A preliminary version of this paper appears in *Proceedings of EUROCRYPT '99* (J. Stern, ed.), Lecture Notes in Computer Science, Springer, 1999.

Computationally Private Information Retrieval with Polylogarithmic Communication

Christian Cachin* Silvio Micali† Markus Stadler‡

February 11, 1999

Abstract

We present a single-database computationally private information retrieval scheme with polylogarithmic communication complexity. Our construction is based on a new, but reasonable intractability assumption, which we call the Φ-Hiding Assumption (ΦHA): essentially the difficulty of deciding whether a small prime divides $\phi(m)$, where m is a composite integer of unknown factorization.

Keywords: Integer factorization, Euler's function, Φ-hiding assumption, private information retrieval.

1 Introduction

Private information retrieval. The notion of *private information retrieval* (PIR for short) was introduced by Chor, Goldreich, Kushilevitz and Sudan [CGKS95] and has already received a lot of attention. The study of PIR is motivated by the growing concern about the user's privacy when querying a large commercial database. (The problem was independently studied by Cooper and Birman [CB95] to implement an anonymous messaging service for mobile users.)

Ideally, the PIR problem consists of devising a communication protocol involving just two parties, the database and the user, each having a secret input. The database's secret input is called the *data string*, an n-bit string $B = b_1 b_2 \cdots b_n$. The user's secret input is an integer i between 1 and n. The protocol should enable the user to learn b_i in a communication-efficient way and at the same time hide i from the database. (The trivial and inefficient solution is having the database send the entire string B to the user.)

Information-theoretic PIRs (with database replication). Perhaps surprisingly, the original paper [CGKS95] shows that the PIR problem is solvable efficiently in an information-theoretic setting if the database does not consist of a single player, but of multiple players, each holding the same data string B, who can communicate with the user but not with each other (a model

*IBM Zurich Research Laboratory, CH-8803 Rüschlikon, Switzerland, cachin@acm.org. Research done at Laboratory for Computer Science, MIT.

†Laboratory for Computer Science, MIT, Cambridge, MA 02139, USA.

‡Crypto AG, P.O. Box 460, CH-6301 Zug, Switzerland, markus.stadler@acm.org.

reminiscent of the multi-prover proof systems of [BGKW88]). By saying that this model offers an information-theoretic solution, we mean that an individual database player cannot learn i at all, no matter how much computation it may perform, as long as it does not collude with other database players.

Several solutions in this model are presented in the paper of Chor *et al.* For example, (1) there are two-database information-theoretic PIRs with $O(n^{1/3})$ communication complexity, and (2) there are $O(\log n)$-database information-theoretic PIRs with polylog(n) communication complexity. In subsequent work, Ambainis gives a construction for k-database information-theoretic PIRs with $O(n^{1/(2k-1)})$ communication complexity [Amb97].

COMPUTATIONAL PIRs (WITH DATABASE REPLICATION). Notice that the latter two information-theoretic PIRs achieve subpolynomial communication complexity, but require more than a constant number of database servers. Chor and Gilboa [CG97], however, show that it is possible to achieve subpolynomial communication complexity with minimal database replication if one requires only computational privacy of the user input—a theoretically weaker though practically sufficient notion. They give a two-database PIR scheme with communication complexity $O(n^{\varepsilon})$ for any $\varepsilon > 0$. Their system makes use of a security parameter k and guarantees that, as long as an individual database performs a polynomial (in k) amount of computation and does not collude with the other one, it learns nothing about the value i.

COMPUTATIONAL PIRs (WITHOUT DATABASE REPLICATION). Though possibly viable, the assumption that the database servers are separated and yet mirror the same database contents may not be too practical. Fortunately, and again surprisingly, Kushilevitz and Ostrovsky [KO97] show that replication is not needed. Under a well-known number-theoretic assumption, they prove the existence of a *single-database* computational PIR with subpolynomial communication. More precisely, under the quadratic residuosity assumption [GM84], they exhibit a CPIR protocol between a user and one database with communication complexity $O(n^{\varepsilon})$, for any $\varepsilon > 0$, where again n represents the length of the data string. (For brevity, we refer to such a single-database, computational PIR, as a CPIR.)

It should be noted that the CPIR of [KO97] has an additional communication complexity that is polynomial in the security parameter k, but this additional amount of communication is *de facto* absorbed in the mentioned $O(n^{\varepsilon})$ complexity, because for all practical purposes k can be chosen quite small.

This result has raised the question of whether it is possible to construct CPIRs with lower communication complexity.

MAIN RESULT. We provide a positive answer to the above question based on a new but plausible number-theoretic assumption: the Φ *Assumption*, or ΦA for short. The ΦA consists of two parts, the *Φ-Hiding Assumption (ΦHA)* and the *Φ-Sampling Assumption (ΦSA)*.

Informally, the ΦHA states that it is computationally intractable to decide whether a given small prime divides $\phi(m)$, where m is a composite integer of unknown factorization. (Recall that ϕ is Euler's totient function, and that computing $\phi(m)$ on input m is as hard as factoring m.) The ΦSA states that it is possible to efficiently find a random composite m such that a given prime p divides $\phi(m)$.

The ΦA is attractively simple and concrete. Finding crisp and plausible assumptions is an important task in the design and analysis of cryptographic protocols, and we believe that the ΦA will prove useful in other contexts and will attract further study. Based on it we prove the following

Main Theorem: Under the ΦA, there is a two-round CPIR whose communication complexity is polylogarithmic in n (and polynomial in the security parameter).

We note that our CPIR is "essentially optimal" in several ways:

Communication complexity. Disregarding the privacy of the user input altogether, in order for the user to obtain the ith bit of an n-bit data string, at least $\log n$ bits have to be communicated between the user and the database in any case.

Computational complexity. Our CPIR is also very efficient from a computational-complexity point of view. Namely, (1) the user runs in time polynomial in $k \log n$ and (2) the database runs in time proportional to n times a polynomial in k. Both properties are close to optimal in our context. The user computational complexity is close to optimal because, as already mentioned, in any scheme achieving sub-linear communication, the user must send at least $\log n$ bits of information, and thus perform at least $\log n$ steps of computation. The database computational complexity is close to optimal because the database must read each bit of its data string in any single-database PIR. (Otherwise, it would know that the user cannot possibly have received any of the unread bits and therefore gain some information about the user input i.)

Round complexity. The round complexity of our CPIR is essentially optimal because, as long as the user can choose his own input i at will in each execution, no single-round CPIR exists[1].

Privacy model. Our CPIR achieves computational privacy. Although information-theoretic privacy is stronger, our scheme is optimal among single-database PIRs since there are no single-database PIRs with information-theoretic privacy (other than sending the entire data string).

2 Preliminaries and Definitions

2.1 Notation

Integers. We denote by $\mathbb{N}$ the set of natural numbers. Unless otherwise specified, a natural number is presented in its binary expansion whenever given as an input to an algorithm. If $n \in \mathbb{N}$, by 1^n we denote the unary expansion of n, that is, the concatenation of n 1's. If $a, b \in \mathbb{N}$, we denote that a evenly divides b by writing $a|b$. Let $\mathbb{Z}_m$ be the ring of integers modulo m and $\mathbb{Z}_m^*$ its multiplicative group. The *Euler totient function* of an integer m, denoted by $\phi(m)$, is defined as the number of positive integers $\leq m$ that are relatively prime to m.

Strings. If σ and τ are binary strings, we denote σ's length by $|\sigma|$, σ's ith bit by σ_i, and the concatenation of σ and τ by $\sigma \circ \tau$.

[1]We do not rule out the possibility of single-round CPIRs in alternative models, for example, in a model where the user always learns the bit in position i in any execution in which the data string has at least i bits.

Computation models. By an *algorithm* we mean a (probabilistic) Turing machine. By saying that an algorithm is *efficient* we mean that, for at most but an exponentially small fraction of its random tapes, it runs in fixed polynomial time. By a *k-gate circuit* we mean a finite function computable by an acyclic circuitry with k Boolean gates, where each gate is either a NOT-gate (with one input and one output) or an AND gate (with two binary inputs and one binary output).

Probability spaces. (Taken from [BDMP91] and [GMR88].) If $A(\cdot)$ is an algorithm, then for any input x, the notation "$A(x)$" refers to the probability space that assigns to the string σ the probability that A, on input x, outputs σ.

If S is a probability space, then "$x \stackrel{R}{\leftarrow} S$" denotes the algorithm which assigns to x an element randomly selected according to S. If F is a finite set, then the notation "$x \stackrel{R}{\leftarrow} F$" denotes the algorithm which assigns to x an element selected according to the probability space whose sample space is F and uniform probability distribution on the sample points.

If $p(\cdot, \cdot, \cdots)$ is a predicate, the notation

$$PROB[x \stackrel{R}{\leftarrow} S; y \stackrel{R}{\leftarrow} T; \cdots : p(x, y, \cdots)]$$

denotes the probability that $p(x, y, \cdots)$ will be true after the ordered execution of the algorithms $x \stackrel{R}{\leftarrow} S$, $y \stackrel{R}{\leftarrow} T, \cdots$.

2.2 Fully Polylogarithmic CPIR

Our proposed CPIR works in only two rounds and achieves both polylogarithmic communication complexity and polylogarithmic user computational complexity. For the sake of simplicity, we formalize only such types of CPIRs below.

Definition: Let $D(\cdot,\cdot,\cdot)$, $Q(\cdot,\cdot,\cdot)$ and $R(\cdot,\cdot,\cdot,\cdot,\cdot)$ be efficient algorithms. We say that (D, Q, R) is a *fully polylogarithmic computationally private information retrieval scheme* (or polylog CPIR for short) if there exist constants $a, b, c, d > 0$ such that,

1. (Correctness) $\forall n$, $\forall$ n-bit strings B, $\forall i \in [1, n]$, and $\forall k$,

$$PROB[(q, s) \stackrel{R}{\leftarrow} Q(n, i, 1^k) \; ; \; r \stackrel{R}{\leftarrow} D(B, q, 1^k) \; : \; R(n, i, (q, s), r, 1^k) = B_i] \; > \; 1 - 2^{-ak}$$

2. (Privacy) $\forall n$, $\forall i, j \in [1, n]$, $\forall k$ such that $2^k > n^b$, and $\forall$ 2^{ck}-gate circuits A,

$$\big| PROB[(q, s) \stackrel{R}{\leftarrow} Q(n, i, 1^k) \; : \; A(n, q, 1^k) = 1] \; - \; PROB[(q, s) \stackrel{R}{\leftarrow} Q(n, j, 1^k) \; : \; A(n, q, 1^k) = 1] \big| \; < \; 2^{-dk}.$$

We call a, b, c, and d the *fundamental constants* (of the CPIR); B the *data string*; D the *database algorithm*; the pair (Q, R) the *user algorithm*; Q the *query generator*; R the *response retriever*; q the *query*; s the *secret* (associated to q); r the *response*; and k the *security parameter*. (Intuitively, query q contains user input i, and response r contains database bit b_i, but both contents are unintelligible without secret s.)

Remarks.

1. Our correctness constraint slightly generalizes the one of [KO97]: Whereas there correctness is required to hold with probability 1, we require it to hold with very high probability.

2. As mentioned above, the communication complexity of our CPIR is polylogarithmic in n (the length of the data string) times a polynomial in k (the security parameter). Because k is an independent parameter, it is of course possible to choose it so large that the polynomial dependence on k dominates over the polylogarithmic dependence on n. But choosing k is an overkill since our definition guarantees "an exponential amount of privacy" also when k is only polylogarithmic in n.

2.3 Number Theory

Some useful sets. Let us define the sets we need in our assumptions and constructions.

Definition: We denote by $PRIMES_a$ the set of the primes of length a, and by H_a the set of the composite integers that are product of two primes of length a. (For a large, H_a contains the hardest inputs to any known factoring algorithm.)

We say that *a composite integer m ϕ-hides a prime p* if $p|\phi(m)$. Denote by $H^b(m)$ the set of b-bit primes p that are ϕ-hidden by m, denote by $\bar{H}^b(m)$ the set $PRIMES_b - H^b(m)$, and denote by H_a^b the set of those $m \in H_a$ (i.e., products of two a-bit primes) that ϕ-hide a b-bit prime.

Some useful facts. Let us state without proof some basic or well-known number-theoretic facts used in constructing our CPIR.

Fact 1: There exists an efficient algorithm that on input a outputs a random prime in PRIMES_a.

Fact 2: There exists an efficient algorithm that on input a outputs a random element of H_a.

Fact 3: There exists an efficient algorithm that, on input a b-bit prime p and an integer m together with its integer factorization, outputs whether or not $p \in H^b(m)$.

Fact 4: There exists an efficient algorithm that, on inputs x, p, m, and m's integer factorization, outputs whether or not x has a pth root mod m.

Our assumptions.

The Φ-Assumption (ΦA):

$\exists e, f, g, h > 0$ such that

- **Φ-Hiding Assumption (ΦHA):** $\forall k > h$ and $\forall$ 2^{ek}-gate circuits C,

$$PROB[m \stackrel{R}{\leftarrow} H_{kf}^k \; ; \; p_0 \stackrel{R}{\leftarrow} H^k(m) \; ; \; p_1 \stackrel{R}{\leftarrow} \bar{H}^k(m) \; ; \; b \stackrel{R}{\leftarrow} \{0,1\} \; : \; C(m, p_b) = b] \; < \; \frac{1}{2} + 2^{-gk}.$$

- **Φ-Sampling Assumption (ΦSA):** $\forall k > h$, there exists a sampling algorithm $S(\cdot)$ such that for all k-bit primes p, $S(p)$ outputs a random k^f-bit number $m \in H^k_{k^f}$ that ϕ-hides p, together with m's integer factorization.

We refer to e, f, g, and h as the *first, second, third*, and *fourth fundamental constant* of the ΦA, respectively.

REMARKS.

1. Revealing a large prime dividing $\phi(m)$ may compromise m's factorization. Namely, if p is a prime $> m^{1/4}$ and $p|\phi(m)$, then one can efficiently factor m on inputs m and p [Cop98, Cop96b, Cop96a]. Consequently, it is easy to decide whether p divides $\phi(m)$ whenever $p > m^{1/4}$. But nothing similar is known when p is much smaller, and for the ΦHA, it suffices that deciding whether p divides $\phi(m)$ is hard when p is not just a constant fraction shorter than m, but polynomially shorter.

   We further note that if the complexity of factoring is $\Omega(2^{\log m^c})$ for some constant c between 0 and 1, then revealing a prime p dividing $\phi(m)$ cannot possibly compromise m's factorization significantly if $\log p$ is significantly smaller than $(\log m)^c$. Indeed, since p can be represented using at most $\log p$ bits, revealing p cannot contribute more than a speed-up of $2^{\lceil \log p \rceil} \approx p$ for factoring m.

   Note that the ΦHA does not hold for $p = 3$. If $m = Q_1 Q_2$ and $m \equiv 2 \pmod 3$, then one can tell that one of Q_1 and Q_2 is congruent to 1 mod 3 and the other is 2 mod 3. In this case, it's obvious that 3 divides $\phi(m) = (Q_1 - 1)(Q_2 - 1)$.

2. The ΦSA is weaker than the well-known and widely accepted Extended Riemann Hypothesis (ERH). Consider the following algorithm $S(\cdot)$:

   Inputs: a k-bit prime p.

   Output: a k^f-bit integer $m \in H^k_{k^f}$ that ϕ-hides p and its integer factorization.

   Code for $S(p)$:

   (a) Repeatedly choose a random $(k^f - k)$-bit integer q_1 until $Q_1 = pq_1 + 1$ is a prime.
   (b) Choose a random k^f-bit prime Q_2.
   (c) Let $m \leftarrow Q_1 \cdot Q_2$ and return m and (Q_1, Q_2).

   Under the ERH, algorithm S finds a suitable m in expected polynomial time in k^f (see Exercise 30 in Chapter 8 of [BS96]).

3 Our CPIR

3.1 The High-Level Design

At a very high level, the user's query consists of a compact program that contains the user input i in a hidden way. The database runs this program on its data string, and the result of this computation is its response r.

A bit more specifically, this compact program is actually run on the data string in a bit-by-bit fashion. Letting B be the data string, the user sends the database an algorithm A and a k-bit value x_0 (where k is the security parameter), and the database computes a sequence of k-bit values: $x_1 = A(x_0, B_1)$, $x_2 = A(x_1, B_2)$, $\dots$, $x_n = A(x_{n-1}, B_n)$. The last value x_n is the response r. The user retrieves B_i by evaluating on x_n a predicate R_i, which is hard to guess without the secret key of the user.

This high-level design works essentially because the predicate R_i further enjoys the following properties relative to the sequence of values $x_0, \dots, x_n$:

1. $R_i(x_0) = 0$;
2. $\forall j = 1, \dots, i-1$, $R_i(x_j) = 0$;
3. $R_i(x_i) = 1$ if and only if $B_i = 1$; and
4. $\forall j \geq i$, $R_i(x_{j+1}) = 1$ if and only if $R_i(x_j) = 1$.

It follows by induction that $R_i(x_n) = 1$ if and only if $B_i = 1$.

3.2 The Implementation

To specify our polylog CPIR we must give a database algorithm D and user algorithms Q (query generator) and R (response retriever). These algorithms use two common efficient subroutines T and P that we describe first. Algorithm T could be any probabilistic primality test [SS77, Rab81], but we let it be a primality *prover* [GK86, AH87] so as to gain some advantage in the notation and presentation (at the expense of running time).

BASIC INPUTS.

A number $n \in \mathbb{N}$; an n-bit sequence B; an integer $i \in [1, n]$; and a unary security parameter 1^k such that $k > (\log n)^2$.

PRIMALITY PROVER $T(\cdot)$.

Input: an integer z (in binary).

Output: 1 if z is prime, and 0 if z is composite.

Code for $T(z)$: See [AH87].

PRIME(-SEQUENCE) GENERATOR $P(\cdot, \cdot, \cdot)$.

Inputs: an integer $a \in [1, n]$; a sequence of k^3 k-bit strings $Y = (y_0, \dots, y_{k^3-1})$; and 1^k.

Output: a k-bit integer p_a (a prime with overwhelming probability).
Because P is deterministic, for Y and k fixed, it generates a sequence of (probable) primes $p_1, \dots, p_n$ with $a = 1, \dots, n$.

Code for $P(a, Y, 1^k)$:

1. $j \leftarrow 0$.
2. $\sigma_{aj} \leftarrow \bar{a} \circ \bar{j}$, where $\bar{a}$ is the $(\log n)$-bit representation of a and $\bar{j}$ the $(k - \log n)$-bit representation of j.
3. $z_j \leftarrow \sum_{l=0}^{k^3-1} y_l {\sigma_{aj}}^l$, where all strings y_l and σ_{aj} are interpreted as elements of $GF(2^k)$ and the operations are in $GF(2^k)$.
4. If $T(z_j) = 1$ or $j = 2^{k-\log n}$, then return $p_a \leftarrow z_j$ and halt; else, $j \leftarrow j+1$ and go to step 2.

Query generator $Q(\cdot,\cdot,\cdot)$.

Inputs: n; an integer $i \in [1,n]$; and 1^k.

Outputs: a query $q = (m, x, Y)$ and a secret s, where m is a k^f-bit composite (f being the second constant of the ΦA), $x \in \mathbb{Z}_m^*$, Y a k^3-long sequence of k-bit strings, and where s consists of m's prime factorization.

Code for $Q(n, i, 1^k)$:

1. Randomly and independently choose $y_0, \ldots, y_{k^3-1} \in \{0,1\}^k$ and let $Y = (y_0, \ldots, y_{k^3-1})$.
2. $p_i \leftarrow P(i, Y, 1^k)$.
3. Choose a random k^f-bit integer m that ϕ-hides $p_i = P(i, Y, 1^k)$ and let s be its integer factorization.
4. Choose a random $x \in \mathbb{Z}_m^*$.
5. Output the query $q = (m, x, Y)$ and the secret s.

Database algorithm $D(\cdot,\cdot,\cdot)$.

Inputs: B; $q = (m, x, Y)$, a query output by $Q(n, i, 1^k)$; and 1^k.

Output: $r \in \mathbb{Z}_m^*$.

Code for $D(B, q, 1^k)$:

1. $x_0 \leftarrow x$.
2. For $j = 1, \ldots, n$, compute:
   (a) $p_j \leftarrow P(j, Y, 1^k)$.
   (b) $e_j \leftarrow p_j^{B_j}$.
   (c) $x_j \leftarrow x_{j-1}^{e_j} \bmod m$.
3. Output the response $r = x_n$.

Response retriever $R(\cdot,\cdot,\cdot,\cdot,\cdot)$:

Inputs: n; i; $(m, x, Y), s)$, an output of $Q(n, i, 1^k)$; $r \in \mathbb{Z}_m^*$, an output of $D(B, (m, x, Y), 1^k)$; and 1^k.

Output: a bit b. (With overwhelming probability, $b = B_i$.)

Code for $R(n, i, (q, s), r, 1^k)$*:* If r has p_ith roots mod m, then output 1, else output 0.

Theorem: Under the ΦA, (D, Q, R) is a polylog CPIR.

3.3 Proof of the Theorem

Running time (sketch). Subroutine P is efficient because (on inputs i, Y, and 1^k) its most intensive operation consists, for at most k^3 times, of evaluating once a k-degree polynomial over $GF(2^k)$ and running the primality prover T. Algorithm Q is efficient because subroutines P and T are efficient, because p_i is a k-bit prime with overwhelming probability, and because, under the ΦSA, selecting a random $2k^f$-bit composite $\in H^k_{kf}$ ϕ-hiding p_i is efficient. (Notice that, because n and i are presented in binary, Q actually runs in time polylogarithmic in n.) Algorithm D is efficient because it performs essentially one exponentiation mod m for each bit of the data string (and thus runs in time polynomial in k and linear in n). Algorithm R is efficient because of Fact 4 and because it has m's factorization (the secret s) available as an input. (R actually runs in time polynomial in k because m's length is polynomial in k.)

Correctness (sketch). Let us start with a quick and dirty analysis of the prime-sequence generator P. Because the elements of Y are randomly and independently selected, in every execution of $P(a, Y, 1^k)$, the values $z_0, \ldots, z_{2^{k-\log n}}$ are k^3-wise independent. Thus with probability lower bounded by $1 - 2^{O(-k^2)}$, at least one of them is prime, and thus p_a is prime. Because the length n of the data string satisfies $n^2 < 2^k$, with probability exponentially (in k) close to 1, *all* possible outputs $p_1, \ldots, p_n$ are primes. Actually, with probability exponentially (in k) close to 1, $p_1, \ldots, p_n$ consists of *random* and *distinct* primes of length k. Observe that the k^f-bit modulus m can ϕ-hide at most a constant number of primes from a set of randomly chosen k-bit primes except with exponentially (in k) small probability. Thus, with probability still exponentially (in k) close to 1, p_i will be the only prime in our sequence to divide $\phi(m)$.

In sum, because it suffices for correctness to hold with exponentially (in k) high probability, we might as well assume that, in every execution of $Q(n, i, 1^k)$, $p_1, \ldots, p_n$ are indeed random, distinct primes of length k, such that only p_i divides $\phi(m)$. Let R_i be the following predicate on $\mathbb{Z}^*_m$:

$$R_i(x) = \begin{cases} 1 & \text{if } x \text{ has a } p_i\text{th root mod } m \\ 0 & \text{otherwise.} \end{cases}$$

The user retrieves b_i by evaluating $R_i(x_n)$. It is easy to check that properties 1–4 of our high-level design hold as promised:

1. $R_i(x_0) = 0$.

   This property follows from the fact that the function $x \to x^{p_j} \bmod m$ on $\mathbb{Z}^*_m$ is 1-to-1 if p_j is relatively prime to $\phi(m)$, and at least p_j-to-1 otherwise. Because p_i is in $\Theta(2^k)$ except with exponentially (in k) small probability, the probability that a random element of $\mathbb{Z}^*_m$ has a

p_ith root mod m is also exponentially small (in k). Thus we might as well assume that x_0 has no p_ith roots mod m (remember that correctness should hold only most of the time).[2]

2. $\forall j = 1, \ldots, i-1$, $R_i(x_j) = 0$.

   This follows because x_0 has no p_ith roots mod m and because if x has no p_ith roots mod m, for all primes p not dividing $\phi(m)$ also x^p has no p_ith roots mod m. Again because of the size of the primes p_j for $j \neq i$, one can show that except with exponentially small (in k) probability, none of the p_j divides $\phi(m)$.

3. $R_i(x_i) = 1$ if and only if $B_i = 1$.

   If $B_i = 0$, then $x_i = x_{i-1}$. Thus, by property 2 above, x_i has no p_ith roots mod m. If $B_i = 1$, then $x_i = x_{i-1}^{p_i}$ mod m. Thus, x_i has p_ith roots mod m by construction.

4. $\forall j \geq i$, $R_i(x_{j+1}) = 1$ if and only if $R_i(x_j) = 1$.

   The "if part" follows from the fact that if x_j has p_ith roots mod m, then there exists a y such that $x_j = y^{p_i}$ mod m and therefore also $x_{j+1} = x_j^{p_j} = y^{p_i p_j} = (y^{p_j})^{p_i}$ mod m has p_ith roots. For the "only-if part," see the proof of property 2 above.

PRIVACY (SKETCH). Suppose for contradiction that the privacy condition does not hold for (D, Q, R). Then for all $b, c, d > 0$, there exist n, indices i and j ($i \neq j$), $k > \log n^b$, and a 2^{bk}-gate circuit $\widetilde{A}$ (with binary output) such that

$$|\alpha_1 - \alpha_2| \;\geq\; \varepsilon$$

for some $\varepsilon > 2^{-dk}$, where

$$\begin{aligned}
\alpha_1 &= PROB\big[((m,x,Y),s) \overset{R}{\leftarrow} Q(n,i,1^k) \,:\, \widetilde{A}(n,(m,x,Y),1^k) = 1\big], \\
\alpha_2 &= PROB\big[((m,x,Y),s) \overset{R}{\leftarrow} Q(n,j,1^k) \,:\, \widetilde{A}(n,(m,x,Y),1^k) = 1\big].
\end{aligned}$$

(Intuitively, $\widetilde{A}$'s advantage ε is always bigger than any exponentially small in k quantity.) Define now the following probability:

$$\beta \;=\; PROB\big[m \overset{R}{\leftarrow} H^k_{kf} \,;\, x \overset{R}{\leftarrow} \mathbb{Z}^*_m \,;\, Y \overset{R}{\leftarrow} GF(2^k)^{k^3} \,:\, \widetilde{A}(n,(m,x,Y),1^k) = 1\big].$$

(Notice that, in the sequence of experiments defining β, Y still defines a prime p_i and a prime p_j with overwhelming probability, but there is no guarantee that m ϕ-hides either of them.) It follows either $|\alpha_1 - \beta| \geq \varepsilon/2$ or $|\alpha_2 - \beta| \geq \varepsilon/2$. W.l.o.g. assume $|\alpha_1 - \beta| \geq \varepsilon/2$ and also $\alpha_1 - \beta \geq \varepsilon/2$.

We can construct a guessing circuit $\widetilde{C} = \widetilde{C}_{n,i}$ to contradict the ΦHA as follows.

GUESSING CIRCUIT $\widetilde{C}_{n,i}(\cdot,\cdot)$.

Inputs: a number $m \in H^k_{kf}$; and a k-bit prime p.

[2] We choose x_0 at random rather than ensuring that it has no p_ith roots mod m to facilitate proving the privacy constraint.

Output: a bit b (indicating whether m ϕ-hides p).

Code for $\widetilde{C}_{n,i}(m,p)$:

1. Choose k^3 uniformly random k-bit numbers $a_1, \ldots, a_{k^3}$.
2. Run primality prover T on a_j for $j = 1, \ldots, k^3$ and let j' be the smallest j for which $T(a_j) = 1$. If T returns 0 for all a_j, then $j' \leftarrow k^3$.
3. Use Lagrange interpolation to find the coefficients $y_0, \ldots, y_{k^3-1}$ of a polynomial $\xi(\sigma)$ over $GF(2^k)$ with degree $k^3 - 1$ such that $\xi(\sigma_{ij}) = a_j$ for $j = 1, \ldots, j'-1, j'+1, \ldots, k^3$ and $\xi(\sigma_{ij'}) = p$, where $\sigma_{ij} \in GF(2^k)$ corresponds to the k-bit string $i \circ j$ as in the prime-sequence generator P. Let $Y = (y_0, \ldots, y_{k^3-1})$.
4. Choose x at random from $\mathbb{Z}_m^*$ and run $\widetilde{A}(n, (m, x, Y), 1^k)$. If $\widetilde{A}$ returns 0, then return 1, otherwise (if $\widetilde{A}$ returns 1), then return 0.

Notice that $\widetilde{C}$ can be constructed with a number of gates that is at most polynomially (in k) greater than the number of gates of $\widetilde{A}$.

Above we have defined how $\widetilde{C}$ operates for any $m \in H_{k^f}^k$ and any $p \in PRIMES_k$. Let us now analyze $\widetilde{C}$'s behavior on the input distribution required by the ΦHA (i.e., when $m \stackrel{R}{\leftarrow} H_{k^f}^k$ and $p \stackrel{R}{\leftarrow} H^k(m)$ with probability 1/2 and $p \stackrel{R}{\leftarrow} \bar{H}^k(m)$ with probability 1/2) and calculate the probability that $\widetilde{C}$ guesses correctly from which distribution p is drawn.

$$\begin{aligned} PROB[\widetilde{C} \text{ correct}] &= \frac{1}{2} \cdot PROB[\widetilde{C} \text{ correct} | p \stackrel{R}{\leftarrow} H^k(m)] \\ &\quad + \frac{1}{2} \cdot PROB[\widetilde{C} \text{ correct} | p \stackrel{R}{\leftarrow} \bar{H}^k(m)] \\ &= \frac{1}{2} \cdot PROB[\widetilde{C} = 0 | p \stackrel{R}{\leftarrow} H^k(m)] \\ &\quad + \frac{1}{2} \cdot PROB[\widetilde{C} = 1 | p \stackrel{R}{\leftarrow} \bar{H}^k(m)]. \end{aligned}$$

The distribution of the output of $\widetilde{C}$ depends directly on $\widetilde{A}$. If $p \stackrel{R}{\leftarrow} H^k(m)$, then, by construction, $\widetilde{A}$ is run with the same input distribution as in the definition of α_1, except for the case that $\widetilde{C}$ finds no prime among $a_1, \ldots, a_{k^3}$ in step 2 (assume this is not the case for the moment). Let us examine $\widetilde{A}$'s input distribution in $\widetilde{C}$ when $p \stackrel{R}{\leftarrow} \bar{H}^k(m)$ and compare it to $\widetilde{A}$'s input distribution in the definition of β. The experiment leading to β contains three distinct cases for $p_i = P(i, Y, 1^k)$:

1. p_i is composite;
2. $p_i \in H^k(m)$; or
3. $p_i \in \bar{H}^k(m)$.

Note that case 3 is actually how $\widetilde{A}$ is called by our $\widetilde{C}$ in the ΦHA and occurs with overwhelming probability. Let δ_0 be the probability of case 1, which will be computed below, and assume for the moment that p_i is indeed a random k-bit prime. The probability δ_1 that a random element of $PRIMES_k$ is in $H^k(m)$ is upper bounded by $k^f 2^{-k} = O(2^{-k/2})$. (This is the conditional probability of case 2 above given that p_i is prime.) For $\widetilde{C}$, this implies

$$PROB[\widetilde{C} = 1 | p \stackrel{R}{\leftarrow} PRIMES_k] \;\leq\; PROB[\widetilde{C} = 1 | p \stackrel{R}{\leftarrow} \bar{H}^k(m)] + \delta_1.$$

Now consider the case that no prime is detected among $a_1, \ldots, a_{k^3}$ in step 2. Because T is an ideal primality prover, this probability is at most about $(1 - \frac{1}{k})^{k^3}$ and therefore $\delta_0 = O(2^{-k/2})$.

We can now bound $PROB[\widetilde{C}$ correct$]$ as

$$\begin{aligned} PROB[\widetilde{C} \text{ correct}] \;\geq\;& \frac{1}{2} \cdot (1-\delta_0) \cdot PROB[\widetilde{C} = 0 | p \stackrel{R}{\leftarrow} H^k(m)] \\ &+ \frac{1}{2} \cdot (1-\delta_0) \cdot \left(PROB[\widetilde{C} = 1 | p \stackrel{R}{\leftarrow} PRIMES_k] - \delta_1 \right) \\ \geq\;& \frac{1}{2} \cdot (1-\delta_0) \cdot \alpha_1 + \frac{1}{2} \cdot (1-\delta_0) \cdot (1 - \beta - \delta_1) \\ \geq\;& \frac{1}{2} \cdot \left(1 + \alpha_1 - \delta_0 - \beta - \delta_0 - \delta_1 \right) \\ \geq\;& \frac{1}{2} + \frac{\varepsilon}{4} - \delta_0 - \frac{\delta_1}{2}. \end{aligned}$$

The last inequality follows from the assumption $\alpha_1 - \beta \geq \varepsilon/2$.

To conclude, $\widetilde{C}$ distinguishes correctly with probability at least

$$\frac{1}{2} + \frac{\varepsilon}{4} - \delta_0 - \frac{\delta_1}{2}.$$

Intuitively, since δ_1 and δ_0 are exponentially small in k, but ε exceeds any exponentially small quantity, there remains an advantage for $\widetilde{C}$ that is not exponentially small and it is clear that $\widetilde{C}$ violates the ΦHA. ■

Acknowledgments

The authors wish to thank Wenbo Mao for interesting comments on an earlier version of the paper.

References

[AH87] Leonard M. Adleman and Ming-Deh A. Huang, *Recognizing primes in random polynomial time*, Proc. 19th Annual ACM Symposium on Theory of Computing (STOC), 1987, pp. 462–469.

[Amb97] Andris Ambainis, *Upper bound on the communication complexity of private information retrieval*, Proc. 24th ICALP, Lecture Notes in Computer Science, vol. 1256, Springer, 1997.

[BDMP91] Manuel Blum, Alfredo De Santis, Silvio Micali, and Giuseppe Persiano, *Noninteractive zero-knowledge*, SIAM Journal on Computing **20** (1991), no. 6, 1085–1118.

[BGKW88] Michael Ben-Or, Shafi Goldwasser, Joe Kilian, and Avi Wigderson, *Multi prover interactive proofs: How to remove intractability*, Proc. 20th Annual ACM Symposium on Theory of Computing (STOC), 1988, pp. 113–131.

[BS96] Eric Bach and Jeffrey Shallit, *Algorithmic number theory*, vol. 1: Efficient Algorithms, MIT Press, Cambridge, 1996.

[CB95] David A. Cooper and Kenneth P. Birman, *Preserving privacy in a network of mobile computers*, Proc. IEEE Symposium on Security and Privacy, 1995, pp. 26–38.

[CG97] Benny Chor and Niv Gilboa, *Computationally private information retrieval*, Proc. 29th Annual ACM Symposium on Theory of Computing (STOC), 1997, pp. 304–313.

[CGKS95] Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan, *Private information retrieval*, Proc. 36th IEEE Symposium on Foundations of Computer Science (FOCS), 1995.

[Cop96a] Don Coppersmith, *Finding a small root of a bivariate integer equation; factoring with high bits known*, Advances in Cryptology: EUROCRYPT '96 (Ueli Maurer, ed.), Lecture Notes in Computer Science, vol. 1233, Springer, 1996.

[Cop96b] Don Coppersmith, *Finding a small root of a univariate modular equation*, Advances in Cryptology: EUROCRYPT '96 (Ueli Maurer, ed.), Lecture Notes in Computer Science, vol. 1233, Springer, 1996.

[Cop98] Don Coppersmith, personal communication, 1998.

[GK86] Shafi Goldwasser and Joe Kilian, *Almost all primes can be quickly certified*, Proc. 18th Annual ACM Symposium on Theory of Computing (STOC), 1986, pp. 316–329.

[GM84] Shafi Goldwasser and Silvio Micali, *Probabilistic encryption*, Journal of Computer and System Sciences **28** (1984), 270–299.

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest, *A digital signature scheme secure against adaptive chosen-message attacks*, SIAM Journal on Computing **17** (1988), no. 2, 281–308.

[KO97] Eyal Kushilevitz and Rafail Ostrovsky, *Replication is not needed: Single database, computationally-private information retrieval*, Proc. 38th IEEE Symposium on Foundations of Computer Science (FOCS), 1997, pp. 364–373.

[Rab81] Michael O. Rabin, *How to exchange secrets by oblivious transfer*, Tech. Report TR-81, Harvard, 1981.

[SS77] R. Solovay and V. Strassen, *A fast monte-carlo test for primality*, SIAM Journal on Computing **6** (1977), no. 1, 84–85.