Public-Key Locally-Decodable Codes

Brett Hemenway[1] and Rafail Ostrovsky[2]

[1] Department of Mathematics, University of California, Los Angeles.
`bretth@math.ucla.edu` *

[2] Department of Computer Science and Department of Mathematics, University of California, Los Angeles `rafail@cs.ucla.edu` **

Abstract. In this paper we introduce the notion of a Public-Key Encryption Scheme that is also a Locally-Decodable Error-Correcting Code (PKLDC). In particular, we allow any polynomial-time adversary to read the entire ciphertext, and corrupt a constant fraction of the bits of the *entire* ciphertext. Nevertheless, the decoding algorithm can recover any bit of the plaintext with all but negligible probability by reading only a sublinear number of bits of the (corrupted) ciphertext.

We give a general construction of a PKLDC from any Semantically-Secure Public Key Encryption (SS-PKE) and any Private Information Retrieval (PIR) protocol. Since Homomorphic encryption implies PIR, we also show a reduction from any Homomorphic encryption protocol to PKLDC.

Applying our construction to the best known PIR protocol (that of Gentry and Ramzan), we obtain a PKLDC, which for messages of size n and security parameter k achieves ciphertexts of size $\mathcal{O}(n)$, public key of size $\mathcal{O}(n+k)$, and locality of size $\mathcal{O}(k^2)$. This means that for messages of length $n = \omega(k^{2+\epsilon})$, we can decode a bit of the plaintext from a corrupted ciphertext while doing computation sublinear in n.

Keywords: Public Key Cryptography, Locally Decodable Codes, Error Correcting Codes, Bounded Channel Model, Chinese Remainder Theorem, Private Information Retrieval.

1 Introduction

Error correction has been an important field of research since Shannon laid the groundwork for a mathematical theory of communication in the nineteen forties, and active research continues until this day. An error correcting code is a pair of algorithms C and D such that given a message x, $C(x)$ is a codeword such that, given a string y, if the Hamming Distance between $d(C(x), y)$ is "small", then $D(C(x)) = x$. When speaking of an error correcting code, two of

* Part of this research was done while the author was visiting IPAM. This research was supported in part by VIGRE and NSF grants 0716835 and 0716389.

** Part of this research was done while visiting IPAM. This research was supported in part by IBM Faculty Award, Xerox Innovation Group Award, NSF grants 0430254, 0716835, 0716389 and U.C. MICRO grant.

its most important characteristics are the *information rate*, which is the ratio of the message size to the codeword size $\frac{|x|}{|C(x)|}$, and the *error rate* which is the smallest ϵ such that if $d(C(x), y) > \epsilon|C(x)|$ then $D(C(x))$ fails to recover x uniquely. Since the field's inception, many codes have been found that exhibit both constant information rate, and constant error rate, which, in a sense, is optimal. These codes all share the property that to recover even a small portion of the message x from the codeword y, the receiver must decrypt the entire codeword. In [1], Katz and Trevisan posed the question: can codes be found in which a single bit of the message can be recovered by decoding only a small number of bits from the codeword? Codes of this type are called *locally-decodable*, and would be immensely useful in encoding large amounts of data which only need to be recovered in small portions, for example any kind of database or archive. Currently the best known locally-decodable codes are due to Yekhanin [2]; they can tolerate a constant error rate, but achieve only slightly better than exponentially small information rates[3].

In 1994, Lipton examined the notion of error-correction in the computationally bounded channel model [3]. In this model, errors are not introduced in codewords at random, but in a worst case fashion *by a computationally bounded adversary* who can corrupt up to a constant fraction of the entire codeword. This realistic restriction on the power of the channel allowed for the introduction of cryptographic tools into the problem of error correction. In Lipton [3] and Gopalan, Lipton, Ding [4] it was shown how, assuming a shared private key, one can use hidden permutations to achieve improved error correcting codes in the private key setting. Recently, Micali, Peikert, Sudan and Wilson used the computationally bounded channel model to show how existing error correcting codes could be improved in the public-key setting [5]. After seeing the dramatic improvement of error-correcting codes in the computationally bounded channel model, a natural question then becomes whether locally-decodable codes can also be improved in this model.

The first progress in this setting was by Ostrovsky, Pandey and Sahai [6], where they construct a constant information-rate, constant error-rate locally-decodable code in the case where the sender and receiver share a private key. This left open the question whether the same can be accomplished in the Public-Key setting, which does not follow from their results. Indeed, a naïve proposal (that does not work) would be to encrypt the key needed by [6] separately and then switch to the private-key model already solved by [6]. This however leaves unresolved the following question: how do you encrypt the private key from [6] in a locally-decodable fashion? Clearly, if we allow the adversary to corrupt a constant fraction of all the bits (including encryption of the key and the message), and we encrypt the key separately, then the encryption of the key must consume a constant fraction of the message, otherwise it can be totally corrupted by an Adversary. But if this is the case all hope for local decodability is lost. Another

[3] Yekhanin achieves codewords of size $2^{n^{1/\log\log n}}$ for messages of length n, assuming there exist infinitely many Mersenne primes.

suggestion is to somehow hide the encryption of the key inside the encryption of the actual message, but it is not clear how this can be done.

A more sophisticated, but also flawed, idea is to use Lipton's code-scrambling approach [3]. In his paper, Lipton uses a private shared permutation to "scramble" the code and essentially reduce worst-case error to random error. A first observation is that we can use PIR to implement a random permutation in the public-key setting. We would then proceed as follows: the receiver would generate a random permutation $\sigma \in S_r$, and the receiver's public key would be a set of PIR queries $Q_1, \ldots, Q_r$, where Q_i is a PIR query for the $\sigma(i)$th block of an r block database, using some known PIR protocol. The sender would then break their message x into blocks, $x_1, \ldots, x_r$, apply standard error correction to each block, calculate the $Q_1, \ldots, Q_r$ on their message, apply standard error correction to each PIR response $R_i = Q_i(\mathsf{ECC}(x))$, and send the message $\mathsf{ECC}(R_1), \ldots, \mathsf{ECC}(R_r)$. If ECC and PIR have constant expansion rates, as is the case with many ECCs and the Gentry-Ramzan PIR [7], the resulting code has only constant expansion rate. But an adversary can still destroy a single block, by focusing damage on a single PIR response. If we add redundancy by copying the message c times, and publishing cr PIR queries, the adversary can still destroy a block with non-negligible probability by destroying constant number of blocks at random, and with non-negligible probability the adversary will destroy all c responses corresponding to the same block, and the information in that block will be lost. Recall that we demand that no bit of information should be destroyed except with negligible probability. Hence this method does not work. Of course, this can be fixed by increasing the redundancy beyond a constant amount, but then the codeword expansion becomes more than constant as does the public key size. Thus, this solution does not work, and new ideas are needed. Indeed, in this paper, we use PIR to implement a hidden permutation, but we achieve a PKLDC which can recover from constant error-rate with only *constant* ciphertext expansion.

1.1 Previous Work

The first work on error correction in the computationally bounded channel model was done by Lipton in [3]. In Lipton [3] and Gopalan, Lipton, Ding [4] it was shown how to use hidden permutations to achieve improved error correcting codes in the private key setting. In [5], Micali, Peikert, Sudan and Wilson demonstrate a class of binary error correcting codes with positive information rate, that can uniquely decode from $\frac{1}{2} - \epsilon$ error rate, under the assumption that one-way functions exist. These codes decode from an error rate *above* the proven upper bound of $\frac{1}{4} - \epsilon$ in the (unbounded) adversarial channel model. The first application of the computationally bounded channel to Locally Decodable Codes was given by Ostrovsky, Pandey and Sahai [6], although their work was in the private-key setting, and does not extend to the public-key setting.

In addition to extending the work in the computationally bounded channel model, our work draws heavily from the field of Computational Private Information Retrieval (PIR). The first computational PIR protocol was given by

Ostrovsky and Kushilevitz [8], and since then there has been much progress. For a survey of work relating to computational PIR see [9].

1.2 Our Results

In this paper, we present a general reduction from semantically-secure encryption and a PIR protocol to a Public Key Encryption system with local decodability (PKLDC). We also present a general reduction from any homomorphic encryption to a PKLDC. In §5 we present the first Locally Decodable Code with constant information-rate which does not require the sender and receiver to share a secret key. To achieve this, we work in the Computationally Bounded Channel Model, which allows us to use cryptographic tools that are not available in the Adversarial Channel Model. Our system presents an improvement in communication costs over the best codes in the information-theoretic setting. We create codes with constant information-rate, as compared with the best known locally decodable codes [2] in the information-theoretic setting which have an almost exponentially small information rate.

Informally, our results can be summarized as follows,

Main Theorem (informal). *Given a computational PIR protocol with query size $|Q|$, and response size $|R|$ which retrieves dk bits per query, and a semantically-secure encryption scheme, there exists a Public Key Locally Decodable Code which can recover from a constant error-rate in the bits of the message, which has public key size $\mathcal{O}(n|Q|/(dk^2)+k)$ and ciphertexts of size $\mathcal{O}(n|R|/(dk^2))$, where n is the size of the plaintext and k is the security parameter. The resulting code has locality $\mathcal{O}(|R|k/d)$, i.e. to recover a single bit from the message we must read $\mathcal{O}(|R|k/d)$ bits of the codeword.*

Combining the main theorem with the general reduction from homomorphic encryption to PIR, we obtain

Corollary 1. *Under any homomorphic encryption scheme which takes plaintexts of length m to ciphertexts of length αm, there is a Public-Key Locally Decodable Code which can recover from a constant error-rate in the bits of the message, with public key size $\mathcal{O}(nk\beta\sqrt[\beta]{n})$ and ciphertexts of size $\mathcal{O}(n\alpha^{\beta-1}k)$, for any $\beta \in \mathbb{N}$, where n is the size of the plaintext and k is the security parameter. The resulting code has locality $\mathcal{O}(\alpha^{\beta-1}k^2)$, i.e. to recover a single bit from the message we must read $\mathcal{O}(\alpha^{\beta-1}k^2)$ bits of the codeword.*

We can further improve efficiency if we have a Length-Flexible Additively Homomorphic Encryption like Dåmgard-Jurik [10], using this cryptosystem we obtain

Corollary 2. *Under the Decisional Composite Residuousity Assumption [11] there is a Public-Key Locally Decodable Code which can recover from a constant error-rate in the bits of the message, with public key size $\mathcal{O}(n\log^2(n)+k)$ and ciphertexts of size $\mathcal{O}(n\log(n))$, where n is the size of the plaintext and k is the security parameter. The resulting code has locality $\mathcal{O}(k^2\log(n))$, i.e. to recover a single bit from the message we must read $\mathcal{O}(k^2\log(n))$ bits of the codeword.*

We also give a specific construction of a system based on the Φ-hiding assumption first introduced by Cachin, Micali and Stadler in [12], and later used by Gentry and Ramzan in [7]. Under this assumption we obtain

Corollary 3. *Under the Small Primes Φ-Hiding Assumption there is a Public-Key Locally Decodable Code which can recover from a constant error-rate in the bits of the message, with public key size $\mathcal{O}(n)$ and ciphertexts of size $\mathcal{O}(n)$, where n is the size of the plaintext and k is the security parameter. The resulting code has locality $\mathcal{O}(k^2)$, i.e. to recover a single bit from the message we must read $\mathcal{O}(k^2)$ bits of the codeword.*

Note that in full generality, our main result requires two assumptions, the existence of a PIR protocol and a semantically-secure encryption protocol. In practice, however, two separate assumptions are usually not needed, and all the corollaries apply under a single hardness assumption.

Our construction does have a few disadvantages over the information-theoretic codes. First, our channel is computationally limited. This assumption is fairly reasonable, but it is also necessary one for any type of public key encryption. In [5], Micali et al. show that if a true adversarial channel exists, which can always introduce errors in a worst-case fashion, then one-way functions cannot exist. Second, our code has a larger "locality" than most information-theoretic codes. For example, in Yekhanin's Codes, the receiver is only required to read three letters of the codeword to recover one letter of the message. In our code in §5 the receiver must read $\mathcal{O}(k^2)$ bits to recover 1 bit of the plaintext, where k is the security-parameter. It should be noted, however, that to maintain the semantic security of the cryptosystem, the receiver must read $\omega(\log k)$ bits to recover any single bit of the message. It is an interesting question whether the locality of our code can be reduced from $\mathcal{O}(k^2)$ to $\mathcal{O}(k)$. For long messages (i.e. $n = \omega(k^{2+\epsilon})$) our code still presents a very significant improvement in locality over standard error correcting codes.

2 Computationally Locally Decodable Codes

2.1 Modelling Noisy Channels

When discussing error correcting, or locally-decodable codes, it is important to consider how the errors are introduced by the channel. While it may be natural to assume the errors are introduced "at random", small changes in the exact nature of these errors can result in substantial changes in the bounds on the best possible codes.

The first definition of a noisy channel is due to Claude Shannon [13]. Shannon defined the *symmetric channel* where each message symbol is independently changed to a random different symbol with some fixed probability, called the error rate. An alternative definition of a noisy channel is Hamming's *adversarial channel*, where one imagines an adversary corrupting bits of the message in a worst-case fashion, subject only to the total number of bits that can be corrupted per block.

In 1994, Lipton [3] observed that the adversarial channel model assumes that the adversarial channel itself is computationally unbounded. In that paper, Lipton proposed a new model of *computationally bounded noise*, which is similar to Hamming's adversarial channel, except the adversary is restricted to computation which is polynomial in the block length of the code. This restriction on the channel's ability to introduce error is a natural one, and it is implied by the existence of any one-way function [5]. Throughout this paper, we use Lipton's model.

2.2 Definitions

We use the standard definition of computational indistinguishability for public key encryption, where we also view the size of the plaintext as a function of the security parameter. That is, we set the plaintext x to be of length k^α, where k is the security parameter and $\alpha > 1$.

The primary difference between our definition and the standard definition of semantic security is the local decodability property of the cryptosystem. Roughly, this says that given an encryption c of a message x, and a corrupted encryption c' such that the hamming distance of c and c' is less than $\delta|c|$, the time it takes the decoder to decode any bit x_i of the plaintext x from c' is much shorter than the length of the message, and does not increase as the message length increases.

Definition 1. *We call a* Public Key Cryptosystem semantically-secure (in the sense of indistinguishability) and δ-computationally locally-decodable *if there is a triple of probabilistic polynomial-time algorithms* (G, E, D)*, such that for all* k *and for all* α *sufficiently large*

- $(PK, SK) \leftarrow G(1^k, \alpha)$,
- $c \leftarrow E(PK, x, r)$ *(where* $|x| = k^\alpha$ *is a plaintext message of length polynomial in* k*, and* r *is the randomness of the encryption algorithm);*
- $b' \leftarrow D(SK, c', i)$

so that for all probabilistic polynomial-time adversaries A, A'*:*

$$\Pr[(PK, SK) \leftarrow G(1^k, \alpha); \{x^0, x^1, \gamma\} \leftarrow A(PK); A'(E(PK, x^b, r), \gamma) = b] < \frac{1}{2} + \nu(k),$$

where x^0 *and* x^1 *must both be of length* k^α*, and the probability is taken over the key generation algorithm's randomness,* b*, randomness* r *used in the encryption algorithm* E *and the internal randomness of* A *and* A'*.*[4] *Furthermore, it is computationally, locally-decodable. That is, for all probabilistic polynomial-time adversaries* A'' *and* A'''*,*

$$\begin{aligned} \Pr[(PK, SK) \leftarrow G(1^k, \alpha); (x, \gamma) \leftarrow A''(PK);& \\ c \leftarrow E(PK, x, r); \{c', i\} \leftarrow A'''(c, \gamma) :& \\ D(SK, c', i) = x_i] > 1 - \nu(k),& \end{aligned}$$

[4] As is standard practice, we allow the adversary A to pass state information γ, which could include information about the plaintexts x^0, x^1, which might be of use in determining which plaintext is encrypted by $E(PK, x^b, r)$.

where x_i denotes the ith bit of x, x must be of the length k^α, c' and c must be of the same length and the hamming distance between c' and c is at most $\delta|c|$, and where the probability is taken over the key generation algorithm's randomness, the randomness r used in the encryption algorithm E and the internal randomness of both A'' and A'''. The information-rate is $\frac{|m|}{|c|}$ and we call the decryption algorithm locally-decodable *if its running time is $o(k^\alpha)$, and the* efficiency *of the local decodability is measured as a function of k and α.*

3 Building Blocks

Our construction relies on a number of standard cryptographic tools and for completeness we briefly review them here.

3.1 Private Information Retrieval

A computational Private Information Retrieval protocol (PIR) is a protocol in which a user or client wants to query a position from a database, while keeping the position queried hidden from the server who controls the database. In particular the user generates a decryption key D_{PIR}, picks a position j and generates a query Q_j. Then, given Q_j, the server who has a database (or message) x, can execute query Q_j on x and obtain a response R_j. The privacy requirement is that the server cannot guess the position j with probability noticeably greater than random. The correctness requirement is that given D_{PIR}, and R_j the user can correctly recover the jth position of the message x. The efficiency of a PIR protocol is measured in the communication complexity, i.e. the sizes of Q and R. Currently, the most efficient PIR protocol is that of Gentry and Ramzan [7], which has $|Q| = |R| = \mathcal{O}(k)$ where k is a security parameter, and each query successfully retrieves approximately $k/4$ bits of the message x.

Formal definitions and concrete constructions of computational Private Information Retrieval protocols can be found in [8], [14], [12], [15] or [7].

3.2 Semantically-Secure Public Key Encryption

Our construction requires a semantically-secure encryption protocol, SSE. The only requirement we make on the protocol SSE, is that for a message x, $|\mathsf{SSE}(x)| = \mathcal{O}(|x|)$. For concreteness, we assume $|\mathsf{SSE}(x)| = c_1|x|$ for some constant c_1. This is achieved by many cryptosystems for example [11], [10], [16], [17], or the Φ-hiding based scheme in described §5.1.

To avoid making additional intractability assumptions, it is natural to choose a hardness assumption that yields both a semantically-secure encryption protocol as well as a PIR protocol. In practice this is almost always the case, for example Paillier's Cryptosystem [11] and Chang's PIR [15], or Gentry-Ramzan [7] (or Cachin-Micali-Stadler PIR [12]) and the encryption protocol outlined in Section 5.1. It is also worth noting that since [14] shows that any homomorphic encryption protocol immediately yields a PIR protocol, if we have a homomorphic encryption, we need not make an additional assumption to obtain a PIR protocol.

3.3 Reed-Solomon Codes

The Reed-Solomon Error Correcting Code (RS-ECC) works as follows: first we fix a prime p of length k, and all computations are done in the field $\mathbb{Z}/p\mathbb{Z}$. Then, given a plaintext x of length n, we represent x as a polynomial f_x of degree $n/k-1$ over $\mathbb{Z}/p\mathbb{Z}$. This can be done in many ways, perhaps the simplest is to break x into blocks of size k and view these as the coefficients of f_x. Then, the encoding of x is simply the evaluation of f_x at a number of points in $\mathbb{Z}/p\mathbb{Z}$. We need at least n/k evaluations to uniquely determine a polynomial of degree $n/k-1$, the RSECC adds redundancy by evaluating f_x at more points, $\mathsf{RSECC}(x) = (f_x(1), \ldots, f_x(\rho n/k))$ for some $\rho > 1$. For distinct plaintexts x, y, we have $f_x - f_y \neq 0$. Since a nonzero polynomial of degree $n/k-1$ has at most $n/k-1$ zeros, and $\mathsf{RSECC}(x)$ and $\mathsf{RSECC}(y)$ must have hamming distance at least $(\rho-1)n/k+1$, this code can recover from $(\rho-1)n/(2k)$ errors in the evaluation points, i.e. it can recover from an error rate of $\frac{1}{2} - \frac{1}{2\rho}$ in the digits of the code.

From now on we will view $\mathsf{RSECC}(x)$ as a $\rho n/k$-tuple which can be successfully decoded from an error rate of $\frac{1}{2} - \frac{1}{2\rho}$ in its digits.

3.4 Binary Error Correction

A desirable property of any error-correcting code is the ability to recover from a constant fraction of errors among the *bits* of the codeword. A drawback of many error-correcting codes, and locally-decodable codes, is that they are defined over large alphabets, and can only recover from a constant fraction of errors in the alphabet of the code. The natural alphabet of the RSECC described above is the field $\mathbb{Z}/p\mathbb{Z}$. In practice, all these codes are implemented on computers, where the natural alphabet is $\{0, 1\}$. Thus when we say that a code like the Reed-Solomon code can tolerate a constant fraction of errors, we mean a constant fraction of errors in their natural alphabet. In the Reed Solomon code, if one bit of each evaluation point is corrupted, there are no guarantees that the message will not be corrupted. Binary error correcting codes do exist, but they are generally not as efficient as codes over larger alphabets.

To allow our code to tolerate a constant fraction of errors in the *bits* of the ciphertext, we will make use of a binary error correcting code ECC, with two properties. First, $|\mathsf{ECC}(x)| = c_2|x|$ for some constant c_2, and second ECC can recover from an error-rate of $\frac{1}{2}-\delta$ in the *bits* of $\mathsf{ECC}(x)$. Such codes exist, for $\delta > \frac{1}{4}$ in the unbounded adversarial channel model, and $\delta > 0$ in the computationally bounded channel model. See the full version of this paper for a more in-depth discussion.

4 Construction

4.1 High Level Outline of Our Construction

A public key will be a list of t PIR queries $Q_1, \ldots, Q_t$, along with the public key to the semantically-secure encryption SSE. The private key will be the private key for the semantically-secure encryption, the private key for the PIR protocol and a permutation $\sigma \in S_t$ such that Q_j is a query for the $\sigma(j)$th position of the message. To encrypt an n-bit message X, we first divide X into r blocks $X_1, \ldots, X_r$, then we encrypt each block using our semantically-secure encryption (this can be done by further subdividing the block if necessary). Then we encode each block using the Reed-Solomon code, thus obtaining a list of evaluation points that constitute the Reed-Solomon encoding of this block. Next, we concatenate the evaluation points for all the blocks, and, treating this list as a single database, we evaluate all t PIR queries on it. Finally, we encode each PIR response with a standard binary error correcting code ECC.

In more detail, we assume that when we evaluate a PIR query Q on a message X, the PIR response R encodes dk bits of X where k is our security parameter and d depends on the specific PIR protocol used. For example the Gentry-Ramzan protocol has $d \approx \frac{1}{4}$, while a PIR protocol like [12] which only retrieves a single bit at a time has $d = 1/k$. Next, we fix a prime p of length k which will determine the base-field of the RSECC. Then, we set $r = n/(\ell k)$, thus each block X_i has $|X_i| = \ell k$, where ℓ is the parameter that will determine the "spread" of our code. Next we encrypt each block X_i using SSE, obtaining $\mathsf{SSE}(X_1), \ldots, \mathsf{SSE}(X_r)$ where $|\mathsf{SSE}(X_i)| = c_1 \ell k$. Then we encode each encrypted block as $c_1 \rho \ell$ field elements in $\mathbb{Z}/p\mathbb{Z}$ using RSECC. Thus we can recover any block X_i as long as no more than $\frac{1}{2} - \frac{1}{2\rho}$ of the field elements that encode it are corrupted. Finally, we concatenate all $c_1 r \rho \ell$ field elements, thus at this point our "database" is $c_1 r \rho \ell k = c_1 n \rho$ bits. Next we evaluate all t queries $Q_1, \ldots, Q_t$ on this database. Since we wish to retrieve *all* the information in X, we need $t = c_1 n \rho/(dk)$. Thus we obtain t PIR responses $R_1, \ldots, R_t$. Finally, we send the t-tuple $(\mathsf{ECC}(R_1), \ldots, \mathsf{ECC}(R_t))$.

Thus our final encryption is of size $c_1 c_2 n \rho |R_j|/(dk)$. If $|R_j| \approx k$ as is case in [12], [15], [7], then our encryption will be of length $c_1 c_2 \rho n/d$. If we use the PIR protocol in [7] then, d will be constant, thus our code will have constant information rate. Notice that the spread parameter ℓ has no effect on the length of the encryption. This encryption is error correcting because as long as no more than $\frac{1}{2} - \frac{1}{2\rho}$ of the responses that encode a given block are corrupted, the block can be recovered correctly by first decoding each point using ECC, and then reconstructing the block using the RSECC. This cryptosystem is also locally-decodable since to decrypt a given block, it suffices to read the $\frac{c_1 \rho \ell}{dk}$ PIR responses that encode it.

4.2 Error Correcting Public Key Encryption

We now define a triple of algorithms G, E, D for our encryption scheme.

Key Generation: $G(1^k, \alpha)$.

- Fix a prime p of length k.
- Generate public-key private-key pair for SSE, PK_E, SK_E.
- Generate a PIR decryption key D_{PIR}.
- Generate a random permutation $\sigma \in S_t$.
- Generate t PIR queries $Q_1, \ldots, Q_t$, where Q_j queries the block of dk bits at position $(\sigma(j)-1)c_1dk + 1$ of a $c_1n\rho$ bit database.

The public key will then be

$$PK = (PK_E, Q_1, \ldots, Q_t)$$

and the secret key will be

$$SK = (\sigma, SK_E, D_{PIR})$$

Thus the public key will be of length $t|Q| + |SK_E| = c_1n\rho|Q|/(dk)$. If we use [7], then $|Q| = k$ and d is constant, so assuming $|SK_E| = \mathcal{O}(k)$, we obtain $|PK| = \mathcal{O}(n+k)$.
Encryption: given an n-bit message X,

- Break X into $r = \frac{n}{\ell k}$ blocks X_i of size ℓk.
- Encrypt each block using SSE. If SSE can only encrypt strings of length k, we simply divide X_i into shorter strings, encrypt the shorter strings and then concatenate the encryptions.
- For each encrypted block, $\mathsf{SSE}(X_i)$ we encode it as a list of $c_1\rho\ell$ field elements $Z_{i,1}, \ldots, Z_{i,c_1\rho\ell}$ in $\mathbb{Z}/p\mathbb{Z}$ using the RSECC.
- Concatenate all the evaluations, creating $\tilde{X} = Z_{1,1}, \ldots, Z_{1,c_1\rho\ell}, \ldots, Z_{r,1}, Z_{r,c_1\rho\ell}$. Thus $|\tilde{X}| = rc_1\rho\ell k = c_1n\rho$ bits, and we run each PIR query $\{Q_1, \ldots, Q_t\}$ on $\tilde{X}$ receiving responses $R_1, \ldots, R_t$. Since each PIR query recovers dk bits, we will need c_1/d queries to recover each field element Z.
- Encode each R_j individually using the binary error correcting code ECC.
- The encryption is then the t-tuple $(\mathsf{ECC}(R_1), \ldots, \mathsf{ECC}(R_t))$.

Decryption: to recover the ith block, of a message X from the t-tuple $(\mathsf{ECC}(R_1), \ldots, \mathsf{ECC}(R_t))$

- We wish to retrieve the encoding $Z_{i,1}, \ldots, Z_{i,c_1\rho\ell}$, which are the bits of $\tilde{X}$ in positions $(i-1)c_1\rho\ell/d + 1, \ldots, ic_1\rho\ell/d$, Thus we select the $c_1\rho\ell/d$ responses that encode X_i, $\{\mathsf{ECC}(R_{\sigma^{-1}((i-1)c_1\rho\ell/d+1)}), \ldots, \mathsf{ECC}(R_{\sigma^{-1}(ic_1\rho\ell/d)})\}$.
- Decode each $\mathsf{ECC}(R_j)$ to obtain $\{R_{\sigma^{-1}((i-1)c_1\rho\ell/d+1)}, \ldots, R_{\sigma^{-1}(ic_1\rho\ell/d)}\}$.
- Decode each of the $c_1\rho\ell/d$ PIR responses R_j to obtain $Z_{i,1}, \ldots, Z_{i,c_1\rho\ell}$.
- Using the RSECC reconstruct $\mathsf{SSE}(X_i)$ from $Z_{i,1}, \ldots, Z_{i,c_1\rho\ell}$.
- Decrypt $\mathsf{SSE}(X_i)$.

Notice that to recover block X_i we only need to read $c_1c_2|R|\rho\ell/d$ bits of the encryption. In the Gentry-Ramzan PIR $|R| = k$ and $d = 1/4$, so we are reading only $\mathcal{O}(\ell k)$ bits of the message. For correctness we will choose $\ell = k$, thus in this case our scheme will achieve locality $\mathcal{O}(k^2)$.

4.3 Local-Decodability

One of the most interesting features of our construction is the local-decodability. To recover a small portion of the message X, only a small portion of the ciphertext $(\mathsf{ECC}(R_1), \ldots, \mathsf{ECC}(R_t))$ needs to be decoded. During encryption the message X is broken into blocks of length ℓk bits, and this is the smallest number of bits that can be recovered at a time. To recover a single bit of X, or equivalently the entire block X_i that contains it, we must read $c_1\rho\ell/d$ blocks of the ciphertext $\{\mathsf{ECC}(R_{\sigma^{-1}((i-1)c_1\rho\ell/d+1)}), \ldots, \mathsf{ECC}(R_{\sigma^{-1}(ic_1\rho\ell/d)})\}$. Since $|\mathsf{ECC}(R_j)| = c_2|R_j|$, we must read a total of $c_1c_2|R|\rho\ell/d$ bits. Since the probability of error will be negligible in ℓ, we will set $\ell = k$. Here c_2 and ρ are parameters that determine the error-rate of our code.

Using the Gentry-Ramzan PIR, we have $|R| = k$ and $d = 1/4$, so the locality is $\mathcal{O}(k^2)$. Using the Chang's PIR [15] based on Paillier's cryptosystem [11] we have $|R| = 2k$ and $d = 1/2$ so we achieve the same encryption size and locality, although in this situation the public key size is $\mathcal{O}(n^{3/2})$ instead of $\mathcal{O}(n)$ in the Gentry-Ramzan case.

4.4 Proof of Security

The semantic security of our scheme follows immediately from the semantic security of the underlying encryption SSE. The full proof of the correctness (i.e. local decodability) of our scheme requires some care. The formal proof can be found in the full version of this paper. Here, we outline only the high-level ideas of the proof. The structure of the proof is as follows. Given an encryption $(\mathsf{ECC}(R_1), \ldots, \mathsf{ECC}(R_t))$, the outer ECC forces an adversary to concentrate their errors among only a few R_j. Thus, we may assume that the adversary is only allowed to introduce errors into a constant fraction of the R_j. Then, we note that any polynomial-time adversary cannot tell which remainders R_j encode which block X_i by the privacy of the PIR protocol. Thus any errors introduced in the R_j will be essentially uniform among the Z's that make up the Reed-Solomon encryptions. Next, we show that our code has sufficient "spread" so that errors introduced uniformly among the R_j will cluster on the R_j encoding a given block X_i with only negligible probability. Finally, if the errors are not clustered among the R_j that encode a given block, we show that the RSECC will correctly recover that block.

Thus we arrive at the following result

Main Theorem. *Given a computational PIR protocol with query size $|Q|$, and response size $|R|$ which retrieves dk bits per query, and a semantically-secure encryption protocol* SSE*, there exists a Public Key Locally Decodable Code which can recover from a constant error-rate in the bits of the message, which has public key size $\mathcal{O}(n|Q|/(dk^2) + k)$ and ciphertexts of size $\mathcal{O}(n|R|/(dk^2))$, where n is the size of the plaintext and k is the security parameter. The resulting code has locality $\mathcal{O}(|R|k/d)$, i.e. to recover a single bit from the message we must read $\mathcal{O}(|R|k/d)$ bits of the codeword.*

4.5 Extensions

For convenience, in our proof of correctness, we set the parameter ρ equal to $1/2$. It should be clear that this value is somewhat arbitrary and that by increasing ρ we increase the error tolerance of the code along with the ciphertext expansion. Similarly, in our proof we set the parameter ℓ to be the security parameter k. We can change ℓ, and an increase in ℓ corresponds to a decrease in the probability that the channel succeeds in introducing an error, and a decrease in the locality of the code. In particular our code fails with probability that is negligible in ℓ, and the smallest number of bits that can be recovered from the message is $\mathcal{O}(\ell k)$.

Our protocol also benefits nicely from the idea of Batch Codes [18]. Since our protocol requires making multiple PIR queries to the same message, this is an ideal application of Batch Codes, which can be used to amortize the cost of making multiple PIR queries to a fixed database. By first "batching" the message $\tilde{X}$ in §4.2, we can significantly decrease server computation by slightly increasing ciphertext expansion, or we can decrease ciphertext expansion by paying a slight increase in server computation. It should be noted that batch codes are perfect, in the sense that batching the message in this way does not change the probability of correctness.

We can also increase the efficiency of our construction by further taking advantage of the bounded channel model. If in addition to the sender knowing the receiver's public key, we assume that the receiver knows the verification key to the senders signature algorithm (a reasonable assumption since anyone receiving messages from the sender should be able to verify them), our scheme benefits nicely from the sign and list-decode methods described in [5]. The use of digital signatures before applying the RSECC or the binary ECC has the effect of increasing the maximum tolerable error-rate, and decreasing the codeword expansion. Unlike the application of Batch Codes above, this sign and list-decode technique will slightly increase the probability that a message fails to decrypt, although it still remains negligible.

4.6 Constructions Based on Homomorphic Encryption

It was shown in [14] that any homomorphic encryption protocol yields a PIR protocol, thus our construction can be achieved based on any homomorphic encryption protocol. In this situation, it is unnecessary to first encrypt each block X_i before applying the RSECC since the PIR protocol described in [14] is already semantically-secure. Thus the idea of coupling encryption and error-correction is even more natural in this situation. Using the construction in [9] to construct a PIR protocol from a homomorphic encryption protocol and then applying our construction yields

Corollary 1. *Under any homomorphic encryption protocol which takes plaintexts of length m to ciphertexts of length αm, there is a Public-Key Locally Decodable Code which can recover from a constant error-rate in the bits of the* *message, with public key size $\mathcal{O}(nk\beta\sqrt[\beta]{n})$ and ciphertexts of size $\mathcal{O}(n\alpha^{\beta-1}k)$, for any $\beta \in \mathbb{N}$, where n is the size of the plaintext and k is the security parameter. The resulting code has locality $\mathcal{O}(\alpha^{\beta-1}k^2)$, i.e. to recover a single bit from the message we must read $\mathcal{O}(\alpha^{\beta-1}k^2)$ bits of the codeword.*

Using a Length-Flexible Additively Homomorphic Encryption protocol such as the one described in [10] yields an even more efficient PIR protocol. Using the methods outlined in [9] and applying our construction we arrive at the following result

Corollary 2. *Under the Decisional Composite Residuousity Assumption [11] there is a Public-Key Locally Decodable Code which can recover from a constant error-rate in the bits of the message, with public key size $\mathcal{O}(n\log^2(n)+k)$ and ciphertexts of size $\mathcal{O}(n\log(n))$, where n is the size of the plaintext and k is the security parameter. The resulting code has locality $\mathcal{O}(k^2\log(n))$, i.e. to recover a single bit from the message we must read $\mathcal{O}(k^2\log(n))$ bits of the codeword.*

5 A Concrete Protocol Based on Φ-Hiding

We now present a concrete example of our reduction based on the Gentry-Ramzan [7] PIR protocol. A straightforward application of our main construction in §4.2 already yields a PKLDC with public key size $\mathcal{O}(n)$ and constant ciphertext expansion, but the Gentry-Ramzan PIR protocol has many nice properties which can be exploited to simplify the construction and further increase the efficiency of the protocol. The construction we present here differs from the straightforward application of our general reduction to the Gentry-Ramzan protocol in two ways. First, we are able to integrate the basic semantically-secure encryption protocol into our construction, thus reducing the ciphertext expansion by a constant factor, and eliminating the need for another hardness assumption. Second, we use the Chinese Remainder Theorem Error Correcting Code (CRT-ECC) instead of the Reed-Solomon code used in the general construction. This is because the Φ-hiding assumption allows us to do hidden chinese-remaindering, and so it is a more natural code to use in this context. This does not change the arguments in any substantial way, since from the ring-theoretic perspective, the CRT-ECC and the Reed-Solomon ECC are exactly the same.[5]

5.1 A Φ-hiding based Semantically-Secure Encryption Protocol

Here, we describe a simple semantically-secure public key encryption scheme, BasicEncrypt that will be an essential building block of our construction. The encryption protocol consists of three algorithms, G, E, D described below.

To generate the keys, $G(1^k)$ first selects a small prime-power π, then generates $m \in \mathcal{H}_k^\pi$, i.e. $m = pq$, where $p, q \in_R \mathcal{P}_k$[6], subject to $\pi \bigm| p-1$. The public key

[5] See the full version for a more detailed discussion of this point.
[6] We use the notation $\in_R$ to denote selecting uniformly at random from a set.

will be $PK = (g, m, \pi)$ where g is a generator for the cyclic group G_m, and $SK = \frac{\varphi(m)}{\pi}$.

To encrypt a message $x \in \mathbb{Z}/\pi\mathbb{Z}$, we have

$$E(x) = g^{x+\pi r} \mod m,$$

for a random $r \in \mathbb{Z}/m\mathbb{Z}$. To decrypt, we do

$$D(y) = y^{\varphi(m)/\pi} = g^{x\varphi(m)/\pi \mod \varphi(m)} \mod m = \left(g^{\varphi(m)/\pi}\right)^x \mod m,$$

then, using the Pohlig-Hellman algorithm to compute the discrete logarithm in the group $\langle g^{\varphi(m)/\pi}\rangle$, we can recover $x \mod \pi = x$. If a is a small prime, and $\pi = a^c$, the Pohlig-Hellman algorithm runs in time $c\sqrt{a}$. Thus the decryption requires $\mathcal{O}(\log(m/\pi)+c\sqrt{a})$ group operations in G_m which is acceptable for small primes a. In our locally decodable code, we will require multiple different prime powers $\pi_1, \ldots, \pi_t$, and we will choose the small primes a, as the first primes, i.e. $\pi_1 = 5^{e_1}, \pi_2 = 7^{e_2}, \pi_3 = 11^{e_3}$. If we require t prime powers π_i, the Prime Number Theorem implies that the largest a will be approximately $t \log t$. Since t will be less than the message length, n, $\sqrt{a}$ will be polynomial in the message length, and hence polynomial in the security parameter k.

It is worth noticing that this scheme is additively homomorphic over the group $\mathbb{Z}/\pi\mathbb{Z}$, although we do not have an explicit use for this property. When $\pi = 2$, this is just Goldwasser-Micali Encryption [19], for larger π it was described in [20] and [21]. An extension of this scheme is described in [16].

While this protocol is not new, none of the previous descriptions of this protocol make use of the Φ-hiding assumption, and instead their security is based on some form of composite residuousity assumption, i.e. it is impossible to tell whether a random group element h belongs to the subgroup of order π in G_m. We are able to prove security under the Φ-hiding assumption because the Φ-hiding assumption is strictly stronger than these other assumptions. The proof that this protocol is semantically-secure under the Φ-hiding assumption is in the full version [22].

5.2 Outline of Our Φ-hiding based Construction

We begin by fixing a list of t prime powers $\{\pi_1, \ldots, \pi_t\}$ as part of the public parameters. For concreteness we choose $\pi_1 = 5^{e_1}$, $\pi_2 = 7^{e_2}$, ... as in §5.1. A public key will be a list of t RSA moduli $\{m_1, \ldots, m_t\}$, such that each m_j Φ-hides some prime power $\pi_{j'}$. The Private key will be the factorizations of the m_j, more specifically $\varphi(m_1), \ldots, \varphi(m_t)$, along with a random permutation $\sigma \in S_t$ such that m_j Φ-hides $\pi_{\sigma(j)}$. To encrypt a message $X \in \{0,1\}^n$, we first divide X into blocks X_i of size ℓk. Where k is the security parameter, and ℓ is a parameter determining the "spread" of the code. As in the Gentry-Ramzan PIR scheme, we view each block as a number in the range $\left\{0 \ldots 2^{\ell k}\right\}$. Our public key will be $t = \frac{\rho n}{dk}$ RSA moduli $\{m_1, \ldots, m_{\frac{\rho n}{dk}}\}$ such that each modulus Φ-hides a prime power π_j. We will use $s = \lceil \rho\ell/d \rceil$ of the π_j to encode each block X_i. Since there are $\lceil n/\ell k \rceil$ blocks, and for each block we use $\lceil \rho\ell/d \rceil$ prime powers, we use a total of $\frac{n}{\ell k} \cdot \frac{\rho\ell}{d} = \frac{\rho n}{dk} = t$ prime powers. The parameter ρ determines the redundancy of the CRT-ECC, hence increasing ρ increases the error tolerance and also the ciphertext expansion. Recall that d is the information rate of the Gentry-Ramzan PIR, so d is some fixed constant less than $1/4$, for concreteness we may assume $d = 1/5$. Exactly which prime is hidden by which modulus will be chosen at random at the time of key generation, and is part of the receiver's secret key. For each block X_i, the sender encrypts X_i modulo the s prime powers $\{\pi_{(i-1)s+1}, \ldots, \pi_{is}\}$, where each π_j is roughly of size dk. Notice here that we have used ρ times as many moduli π_j as necessary to encode each block, thus for each block X_i we have effectively calculated an encoding of X_i under the CRT-ECC which can tolerate $\left(\frac{1}{2} - \frac{1}{2\rho}\right)\frac{\ell}{d}$ corrupted moduli.[7] We do this for each block, and thus the resulting encryption is $\frac{\rho\ell}{d} \cdot \frac{n}{\ell k}$ residues. Since each residue is of size k, the encryption of the whole message is now $\frac{n}{\ell k}\frac{\rho\ell}{d} = \frac{\rho n}{dk}$ encryptions of size k. Finally, we encode each of the $\rho n/(kd)$ encryptions independently using the error correcting code in §3.4. So our final encryption is of size $\rho c_2 n/d$ bits, which is a constant multiple of n. This encryption is error correcting because as long as no more than $\frac{1}{2} - \frac{1}{2\rho}$ of the residues that encode a given block are corrupted, the block can be recovered correctly by first decrypting each residue, and then reconstructing the CRT-ECC. This cryptosystem is also locally-decodable since to decrypt a given block, it suffices to decrypt the $\frac{\rho\ell}{d}$ encryptions that encode it.

5.3 Error Correcting Public Key Encryption Based on Φ-hiding

We now define a triple of algorithms G, E, D for our encryption scheme.

Key Generation: $G(1^k, \alpha)$.

- Let $p_1, \ldots, p_t$ be primes with $5 \le p_1 < p_2 < \cdots < p_t$, and choose $e_j = \left\lfloor \frac{k}{4 \log p_j} \right\rfloor$, thus e_j is the largest integer such that $\log\left(p_j^{e_j}\right) < dk$, for some $d < \frac{1}{4}$. Set $\pi_j = p_j^{e_j}$. To encrypt n-bit messages, we will need to choose $t = \frac{\rho n}{dk}$. Since we assume $n = k^\alpha$, this becomes $t = \frac{\rho k^{\alpha-1}}{d}$.
- Generate a random permutation $\sigma \in_R S_t$, the symmetric group on t elements.
- Generate moduli $m_1, \ldots, m_t$ such that $m_j \in \mathcal{H}_k^{\pi_{\sigma(j)}}$, i.e. m_j Φ-hides $\pi_{\sigma(j)}$.
- Find generators $\{g_j\}$ of the cyclic groups $\{G_{m_j}\}$.

The public key will then be

$$PK = ((g_1, m_1, \pi_1), \ldots, (g_t, m_t, \pi_t)),$$

and the secret key will be

$$SK = \left(\sigma, \frac{\varphi(m_1)}{\pi_{\sigma(1)}}, \ldots, \frac{\varphi(m_t)}{\pi_{\sigma(t)}}\right).$$

[7] See the full version for a more in-depth discussion of the error tolerance of the CRT-ECC

Encryption: given an n-bit message X,

- Break X into $\frac{n}{\ell k}$ blocks X_i of size ℓk, and treat each X_i as an integer in the range $\{0 \dots 2^{\ell k}\}$.
- For block X_i, we will use the s prime powers $\pi_{(i-1)s+1}, \dots, \pi_{is}$ to encode X_i. Since the moduli $m_{\sigma^{-1}((i-1)s+1)}, \dots, m_{\sigma^{-1}(is)}$ that correspond to these π's is unknown to the sender, he must apply the Chinese Remainder Theorem using all the π_j's. Thus for each block X_i, using the CRT, the sender generates $\tilde{X}_i \in [1, \dots, (\pi_1 \cdots \pi_t)]$, such that

$$\tilde{X}_i = \begin{cases} X_i \mod \pi_j & \text{for } j \in [(i-1)s+1, \dots, is], \\ 0 \mod \pi_j & \text{for } j \in [1, \dots, (i-1)s] \cup [is+1, \dots, t]. \end{cases}$$

  To recover from error-rate $\frac{1}{2} - \frac{1}{2\rho}$, we set $s = \frac{\rho \ell}{d}$.
- The sender then sets $\tilde{X} = \sum_{i=1}^{\frac{n}{\ell k}} \tilde{X}_i$. Thus for each j, $\tilde{X} = X_i \mod \pi_{\sigma(j)}$ for the unique i such that $(i-1)s+1 \leq \sigma(j) \leq is$.
- For $j \in [1, \dots, t]$, generate a random $r_j \in \{0, \dots, \pi_1 \cdots \pi_t\}$.
- Then calculate $h_j = g_j^{\tilde{X} + r_j \pi_1 \cdots \pi_t} \mod m_j$ for each $j \in \{1, \dots, t\}$. Thus

$$h_j = E\left(\tilde{X} \mod \pi_{\sigma(j)}\right) = E(X_i \mod \pi_{\sigma(j)}),$$

  where $(i-1)s+1 \leq \sigma(j) \leq is$, and E is the encryption protocol described in §5.1. At this point, partial information about the block X_i is spread over s of the h_j's.
- Apply the binary Error Correcting Code ECC to each h_j individually.
- The encryption is then the t-tuple $(\mathsf{ECC}(h_1), \mathsf{ECC}(h_2), \dots, \mathsf{ECC}(h_t))$.

Decryption: to recover the ith block, of a message X from the t-tuple $(h_1, \dots, h_t)$

- Select the s encryptions that encode X_i, $\{\mathsf{ECC}(h_{\sigma^{-1}((i-1)s+1)}), \dots, \mathsf{ECC}(h_{\sigma^{-1}(is))}\}$.
- Decode each $\mathsf{ECC}(h_j)$ to find obtain $\{h_{\sigma^{-1}((i-1)s+1)}, \dots, h_{\sigma^{-1}(is)}\}$.
- Decrypt each of the s encryptions using the decryption algorithm from §5.1. This gives $a_1, \dots, a_s$ where $a_j = X_i \mod (\pi_{(i-1)s+j})$.
- Using the Chinese Remainder Code Decoding Algorithm, reconstruct X_i from the s remainders $a_1, \dots, a_s$. Note that if there are no errors introduced, this step can be replaced by simple Chinese Remaindering.

5.4 Analysis

The proof of security remains essentially the same as in the general setting.

For the locality, we note that to recover a single bit of X, or equivalently the entire block X_i that contains it, we must read s blocks of the ciphertext $\{\mathsf{ECC}(h_{\sigma^{-1}((i-1)s+1)}), \dots, \mathsf{ECC}(h_{\sigma^{-1}(is)})\}$. Since $|h_j| = k$ and $|\mathsf{ECC}(h_j)| = c_2 k$, we must read a total of $sc_2 k = \frac{\rho c_2 \ell k}{d}$ bits. Since the probability of error will be negligible in ℓ, we set $\ell \approx k$, and since $d < \frac{1}{4}$, we find that we need to read

$5c_2\rho k^2$ bits of the ciphertext to recover one bit of the plaintext, where c and ρ are parameters that determine the error-rate of our code. Thus our system only achieves local-decodability for $n = \Omega(k^{2+\epsilon})$. For $n \approx k^3$, our system already offers a significant improvement over standard error-correcting codes.

Thus we arrive at the following result

Corollary 3. *Under the Small Primes Φ-Hiding Assumption there is a Public-Key Locally Decodable Code which can recover from a constant error-rate in the bits of the message, with public key size $\mathcal{O}(n)$ and ciphertexts of size $\mathcal{O}(n)$, where n is the size of the plaintext and k is the security parameter. The resulting code has locality $\mathcal{O}(k^2)$, i.e. to recover a single bit from the message we must read $\mathcal{O}(k^2)$ bits of the codeword.*

References

1. Katz, J., Trevisan, L.: On the efficiency of local decoding procedures for error-correcting codes. In: STOC '00: Proceedings of the 32nd Annual Symposium on the Theory of Computing. (2000) 80–86
2. Yekhanin, S.: Towards 3-query locally decodable codes of subexponential length. In: Proceedings of the 39th ACM Symposiom on the Theory of Computinng (STOC). (2007)
3. Lipton, R.J.: A new approach to information theory. In: STACS '94: Proceedings of the 11th Annual Symposium on Theoretical Aspects of Computer Science, London, UK, Springer-Verlag (1994) 699–708
4. Gopalan, P., Lipton, R.J., Ding, Y.Z.: Error correction against computationally bounded adversaries. Manuscript (2004)
5. Micali, S., Peikert, C., Sudan, M., Wilson, D.A.: Optimal error correction against computationally bounded noise. In Kilian, J., ed.: TCC. Volume 3378 of Lecture Notes in Computer Science., Springer (2005) 1–16
6. Ostrovsky, R., Pandey, O., Sahai, A.: Private locally decodable codes. In: ICALP '07 : Proceedings of the 34th International Colloquium on Automata, Languages and Programming. Volume 4596 of Lecture Notes in Computer Science., Springer (2007) 387–298
7. Gentry, C., Ramzan, Z.: Single-database private information retrieval with constant communication rate. In: Automata, Languages and Programming. Volume 3580 of Lecture Notes in Computer Science., Springer Berlin / Heidelberg (2005) 803–815
8. Kushilevitz, E., Ostrovsky, R.: Replication is not needed: Single database, computationally-private information retrieval. In: IEEE Symposium on Foundations of Computer Science. (1997) 364–373
9. Ostrovsky, R., III, W.E.S.: A survey of single-database private information retrieval: Techniques and applications. In Okamoto, T., Wang, X., eds.: Public Key Cryptography. Volume 4450 of Lecture Notes in Computer Science., Springer (2007) 393–411
10. Damgård, I., Jurik, M.: A generalisation, a simplification and some applications of paillier's probabilistic public-key system. In: PKC '01: Proceedings of the 4th International Workshop on Practice and Theory in Public Key Cryptography, London, UK, Springer-Verlag (2001) 119–136

11. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Advances in Cryptology – EUROCRYPT '99. Volume 1592 of Lecture Notes in Computer Science., Springer Berlin / Heidelberg (1999) 223–238
12. Cachin, C., Micali, S., Stadler, M.: Computationally private information retrieval with polylogarithmic communication. In: Advances in Cryptology: EUROCRYPT '99. Volume 1592 of Lecture Notes in Computer Science., Springer Verlag (1999) 402–414
13. Shannon, C.E.: A mathematical theory of communication. Bell System Technical Journal **27** (1948) 379–343, 623–656
14. Ishai, Y., Kushilevitz, E., Ostrovsky, R.: Sufficient conditions for collision resistant hashing. In: Theory of Cryptography. Volume 3378., Springer Berlin / Heidelberg (2005) 445–456
15. Chang, Y.C.: Single database private information retrieval with logarithmic communication. In: Information Security and Privacy. Volume 3108 of Lecture Notes in Computer Science., Springer Berlin / Heidelberg (2004)
16. Naccache, D., Stern, J.: A new public key cryptosystem based on higher residues. In: CCS '98: Proceedings of the 5th ACM conference on Computer and communications security, New York, NY, USA, ACM Press (1998) 59–66
17. Gamal, T.E.: A public key cryptosystem and a signature scheme based on discrete logarithms. In: Proceedings of CRYPTO 84 on Advances in cryptology, New York, NY, USA, Springer-Verlag New York, Inc. (1985) 10–18
18. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Batch codes and their applications. In: STOC '04: Proceedings of the thirty-sixth annual ACM symposion the theory of computing, New York, NY, USA, ACM Press (2004) 373–382
19. Goldwasser, S., Micali, S.: Probabilistic encryption. Journal of Computer and System Sciences **28**(2) (1984) 270–299
20. Benaloh, J.D.C.: Verifiable secret-ballot elections. PhD thesis, Yale University (1987)
21. Benaloh, J.C.: Dense probabilistic encryption. In: Proceedings of the Workshop on Selected Areas in Cryptography. (1994) 120–128
22. Hemenway, B., Ostrovsky, R.: Public key locally decodable codes. http://eprint.iacr.org/2007/083/ (2007)