Private Information Retrieval

Aleksandr Grebennik

November 7, 2005

1 Introduction

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 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$i$.

Preliminaries

All logarithms in this paper will be on base 2, unless explicitly mentioned.

In this paper, ℓ_m denotes the length of number m in its binary representation, i.e. $\ell_m = \lceil \log m \rceil$.

A number z is said to be an M-th residue modulo N if there exists a number $y \in \mathbb{Z}_N$ such that $z \equiv y^M \pmod N$.

Let r and s be positive integers which are relatively prime and let a and b be any two integers. Chinese Remainder Theorem claims that there is an integer N such that $N \equiv a \pmod r$ and $N \equiv b \pmod s$ and that N is uniquely determined modulo rs.

Group G of order m is cyclic if $\exists g \in G : G = \{g, g^2, \ldots, g^{m-1}, g^m = g^0\})$.

For a positive integer n, Euler's totient function $\phi(n) = |\{x \in \mathbb{N} : x \leq n, \gcd(x,n) = 1\}|$; if n factorization is $n = p_1^{a_1} p_2^{a_2} \ldots p_m^{a_m}$, then $\phi(n) = \prod_i^m (p_i^{a_i} - p_i^{a_i - 1})$.

Baby-step giant-step algorithm refers to a series of well defined steps to compute the discrete logarithm in a cyclic group G. The problem is to find x in $a^x = b \pmod n$ where a, b and n are given. The baby-step giant-step algorithm is based on rewriting x as $x = im + j$ with $m = \lceil \sqrt{n} \rceil$ and $0 \leq i, j < m$. Therefore, we have: $b(a^{-m})^i = a^j \pmod n$. The algorithm precomputes $a^j \mod n$ for all $0 \leq j < m$ and stores the pairs (j, a^j) in a table. Then it tries values of $0 \leq i < m$ until the congruence is satisfied with some of precomputed a^j.

2 First Single-Database PIR

Cachin, Micali and Stadler constructed the first single-database PIR scheme with *poly-logarithmic* communication complexity. The security of their scheme (CMS) is based on "Φ-hiding" assumption — roughly, that it 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 $B = b_1 \ldots b_n$, the user sends a composite (hard-to-factor) modulus m such that p_i divides $\phi(m)$ and a generator $x \in \mathbb{Z}_m^*$ ($|\mathbb{Z}_m^*| = \phi(m)$). The server sends back $r \equiv x^P \pmod m$ for $P = \prod_j p_j^{b_j}$. The user concludes that $b_i = 1$ if r is a p_i-residue modulo m; otherwise, $b_i = 0$. The communication complexity of (this simplified version of) CMS is $3\ell_m$ to recover 1 database bit. The authors recommend the value of ℓ_m to be in the order of $\mathcal{O}(\log^8 n)$.

3 Gentry-Ramzan Private Block Retrieval Scheme

General description

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 $S = \pi_1, \ldots, \pi_t$ of prime powers $\pi_i = p_i^{c_i}$, where $c_i = \lceil \ell / \log_2 p_i \rceil$ (so that $p_i^{c_i} \geq 2^\ell$). The server partitions the database B into t blocks $B = C_1 \| C_2 \| \ldots \| C_t$ of size at most ℓ. In this scheme, the user will retrieve the entire ℓ-bit block that contains 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 $e \equiv C_i \pmod{\pi_i}$, 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 $e \mod \pi_i$.

Roughly speaking, to query the value of $e \mod \pi_i$, the user generates an appropriate cyclic group $G = \langle g \rangle$ with order $|G| = q\pi_i$ for some suitable integer q. He 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.

The server responds with $g_e = g^e \in G$. The user then obtains $e \mod \pi_i$ by setting $h_e = g_e^q \in H$ ($g_e^q \in H$ because $g_e^q = (g^e)^q = (g^q)^e = h^e \in H$) and performing a (tractable) discrete logarithm computation: $\log_h h_e \equiv e \pmod{\pi_i}$ using Pohlig-Hellman algorithm.

Let us describe this algorithm a little more in detail. We are searching for $x = C_i = log_h h_e$ in group H (equality $C_i = log_h h_e$ is proved in the Correctness of Response Retrieval section below). As C_i is a number modulo $\pi_i = p_i^{c_i}$, we can write x in base p_i: $x = x_0 + x_1 p + \ldots + x_{c-1} p^{c-1}, 0 \leq x_i < p$. Now, define $y := h_e$. Notice that $y^{|H|/p} = h^{C_i |H|/p} = h^{x|H|/p} = h^{[x_0|H|/p] + [(x_1 p + \ldots + x_{c-1} p^{c-1})|H|/p]} = h^{x_0|H|/p} h^{(x_1 + \ldots + x_{c-1} p^{c-2})|H|} = h^{x_0|H|/p}$. We can now just try all $0 \leq x_0 < p$ candidates and thus get first "digit" x_0. This is feasible because p is small due to the construction of this scheme. Now, assume $y_1 := h_e h^{-x_0}$ and compute $y_1^{|H|/p^2}$, thus finding second "digit" x_1. Next, define $y_2 := h_e h^{-x_0 - x_1 p}$ and compute $y_1^{|H|/p^3}$, thus finding third "digit" x_2. Repeat until all "digits" $\{x_0, \ldots, x_{c-1}\}$ are found.

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.

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).

Correctness of Response Retrieval

Let $e_{\pi_i} \in [0, \pi_i - 1]$ satisfy $e_{\pi_i} \equiv e \pmod{\pi_i}$; 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 $e = e_{\pi_i} + \pi_i E$, for some $E \in \mathbb{Z}$. Now: $h_e = g_e^{|\langle g \rangle|/\pi_i} = g^{e|\langle g \rangle|/\pi_i} = g^{e_{\pi_i}|\langle g \rangle|/\pi_i} g^{E|\langle g \rangle|} = g^{e_{\pi_i}|\langle g \rangle|/\pi_i} = h^{e_{\pi_i}}$.

Computational Complexity

The dominant component of the querier's computation is in computing the discrete logarithm of h_e for base h. The authors state that the querier's computation is no more than $4\sqrt{n\ell}$ operations, where ℓ must be less than $\log |G|$

The dominant component of the database's computation is in counting $g^e \mod m$. 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 $\Theta(n)$ operations.

Computational Assumption

This scheme's 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 integers.

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 $\mathcal{O}(\ell_m)$ bits.) Then, the total communication complexity is $3\ell_G$. The size of ℓ_G depends, in part, on security considerations pertaining to the particular instantiation of our general scheme. 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.

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 in using a composite modulus, as in CMS scheme. 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, π divides $\phi(m)$ because $\phi(m) = \phi(Q_0Q_1) = \phi(Q_0)\phi(Q_1) = (Q_0 - 1)(Q_1 - 1) = 4d\pi q_0 q_1$ Also, m should have good uniformity properties, even modulo the primes dividing π.

4 Lipmaa Oblivious Transfer Protocol with Log-Squared Communication

I was not able to get into details of this scheme, though in the article it was twice stated that this protocol is "simple to understand and implement". The underlying crypto primitive in use is length-flexible additively homomorphic public-key cryptosystem that should be IND-CPA secure.

Additively homomorphic public-key scheme

A length-flexible additively homomorphic (LFAH) public-key cryptosystem is a tuple $\Pi = (Gen, Enc, Dec)$, where (a) Gen is a key generation algorithm, that on input 1^k returns (sk, pk), where sk is a secret key and pk is a public key, (b) Enc is an encryption algorithm, that on input (pk, s, m, r), where $s \in \mathbb{Z}^+$ is a length parameter, m is a plaintext and r is a random coin, returns a ciphertext $Enc^s_{pk}(m; r)$, and (c) Dec is a description algorithm that on input (sk, s, c), where s is a length parameter and c is a ciphertext, returns a plaintext $Dec^s_{sk}(c)$. For any $s \in \mathbb{Z}^+$, $Enc^s_{pk} : P_s \times R \rightarrow C_s$ and $Dec^s_{pk} : C_s \rightarrow P_s$, where C_s is the ciphertext space and P_s is the plaintext space corresponding to s, and R is the s-independent randomness space. We require that for some positive integer a, $C_s \subseteq P_{s+a}$ for every s; we assume that ξ is the minimal among such a's. An LFAH public-key cryptosystem Π is *additively homomorphic* if for any key pair (sk, pk), any length parameter s, any $m, m' \in P_s = \mathbb{Z}_{|P_s|}$, and any $r, r' \in R$, $Enc^s_{pk}(m; r) \cdot Enc^s_{pk}(m'; r') = Enc^s_{pk}(m + m'; r \circ r')$, where $\cdot$ is a multiplicative group operation in C_s, $+$ is addition in $\mathbb{Z}_{|P_s|}$, and $\circ$ is a groupoid operation in R.

An example of IND-CPA secure LFAH public-key cryptosystem is the Damgård-Jurik protocol named DJ01. Assume that $N = p_1p_2$ is an RSA modulus. Here, for a fixed length parameter s, $P_s = \mathbb{Z}_{N^s}, R = \mathbb{Z}^*_N$ and $C_s = \mathbb{Z}^*_{N^{s+1}}$. Encryption is defined by $Enc^s_{pk}(m; r) := (1 + N)^m \cdot r^{N^s} \bmod N^{s+1}$, where $r \leftarrow \mathbb{Z}^*_N$. DJ01 is additively homomorphic.

Communication complexity

Communication complexity for d-bit blocks is stated as $\Theta(\ell_m \cdot \log^2 n + d \cdot \log n)$, where $\ell_m = \Omega(\log^{3-o(1)} n)$. Thus, Lipmaa's scheme has a good communication complexity to amount of bits trasferred ratio, namely $1/(\log n)$ for large blocks. 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 $\ell_m \cdot \log^2 n$.

References

[1] H. Lipmaa. An Oblivious Transfer Protocol with Log-Squared Communication. Cryptology ePrint Archive, 2004/063.

[2] C. Gentry and Z. Ramzan. Single-Database Private Information Retrieval with Constant Communication Rate.

[3] S. Pohlig and M. Hellman. An Improved Algorithm for Computing Algorithms over $GF(p)$ and its Cryptographic Significance.