrak{L}}_v$ is set equal to the concatenation of the replies of t_d parallel CPIR protocols, where the zth $(w,1)$-CPIR protocol is applied to client's input $x_{\mathsf{ind}(v)}$ and the database $\vec{f}_{(v,z)}$. Intuitively, $\vec{\mathfrak{L}}_v$ is a "garbled" version of $\vec{\mathfrak{L}}_{\mathsf{child}(v,x_{\mathsf{ind}(v)})}$.

This means that at the end of the protocol, $\vec{\mathfrak{L}}_{\mathsf{size}(P_f)}$ is equal to the m-times recursive (and parallel) application of Rep to f_x. From this, the decrypter can obtain f_x from $\vec{\mathfrak{L}}_{\mathsf{size}(P_f)}$ by recursively applying $\mathsf{Ans}_{\mathsf{sk}}$ to it. In our case, $\vec{\mathfrak{L}}_{\mathsf{size}(P_f)}$ (and the intermediate values) is interpreted as a concatenation of t_d bitstrings, and $\mathsf{Ans}_{\mathsf{sk}}$ is applied to each piece separately. The answers are concatenated again, and the result is given as an input to $\mathsf{Ans}_{\mathsf{sk}}$ of the next level. The algorithms $\mathsf{KG}(1^k)$, $\mathsf{Enc}^m_{\mathsf{pk}}(x;\cdot)$, $\mathsf{Eval}_{\mathsf{pk}}(P_f, c; \hat{r})$, and $\mathsf{Dec}^m_{\mathsf{sk}}(C)$ are formally described by Fig. 3. The required constraints are satisfied by Lipmaa's $(w,1)$-CPIR, see Sect. 2.

Theorem 1. *Let Π be the leveled LHE scheme from Fig. 3. Π is perfectly branching program private. If Γ is CPA-secure, then Π is CPA-secure.*

Proof. The correctness of the construction is obvious. The branching program privacy is clear, since the decrypter only sees a number of $(m-1)$-times application of Rep to an output of Que, and it is guaranteed by the definition of privacy that the input to the query does not depend on the branching program.

Next, if an adversary is able to break the CPA-security of the leveled LHE scheme, then via a standard hybrid argument she is also able to break the CPA-security of the underlying CPIR protocol. □

Theorem 2. *Let Π be the leveled LHE scheme from Fig. 3. The computation of $\mathsf{Eval_{pk}}$ is dominated by $\sum_{d=1}^{m} t_d \cdot \mathsf{size}(P_f, d) \cdot T_{\mathsf{Rep}}(s_d, w)$, where $T_{\mathsf{Rep}}(s_d, w)$ is the computational complexity of $\mathsf{Rep_{pk}}$ with given parameters. Write $\vec{s} = (s_1, \dots, s_m)$. The communication $|\mathsf{Enc}| + |\mathsf{Eval}|$ of Π is equal to*

$$\mathsf{com}(\chi, w, m, \vec{s}, k, \ell) = (w-1)k\left(\sum_{i=1}^{\chi} s_i^{\max} + \chi\right) + \ell \cdot \prod_{d=0}^{m-1}\left(1 + \frac{1}{s_d}\right) . \tag{5}$$

Proof. The computational complexity is obvious. For the communication complexity, clearly,

$$\begin{aligned}|\mathsf{Enc}^m_{\mathsf{pk}}(x; r)| &= \sum_{i=1}^{\chi}(w-1)(s_i^{\max}+1)k \\ &= (w-1)(\sum_{i=1}^{\chi} s_i^{\max} + \chi)k\end{aligned}$$

bits. For $d > 0$,

$$\begin{aligned}t_d &= \frac{s_{d-1}+1}{s_d} t_{d-1} = \prod_{i=0}^{d-1} \frac{s_i + 1}{s_{i+1}} \cdot t_0 \\ &= \prod_{i=0}^{d-1}(s_i+1) \cdot \prod_{i=1}^{d} \frac{1}{s_i} \cdot \frac{\ell}{s_0 k} \\ &= \frac{\ell}{(s_d+1)k} \cdot \prod_{i=0}^{d}\left(1 + \frac{1}{s_i}\right) .\end{aligned}$$

Thus, $|\mathsf{Eval_{pk}}(P_f, c; \hat{r})| = t_m s_m k$ is equal to

$$\begin{aligned}t_m s_m k &= \frac{\ell}{(s_m+1)k} \cdot \prod_{d=0}^{m}\left(1 + \frac{1}{s_d}\right) \cdot s_m k \\ &= \ell \cdot \prod_{d=0}^{m-1}\left(1 + \frac{1}{s_d}\right) .\end{aligned}$$

This gives the claimed communication complexity. □

5 Finding Optimal Parameters

Next, we find the optimal parameters that result in the best possible rate for the leveled LHE scheme from Sect. 4. More precisely, our goal is to find optimal length parameters s_d, as a function of ℓ. As we will see, this optimization problem has quite an unexpected solution.

We now briefly summarize our strategy. First, we show by using standard methods of multivariable calculus that the communication is minimized when the length parameters s_d used at every level are all equal, $s_1 = \cdots = s_m =: s$. Second, we show that the optimal s is defined as the unique positive root of a certain degree-$(m+1)$ polynomial. Third, since there is no general algebraic solution to this polynomial (except for $m < 4$), we find a Puiseux series for the unique positive root s (and also for the communication and rate, given optimal s). Fourth, we describe an efficient $\log m$-time algorithm to find an integer approximation for the optimal value s. As we will show, this results in rate that is very close to 1 in practically relevant scenarios.

5.1 Rewording the Optimization Problem

In App. A, we show that

$$\frac{\partial \mathsf{com}}{\partial s_1} = \cdots = \frac{\partial \mathsf{com}}{\partial s_m} = 0 .$$

Since all s_i are positive, then the global minimum is reached if $s_1 = \ldots = s_m =: s$ for *some* s. (See App. A for a proof; to establish this result we need to assume that the underlying branching program is oblivious and that $\chi = m$.) In particular, this means that the optimal communication complexity does not depend on the fact whether the encrypter knows the values $s_i^{\max}$. Since all s_d-s are equal, we denote

$$\begin{aligned}\mathsf{com}(\chi, w, m, s, k, \ell) :=& \mathsf{com}(\chi, w, m, s, \dots, s, k, \ell) \\ =& (w-1)\chi(s+1)k + \left(1 + \frac{1}{s}\right)^m \cdot \ell .\end{aligned} \tag{6}$$

Now, $\partial\mathsf{com}/\partial s = (w-1)\chi k - m(s+1)^{m-1}/s^{m+1} \cdot \ell$. Denoting

$$\sigma := \frac{(w-1)\chi k}{m\ell} , \tag{7}$$

$\partial\mathsf{com}/\partial s = 0$ if and only if s is a root of the univariate polynomial $f_m(\cdot, \sigma)$, where

$$f_m(x, y) := yx^{m+1} - (x+1)^{m-1} . \tag{8}$$

According to the Descartes' rule of signs, $f_m(\cdot, \sigma)$ has exactly one positive real root for each $m > 0$. Thus, this unique positive real root s minimizes the function com.

5.2 Computing Puiseux Series

The famous theorem of Abel-Ruffini states that polynomial equations of degree $n \geq 5$ are not solvable by radicals (see, e.g., [35]). In particular, by using Galois theory we can show that the polynomial $f_m(x, 1)$ is not solvable in radicals for say $m = 4$ (see Sect. 6); one can prove a similar result for any even $m \geq 4$.

Since we are interested in a solution s of $f_m(s, \sigma)$ for *all* m and well-chosen σ, we apply the classical Newton-Puiseux method [39] which can be used given any polynomial f. The polynomial $f_m(x, y) \in \mathbb{R}[x, y]$ can be considered as an algebraic curve on a plane. The Puiseux series are power series with fractional exponents. Computing a power series expansion for y can be seen as solving a polynomial equation in one variable over the field of Puiseux series. Since the field of formal Puiseux series is algebraically closed (see Theorem 3.1 from [39]), a root can always be found. Although this method involves a power series substitution and solving equations at the determination of each expansion coefficient, it is rather simple in our case of the unique root, demanding only two iterations.

More precisely, the *Puiseux series* of $g(x)$ is a series of type $g(x) = \sum_{i=0}^{\infty} a_i x^{i/n}$, where n is some integer. We use the Newton-Puiseux algorithm [39] to find the Puiseux series for the unique positive root s of f_m. By the previous discussion, this will also be the Puiseux series for the value of s that minimizes com. First, it is known [39] that the Puiseux series exists, i.e., $s = \sum_{i=0}^{\infty} a_i \sigma^{i/n}$ for *some* a_i and n. We find this series by assuming that

$$s = c_0\sigma^{\gamma_0} + c_1\sigma^{\gamma_0+\gamma_1} + c_2\sigma^{\gamma_0+\gamma_1+\gamma_2} + \dots \ ,$$

and then finding c_i and γ_i one by one. The exponents γ_i are defined as certain slopes of the Newton polygon [39] for $f_m(s, \sigma) = 0$. After a while we form a hypothesis about the general formula for c_i and prove it.

See Appendix B for a more detailed description of the Newton-Puiseux algorithm and for the proofs of the following theorems.

Theorem 3. *Let σ be as in Eq. (7). The Puiseux series of the unique positive root s of $f_m(\cdot, \sigma)$ is*

$$s = \sum_{i=0}^{\infty} c_i \sigma^{(i-1)/2} = \sigma^{-1/2} + \frac{m-1}{2} - \frac{m^2-1}{8} \cdot \sqrt{\sigma} + O(\sigma) \ , \tag{9}$$

where

$$c_i = (-1)^{i+1} \frac{(m-1)}{2^i i!} \frac{((i-1)(m+1))!!}{((i-1)(m-1))!!} \ .$$

Denote by

$$s_{(i)} := \sum_{j=0}^{i-1} c_j \sigma^{(j-1)/2}$$

the sum of the first i elements of this Puiseux series. In practice (see Sect. 5.3), it suffices to know the values

$$s_{(1)} = \sigma^{-1/2} \quad \text{and} \quad s_{(2)} = \sigma^{-1/2} + (m-1)/2 \ .$$

Knowing the Puiseux series of s, we may substitute it into the communication function com of Eq. (6), deriving thus the Puiseux series for the communication.

Theorem 4. *Let $f : \{0, \dots, w-1\}^{\chi} \to \{0, 1\}^{\ell}$ be computable by a polynomial-size w-ary branching program P_f of length m. Consider parameters fixed in the current section. The leveled LHE scheme of Sect. 4 for f has communication*

$$\ell + 2\sqrt{(w-1)\chi mk\ell} + \frac{w-1}{2}\chi(m+1)k + O(\ell^{-1/2})$$

and rate

$$1 - 2\sqrt{\frac{(w-1)\chi mk}{\ell}} + O(\ell^{-1}) \ .$$

See App. C for a detailed statement (with a precise series expression) and a proof. According to the preceding discussion, the above rate is the best possible that one can achieve by using the leveled LHE scheme of Sect. 4, given that the underlying branching program is oblivious and $\chi = m$. However, in the upper bound of Thm. 4 we only need the branching program to be leveled.

5.3 Algorithm for Approximation of Root

Next, we propose a simple algorithm that finds the best integer approximation to the unique positive root s of Eq. (8) in $\approx \log_2 m$ steps. Clearly, in our application, an integer approximation is sufficient. Let σ be as defined in Eq. (7). First, we show that for partial sums $s_{(1)} = \sigma^{-1/2}$ and $s_{(2)} = \sigma^{-1/2} + (m-1)/2$ as defined after Thm. 3, $f_m(s_{(1)}, \sigma)$ is negative and $f_m(s_{(2)}, \sigma)$ is positive. Since $0 < s_{(1)} < s_{(2)}$, we know that the only positive root s of $f_m(x, \sigma)$ is in the interval $(s_{(1)}, s_{(2)})$ of length $(m-1)/2$. We compute the integer approximation to s (that is sufficient for our purposes) by using binary search over this interval, see Fig. 4.

Lemma 1. *$f_m(s_{(1)}, \sigma) < 0$ and $f_m(s_{(2)}, \sigma) \geq 0$. Moreover, $f_1(s_{(2)}, \sigma) = 0$ and $f_m(s_{(2)}, \sigma) > 0$ for $m > 1$.*

Proof. The case $m = 1$ is trivial. Assume that $m \geq 2$. Then $f_m(s_{(1)}, \sigma) < 0$, since $\sigma x^{m+1} = x^{m-1} < (1+x)^{m-1}$ for $x = s_{(1)} = \sigma^{-1/2}$.

```
1. $s_L \leftarrow \lfloor s_{(1)} \rfloor$, $s_H \leftarrow \lceil s_{(2)} \rceil$.
2. While $s_H > s_L + 1$:
   – $s_M \leftarrow \lfloor (s_L + s_H)/2 \rfloor$.
   – If $f_m(s_M, \sigma) > 0$ then $s_H \leftarrow s_M$
     else $s_L \leftarrow s_M$.
3. If $\mathsf{com}(\chi, w, m, s_L, k, \ell) < \mathsf{com}(\chi, w, m, s_H, k, \ell)$
   then return $s \leftarrow s_L$
   else return $s \leftarrow s_H$.
```

Fig. 4. Finding integer approximation to root s

It remains to show that $f_m(s_{(2)}, \sigma) \geq 0$. Since

$$f_m(x, \sigma) = x^{m+1}/s_{(1)}^2 - (x+1)^{m-1} \ ,$$

it suffices to show $s_{(2)}^{m+1}/s_{(1)}^2 \geq (s_{(2)} + 1)^{m-1}$. This follows from the following estimation:

$$\begin{aligned}\left(\frac{s_{(1)}}{s_{(2)}}\right)^2 \left(1 + \frac{1}{s_{(2)}}\right)^{m-1} &= \left(1 - \frac{m-1}{2s_{(2)}}\right)^2 \left(1 + \frac{1}{s_{(2)}}\right)^{m-1} \\ &\leq e^{-\frac{m-1}{s_{(2)}}} e^{\frac{m-1}{s_{(2)}}} = 1.\end{aligned}$$

□

Theorem 5. *The algorithm on Fig. 4 finds the best integer approximation of s in $\approx \log_2((m-1)/2) \approx \log_2 m$ steps. Its computational complexity is dominated by $\approx \log_2 m$ evaluations of f_m.*

Proof. The algorithm finds the unique unit interval $[s^*, s^* + 1]$ in the original interval that contains the root. It does so by using binary search. The number of the steps is clearly (approximately) logarithmic in the length of the interval, $(m-1)/2$. □

6 On Solvability in Radicals

By using well-known methods of the Galois theory, see [27] for more details, we now give a proof sketch that the Galois group of

$$g(x) := f_4(x, 1) = x^5 - (x+1)^3$$

is the symmetric group S_5. Since S_5 is not a solvable group, this means that $g(x)$ is not solvable in radicals.

First, some background. Let F be a field and $f(x) \in F[X]$ be a polynomial of degree n. A *splitting field E/F of a polynomial f over F* is the smallest field extension E, $F \subset E$, over which the polynomial f decomposes into linear factors, i.e., exist $\alpha_1, \ldots, \alpha_n \in E$ such that

$$f(x) = k(x - \alpha_1) \ldots (x - \alpha_n)$$

and

$$E = F(\alpha_1, \ldots, \alpha_n) \ .$$

A polynomial f is called *separable* if it has n distinct roots in the splitting field E/F of f over F. A splitting field of a separable polynomial f is called *Galois extension*. The group $\mathrm{Aut}_F(E)$ of all automorphisms γ of a Galois extension E/F that fix F pointwise, i.e., $\gamma(x) = x$ for each $x \in F$, is called the *Galois group of the extension E/F* and is denoted by $\mathrm{Gal}(E/F)$. *The Galois group* $\mathrm{Gal}(f(x))$ *of the polynomial* f is then the Galois group of $\mathrm{Gal}(E/F)$. Thus, elements of Galois group are interpreted as permutations of the roots of some polynomial that decomposes over E. It is a well known fact that the Galois group $\mathrm{Gal}(f(x))$ is isomorphic to a subgroup of the symmetric group S_n.

While it is hard to determine Galois groups in general, special arguments can be exploited sometimes. By *discriminant* of polynomial $f(x)$ we call the value $D_f = \delta^2$, where

$$\delta = \Pi_{1 \leq j < i \leq n} (\alpha_i - \alpha_j) \ .$$

The Galois group $\mathrm{Gal}(E/F) \leq S_n$ is contained in the group of even permutations A_n if and only if D_f is a square in F. Now, consider the polynomial $g(x) := f_4(x, 1) \in \mathbb{Z}[X]$; it has discriminant $3017 = 7 \cdot 431$. Clearly, $g(x)$ is irreducible modulo 2, so it is irreducible over $\mathbb{Q}$.

The Dedekind theorem (see [27], Thm. F13) states that if a polynomial $g(x) \in \mathbb{Z}[x]$ is factored into irreducible factors modulo a prime not dividing the discriminant, then the Galois group $\mathrm{Gal}(g(x))$, considered as a subgroup of S_n, contains a permutation whose cycle type corresponds to the degrees of the irreducible factors. According to Sylow Theorems (see [27], p. 100), S_5 has plenty different subgroups. Since

$$g(x) \equiv (x^2 + x + 2)(x^3 + 2x^2 + x + 1) \pmod 3 \ ,$$

by the Dedekind theorem, $\mathrm{Gal}(g(x))$ contains a permutation of the roots with the cycle type $(2, 3)$, i.e., two cycles with length 2 and 3 respectively.

Since the order of a permutation of the cycle type $(2, 3)$ is 6, and since the degree of the irreducible polynomial $g(x)$ is 5, the Galois group of $g(x)$ over $\mathbb{Q}$ has order divisible by $5 \cdot 6 = 30$. As $\mathrm{Gal}(g(x))$ embeds into S_5, $|\mathrm{Gal}(g(x))|$ is either 30, 60, or 120. Since $g(x)$'s discriminant is not a rational square, $\mathrm{Gal}(g(x))$ is not the only subgroup of order 60, that is A_5. Because there are no subgroups of S_5 with order 30, this proves that $\mathrm{Gal}(g(x)) = S_5$.

7 Applications

7.1 Rate-Optimal CPIR

Given the new leveled LHE scheme, the construction of a rate-optimal $(n, 1)$-CPIR is straightforward. Following [26], in the $(n, 1)$-CPIR protocol, we let the client first generate a new Damgård-Jurik public and secret key pair, and then send to the server the public key together with an encryption of every individual bit of the index x. The server represents her database $\vec{f}$ as a compact leveled branching program P_f that computes the function f where $f(x) := f_x$, and then evaluates securely the client's query on top of it, i.e., P_f has n sink nodes, each with an ℓ-bit label. The client obtains the encrypted source value, and then decrypts it.

When using the new leveled LHE scheme, the computational and communication complexity of the CPIR protocol are as per Thm. 2. Thus, the resulting CPIR protocol has both optimal rate (when using the parameters derived in Sect. 5), and (given the database is sufficiently redundant) sublinear-in-n computational complexity. If the database is not redundant, then the server represents it as an w-ary tree of length m.

In most of the applications of the LHE, $w = 2$, which is also often the optimal case. The following corollary shows however that this not always the case.

Corollary 1. *Assume that the DCR assumption [33] is true. There exists a CPA-secure* $(n, 1)$*-CPIR protocol with communication*

$$\ell + 1.72 \cdot \log_2 n \cdot \sqrt{k\ell} + 2(\log_5^2 n + \log_5 n)k + O(\ell^{-1/2})$$

and rate

$$1 - 1.72 \cdot \log_2 n \cdot \sqrt{k/\ell} + O(\ell^{-1}) \ .$$

Proof. Follows from preceding discussion and Thm. 4 by setting $m = \chi = \log_w n$, and considering the full w-ary decision tree (note that it is leveled). Thus, the $(n, 1)$-CPIR protocol has communication

$$\ell + \frac{2\sqrt{w-1}}{\log_2 w} \log_2 n \cdot \sqrt{k\ell} + O(1) \ .$$

Since w is an integer, the second coefficient in this series is minimized when $w = 5$. □

7.1.1 Numerical Examples

Next, we provide examples with concrete parameters. Consider the setting of $(n, 1)$-CPIR, where each database element is a movie, and a paying client wishes to obtain privately one movie. Differently from Eq. (4), we also assume here that all the intermediate values t_d are integers, that is,

$$t_0 = \lceil \ell/(s \cdot k) \rceil \ , \quad \text{and}$$
$$t_d = \left\lceil \left(1 + \frac{1}{s}\right) t_{d-1} \right\rceil \quad \text{for } d \in [1, m] \ . \tag{10}$$

For t_m defined accordingly, and recalling that here $w = 5$ and $\chi = m = \log_w n$ (and, w.l.o.g., assuming that n is a power of w), this changes the communication function from Eq. (6) to

$$\mathsf{com}(n, s, k, \ell) := 4m(s + 1)k + t_m s \cdot k \ . \tag{11}$$

Assume that $k = 2048$, $\ell = 10^6 k$ (about 256 Megabytes), $m = \chi = 7$, and $w = 5$. (Thus $n = w^m = 5^7 = 78\,125$, which allows to select between more movies than any of the current commercial online store is offering.) The algorithm on Fig. 4 starts with the interval (s_L, s_H), where $s_L = 500$ (communication of 2111545344 bits) and $s_H = 503$ (2107731968 bits). After only 2 steps, the interval is (s_L, s_H), where $s_L = 502$ (2109710336 bits) and $s_H = 503$, and thus the algorithm outputs $s_H = 503$ as the desired integer approximation of the optimal length parameter s. This results in communication of 2107731968 bits. The achieved rate is 0.971661. The preceding theoretical analysis, i.e., with communication function as in Eq. (6) and not as in Eq. (11), would give that the communication is 2105572921 bits and the rate is 0.972657; thus, our earlier theoretical analysis is very precise.

In Tbl. 1, we have tabulated the resulting rate (together with the final value of s) for some other values of ℓ. In all cases, $k = 2048$, $m = \chi = 7$, $w = 5$, and thus $n = w^m = 5^7 = 78\,125$. The case $\ell = 10^7 k$ corresponds to the realistic size of a Bluray movie, while the cases $\ell = 200k$, $\ell = 1200k$, and $\ell = 6.95 \cdot 10^4 k$ correspond to the rate 1/4 (as achieved by the CPIR protocol of [18]), 1/2 (as achieved by the CPIR protocol of [26]), and 0.9, respectively. In all cases, the approximation algorithm from Fig. 4 determinates after 2 steps.

7.2 Rate-Optimal Oblivious Transfer

A 1-*out-of-n oblivious transfer (OT)* protocol is a 1-out-of-n CPIR protocol that also satisfies server's privacy. That is, even a malicious client will get information only about one database element, and not more. There exist various security definitions for OT, and for simplicity we only consider the following one. A 1-out-of-n OT

ℓ	s	rate
$200k = 409.6$KB	10	0.271013
$1200k = 2.4576$MB	20	0.511077
$10^4k = 20.48$MB	53	0.765346
$6.95 \cdot 10^4k \approx 142.3$MB	135	0.901275
$10^5k = 204.8$MB	162	0.915617
$10^6k = 2.048$GB	503	0.971661
$10^7k = 20.48$GB	1585	0.991067

Table 1. The final value of s and the rate for some ℓ

protocol is *semi-simulatable* [29], if it is a CPIR protocol (i.e., it satisfies client's CPA privacy), and in addition, it satisfies server's privacy in the sense of simulatability. That is, we make a comparison to the ideal model, where there exists a TTP that gets the client's input x, server's input $\vec{f}$, and outputs f_x. See [29] for more details.

Naor and Pinkas [29] proposed an efficient transformation from any 1-out-of-n CPA-secure CPIR protocol to a semi-simulatable 1-out-of-n OT protocol that makes one call to the CPIR protocol, and a logarithmic number of calls to 1-out-of-2 OT protocols and a linear number of calls to a pseudorandom function.

More precisely, assume the server's database is $\vec{f} = (f_1, \dots, f_n)$, where $|f_i| = \ell$ and the client input is $x = (x_1, \dots, x_\ell)$, where $x_i \in \{0, 1\}$. The server first selects randomly 2ℓ keys $(k_1^0, k_1^1), \dots, (k_\ell^0, k_\ell^1)$, where $|k_i^j| = \kappa$, and computes $f'_1, \dots, f'_n$, where

$$f'_x = f_x \oplus \bigoplus_{i=1}^{\ell} \mathrm{PRF}(k_i^{x_i}, x)$$

for $i \in [n]$. (See [29] for possible optimizations.) The two parties first run the optimal-rate CPIR protocol to transmit f'_x; then they run ℓ 1-out-of-2 OT protocols to transmit the corresponding keys $k_i^{x_i}$. It is known that 1-out-of-2 OT protocols can be constructed with a constant rate [30]; moreover, the key size is much smaller than the ℓ. It is easy to see that this transformation preserves the rate.

Corollary 2. *Assume that the DCR assumption [33] is true, there exists a 1-out-of-2 OT protocol for κ-bit strings with communication complexity $\Theta(\kappa)$, and there exists a pseudorandom function* PRF : $\{0,1\}^\kappa \times \{1, \dots, n\} \to \{0,1\}^\ell$. *Then there exists a two-message semi-simulatable $(n, 1)$-OT protocol (with computational server's privacy) with the same asymptotic communication and rate as the CPIR protocol of Cor. 1.*

Proof. Follows from Cor. 1 and [29]. □

The Naor-Pinkas transformation only offers computational server privacy due to the reliance on pseudorandom number generators. Aiello, Ishai, and Reingold [1] proposed another rate-preserving CPIR-to-OT transformation that results in information-theoretic server's privacy. Their transformation assumes that the CPIR protocol uses a homomorphic public-key cryptosystem where the plaintext group has a prime order. Laur and Lipmaa [23] modified the Aiello-Ishai-Reingold transformation to work with the Damgård-Jurik cryptosystem. However, the Laur-Lipmaa transformation — that relies on concrete properties of the Damgård-Jurik cryptosystem — is not rate preserving. We leave the construction of a CPIR-to-OT transformation that is simultaneously rate-preserving, guarantees information-theoretic server's privacy, and works on top of Damgård-Jurik cryptosystem, as an open problem.

Finally, to achieve the stronger security notion of simulatability [5], instead of using the Naor-Pinkas transformation, the encrypter can accompany each of his χ ciphertexts with a standard zero-knowledge proof [10, 11] that it encrypts a Boolean value, and then prove that their sum encrypts 1. The total communication complexity of the zero-knowledge proofs is $\Theta(\chi sk) = \Theta(\log n \cdot \sqrt{\ell k}) = o_\ell(\ell)$. (This approach works when $w = 2$, in the case $w > 2$ one can use efficient range proofs [4, 6, 24].) The main drawback of this approach, compared to the Naor-Pinkas transformation, is reliance on zero-knowledge proofs. This either increases the number of rounds, or forces one to rely on the random oracle model.

7.3 Rate-Optimal SCOT

A 1-*out-of-n strong conditional oblivious transfer* ($(n, 1)$-SCOT, [2]) protocol for function Q (s.t. $Q(x, y) \in [n]$) implements securely the following functionality:

$$\mathcal{F}_{Q-\mathrm{SCOT}}(x, (y, f_1, \dots, f_n)) = (f_{Q(x,y)}, \perp) \ .$$

That is, on client's input x and server's input $\vec{f}$, the client obtains $f_{Q(x,y)}$ and the server outputs nothing. In the case of OT, $Q(x, y) = x$. The semi-simulatable security of the SCOT is defined similarly like the semi-simulatable security of OT.

One can use the new rate-optimal leveled LHE scheme to construct an efficient SCOT protocol for the functionality $\mathcal{F}_{Q-\mathrm{SCOT}}(x, (y, \vec{f}))$, where $Q \in \mathbf{LBP}$ has a polynomial-size large-output branching program P'_Q, as follows. Let P_Q be the large-output branching program, obtained from P'_Q by just replacing each leaf value

$i \in \{1, \ldots, n\}$ with f_i. It is easy to see that applying the new LHE scheme with P_Q results in a SCOT protocol that implements $\mathcal{F}_{Q-\mathrm{SCOT}}$.

Corollary 3. *Make the same three assumptions as in Cor. 2. Then there exists a semi-simulatable $(n, 1)$-SCOT protocol (with computational server's privacy) with the same asymptotic communication complexity and rate as the CPIR protocol of Cor. 1.*

Proof. Follows from Cor. 2 and preceding discussion. □

The rate efficiency in a SCOT protocol is equally important for practical use when the data size is large. We note that optimal-rate SCOT can find applications in a similar setting as our optimal-rate CPIR (but for more complex content selection strategies); for instance consider a client that wishes to watch one out of q variants of a video stream where each variant has inserted different style of advertisements. If $Q(x)$ is a BP that produces the index of the video stream variant based on client preferences x then, using our SCOT, the client can stream the video that matches her advertisement preferences without revealing them to the server.

In a recent paper [21], the authors showed how to construct an optimal-rate asymmetric fingerprinting protocol from a rate optimal SCOT for the relation $Q(x, y) = [x \leq y]$, thus answering an open question related to efficient asymmetric fingerprinting codes.

8 On Computational Complexity

In this section, we will analyze the computation complexity of the new CPIR protocol (analysis of the LHE can be done similarly). We show that the new CPIR protocol is significantly more computation efficient for the server than the protocols from [25, 26], an that one can further improve it significantly while decreasing the rate only by a small factor. We emphasize that the following analysis is theoretical, since we lack an implementation.

The server's computational complexity of the previously known variants of the $(n, 1)$-CPIR protocol from [25, 26] and of the resulting leveled LHE schemes [20] is quite high, due to the need to encrypt larger and larger plaintexts during the processing of the branching program. Since one Damgård-Jurik encryption with ℓ-bit modulus takes time $\Omega(\ell^2 \log \ell \log \log \ell)$, when using FFT-based multiplication (in our applications, ℓ is large enough for FFT-based multiplication to perform faster than Karatsuba), elongation of plaintexts is extremely detrimental to computational complexity.

In the new CPIR protocol, we encrypt many shorter plaintexts. Next, we will give an estimate on the server's computation. It is dominated by $n - 1$ encryptions. More precisely, assuming that an encryption (resp., an exponentiation) with plaintext length $s_d k$ takes time $T_{\mathsf{Enc}}(s_d, k)$ (resp., $T_{\exp}(s_d, k)$), the server's computation is dominated by

$$\sum_{d=1}^{m} t_d w^{m-d} (T_{\mathsf{Enc}}(s_d, k) + (w - 1) T_{\exp}(s_d, k))$$

bit-operations.

According to [13], after several optimizations, a single Damgård-Jurik encryption takes $k/4 + 2s_d$ multiplications (modulo N^{s_d+1}). Finally, an exponentiation takes $\frac{1}{2} s_d k$ multiplications. Assuming that one uses FFT-based multiplication, a multiplication modulo N^{s_d+1} takes time $T_*(s_d, k) = \Omega((s_d k)^{1.58} \log(s_d k))$. (For the sake of simplicity, we omit the $\log \log(s_d k)$ term.)

To estimate the server's computation, we note that for large ℓ, $t_d = \Theta(\sqrt{\ell/k})$. Thus, in bit operations, the server's computational complexity is dominated by

$$\begin{aligned}
&\sum_{d=0}^{m-1} (T_{\mathsf{Enc}}(s_d, k) + (w - 1) T_{\exp}(s_d, k)) w^{m-d-1} t_d \\
&= \Theta((T_{\mathsf{Enc}}(s, k) + (w - 1) T_{\exp}(s, k)) \sqrt{\ell/k}) \cdot \\
&\quad \sum_{d=0}^{m-1} w^{m-d-1} = \Theta(T_{\exp}(s, k) \sqrt{\ell/k} \cdot n) \\
&= \Theta(\sqrt{\ell/k} \cdot k T_*(s, k) \sqrt{\ell/k} \cdot n) \\
&= \Theta(\ell^{1.5} \sqrt{k} n \cdot \log(\ell k)) \ .
\end{aligned}$$

For example, the server computation in the CPIR protocol of [25] with the parameters ($k = 2048$, $\ell = 10^6 \cdot k$, $n = 5^7$) given in Sect. 7.1.1, is (when generously forgetting all constants and the log log-term) *at least* $\ell^2 n \log \ell \approx 2^{83.0676}$ bit-operations. The rate-1/2 variant of this CPIR, proposed in [26], has server's computation of *at least* $(\ell / \log n)^2 n / \log n \cdot \log(\ell / \log n) \approx 2^{78.3152}$ bit-operations. With the same parameters, the new CPIR protocol has server computation $\approx 2^{73.5408}$. While this number is still huge, omitting small constant factors, it is approximately 2^{10} (resp., 2^5) times smaller than the computation in the CPIR protocol of [25] (resp., [26]).

8.1 Optimization by Parallelization

Since the server's computation is super-linear in ℓ, a simple solution to reduce it is to execute the CPIR protocol

λ	$\ell = 10^4 \cdot k$			$\ell = 10^6 \cdot k$		
	s	rate	log.comp	s	rate	log.comp
1	53	0.765346	22.8	503	0.971661	26.4
10	53	0.653937	17.7	503	0.952116	21.3
50	52	0.332403	14.1	502	0.837672	17.7
250	—	—	—	502	0.527264	14.1

Table 2. The value of s, rate and logarithm of server's computation (per database bit) for some ℓ and λ

in parallel λ times on $\ell' := (\ell/\lambda)$-bit chunks of the data, for a well-chosen λ; this results in server computation of $\Theta(\ell^{1.5}k^{0.5}\lambda^{-1.5}n\log(\ell k/\lambda))$. Crucially, see Eq. (6), the client's communication does not depend on ℓ while the server's communication is linear in ℓ. Thus, asymptotically, the modified *λ-parallel CPIR protocol* will have exactly the same communication and thus also the rate as in Eq. (6), as long as ℓ' is not too small.

In concrete terms, one has to take into account that the fact that the values t_d must be integers has a larger influence on the rate if t_d is smaller (i.e., λ is larger). More precisely, the communication complexity of the λ-parallel CPIR is equal to

$$\mathsf{com}_\lambda(\chi, w, m, s, k, \ell) := (w-1)\chi(s+1)k + \lambda t'_m s \cdot k \enspace ,$$

where for s computed as on Fig. 4, t'_m is computed recursively from

$$t'_0 = \lceil \ell'/(s \cdot k) \rceil \quad , \quad \text{and}$$
$$t'_d = \left\lceil \left(1 + \frac{1}{s}\right) t'_{d-1} \right\rceil \quad \text{for } d \in [1, m] \enspace .$$

Here, we assume that $\lambda \mid \ell$.

Using the same parameters as in Sect. 7.1.1 ($k = 2048$, $n = 5^7$, and either $\ell = 10^4 \cdot k$ or $\ell = 10^6 \cdot k$; we note that the case $\lambda > s$ does not make sense), we calculated the rate and logarithm of the server's computation for some values of λ, see Tbl. 2. More precisely, the last column of Tbl. 2 has the value $\log_2(\ell^{1.5}k^{0.5}\lambda^{-1.5}n\log(\ell k/\lambda)) - \log_2(\ell n)$, i.e., the number of server's bit-operations per database bit (clearly, the database has ℓn bits).

If $\ell = 10^6 \cdot k$, then even with $\lambda = 250$ the new CPIR has rate $> 1/2$. On the other hand, for $\ell = 10^6 k$, the 250-parallel version is $\approx 2^{26.4-14.1} = 2^{12.3}$ times faster than the 1-parallel version, and thus $\approx 2^{12} \cdot 2^5 \approx 2^{17}$ times faster than the rate 1/2 CPIR protocol of [26].

As seen from Tbl. 2, the 250-parallel version spends $\approx 2^{61-47} = 2^{14} \approx 16\,000$ bit-operations per database bit. To compare, a single 2048-bit exponentiation (when using FFT-based multiplication) takes approximately $2048^2 \cdot \log_2 2048 \approx 2^{25}$ bit-operations. Hence, in the parallel version of the CPIR protocol, the server roughly has to compute a single 2048-bit exponentiation per each 2048 database bits.

The preceding discussion indicates that the actual rate function (and not only the fact that it is $1 - o(1)$) of the CPIR protocol matters: if the rate function would be even slightly smaller, it might happen that using (say) $\lambda = 250$ would result in the much worse rate. With the new CPIR protocol, choosing a large λ introduces only a minor change to the rate.

8.2 Further Work

We expect that one can apply many more optimizations, but we leave their study (together with an optimized implementation) for a future work. We only mention that one can implement the λ-parallel version on λ different servers, as done say in [34] in the context of the CPIR protocol from [18]. This reduces the computation of every single server by an additional factor of λ; however, in this case the client's communication complexity will increase λ times to $\lambda(w-1)\chi(s+1)k \approx 2\lambda \cdot \sqrt{\ell k}\log_2 n$. See [14, 15, 31] and references therein for other possible optimizations. We will leave it as an another open question to construct a version of the new CPIR protocol that has the best trade-offs between rate and server's computation.

To conclude, we think that the new CPIR protocol (if considered as a *reduction* that utilizes an existing cryptosystem) by itself is also computationally efficient. The main bottleneck is clearly in the underlying cryptosystem. Hence, an important open question posed by the current work is to construct an optimal-rate additively homomorphic cryptosystem with significantly faster computational (encryption) complexity compared to the Damgård-Jurik cryptosystem.

Acknowledgements

We would like to thank Amir Herzberg and anonymous reviewers for useful comments. This research was performed while Leonardos and Tang were at the University of Athens. Kiayias, Leonardos and Tang were supported by ERC project CODAMODA. Lipmaa was supported by the Estonian Research Council, and European Union through the European Regional Development Fund. Pavlyk was supported by the institutional research funding IUT20-57 of the Estonian Ministry of Education and Research.

References

[1] Aiello, W., Ishai, Y., Reingold, O.: Priced Oblivious Transfer: How to Sell Digital Goods. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 119–135. Springer, Heidelberg, Innsbruck, Austria (May 6–10, 2001)

[2] Blake, I.F., Kolesnikov, V.: Strong Conditional Oblivious Transfer and Computing on Intervals. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 515–529. Springer, Heidelberg, Jeju Island, Korea (Dec 5-9 2004)

[3] Cachin, C., Micali, S., Stadler, M.: Computational Private Information Retrieval with Polylogarithmic Communication. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 402–414. Springer, Heidelberg, Prague, Czech Republic (May 2–6, 1999)

[4] Camenisch, J., Chaabouni, R., shelat, a.: Efficient Protocols for Set Membership and Range Proofs. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 234–252. Springer, Heidelberg, Melbourne, Australia (Dec 7–11, 2008)

[5] Camenisch, J., Neven, G., shelat, a.: Simulatable Adaptive Oblivious Transfer. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 573–590. Springer, Heidelberg, Barcelona, Spain (May 20–24, 2007)

[6] Chaabouni, R., Lipmaa, H., shelat, a.: Additive Combinatorics and Discrete Logarithm Based Range Protocols. In: Steinfeld, R., Hawkes, P. (eds.) ACISP 2010. LNCS, vol. 6168, pp. 336–351. Springer, Heidelberg, Sydney, Australia (Jul 5–7, 2010)

[7] Chor, B., Goldreich, O., Kushilevitz, E., Sudan, M.: Private Information Retrieval. In: FOCS 1995. pp. 41–50. IEEE, Milwaukee, Wisconsin, USA (Oct 23–25 1995)

[8] Choudhury, A., Loftus, J., Orsini, E., Patra, A., Smart, N.P.: Between a Rock and a Hard Place: Interpolating between MPC and FHE. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013 (2). LNCS, vol. 8270, pp. 221–240. Springer, Heidelberg, Bangalore, India (Dec 1–5, 2013)

[9] Cobham, A.: The Recognition Problem for the Set of Perfect Squares. In: FOCS 1966. pp. 78–87. IEEE Computer Society, Berkeley, California (Oct 23–25, 1966)

[10] Cramer, R., Damgård, I., Schoenmakers, B.: Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 174–187. Springer, Heidelberg, Santa Barbara, USA (Aug 21–25 1994)

[11] Damgård, I., Jurik, M.: A Generalisation, a Simplification and Some Applications of Paillier's Probabilistic Public-Key System. In: Kim, K. (ed.) PKC 2001. LNCS, vol. 1992, pp. 119–136. Springer, Heidelberg, Cheju Island, Korea (Feb 13–15, 2001)

[12] Damgård, I., Jurik, M.: A Length-Flexible Threshold Cryptosystem with Applications. In: Safavi-Naini, R. (ed.) ACISP 2003. LNCS, vol. 2727, pp. 350–364. Springer, Heidelberg, Wollongong, Australia (Jul 9-11, 2003)

[13] Damgård, I.B., Jurik, M.J., Nielsen, J.B.: A Generalization of Paillier's Public-key System with Applications to Electronic Voting. Int. J. Inf. Sec. 9(6), 371–385 (2010)

[14] Demmler, D., Herzberg, A., Schneider, T.: RAID-PIR: Practical Multi-Server PIR. In: Oprea, A., Safavi-Naini, R. (eds.) ACM CCSW 2014. ACM Press, Scottsdale, Arizona, USA (Nov 7, 2014)

[15] Devet, C., Goldberg, I.: The Best of Both Worlds: Combining Information-Theoretic and Computational PIR for Communication Efficiency. In: Cristofaro, E.D., Murdoch, S.J. (eds.) PETS 2014. LNCS, vol. 8555, pp. 63–82. Springer, Heidelberg, Amsterdam, The Netherlands (Jul 16–18, 2014)

[16] Gentry, C.: A Fully Homomorphic Encryption Scheme. Ph.D. thesis, Stanford (Sep 2009)

[17] Gentry, C.: Fully Homomorphic Encryption Using Ideal Lattices. In: Mitzenmacher, M. (ed.) STOC 2009. pp. 169–178. ACM Press, Bethesda, Maryland, USA (May 31 — Jun 2, 2009)

[18] Gentry, C., Ramzan, Z.: Single-Database Private Information Retrieval with Constant Communication Rate. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 803–815. Springer, Heidelberg, Lisboa, Portugal (2005)

[19] Groth, J., Kiayias, A., Lipmaa, H.: Multi-Query Computationally-Private Information Retrieval with Constant Communication Rate. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 107–123. Springer, Heidelberg, Paris, France (May 26–28, 2010)

[20] Ishai, Y., Paskin, A.: Evaluating Branching Programs on Encrypted Data. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 575–594. Springer, Heidelberg, Amsterdam, The Netherlands (Feb 21–24, 2007)

[21] Kiayias, A., Leonardos, N., Lipmaa, H., Pavlyk, K., Tang, Q.: Communication Optimal Tardos-based Asymmetric Fingerprinting. In: Nyberg, K. (ed.) CT-RSA 2015. LNCS, vol. 9048, pp. 469–486. Springer, Heildeberg, San Franscisco, CA, USA (Apr 20–24, 2015)

[22] Kushilevitz, E., Ostrovsky, R.: Replication is Not Needed: Single Database, Computationally-Private Information Retrieval. In: FOCS 1997. pp. 364–373. IEEE Computer Society, Miami Beach, Florida (Oct 20–22, 1997)

[23] Laur, S., Lipmaa, H.: A New Protocol for Conditional Disclosure of Secrets And Its Applications. In: Katz, J., Yung, M. (eds.) ACNS 2007. LNCS, vol. 4521, pp. 207–225. Springer, Heidelberg, Zhuhai, China (Jun 5–8, 2007)

[24] Lipmaa, H.: On Diophantine Complexity and Statistical Zero-Knowledge Arguments. In: Laih, C.S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 398–415. Springer, Heidelberg, Taipei, Taiwan (Nov 30–Dec 4, 2003)

[25] Lipmaa, H.: An Oblivious Transfer Protocol with Log-Squared Communication. In: Zhou, J., Lopez, J. (eds.) ISC 2005. LNCS, vol. 3650, pp. 314–328. Springer, Heidelberg, Singapore (Sep 20–23, 2005)

[26] Lipmaa, H.: First CPIR Protocol with Data-Dependent Computation. In: Lee, D., Hong, S. (eds.) ICISC 2009. LNCS, vol. 5984, pp. 193–210. Springer, Heidelberg, Seoul, Korea (Dec 2–4, 2009)

[27] Lorenz, F.: Algebra. Volume I: Fields and Galois Theory. Universitext, Springer (Dec 8, 2005)

[28] Meyer, C.D.: Matrix Analysis and Applied Linear Algebra. SIAM, 1 edn. (Jun 1, 2001)
[29] Naor, M., Pinkas, B.: Oblivious Transfer And Polynomial Evaluation. In: STOC 1999. pp. 245–254. ACM Press, Atlanta, Georgia, USA (May 1–4, 1999)
[30] Naor, M., Pinkas, B.: Efficient Oblivious Transfer Protocols. In: SODA 2001. pp. 448–457. ACM Press, Washington, DC, USA (Jan 7–9, 2001)
[31] Olumofin, F.G., Goldberg, I.: Revisiting the Computational Practicality of Private Information Retrieval. In: Danezis, G. (ed.) FC 2011. LNCS, vol. 7035, pp. 158–172. Springer, Heidelberg, Gros Islet, St. Lucia (Feb 28–Mar 4, 2011)
[32] Ostrovsky, R., Skeith III, W.E.: Communication Complexity in Algebraic Two-Party Protocols. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 379–396. Springer, Heidelberg, Santa Barbara, USA (Aug 17–21, 2008)
[33] Paillier, P.: Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg, Prague, Czech Republic (May 2–6, 1999)
[34] Papadopoulos, S., Bakiras, S., Papadias, D.: pCloud: A Distributed System for Practical PIR. IEEE Trans. Dependable Sec. Comput. 9(1), 115–127 (2012)
[35] Pesic, P.: Abel's Proof: An Essay on the Sources and Meaning of Mathematical Unsolvability. MIT Press (Feb 27, 2004)
[36] Pippenger, N.: On Simultaneous Resource Bounds. In: FOCS 1979. pp. 307–311. IEEE Computer Society Press, San Juan, Puerto Rico (Oct 29–31 1979)
[37] Stern, J.P.: A New And Efficient All Or Nothing Disclosure of Secrets Protocol. In: Ohta, K., Pei, D. (eds.) ASIACRYPT 1998. LNCS, vol. 1514, pp. 357–371. Springer, Heidelberg, Beijing, China (Oct 18–22, 1998)
[38] Stewart, J.: Multivariable Calculus. Cengage Learning, 7 edn. (Jan 1, 2011)
[39] Walker, R.J.: Algebraic Curves. Springer (Oct 4, 2013)
[40] Wegener, I.: Branching Programs and Binary Decision Diagrams: Theory and Applications. Monographs on Discrete Mathematics and Applications, Society for Industrial Mathematics (2000)

A Derivation of Global Minimum

Here we give some details and explanations to Section 5. To determine a minimum of the multivariable communication function com we use the generalization of the second derivative test for multivariable functions [38].

Assuming that the rest of the parameters are fixed, the function com of Eq. (5) has a critical point at $\vec{s} = (s_1, \dots, s_m)$, if *the gradient* (the vector of partial derivatives) of the function com $\nabla\mathsf{com}(\chi, w, m, \vec{s}, k, \ell)$ is 0 at $\vec{s}$, i.e.,

$$\partial\mathsf{com}/\partial s_1 = \cdots = \partial\mathsf{com}/\partial s_m = 0 \ .$$

Recall $m = \chi$. Assume for the sake of simplicity that the branching program is oblivious. Without this assumption, the number of ciphertexts with some length parameter can depend on the result of the following optimization, and this results in a vicious circle. This may mean that there exist cases of non-oblivious branching programs, where the derived result is not optimal. However, as emphasized before, in all our applications we in fact have oblivious branching programs.

Under this assumption, the client will send one ciphertext with every possible length parameter s_d. Thus, it is easy to see that

$$\begin{aligned}\frac{\partial\mathsf{com}}{\partial s_j} &= (w-1)k - \frac{\ell}{s_j^2} \cdot \frac{\prod_{i=1:i\neq j}^{m}(s_i+1)}{\prod_{i=1:i\neq j}^{m} s_i} \\ &= (w-1)k - \frac{\ell}{(s_j+1)s_j} \cdot \frac{\prod_{i=1}^{m}(s_i+1)}{\prod_{i=1}^{m} s_i} = 0\end{aligned}$$

hence,

$$(s_j+1)s_j = \frac{\ell}{(w-1)k} \cdot \frac{\prod_{i=1}^{m}(s_i+1)}{\prod_{i=1}^{m} s_i} \ ,$$

for every j. Then, for every $i \neq j$,

$$s_i(s_i+1) = s_j(s_j+1) \ .$$

and thus either $s_i = s_j$, or $s_i = -1 - s_j$. But since s_i and s_j both have to be positive, we get that

$$s_1 = \cdots = s_m =: s$$

for some s.

Substituting equal values of s_i into com we find $\vec{s}$ as a root of polynomial $f_m(s,\sigma)$ (Eq. (8)). Let $\vec{s}^*$ be a solution of Eq. (8). Next step is to determine if a critical point $\vec{s}^*$ is a local/global minimum of com. The following test can be applied at the critical point of com: if the Hessian matrix which describes the local curvature of a function of many variables

$$H(m,\vec{s}) := \begin{pmatrix} \frac{\partial^2\mathsf{com}}{\partial s_1\partial s_1} & \frac{\partial^2\mathsf{com}}{\partial s_1\partial s_2} & \cdots & \frac{\partial^2\mathsf{com}}{\partial s_1\partial s_m} \\ \frac{\partial^2\mathsf{com}}{\partial s_2\partial s_1} & \frac{\partial^2\mathsf{com}}{\partial s_2\partial s_2} & \cdots & \frac{\partial^2\mathsf{com}}{\partial s_2\partial s_m} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2\mathsf{com}}{\partial s_m\partial s_1} & \frac{\partial^2\mathsf{com}}{\partial s_m\partial s_2} & \cdots & \frac{\partial^2\mathsf{com}}{\partial s_m\partial s_m} \end{pmatrix}$$

is positive definite at $\vec{s}^*$, then com attains a local minimum at $\vec{s}^*$. Clearly, a local minimum of a convex function is also a global minimum. A continuous twice differentiable function of several variables is convex on a convex set if and only if its Hessian matrix is positive semidefinite on the interior of the convex set (see [28], Sect. 7.6). Since a positive definite matrix $H(m,\vec{s}^*)$ is also positive semidefinite, $\mathsf{com}(\chi, w, m, \vec{s}, k, \ell)$ is a convex function on a convex domain $\vec{s} > 0$ and $\vec{s}^* = (s^*, \ldots, s^*)$ is indeed its global minimum.

We will not define positive and semipositive definite matrices. Instead, we apply the Sylvester's criterion of positive definitiveness: a matrix M is positive definite iff the determinants associated with all upper-left submatrices of M are positive, [28]. We first compute

$$\frac{\partial^2 \mathsf{com}}{\partial s_i^2} = \frac{2\ell \prod_{\kappa \neq i}(s_\kappa + 1)}{s_i^2 \prod_\kappa s_\kappa} \quad \text{and}$$
$$\frac{\partial^2 \mathsf{com}}{\partial s_i \partial s_j} = \frac{\ell \prod_{\kappa \neq i, \kappa \neq j}(s_\kappa + 1)}{s_i s_j \prod_\kappa s_\kappa} \quad \text{for } i \neq j \ .$$

Note that those values do not depend on m. All upper-left submatrices of $H(m, \vec{s})$ are circulant matrices with determinants of next form

$$\begin{vmatrix} a & b & b & \vdots & b \\ b & a & b & \vdots & b \\ \cdots & \cdots & \cdots & \ddots & \cdots \\ b & b & b & \vdots & a \end{vmatrix} = b(a-b)^\kappa \left(\frac{1}{b} + \frac{\kappa}{a-b}\right) \ ,$$

where $\kappa \in \{0, \ldots, m-1\}$,

$$a = \frac{\partial^2 \mathsf{com}(\chi, w, m, \vec{s}^*, k, \ell)}{\partial s_1^2} = \ldots$$
$$= \frac{\partial^2 \mathsf{com}(\chi, w, m, \vec{s}^*, k, \ell)}{\partial s_m^2} = \frac{2\ell(s^*+1)^{\kappa-1}}{(s^*)^{\kappa+2}} \quad \text{and}$$
$$b = \frac{\partial^2 \mathsf{com}(\chi, w, m, \vec{s}^*, k, \ell)}{\partial s_i \partial s_j} = \frac{\ell(s^*+1)^{\kappa-2}}{(s^*)^{\kappa+2}} \ ,$$

for all $i, j \in \{0, \ldots, \kappa - 1\}$. Since $a, b > 0$ and $a - b > 0$, the Hessian matrices $H(m, \vec{s}^*)$ are positive definite for all $m > 1$ and $\vec{s}^*$ is an absolute minimum of com.

B Proof of Thm. 3

Proof. Given a polynomial $f(x, y)$, it is always possible to solve it for y in terms of x by means of a fractional power series $y = \sum_{i=0}^\infty c_i x^{\Sigma_{j=0}^i \gamma_j}$, so called *Puiseux series.* This method is based on the algebraic closure of the field of fractional power series (the Puiseux's Theorem), while the proof of Puiseux Theorem can be given constructively by the Newton polygon method. We give first a general description of the *Newton-Puiseux algorithm* and then apply it to Eq. (8). More details can be found in Chapter IV, Sect. 1-3 of [39].

Let

$$f(x, y) = a_0(x) + a_1(x)y + \cdots + a_n(x)y^n$$

be a polynomial of degree $n > 0$, with $a_n \neq 0$. We will recursively construct a solution $y(x)$, a Puiseux series in x, of the equation $f(x, y) = 0$. The solution $y(x)$ must have the form $y(x) = c_0 x^{\gamma_0} + y_1(x)$ for

$$y_1(x) = c_1 x^{\gamma_0 + \gamma_1} + c_2 x^{\gamma_0 + \gamma_1 + \gamma_2} + \cdots \ ,$$

with $c_j \neq 0$, $\gamma_j \in \mathbb{Q}$, for all j. In order to get necessary conditions for c_0 and γ_0, we substitute $y(x) = c_0 x^{\gamma_0} + y_1(x)$ for y in $f(x, y)$, getting

$$f(x, y(x)) = \sum b_{ij} x^i y_1^j = 0 \ .$$

The terms of lowest order must cancel, whence we obtain c_0. One can determine the possible value for γ_0 by considering the *Newton polygon* of f. This is the smallest convex polygon in the affine plane over $\mathbb{Q}$, which contains all the points $P_i = (i, j)$ from the expansion of $f(x, y(x))$. Those faces of the Newton polygon, s.t. all the P_i's lie on or above the corresponding line, have possible values for γ_0 as their negative slopes. After γ_0 and c_0 have been determined, the same process is performed on $y_1(x)$, which must be a root of the equation $f_1(x, y_1) = f(x, c_0 x^{\gamma_0} + y_1) = 0$. The Newton polygon may again be used to derive necessary conditions on c_1 and γ_1. This recursive process can be iterated until the desired number of terms is computed, or no further splitting of solutions is possible.

Now, we rewrite Eq.(8) as

$$f_m(s, \sigma) := \sum_{i,j=0}^{m+1} b_{ij} s^i \sigma^j = 0 \qquad (12)$$

and construct the solution $s(\sigma)$ of Eq. (12) recursively as a Puiseux series. Let $s(\sigma) = c_0 \sigma^{\gamma_0} + c_1 \sigma^{\gamma_0 + \gamma_1} + c_2 \sigma^{\gamma_0 + \gamma_1 + \gamma_2} + \ldots$, with $c_i \neq 0$, $\gamma_i \in \mathbb{Q}$, $\gamma_i > 0$ for all i, which can be also rewritten as

$$s(\sigma) = \sigma^{\gamma_0}(c_0 + s_1) \ ,$$

where

$$s_1 = c_1 \sigma^{\gamma_1} + c_2 \sigma^{\gamma_1 + \gamma_2} + c_3 \sigma^{\gamma_1 + \gamma_2 + \gamma_3} + \ldots \ .$$

To find γ_0 we construct polygon in the affine plane over $\mathbb{Q}$, which contains all the points $P_i = (i, j)$, where (i, j) are the pairs of indices (i, j) of $f_m(s, \sigma) = \sum b_{ij} s^i \sigma^j$. Those faces of the polygon, such that all the P_i's lie on or above the corresponding line, have possible values for γ_0 as their negative slopes. Since, $f_m(s, \sigma)$ has a non-zero b_{ij} iff $(i, j) = (i, 0)$ for $i \in [0, m-1]$, or $(i, j) = (m+1, 1)$, the Newton polygon in this case has one non-horizontal slope, from $(m-1, 0)$ to $(m+1, 1)$. It's equation is

$$y = -\gamma_0 x + \beta_0 = \frac{1}{2}x - \frac{m-1}{2}.$$

Hence $\gamma_0 = -\frac{1}{2}$, $\beta_0 = -\frac{m-1}{2}$ and

$$s(\sigma) = \sigma^{-1/2}(c_0 + s_1) \ .$$

To find c_0 we substitute $\sigma^{-1/2}(c_0 + s_1)$ for s into

$$f_m(s,\sigma) = \sigma s^{m+1} - (s+1)^{m-1} \ ,$$

getting

$$\begin{aligned} f_m(s,\sigma) =& \sigma(\sigma^{-1/2}(c_0+s_1))^{m+1} - \\ & (\sigma^{-1/2}(c_0+s_1)+1)^{m-1} \\ =& \sigma^{-\frac{m-1}{2}} \sum_{i=0}^{m+1} \binom{m+1}{i} c_0^{m+1-i} s_1^i - \\ & \sum_{i=0}^{m-1} \binom{m-1}{i} \sigma^{-\frac{i}{2}} \sum_{j=0}^{i} \binom{i}{j} c_0^{i-j} s_1^j \\ =& \sigma^{-\frac{m-1}{2}} \left(\sum_{i=0}^{m+1} \binom{m+1}{i} c_0^{m+1-i} s_1^i - \right. \\ & \left. \sum_{j=0}^{m-1} \binom{m-1}{j} c_0^{m-1-j} s_1^j \right) - \\ & \sum_{i=0}^{m-2} \binom{m-1}{i} \sigma^{-\frac{i}{2}} \sum_{j=0}^{i} \binom{i}{j} c_0^{i-j} s_1^j \ . \end{aligned}$$

Now, let the terms of lowest order $\sigma^{-(m-1)/2}$ be equal to zero. Then

$$c_0^{m+1} - c_0^{m-1} = 0,$$

or $c_0 \in \{-1, 0, 1\}$. Since Eq. (12) has exactly one positive real root, we skip $c_0 = -1$, considering $c_0 = 1$. Thus,

$$s(\sigma) = \sigma^{-1/2}(1+s_1) = \sigma^{-1/2}(1+c_1\sigma^{\gamma_1}+c_2\sigma^{\gamma_1+\gamma_2}+\dots) \ .$$

In the next recursive step of Newton-Puiseux algorithm γ_2 and c_2 will be determined. In particular, γ_2 is a slope of Newton polygon for $f_m^{(1)}(s_1,\sigma)$, where

$$f_m^{(1)}(s_1,\sigma) = \sigma^{(m-1)/2} f_m(\sigma^{-1/2}(1+s_1),\sigma) \ .$$

is obtained from $f_m(\sigma^{-1/2}(1+s_1),\sigma)$ multiplying it with $\sigma^{-\beta_0} = \sigma^{(m-1)/2}$. To construct Newton polygon for $f_m^{(1)}(s_1,\sigma)$, we rewrite this polynomial as

$$\begin{aligned} & f_m^{(1)}(s_1,\sigma) \\ & = \sum_{i=0}^{m+1} \binom{m+1}{i} s_1^i - \sum_{i=0}^{m-1} \binom{m-1}{i} \sigma^{\frac{m-1-i}{2}} (s_1+1)^i \\ & = \left(s_1^{m+1} + (m+1)s_1^m + \sum_{i=0}^{m-1} \binom{m+1}{i} s_1^i \right) - \\ & \left((s_1+1)^{m-1} + \sum_{i=0}^{m-2} \binom{m-1}{i} \sigma^{\frac{m-1-i}{2}} (s_1+1)^i \right) \ , \end{aligned}$$

which is equal to

$$\begin{aligned} =& s_1^{m+1} + (m+1)s_1^m + \\ & \sum_{i=1}^{m-1} \left(\binom{m+1}{i} - \binom{m-1}{i} \right) s_1^i - \\ & \sum_{i=0}^{m-2} \binom{m-1}{i} \sigma^{\frac{m-1-i}{2}} \sum_{j=0}^{i} \binom{i}{j} s_1^j \ . \end{aligned}$$

This polynomial $f_m^{(1)}(s_1,\sigma) = \sum b_{ij} s_1^i \sigma^j$ has a non-zero coefficient b_{ij} iff

$(i,j) = (x, 0)$ for $x \in [1, m+1]$, or
$(i,j) = \left(x, \frac{m-1-y}{2}\right)$ for $x \in [0, y]$ and $y \in [0, m-2]$.

As seen from Fig. 5, the Newton polygon of $f_m^{(1)}(s_1,\sigma)$ consists of a single segment with only two vertices, one on each axis. This segment has equation

$$y = -\gamma_1 x + \beta_1 = -\frac{1}{2}x + \frac{1}{2} \ .$$

According to algorithm, such situation allows to compute the rest c_i by letting different powers of σ cancel. By the Newton-Puiseux algorithm [39], from now on the powers of σ are going to jump by the denominator of γ_1; in other words

$$\gamma_1 = \gamma_2 = \gamma_3 = \dots = \frac{1}{2} \ .$$

Now, we can calculate the rest of the c_i directly from the function obtained from substitution of

$$s_1 = c_1\sigma^{1/2} + c_2\sigma^{2/2} + c_3\sigma^{3/2} + \dots$$

into $f_m^{(1)}(s_1,\sigma)$ and equating to zero successively all terms which contain σ of degree $\frac{n}{2}$, for all $n \geq 1$.

Next we have to do some technical computations. After substitution we get

$$\begin{aligned} f_m^{(1)}(s_1,\sigma) &= \sigma^{\frac{m-1}{2}} f_m(\sigma^{-1/2}(s_1+1),\sigma) \\ &= (s_1+1)^{m+1} - (s_1+1+\sigma^{1/2})^{m-1} \ , \end{aligned}$$

where

$$s_1 + 1 = \sum_{i=0}^{\infty} c_i \sigma^{i/2} \ , \quad s_1 + 1 + \sigma^{1/2} = \sum_{i=0}^{\infty} c'_i \sigma^{i/2} \ ,$$

and $c'_i = c_i$ for $i \neq 1$, and $c'_1 = c_1 + 1$.

Recall, that if n is a natural number and i and a_0 are invertible, then

$$\left(\sum_{k=0}^{\infty} a_k X^k \right)^n = \sum_{k=0}^{\infty} b_k X^k \ ,$$

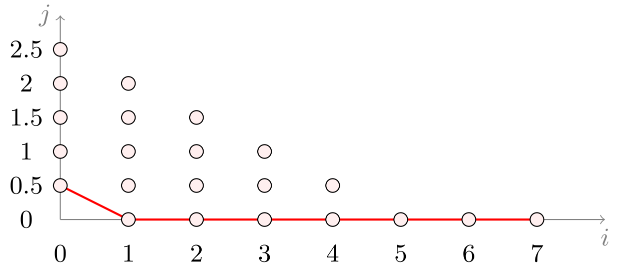

Fig. 5. Newton polygon for $f_m^{(1)}$, where $m = 6$

where $b_0 = a_0^n$, and

$$b_i = \frac{1}{ia_0} \cdot \sum_{k=1}^{i} (kn - i + k) a_k b_{i-k}, \quad i \geq 1 \ .$$

Since

$$(s_1 + 1)^{m+1} = \sum_{i=0}^{\infty} \delta_i \sigma^{i/2} \ ,$$

$$(s_1 + \sigma^{1/2} + 1)^{m-1} = \sum_{i=0}^{\infty} \delta_i' \sigma^{i/2} \ ,$$

where $\delta_0 = \delta_0' = 1$,

$$\delta_i = \frac{1}{i} \sum_{k=1}^{i} \left(k(m+1) - i + k\right) c_k \delta_{i-k} \ , \qquad (13)$$

$$\delta_i' = \frac{1}{i} \sum_{k=1}^{i} \left(k(m-1) - i + k\right) c_k' \delta_{i-k}' \ , \qquad (14)$$

for $i > 0$, we get

$$f_m^{(1)}(s_1, \sigma) = \sum_{i=0}^{\infty} \left(\delta_i - \delta_i'\right) \sigma^{i/2} \ .$$

Collecting terms with $\sigma^{i/2}$, $i \geq 1$ in $f_m^{(1)}(s_1, \sigma)$, we get equations for calculating the rest of c_{i+1}. More precisely, we first collect terms that correspond $i = 1$, and then derive a recursive formula for the case $i > 2$. Via this recursive formula, we compute c_i for small i, and then guess c_i.

Thus, consider terms containing $\sigma^{1/2}$: $\delta_1 = \delta_1'$. According to Eq. (13) and (14)

$$\delta_1 - \delta_1' = (m+1)c_1 - (m-1)(c_1 + 1) = 0 \ ,$$

and

$$c_1 = \frac{m-1}{2}, \qquad \delta_1 = (m+1)c_1 = \frac{(m-1)(m+1)}{2} \ .$$

We assume recursively that $\delta_j = \delta_j'$ for all $j < i$. Then, assuming $i \geq 2$, $0 = i(\delta_i - \delta_i') = \sum_{k=1}^{i}(k(m+1) - i + k)c_k\delta_{i-k} - \sum_{k=1}^{i}(k(m-1) - i + k)c_k'\delta_{i-k}' = ((m+1) - i + 1)c_1\delta_{i-1} + \sum_{k=2}^{i}\big(k(m+1) - i + k\big)c_k\delta_{i-k} - ((m-1) - i + 1)c_1'\delta_{i-1}' - \sum_{k=2}^{i}\big(k(m-1) - i + k\big)c_k'\delta_{i-k}' = (m+2-i)c_1\delta_{i-1} - (m-i)(c_1+1)\delta_{i-1} + 2\sum_{k=2}^{i} kc_k\delta_{i-k} = 2ic_i + 2\sum_{k=1}^{i-1} kc_k\delta_{i-k} - (m-i)\delta_{i-1}$. Hereof,

$$c_i = -\frac{1}{i}\left(\frac{(i-1)\delta_{i-1}}{2} + \sum_{k=2}^{i-1} kc_k\delta_{i-k}\right) \ . \qquad (15)$$

From Eq. (15) we can compute c_i, given that we have already computed c_j and δ_j (using Eq. (13)) for $j \leq i$. For example, for $i = 2$ we get

$$c_2 = -\frac{\delta_1}{4} = -\frac{(m-1)(m+1)}{2^2 \cdot 2!} \ ,$$

and

$$\begin{aligned} \delta_2 =& \frac{1}{2}\sum_{k=1}^{2}\big(k(m+1) - 2 + k\big)c_k\delta_{2-k} \\ =& \frac{(m-1)(m+1)(m^2 - 2m - 1)}{2^2 \cdot 2!} \ . \end{aligned}$$

Further, for $i = 3$,

$$\begin{aligned} c_3 =& \frac{1}{3} \cdot \left(\frac{(1-3)\delta_2}{2} - \sum_{k=2}^{2} kc_k\delta_{3-k}\right) \\ =& \frac{(m-1)2m(2m+2)}{2^3 \cdot 3!} \ , \end{aligned}$$

and

$$\begin{aligned} \delta_3 =& \frac{1}{3} \cdot \sum_{k=1}^{3}\big(k(m+1) - 3 + k\big)c_k\delta_{3-k} \\ =& \frac{(m-1)(m+1)(m^2 - 3m - 2)(m^2 - 3m)}{2^3 \cdot 3!} \ , \end{aligned}$$

and so on.

The process can be continued to calculate more terms. However, the already calculated terms give us a good guess about the nature of both c_i and δ_i.

Thus,

$$c_i = (-1)^{i+1}\frac{(m-1)}{2^i i!}\frac{((i-1)(m+1))!!}{((i-1)(m-1))!!} \ , \qquad (16)$$

Moreover,

$$\delta_i = \frac{(m-1)(m+1)}{2^i \cdot i!}\prod_{j=0}^{i-1}(m^2 - i(m+1) + 1 + 2j).$$

Verifying that these two equations satisfy the recursions Eq. (13) and Eq. (15) is rather tedious, and since it plays no importance in practice, we will omit it. □

C Proof of Thm. 4

Proof. Recall that the communication function of our leveled LHE scheme, $\mathsf{com}(\chi, w, m, s, k, \ell)$, is given by Eq. (6), where

$$s = \sum_{i=0}^{\infty} c_i \sigma^{\frac{i-1}{2}} \ ,$$

and c_i are defined as in Thm. 3. Substituting the expression for s (see Eq. (9)) into $\mathsf{com}(\chi, w, m, \vec{s}, k, \ell)$, we will get required communication.

For that, we first find

$$s^{-1} = \sum_{j=0}^{\infty} c'_j \sigma^{\frac{j+1}{2}}$$

from the condition $ss^{-1} = 1$, obtaining

$$c'_i = (-1)^i \frac{(m-1)}{2^i \cdot i!} \frac{((i+1)m + (i-3))!!}{((i+1)(m-1))!!} \ .$$

In particular, $c'_0 = 1$, $c'_1 = -(m-1)/2$,

$$c'_2 = \frac{(m-1)(3m-1)}{2^2 \cdot 2!} \ ,$$
$$c'_3 = -\frac{(m-1)2m(2m-1)}{2^2 \cdot 3!} \ ,$$

and so on. Then $1 + s^{-1} = \sum_{i=0}^{\infty} d_i \sigma^{i/2}$, where $d_0 = 1$, $d_k = c'_{k-1}$ for $k \geq 1$. Raising power series $1 + s^{-1}$ to the m-th power, we obtain

$$\left(\sum_{i=0}^{\infty} d_i \sigma^{i/2}\right)^m = \sum_{i=0}^{\infty} u_i \sigma^{i/2} \ ,$$

where $u_0 = 1$, $u_p = \frac{1}{p} \sum_{t=1}^{p} (tm - p + t) d_t u_{p-t}$. In particular, $u_1 = m$, $u_2 = 0$, and so on. Then

$$\begin{aligned}
&\mathsf{com}(\chi, w, m, \vec{s}, k, \ell) \\
&=(w-1)k\chi \left(1 + \sigma^{-1/2} + c_1 + \sum_{i=1}^{\infty} c_{i+1} \sigma^{\frac{i}{2}}\right) + \\
&\quad \ell \left(1 + \sum_{j=1}^{\infty} u_j \sigma^{\frac{j}{2}}\right) \\
&=\ell + 2\sqrt{(w-1)\chi mk\ell} + \frac{1}{2}\chi k(m+1)(w-1) + \\
&\quad \frac{1}{\sqrt{\ell}} \frac{((w-1)\chi k)^{3/2} c_2}{\sqrt{m}} + \frac{1}{\ell} \frac{((w-1)\chi k)^2 c_3}{m} + \\
&\quad \sum_{i=3}^{\infty} C_i \sigma^{\frac{i}{2}} \ ,
\end{aligned}$$

where C_t are defined as $C_t = (w-1)k\chi c_{t+1} + \ell u_t$, and $t \geq 3$. Thus,

$$\mathsf{com}(\chi, w, m, \vec{s}, k, \ell) = \ell + 2\sqrt{(w-1)\chi mk\ell} + \frac{1}{2}\chi k(m+1)(w-1) + O(\ell^{-1/2}) \ .$$

The rate is equal to

$$(\chi \log_2 w + \ell) \cdot \mathsf{com}^{-1}(\chi, w, m, \vec{s}, k, \ell) \ .$$

To find com^{-1}, we write

$$\mathsf{com}^{-1}(\chi, w, m, \vec{s}, k, \ell) = \sum_{i=0}^{\infty} a_i \ell^{\frac{2-i}{2}} \ ,$$

then from $\mathsf{com}(\chi, w, m, \vec{s}, k, \ell) \cdot \mathsf{com}^{-1}(\chi, w, m, \vec{s}, k, \ell) = 1$ we get $a_0 = \ldots = a_3 = 0$, $a_4 = 1$, $a_5 = -2\sqrt{(w-1)\chi mk}$, $a_6 = \frac{1}{2}(7m-1)(w-1)\chi mk$, and so on, thus

$$\begin{aligned}
&(\ell + \chi \log w) \cdot \mathsf{com}^{-1}(\chi, w, m, \vec{s}, k, \ell) \\
&=(\ell + \chi \log w) \cdot \\
&\quad \left(\frac{1}{\ell} - \frac{2\sqrt{(w-1)\chi mk}}{\ell\sqrt{\ell}} + \frac{\chi k(7m-1)(w-1)}{2\ell^2} + \right. \\
&\quad \left. O(\ell^{-5/2})\right) \\
&=1 - \frac{2\sqrt{(w-1)\chi mk}}{\sqrt{\ell}} + \\
&\quad \frac{\chi\,(k(7m-1)(w-1)/2 + \log_2 w)}{\ell} + O(\ell^{-\frac{3}{2}}) \ .
\end{aligned}$$

□