Single-Database Private Information Retrieval with Constant Communication Rate

Craig Gentry and Zulfikar Ramzan

DoCoMo Communications Laboratories USA, Inc.
{cgentry, ramzan}@docomolabs-usa.com

Abstract. We present a single-database private information retrieval (PIR) scheme with communication complexity $\mathcal{O}(k+d)$, where $k \geq \log n$ is a security parameter that depends on the database size n and d is the bit-length of the retrieved database block. This communication complexity is better asymptotically than previous single-database PIR schemes. The scheme also gives improved performance for practical parameter settings whether the user is retrieving a single bit or very large blocks. For large blocks, our scheme achieves a constant "rate" (e.g., 0.2), even when the user-side communication is very low (e.g., two 1024-bit numbers). Our scheme and security analysis is presented using general groups with hidden smooth subgroups; the scheme can be instantiated using composite moduli, in which case the security of our scheme is based on a simple variant of the "Φ-hiding" assumption by Cachin, Micali and Stadler [2].

1 Introduction

Problem Statement and Background. Private Information Retrieval (PIR) schemes allow a user to retrieve the i^{th} bit of an n-bit database, without revealing to the database the value of i. The "trivial" solution is for the user to retrieve the entire database, but this approach may incur enormous communication costs. A good PIR scheme, on the other hand, should have considerably lower (certainly sub-linear) communication complexity. Private Block Retrieval (PBR) is a natural and more practical extension of PIR in which, instead of retrieving only a single bit, the user retrieves a d-bit block that begins at index i.

PIR and PBR have been studied extensively; here, we only mention the work most relevant to us. The notion of PIR was introduced by Chor et al. [5], who focused on the *information-theoretic* case, where one requires that the user's query give absolutely no information about i. They proved that if only a single database is used, then n bits must be communicated. On the other hand, if the database is replicated in k servers, and if the user is allowed to give a separate query to each server, one can construct a PIR scheme with k user queries each being $\mathcal{O}(n^{1/k})$-bits and k single-bit server responses.[1] However, to ensure user privacy in the multi-server setting, the servers must be trusted not to collude.

[1] Currently, the lowest asymptotic total communication complexity for information-theoretic PIR is $\mathcal{O}(n^{\log\log k/k\log k})$ [1].

L. Caires et al. (Eds.): ICALP 2005, LNCS 3580, pp. 803–815, 2005.

Chor et al. also introduced PBR. They showed that any PIR scheme with $\alpha_k(n)$-bit queries and $\beta_k(n)$-bit responses can be converted into a PBR scheme for d-bit blocks with $\alpha_k(n)$-bit queries and $d\beta_k(n)$-bit responses. This means that, for a constant $k \geq 2$ of servers, the above information-theoretic PIR scheme can be converted into a PBR scheme with an asymptotically constant "rate" of $1/k$ – i.e., the ratio of bits retrieved (i.e., d) versus total communication complexity (i.e., $kd + \mathcal{O}(n^{1/k})$) tends towards $1/k$ as n and d increase appropriately. Increasing the rate to 1 in the information-theoretic setting seems difficult.

Chor and Gilboa studied the problem of whether one could achieve better communication complexity for multi-server PIR by using computational assumptions [4]. Subsequently, Kushilevitz and Ostrovsky showed that one can achieve *single database* PIR under the Quadratic Residuosity assumption with communication $2^{\mathcal{O}\sqrt{\log n \log l_m}}$, where l_m is the bit length of a composite modulus m. Like all current single-database PIR schemes, the server needs $\Omega(n)$ computation to generate a query response. Since the number field sieve [10] can factor an l_m-bit composite number in time $2^{\mathcal{O}(1)l_m^{1/3}(\log l_m)^{2/3}}$ (and hence solve quadratic residuosity), and since it seems reasonable that the server should need at least as much computation to break user privacy as to generate a response, one should set $l_m = \Omega(\log^{3-o(1)} n)$ to ensure adequate security.

Cachin, Micali, and Stadler [2] constructed the first single-database PIR scheme with *poly-logarithmic* communication complexity (about $\mathcal{O}(\log^8 n)$ for their suggested parameters), addressing an open problem left by Kushilevitz and Ostrovsky. The security of their scheme (CMS) is based on the "Φ-hiding" assumption – roughly, that is hard to distinguish which of two primes divides $\phi(m)$ for composite modulus m. Essentially, the scheme works as follows. Each index $j \in [1, n]$ is mapped to a distinct prime p_j. To recover bit b_i from database $B = b_1 \cdots b_n$, the user sends a composite (hard-to-factor) modulus m such that p_i divides $\phi(m)$ and a generator $x \in \mathbb{Z}_m^*$ with order divisible by p_i. The server sends back $r = x^P (\mathrm{mod} m)$ for $P = \prod_j p_j^{b_j}$. The user concludes that $b_i = 1$ if r is a p-residue modulo m; otherwise, $b_i = 0$. The communication complexity of (this simplified version of) CMS is $3l_m$ to recover 1 database bit. Again, $l_m = \Omega(\log^{3-o(1)} n)$ for adequate security, though [2] recommends an even larger value of l_m ($\mathcal{O}(\log^8 n)$).

Recently, Lipmaa [11] gave a PBR scheme with stated $\Theta(l_m \cdot \log^2 n + d \cdot \log n)$ communication complexity for d-bit blocks, where again $l_m = \Omega(\log^{3-o(1)} n)$. Thus, Lipmaa's scheme has a better "rate" – namely $1/(\log n)$ – than CMS for large blocks. In fact, as we describe in the full version of this paper, one can apply Chor et al.'s [5] abovementioned conversion from PIR to PBR to Lipmaa's scheme to get a PBR scheme with rate arbitrarily close to 1. However, for Lipmaa's scheme to achieve a good rate in practice, n and d must be quite large (on the order of gigabits and megabits, respectively) before they begin to offset the large one-time cost represented by the $l_m \cdot \log^2 n$ term.

Our Results. We present a single-database PBR scheme that has, to the best of our knowledge, the lowest asymptotic communication complexity of

$\Theta(k+d)$. The scheme is somewhat similar to CMS [2], but the scheme is described (and its security proven) with respect to general groups that have "hidden subgroups" of smooth order. Our scheme also transforms the CMS technique to *maximize* the number of database bits the user can recover from a short server response. The essential technique is to associate each *block* of bits with a distinct *small* prime (or power of a small prime), rather than allocating a (largish) prime to each bit. The database's response protocol uses the Chinese Remainder Theorem to encode each database chunk modulo its associated prime power. To decode, the user computes a discrete logarithm, but in a subgroup whose order is *smooth* – i.e., a product of small primes. We can carry out this step efficiently in a (somewhat surprising) *constructive* application of the Pohlig-Hellman method [14]. In the full version of the paper, we show that our scheme is secure against generic attacks even when $k = \mathcal{O}(\log n)$ and when the rate of the scheme approaches 1. We provide an oblivious transfer scheme with similar performance characteristics by using the Naor-Pinkas transformation[13].

We describe an instantiation of our scheme that, like CMS, uses a (e.g., 1024-bit) composite modulus m. In CMS as described above, a user sends a $2l_m$-bit query and gets back a l_m-bit response that allows the user to retrieve a single bit; in our scheme, with the same communication complexity, the user can recover $c \cdot l_m$ bits for $c < 1/4$; this is a fairly high constant "rate" – i.e., the communication of the PBR scheme is only a small constant times more than the communication needed to transmit the block with *no privacy at all*. This instantiation has the best known asymptotic communication complexity $\Theta(\log^{3-o(1)} n, d)$ in terms of n and d among single-database PIR schemes and has the lowest complexity for most practical parameters (until it is overtaken by the modified version of Lipmaa's scheme with rate approaching 1). However, this instantiation does not perform as well as our scheme *could* perform according to the generic group model, since it is vulnerable to the number field sieve unless $k = \Omega(\log^{3-o(1)} n)$ and to Coppersmith's attack [6, 7] when $c \geq 1/4$. We speculate on approaches to instantiating the scheme that may achieve better performance.

2 Preliminaries

In the sequel, n denotes the database size in bits. If S is a set of elements, and D is a sampleable probability distribution on S, we let $s \xleftarrow{D} S$ denote the process of picking an element s from S according to distribution D. Throughout, π will denote a prime power. We say that an integer m Φ-hides π if π divides $\phi(m)$.

If A is an algorithm, we let $A(\cdot, \ldots, \cdot)$ denote that A may take one or more inputs. By $Pr[y \leftarrow A(x) : b(y)]$, we denote the probability that $b(y)$ is true after y was generated by A on input x. By $A^{(B)}(\cdot)$, we denote an algorithm that can make oracle queries to B. For $a, b \in \mathbb{Z}$ with $a \leq b$, let $[a, b]$ denote the set of integers between a and b inclusive. Let $[b]$ denote $[1, b]$.

Now, we define polylogarithmic private information retrieval as in [2].

Definition 1 (Polylogarithmic CPIR). *Let $Q(\cdot,\cdot,\cdot)$, $D(\cdot,\cdot,\cdot)$ and $R(\cdot,\cdot,\cdot,\cdot,\cdot)$ be polynomial-time algorithms. We say that (Q, D, R) is a fully polylogarithmic CPIR scheme if there exists constants $a, b, c, d > 0$ such that:*

- *(Correctness)* $\forall n \in \mathbb{N}$, $\forall B \in \{0,1\}^n$, $\forall i \in [1, n]$, *and* $\forall k' \in \mathbb{N}$,

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

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

$$\left|\Pr[(q,s) \xleftarrow{R} Q(n, i, 1^{k'}) : A(n, q, 1^{k'}) = 1] - \Pr[(q,s) \xleftarrow{R} Q(n, j, 1^{k'}) : A(n, q, 1^{k}) = 1]\right| < 2^{-dk'}.$$

Here a, b, c, d are the fundamental constants of the CPIR scheme; B is the contents of the database, D is the database's response algorithm; Q is the user's query-generating algorithm; R is the user's response reconstruction algorithm; q is the user's actual query; s is the user's secret (associated with q); r is the database's response; and k' is a security parameter.

Notice that we have mentioned two security parameters – namely, $k' > b \log n$ above, and k in the Introduction (which may be, e.g., the bit-length of a composite modulus). The two parameters are related by $k = \mathcal{O}(f(k'))$ for some polynomial f. For example, for the modulus-based instantiation, we may have $k = \max\{1024, Ck'^3\}$ for some constant C to ensure that no $(2^{ck'} = poly(n))$-gate circuits A (e.g., a circuit running NFS) can break user privacy with probability $1/poly(n)$. Against generic attacks, $k = k'$ suffices to ensure user privacy. In short, the security parameter k' is useful because it ensures (above) that no algorithms A that are polynomial in n can break user privacy, while allowing us to separately define the security parameter k in the "common parlance" of a particular instantiation. (For example, for cryptosystems related to factoring, the security parameter k is typically defined as the modulus bit-length, even though such schemes have only $\exp(\mathcal{O}(1)k^{1/3}(\log k)^{2/3})$ security against NFS.)

3 Our General Private Block Retrieval Scheme

We now describe our PIR scheme using general groups with hidden smooth-order subgroups; afterwards, once the essential strategy of our scheme has been laid out, we will describe the computational assumption on which user privacy is based (which, by then, will seem relatively natural).

First, we give a high-level description of the scheme. The scheme has some public parameters known to all users, including the database size n, an integer parameter ℓ, a set of $t = \lceil n/\ell \rceil$ (small) distinct prime numbers $\{p_1, \ldots, p_t\}$, and a set $\mathcal{S} = \{\pi_1, \ldots, \pi_t\}$ of prime powers $\pi_i = p_i^{c_i}$, where $c_i = \lceil \ell / \log_2 p_i \rceil$ (i.e., so that $p_i^{c_i} \geq 2^\ell$). The server partitions the database B into t blocks $B = C_1 \| C_2 \| \cdots \| C_t$ of size at most ℓ. In our scheme, the user will retrieve the entire ℓ-bit block that

contains its desired bit. Each block C_i is associated to a prime power π_i. Using the Chinese Remainder Theorem, the server can express the entire database B as an integer e that satisfies $e \equiv C_i (\mathrm{mod}\, \pi_i)$, where the ℓ-bit block C_i is treated as an integer satisfying $0 \leq C_i < 2^\ell \leq \pi_i$. Notice that to retrieve C_i, it suffices to retrieve $e(\mathrm{mod}\, \pi_i)$.

Roughly speaking, to query the value of $e(\mathrm{mod}\, \pi_i)$, the user generates an appropriate cyclic group $G = \langle g \rangle$ with order $|G| = q\pi_i$ for some suitable integer q. It sends (G, g) to the server and keeps q private. Notice that G contains a subgroup H of order π_i, and that $h = g^q$ is a generator of H. (For technical reasons, in the actual scheme below, $\langle g \rangle$ may be a proper subgroup of G.)

The server responds with $g_e = g^e \in G$. The user then obtains $e(\mathrm{mod}\, \pi_i)$ by setting $h_e = g_e^q \in H$ and performing a (tractable) discrete logarithm computation: $\log_h h_e \equiv e(\mathrm{mod}\, \pi_i)$. This discrete logarithm computation, which occurs entirely in the subgroup H of order $p_i^{c_i}$, can actually be quite efficient if p_i is small. Correctness is demonstrated below. Now, we give a more precise description of the general scheme.

For some parameter choices, the user can select G such that $|G|$ is divisible by multiple π_i's. In this case, the user can recover multiple ℓ-bit blocks (note that this does not contradict the security requirements for PIR schemes). However, for simplicity, we focus on the single-block case.

Specification of the Scheme. Let B be an n-bit database. Let $f_1(x, y)$ and $f_2(x, y)$ be functions. Let $k' = \Theta(\log n)$ and $k = f_2(k', \log n)$ be security parameters. Set $\ell = \lfloor f_1(k, \log n) \rfloor$ and $t = \lceil n/\ell \rceil$. For primes $\mathcal{P} = \{p_1, \ldots, p_t\}$, set $\pi_i = p_i^{c_i}$ for $c_i = \lceil \ell / (\log_2 p_i) \rceil$, and $\mathcal{S} = \{\pi_i\}$. Let $\mathcal{G}_i$ be the set of cyclic groups whose order is a number in $[2^k, 2^{k+1}]$ that is divisible by π_i. Let D_i be a distribution under which elements of $\mathcal{G}_i$ can be efficiently sampled. We assume that for $G \xleftarrow{D_i} \mathcal{G}_i$, each $g \in G$ has a unique "normal" representation. (We will discuss the security considerations involved in choosing k', f_1, f_2 and $\{D_i\}$ later.)

Query Generation: Given input $(n, f_1, f_2, \mathcal{S}, \{D_i\}, 1^{k'})$, the user determines the index i of its desired block, and generates a query for block C_i as follows:

1. Generate $G \xleftarrow{D_i} \mathcal{G}_i$ and a uniformly random "quasi-generator" g of G – i.e., g is a random element of G such that $GCD(|G : \langle g \rangle|, \prod_{j=1}^{t} p_j) = 1$;
2. Output query (G, g); keep $q = |\langle g \rangle| / \pi_i$ private; store $h = g^q$ for future use.

Database Response Generation: Given the input $(B, f_1, f_2, \mathcal{S}, G, g, 1^{k'})$, the server responds to the user's query as follows:

1. Express each ℓ-bit database block C_j (after appending zeros to C_t if needed) as a number in $[0, 2^\ell - 1]$ in the obvious fashion;
2. Set e to be the smallest positive integer such that $e \equiv C_j \pmod{\pi_j}$ for all j;
3. Output the response $g_e = g^e \in G$.

Note that steps 1 and 2 are independent of the query, and can be precomputed.

Response Retrieval: Given the input $(\pi_i, g_e, G, q, h, 1^{k'})$, the user retrieves block C_i as follows:

1. Compute $h_e = g_e^q$;
2. Compute C_i as the discrete logarithm $\log_h h_e$ within the subgroup $H \subset G$ of order $\pi_i = p_i^{c_i}$ using Pohlig-Hellman.

Notice that we need p_i to be small (unlike CMS) for the discrete logarithm computation using Pohlig-Hellman to be efficient. Fortunately, as we show below, the Prime Number Theorem will help us ensure that $\max\{p_i\}$ is small, and that response retrieval is efficient.

Correctness of Response Retrieval. Let $e_{\pi_i} \in [0, \pi_i - 1]$ satisfy $e_{\pi_i} \equiv e (\mathrm{mod} \pi_i)$; observe that e_{π_i} is equal to C_i. So, it suffices to show that e_{π_i} is the discrete logarithm of h_e with respect to base h. Write $e = e_{\pi_i} + \pi_i \cdot E$, for some $E \in \mathbb{Z}$. Now:

$$h_e = g_e^{|\langle g \rangle|/\pi_i} = g^{e|\langle g \rangle|/\pi_i} = g^{e_{\pi_i}|\langle g \rangle|/\pi_i} = h^{e_{\pi_i}}.$$

Remark 1. The above scheme has some similarities to CMS, particularly if one instantiates the group G using a composite modulus m. However, for recovering *blocks* of data (a more realistic scenario anyway), our scheme is much more communication efficient; the server's $(\log m)$-bit response uses the Chinese Remainder Theorem to give the user ℓ bits instead of 1 bit. Later, we will see that ℓ can equal $(\log m)/C$ for reasonably small constant C.

Choosing the Set $\mathcal{P}$ Wisely. Recall that $\mathcal{P} = \{p_1, \ldots, p_t\}$ is the set of primes that the scheme uses; let p_t be the largest. As mentioned above, p_t must be reasonably small to ensure efficient response retrieval. Also, since we must have $\log |G| \geq \max\{\pi_i\} \geq p_t$, the size of p_t also affects communication complexity. The following result of Rosser and Schoenfeld related to the Prime Number Theorem [15] gives an upper bound on p_t.

Theorem 1 (Rosser and Schoenfeld). *For $t > 20$, let $\mathcal{P} = \{p_1, \ldots, p_t\}$ be the first t primes, with p_t the largest. Then, $p_t < t(\ln t + \ln \ln t - 1/2)$.*

For technical reasons in the security proof, we need $p_1 \geq 2t$. Nonetheless, in terms of n and ℓ, we easily get that $p_t < 16(n/\ell) \log_2(n/\ell)$ suffices. For the performance analysis below, we assume for convenience that ℓ is chosen so that $2^\ell \geq p_t$.

Computational Complexity. The dominant component of the querier's computation is in computing the discrete logarithm of h_e for base h. This step involves solving c_i discrete logarithm sub-problems in groups of order p_i for $p_i^{c_i} \in [2^\ell, 2^\ell p_t]$. Assuming that each sub-problem involves $\sqrt{p_i}$ group operations – e.g., using baby-step giant-step – the entire discrete logarithm problem requires about $c_i \sqrt{p_i}$ group operations. Considering the curve $y^x = 2^\ell p_t$ for $y \leq p_t$, we see that $x\sqrt{y} = (\sqrt{y}/\log y)(x \log y) = (\sqrt{y}/\log y)(\log(2^\ell p_t))$ takes its maximum at $y = p_t$. As a very rough upper bound, $\sqrt{p_t}/\log p_t < 2\sqrt{n/\ell}$ and $\log(2^\ell p_t) < 2\ell$, so the querier's computation is no more than $4\sqrt{n\ell}$ group operations, where ℓ must be less than $\log |G|$ (which will be polylogarithmic in n). This does not seem

unreasonable given that the database's computation in single-database PIR is *unavoidably linear* in n (since otherwise the database has not included every database bit in the computation, which would imply that it knows at least one bit that the user did not request).

The dominant component of the database's computation is in computing $g^e \bmod m$. This requires (roughly) $\log e$ group operations. Since e is a number modulo $\prod_{i=1}^{t} \pi_i$, we have $\log e \leq \sum_{i=1}^{t} \log \pi_i$. Since, $p_i \leq 2^\ell$ for all i, $\pi_i = p_i^{c_i} < 2^{2\ell}$ for all $c_i = \lceil \ell/(\log p_i) \rceil$. Thus, we have $\sum_{i=1}^{t} \log \pi_i < 2\ell t = 2\ell \lceil n/\ell \rceil$ – i.e., the database needs $\Theta(n)$ group operations, which is about the best we can hope for in single-database PIR.

Communication Complexity. Suppose that the group G and any element of G can be described in $l_G = \Omega(\log |G|)$ bits. (For example, the group generated by g modulo m for composite modulus m can be described in $\mathcal{O}(\log \phi(m))$ bits.) Then, the total communication complexity is $3l_G$. The size of l_G depends, in part, on security considerations pertaining to the particular instantiation of our general scheme; so, we obviously cannot give a general upper bound for l_G. Here, we merely note that, in terms of the scheme's *correctness*, the only constraint on $|G|$ is that it be divisible by (and, hence, at least as large as) π_i. Above, we saw that when $2^\ell > p_t$, $\pi_i < 2^{2\ell}$ for all i. Thus, if we set $\ell = \lceil \log p_t \rceil$, then $\max\{\log \pi_i\} < 2\ell < 4 \log p_t < 8 \log n$. Thus, the mechanics of the scheme do not prevent $\log |G| = \Theta(\log n)$ or $l_G = \Theta(\log n)$.

We stress that l_G may need to be larger to ensure user privacy. However, in our analysis of the scheme's security in the generic group model in Section 6, we find that generic attacks do not prevent our scheme with $l_G = \Theta(\log n)$ from having the security required by CMS's definition of polylogarithmic PIR; any attack that forces l_G to be larger must exploit the encoding of the group or its elements.

Private Block Retrieval. In our scheme, the user already recovers ℓ-bit blocks. This scheme can be converted, using the general transformation described in [5], into a scheme that recovers d ℓ-bit blocks with total communication complexity $(2+d)l_G$, as follows. The user generates a query (G, g) for the ℓ-bit block beginning with index i. To allow the user to retrieve the ℓ-bit block with index $i + x\ell$ for $x \in [0, d-1]$, the server temporarily relabels the database, giving the database bit with index j (for $j \in [n]$) the "temporary index" $j - x\ell (\mathrm{mod} n)$; it then responds to the user's query (G, g) using the temporary indices, rather than the actual ones. The "rate" of our scheme – i.e., the ratio of the number of bits that the user retrieves over the total communication complexity – is $d\ell/(d+2)l_G$, which approaches ℓ/l_G as d increases. We will see that our general scheme is secure against generic group attacks for ℓ/l_G arbitrarily close to 1. When we instantiate the scheme using Φ-hiding and a composite modulus m in the natural way, however, an attack by Coppersmith [7] forces $\ell/l_G < 1/4$.

Oblivious Transfer. Naor and Pinkas [13] describe how to construct 1-out-of-n OT scheme from a PIR scheme for n-bit databases and $\log n$ invocations of a 1-out-of-2 OT scheme. Since the transformation is generic, we omit the

details, except to mention that 1-out-of-2 OT can be accomplished fairly efficiently through the ElGamal encryption scheme. If k'' is the bit-length of group elements in the ElGamal group (e.g., $k'' = 160$), the transformation only adds $6k''(\log n)$ bits to our PIR scheme, regardless of the block size d.

4 Our General Computational Assumption

In our PIR scheme, the server is given not only a description of G (and generator g), but also a promise that one of the prime powers in $\mathcal{S}$ – i.e., the one associated to the user's target block index i – actually divides $|G|$. For our PIR scheme to be user-private, the server should be unable to distinguish which of π_0 or π_1 divides $|G|$ – or, equivalently, to distinguish whether the "smooth" subgroup H hidden inside G has order π_0 or π_1. So, our computational assumption is roughly that, given (π_0, π_1, G) and the promise that π_b divides $|G|$ for one $b \in \{0,1\}$, it is computationally hard (if G is generated appropriately) to distinguish the value of b, even if π_0 and π_1 are not "much smaller" than $|G|$, and even if π_0 and π_1 are "special" integers such as powers of small primes. We formalize this assumption in terms of the following problem.

Definition 2 (The Decision Subgroup Problem). *Let ℓ be an integer and k a parameter. Let $\pi_0, \pi_1 \in [2^\ell, 2^{2\ell} - 1]$ be distinct integers. Let $\mathcal{G}_i$ be the set of cyclic groups whose order is a number in $[2^k, 2^{k+1}]$ that is divisible by π_i. Let D_i be a distribution on $\mathcal{G}_i$. We say that algorithm A has advantage ϵ against the $(\ell, k, \pi_0, \pi_1, D_0, D_1)$-Decision Subgroup Problem if*

$$\left| \Pr\left[b \xleftarrow{R} \{0,1\}, G_b \xleftarrow{D_b} \mathcal{G}_b : A(G_b, \ell, k, \pi_0, \pi_1, \{D_i\}, \{\mathcal{G}_i\}) = b \right] - \frac{1}{2} \right| \geq \epsilon.$$

A solves the problem if it guesses b correctly.

In our PIR scheme, we want the above problem to be hard for each pair $\pi_{i_0}, \pi_{i_1} \in \mathcal{S}$, a set of prime powers. Thus, we state our computational assmption as follows.

Definition 3 (The (Extended) Decision Subgroup Assumption). *Let $f_1(x, y)$ and $f_2(x, y)$ be functions. Let $\mathcal{S}$ be a set of $t \geq 2$ powers of distinct primes. The $(f_1, f_2, \mathcal{S})$-Extended Decision Subgroup Assumption is that there exist constants $b, c, d > 0$ such that, for all $n \in \mathbb{N}$ and all $k' > b \log n$ with $\ell = \lfloor f_1(k', \log n) \rfloor$ and $k = f_2(k', \log n)$, there exist efficiently sampleable distributions $\{D_i : i \in [t]\}$ such that, for all $i_0, i_1 \in [t]$, all circuits $\mathcal{A}$ with $(2^{ck'} + t \cdot f_2(k', \log n) \cdot C_{\{\mathcal{D}_i\}})$ gates have advantage at most $2^{-dk'}$ against the $(\ell, k, \pi_{i_0}, \pi_{i_1}, D_{i_0}, D_{i_1})$-Decision Subgroup Problem, where $C_{\{\mathcal{D}_i\}}$ is an upper bound on the circuit complexity of a group multiplication in groups drawn according to D_i for $i \in [t]$.*

5 Security Proof for Our PIR Scheme

We base the security of our scheme on the extended decision subgroup assumption. The proof is done in the standard model.

Theorem 2. *Suppose that a circuit $\mathcal{A}$ with $2^{ck'}$ gates can break user privacy with advantage $2^{-dk'}$. Then, there is an $\mathcal{A}'$ with $\mathcal{O}(2^{ck'} + t \cdot f_2(k', \log n) \cdot C_{\{\mathcal{D}_i\}})$ gates that solves the extended decision subgroup problem with advantage $\frac{1}{5}2^{-dk'}$.*

Proof. Suppose that the privacy condition fails for (Q, D, R). Then for all $b, c, d > 0$, there exist n, $k' > b \log n$, $B \in \{0,1\}^n$, block indices $i \neq j$, and a circuit $\mathcal{A}$ with $2^{ck'}$ gates, such that $|\alpha_{i,0^t} - \alpha_{j,0^t}| \geq 2^{-dk'}$, where:

$$\alpha_{i,\overline{v}} \triangleq \Pr[((G,g),q) \leftarrow Q(i,T) : \mathcal{A}((G, g^{\prod_{x=1}^{t} p_x^{v_x}}), T) = 1],$$
$$\alpha_{j,\overline{v}} \triangleq \Pr[((G,g),q) \leftarrow Q(j,T) : \mathcal{A}((G, g^{\prod_{x=1}^{t} p_x^{v_x}}), T) = 1],$$

where $\overline{v}$ is a t-element integer vector and 0^t is the zero vector, and where we define $T = (n, f_1, f_2, \mathcal{S}, \{D_i\}, 1^{k'})$ for convenience. We now define two probabilities representing $\mathcal{A}$'s output when g is chosen uniformly at random from G (as opposed to being a random quasi-generator of G):

$$\beta_{i,\overline{v}} \triangleq \Pr[G \xleftarrow{D_i} \mathcal{G}_i; g \xleftarrow{\mathrm{R}} G : \mathcal{A}((G, g^{\prod_{x=1}^{t} p_x^{v_x}}), T) = 1],$$
$$\beta_{j,\overline{v}} \triangleq \Pr[G \xleftarrow{D_j} \mathcal{G}_j; g \xleftarrow{\mathrm{R}} G : \mathcal{A}((G, g^{\prod_{x=1}^{t} p_x^{v_x}}), T) = 1].$$

Let $\overline{e_x}$ be the unit vector in dimension x. If $p_x^{v_x}$ is greater than $2^{f_2(k,\log n)+1}$ (the maximum possible group order), then $\alpha_{i,\overline{v}} = \alpha_{i,\overline{v}-\overline{e_x}}$ since the distributions of the element given to $\mathcal{A}$ are identical. Let $\overline{v_0}$ be s.t. $\epsilon_{\alpha\alpha} = |\alpha_{i,\overline{v_0}} - \alpha_{j,\overline{v_0}}|$ is maximal and s.t. $p_x^{v_{0,x}} \leq 2^{f_2(k,\log n)+1}$ for all $x \in [t]$. Set $\epsilon_{\beta\beta} \triangleq |\beta_{i,\overline{v_0}} - \beta_{j,\overline{v_0}}|$. Then, $\mathcal{A}'$ can solve the decision subgroup problem instance for prime powers π_i, π_j with advantage $\epsilon_{\beta\beta}$ simply by generating random $g \in G$ and passing $(G, g^{\prod_{x=1}^{t} p_x^{v_{0,x}}})$ to $\mathcal{A}$, and then outputting "i" if $\mathcal{A}$ outputs 1 and outputting "j" otherwise.

Let $\overline{w} \geq \overline{v}$ denote $\forall x \in [t], w_x \geq v_x$. We express $\beta_{i,\overline{v}}$ in terms of $\{\alpha_{i,\overline{w}} : \overline{w} \geq \overline{v}\}$ by noting that choosing an element of G uniformly at random is equivalent to choosing a uniformly random quasi-generator and then exponentiating it by $\prod_{x=1}^{t} p_x^{w_x - v_x}$ with probability $(\prod_{x=1}^{t} \frac{p_x - 1}{p_x})/(\prod_{x=1}^{t} p_x^{w_x - v_x})$. We obtain:

$$\beta_{i,\overline{v}} = \left(\prod_{x=1}^{t} \frac{p_x - 1}{p_x} \right) \left(\sum_{\overline{w} \geq \overline{v}} \alpha_{i,\overline{w}} / (\prod_{x=1}^{t} p_x^{w_x - v_x}) \right) .$$

Since $p_1 > 2t$, $\sum_{\overline{w} \geq \overline{v_0}} \alpha_{i,\overline{w}}/(\prod_{x=1}^{t} p_x^{w_x - v_{0,x}}) < \epsilon_{\alpha\alpha} \prod_{x=1}^{t} \frac{p_x}{p_x - 1} < \epsilon_{\alpha\alpha} e^{t/(p_1 - 1)} < \epsilon_{\alpha\alpha}\sqrt{e}$. By the triangle inequality, $|\beta_{i,\overline{v_0}} - \beta_{j,\overline{v_0}}| \geq (\epsilon_{\alpha\alpha} - (\sqrt{e}-1)\epsilon_{\alpha\alpha})(\prod_{x=1}^{t} \frac{p_x - 1}{p_x})$ $\geq \epsilon_{\alpha\alpha}(2 - \sqrt{e})/(\sqrt{e}) > \epsilon_{\alpha\alpha}/5$. So, $\mathcal{A}$ has $\epsilon_{\beta\beta} \geq \epsilon_{\alpha\alpha}/5 \geq (1/5)2^{-dk'}$ advantage against the Decision Subgroup Problem for (π_i, π_j). The circuit complexity of $\mathcal{A}'$ is basically that of $\mathcal{A}$, plus that needed to compute $g^{\prod_{x=1}^{t} p_x^{v_{0,x}}}$. □

6 Lessons from the Generic Group Model

To gain confidence in our computational assumption, we can consider the Decision Subgroup Problem's vulnerability to generic attacks. The following theorem, which is quite similar to a result by Damgard and Koprowski [8] on root extraction in generic groups, roughly states that, as long as the distributions D_0 and D_1 each tend to output a group whose order is divisible by a large evenly-distributed prime, the Decision Subgroup Problem is hard against generic attacks. In other words, the security of the Decisional Subgroup Problem against generic attacks depends less on the value of $|H|$ (the order of the subgroup hidden in G) than it does on the distribution of $|G : H|$.

Theorem 3. *Let A be a generic algorithm for solving the Decision Subgroup Problem on $(\ell, k, \pi_0, \pi_1, D_0, D_1)$ that makes at most m oracle queries. Let S be a set of bit strings of cardinality at least $2^{2\ell}$. For group G_i, let $\theta(G_i)$ be the largest prime divisor of $|G_i|$ that does not divide $\pi_0\pi_1$. Let $\alpha(D_i) = \max_q\{\Pr[\theta(G_i) = q \mid G_i \xleftarrow{D_i} \mathcal{G}_i]\}$. Let $\beta(D_i, M)$ be the probability that $\theta(G_i) \leq M$ for distribution D_i; let $\beta(D, M) = \max\{\beta(D_0, M), \beta(D_1, M)\}$. Now, randomly choose $b \xleftarrow{R} \{0, 1\}$, $G_b \xleftarrow{D_b} \mathcal{G}_b$, and a random mapping $\sigma_b : G_b \to S$. Then,*

$$\left|\Pr\left[A^{(O)}(S, \ell, k, \pi_0, \pi_1, D_0, D_1) = b\right] - \frac{1}{2}\right| \leq m^2\left(m\alpha(D) + \beta(D, M) + \frac{1}{M}\right)/2,$$

where the probability is over the random bits of the oracle and A.

Proof. See full version of this paper.

Let's choose parameters to give Theorem 3 meaning. Suppose $2^k/\max\{\pi_i\} \geq 2^{k'}$, and define D_i to choose $|G_i|$ as follows: choose a uniformly random prime q from $[2^{k'}, 2^{k'+1}] \setminus P$ (where P is the set of primes dividing $\pi_0\pi_1$) and an integer d from the set of integers in the interval $[2^k/q\pi_i, 2^{k+1}/q\pi_i]$ whose prime divisors are all less than q; set $|G_i| = \pi_i qd$. Then, by the Prime Number Theorem, $\alpha(D) \approx 2^{-k'+\log k'} \ln 2$. If we set $M = 2^{k'}$, then $\beta(D, M) = 0$. Once we insert these values into Theorem 3, we find that a generic algorithm for solving the Decision Subgroup Problem for such D_i takes $\Omega(2^{(k'-\log k')/3})$ oracle queries. Thus, when $2^k/\max\{\pi_i\} \geq 2^{k'}$, the Extended Decision Subgroup Assumption is absolutely true in the generic group model.

Now, let's consider how well our scheme could perform, if generic attacks were the only security concern. First, consider the rate of our scheme. If we set $k = \lceil k' + \ell + \log p_t \rceil$, then $2^k/\max\{\pi_i\} \geq 2^{k'}$ as required above, while the rate $f_1(k', \log n)/f_2(k', \log n) = k/\ell$ can be arbitrarily close to 1. Also, since $k' = b \log n$, $\log p_t = \mathcal{O}(\log n)$, and ℓ can be chosen to be $\mathcal{O}(\log n)$, k can also be purely logarithmic in n. Thus, generic attacks do not prevent our scheme from achieving an optimal rate (approaching 1) for blocks, and minimal communication $\mathcal{O}(\log n)$ for private bit retrieval.

7 Instantiating Groups with Hidden Smooth Subgroups

Up to this point, we have discussed our PIR scheme and its performance and security properties in a general way, without discussing in detail how to instantiate the group G securely. One way to instantiate G is using a composite modulus, as in [2]. For example, to construct a modulus m that Φ-hides π, one may choose a random "semi-safe" prime $Q_0 = 2q_0\pi + 1$ for prime q_0 and a random semi-safe prime $Q_1 = 2dq_1 + 1$ for prime q_1 and d chosen uniformly from a large interval, and set $m = Q_0Q_1$. Then, m should have good uniformity properties, even modulo the primes dividing π.

Cachin, Micali and Stadler [2] note that when a divisor $\pi \geq m^{1/4}$ of $(Q_0 - 1)$ is known, however, it is easy to decide whether π divides $\phi(m)$; in particular, given $m = Q_0Q_1$ and divisor $\pi \geq m^{1/4}$ of $(Q_0 - 1)$, one can factor m using Coppersmith's method [7], [6] – a lattice-based attack. An abundance of work relating to Coppersmith's method has appeared in the literature, the most recent being May's Eurocrypt 2005 paper [12], which provides a unifying framework for most of the results. His Corollary 14 applies to the Φ-hiding situation; it states:

Corollary 14 (A. May). *Let $f(x) \in \mathbb{Z}[X]$ be a polynomial of degree δ. Let m be a composite number of unknown factorization with divisor $Q_0 \geq m^\beta$. Then, we can find all points $x_0 \in \mathbb{Z}$ satisfying $f(x_0) = Q_0$ in time polynomial in $\log m$ and δ if $|x_0| \leq m^{\beta^2}$.*

Setting $\beta = 1/2$ and $f(x) = \pi x + 1$, the algorithm will give us the divisor $Q_0 = \pi c + 1$ in polynomial time, since $c \approx Q_0/\pi < m^{1/4}$ for $\pi > m^{1/4}$. As May notes, this $|x_0| \leq m^{1/4}$ bound occurs frequently in the literature on Coppersmith's method. Since the algorithm works well (in polynomial time) when $\pi > m^{1/4}$, one might expect that the algorithm's performance declines only gradually – e.g., so that for $\pi > m^{1/5}$, the algorithm (while not polynomial-time) would be only slightly super-polynomial, perhaps because of the inefficiency of lattice reduction. However, this is not true; when $(\log m)/(\log \pi)$ is larger than 4, the target vector (i.e., the one that would help us factor m) is not even the shortest vector in the lattice; thus, even perfect lattice reduction algorithms would not, by themselves, make the attack work.

These considerations give us confidence that, as long as $(\log m)/(\log \pi) > 4$ (perhaps by a "comfortable" margin), then the Φ-hiding assumption, as outlined above, is hard. Thus, it seems plausible that the bit-length of m only needs to be a constant factor greater than $\log \pi$. This allows our PIR scheme to achieve constant rate when instantiated with groups modulo composite numbers. A drawback of using composite moduli is that, as mentioned before, we need $\log m = \Omega(\log^{3-o(1)} n)$, due to the number field sieve [10]. This makes our PIR scheme somewhat communication-inefficient for short block sizes d, even though it is most efficient among the single-database PIR schemes that currently exist.

Of course, it would be preferable to instantiate our scheme using groups for which the number field sieve is inapplicable if this could be done securely. For example, one might try elliptic curve groups. However, algorithms exist to find the orders of elliptic curves over finite fields; when we try using the compositum of

finite fields, we seem to be reverting back to a factorization problem. Class groups are another interesting alternative, since currently the best known algorithms for attacking class groups (e.g., determining their order) have quadratic-sieve-type complexity. Unfortunately, in our scheme, the user generating the group must know its order for response retrieval; currently, there are no efficient algorithms that would allow the user to generate a class group with known partially-smooth order, as required by our scheme.

8 Conclusion and Open Problems

We described single-database computational block retrieval schemes based on the decision subgroup problem with communication complexity $\mathcal{O}(k+d)$, where d is the size of the block to be retrieved and k is the security parameter. Asymptotically, this is about as good as one might expect since there is only an additive communication overhead of the security parameter k. Indeed, our scheme has better asymptotic performance compared to previous schemes.

We leave it as an open problem to construct an instantiation of our scheme that achieves rate arbitrarily close to 1, while circumventing Coppersmith's attack. Clearly, based on our analysis of the Decision Subgroup Problem in the generic group model, any attack that prevents the scheme from achieving rate close to 1 must exploit the encoding of the elements.

Acknowledgments. We thank Phil Mackenzie, David Woodruff, Helger Lipmaa, Yuval Ishai, and the anonymous referees for fruitful comments.

References

[1] A. Beimel, Y. Ishai, E. Kushilevitz, and J. F. Raymond. *Breaking the $O(n^{1/(2k-1)})$ Barrier for Information-Theoretic Private Information Retrieval,* FOCS 2002.

[2] C. Cachin, S. Micali, M. Stadler, *Computational Private Information Retrieval with Polylogarithmic Communication*, Eurocrypt 1999.

[3] Y. Chang. *Single-Database Private Information Retreival with Logarithmic Communication,* ACISP 2004.

[4] B. Chor and N. Gilboa, *Comput. Private Information Retrieval,* STOC 1997.

[5] B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan, *Private Information Retrieval,* Journal of the ACM, 45, 1998. Earlier version in FOCS 95.

[6] D. Coppersmith, *Finding a Small Root of a Bivariate Integer Equation; Factoring with High Bits Known*, Eurocrypt 1996.

[7] D. Coppersmith, *Finding a Small Root of a Univ. Mod. Equation*, Eurocrypt 1996.

[8] I. Damgard and M. Koprowski, *Generic Lower Bounds for Root Extraction and Signature Schems in General Groups*, Eurocrypt 2002.

[9] E. Kushilevits and R. Ostrovsky, *Replication is not needed: single database, computationally private information Retrieval.* FOCS 1997.

[10] A.K. Lenstra and H.W. Lenstra, Jr., (eds.), *The Development of the Number Field Sieve,* Lecture Notes in Mathematics 1554, Springer-Verlag, 1995.

[11] H. Lipmaa, *An Oblivious Transfer Protocol with Log-Squared Communication.* Cryptology ePrint Archive, 2004/063.
[12] A. May, *A Tool Kit for Finding Small Roots of Bivariate Polynomials over the Integers*, Eurocrypt 2005.
[13] M. Naor and B. Pinkas, *Obl. Transfer and Polynomial Evaluation*, STOC 1999.
[14] S.C. Pohlig and M. Hellman. *An Improved Algorithm for Computing Logarithms Over* GF(p) *and its Crypt. Significance,* IEEE Trans. Inf. Th. IT-24 (1978).
[15] J.B. Rosser and L. Schoenfeld, *Sharper Bounds for Chebyshev Functions* $\theta(x)$ *and* $\psi(x)$, Math. Comput. 29, 243-269, 1975.
[16] J.P. Stern, *A New and Efficient All or Nothing Disclosure of Secrets Protocol,* Asiacrypt 1998.