Public Key Encryption Which is Simultaneously a Locally-Decodable Error-Correcting Code

Brett Hemenway* Rafail Ostrovsky†

February 7, 2008

Abstract

In this paper, we introduce the notion of a Public-Key Encryption (PKE) Scheme that is also a Locally-Decodable Error-Correcting Code. In particular, our construction simultaneously satisfies all of the following properties:

- Our Public-Key Encryption is semantically secure under a certain number-theoretic hardness assumption (a specific variant of the Φ-hiding assumption).
- Our Public-Key Encryption function has *constant expansion*: it maps plaintexts of length n (for any n polynomial in k, where k is a security parameter) to ciphertexts of size $\mathcal{O}(n+k)$. The size of our Public Key is also linear, $\mathcal{O}(n+k)$.
- Our Public-Key Encryption is also a *constant rate* binary error-correcting code against any polynomial-time Adversary. That is, we allow a polynomial-time Adversary to read the entire ciphertext, perform any polynomial-time computation and change an arbitrary (i.e. adversarially chosen) constant fraction of *all* bits of the ciphertext. The goal of the Adversary is to cause error in decoding any bit of the plaintext. Nevertheless, the decoding algorithm can decode **all** bits of the plaintext (given the corrupted ciphertext) while making a mistake on *any* bit of the plaintext with only a negligible in k error probability.
- Our Decoding algorithm has a **Local Decodability** property. That is, given a corrupted ciphertext of $E(x)$ the decryption algorithm, for any $1 \leq i \leq n$ can recover the i'th bit of x (i.e., x_i) with overwhelming probability reading at most $\mathcal{O}(k^2)$ bits of the corrupted ciphertext and performing computation *polynomial in* k . Thus, for large plaintext messages (specifically for $|x| = \omega(k^{2+\epsilon})$), our Public Key Decryption algorithm can decode and error-correct any x_i computation sublinear in $|x|$.

We believe that the tools and techniques developed in this paper will be of independent interest in other settings.

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

*Department of Mathematics, University of California, Los Angeles. E-mail: bretth@math.ucla.edu

†Department of Computer Science and Department of Mathematics, University of California, Los Angeles 90095. E-mail: rafail@cs.ucla.edu, rostrovs@math.ucla.edu.

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. 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 the 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 [18], Katz and Trevisan posed the question: can codes be found in which a single bit of the message can be recovered by decrypting 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 needs 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 [30], they can tolerate a constant error rate, but achieve only slightly better than exponentially small information rates[1].

It was shown by Katz and Trevisan [18], that any information-theoretic Private Information Retrieval (PIR) scheme can be transformed into a locally-decodable code. While this provides a new approach to the problem of constructing efficient locally-decodable codes, so far it has not lead to any codes with significantly sub-exponential size codewords, as we are still unable to construct efficient information-theoretic Private Information Retrieval schemes.

In 1994, Lipton proposed to look at the question of error-correction in the computationally bounded channel model [20]. In this model, errors are not introduced in codewords at random, but in a worst case fashion *by a computationally bounded adversary*. This realistic restriction on the power of the channel allowed for the introduction of cryptographic tools into the problem of error correction. In [20] and [12] 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 shown how in the same computationally bounded model existing error correcting codes could be significantly improved in the public-key setting [24]. After seeing the dramatic improvement of error-correcting codes in this model, a natural question then becomes whether locally-decodable codes can also be improved in the computationally bounded channel model.

The first real progress in this setting was recently accomplished by Ostrovsky, Pandey and Sahai [26], where they showed how to 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 [26] separately and then switch to the private-key model already solved by [26]. This however leaves unresolved the following question: how do you encrypt the private key from [26] 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 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.

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

A somewhat more sophisticated, but also flawed, idea is to use Lipton's private-key permutation approach [20], where the permutation is implemented by publishing Private Information Retrieval (PIR) queries [19] as part of the public key to implement a random permutation. In this case, the receiver will generate a random permutation $\sigma \in S_n$, and the receiver's public key would be a set of PIR queries $Q_1, \ldots, Q_n$, where Q_i is a PIR query for the $\sigma(i)$th block of an n block database, using some known PIR protocol. The sender would then break their message x into blocks, $x_1, \ldots, x_n$, apply standard error correction to each block, calculate the $Q_1, \ldots, Q_n$ on their message, apply standard error correction to each PIR response $R_i = Q_i(\mathrm{ECC}(x))$, and send the message $\mathrm{ECC}(R_1), \ldots, \mathrm{ECC}(R_n)$. If ECC and PIR have constant expansion rates, as is the case with many ECCs and the Gentry-Ramzan PIR [9], 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 cn 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 naïve method does not work either. Of course, this can be fixed by increasing the redundancy beyond a constant amount, but then expansion becomes more than constant as does the public key. Thus, this solution does not work either, and a different approach is needed. Indeed, in this paper, we also use a private permutaiton, but we achieve a Public-Key Encryption scheme with constant information-rate and constant error-rate code with local decodability using a completely different approach.

Intuition Behind Our Construction. Recall that Gentry and Ramzan [9] described a computational PIR scheme that has a very nice algebraic property: the client picks a modulus which is hidden from the database, and constructs a PIR query. The server then responds to this query with an encryption of the database modulo this hidden modulus. If we change terminology to the language of public key encryption, we can view the client as the receiver, and the server as the sender. In this setting, the client publishes a number of PIR queries corresponding to different hidden moduli as his Public Key, the sender can execute these queries on his message and send the response to the receiver. It has long been known that the Chinese Remainder Theorem provides an efficient Error Correcting Code (CRT-ECC), see for example [10]. Thus if the sender sends his message modulo sufficiently many moduli, this demonstrates an error-correcting public-key cryptosystem. This simple construction is *not* locally decodable, and is no more efficient than taking a generic public key cryptosystem and encrypting a message and then encoding the encryption with a generic ECC. To achieve local decodability, we must exploit the fact that chinese remaindering is tightly integrated into Gentry-Ramzan PIR queries. Our approach is as follows: the receiver chooses a secret permutation of the moduli, and publishes PIR queries for these moduli as his public key. The Φ-hiding assumption assures that this permutation remains secret. Next, the sender divides his message into into a number of fairly "long" (but still sublinear) blocks, and executes the PIR queries in such a way that information about each block extends over many PIR responses, but also importantly that each PIR response contains partial information about many blocks of the message. It is this "spreading" of the information about each block which allows our code to achieve both constant message blowup and local decodability. The information in each block is spread over a large number of PIR responses. Which PIR responses encode which blocks is determined by the receivers permutation, and hence hidden from the sender (and the channel). By careful manipulation of the redundancy of the CRT-ECC and the length of the blocks and the exact spread of the code, we achieve a large alphabet locally decodable error-correcting code. Then using the ideas of concatenated codes, we extend this to a public key cryptosystem which is also locally decodable binary error-correcting code with asymptotically optimal rate.

1.1 Previous Work

The first work on error correction in the computationally bounded channel model was done by [20]. In [20] and [12] it was shown how to use hidden permutations to achieve improved error correcting codes in the private key setting. The computationally bounded channel model was first considered in the public key setting only recently. In [24], Micali et al used a generic public key signature scheme combined with list-decoding to 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. Here, again, we emphasize the reasonableness of the computationally bounded channel model, since under the assumption that one-way functions exist, Micali et al show that *all* channels (that don't hold the messages for an exponential amount of time) must be computationally bounded, or they could be used as inverters of the one-way function.

The first application of these ideas to Locally Decodable Codes was in [26], where Ostrovsky, Pandey and Sahai were able to produce private locally decodable codes with constant information rate, and able to recover from a constant error rate in the private key setting. This was a significant improvement over the best known locally-decodable codes in the unbounded adversarial channel model, due to Yekhanin, which require almost exponential size codewords to recover from constant error rate.

We also consider locally-decodable codes in the computationally bounded channel model, but in the *public key setting*. Specifically we demonstrate a cryptosystem that has constant ciphertext expansion, i.e. constant information rate, and is locally decodable from constant error rate in the computationally bounded channel model. Our construction uses PIR to implement a hidden permutation in the public key setting, effectively doing "code-scrambling" similar to [20]. As mentioned above, this does not immediately lead to a locally-decodable code in the public key setting. Instead, we use the underlying machinery of the Gentry-Ramzan PIR, specifically the fact that it allows Chinese Remaindering modulo hidden moduli and the fact that redundant chinese remaindering forms an efficient error-correcting code, this allows us to add redundancy in an efficient way, and create a code that is locally-decodable with only constant ciphertext expansion.

1.2 Our Results

In this paper, we present the first Public Key Encryption system with local decodability. Our system is also 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 a significant improvement in communication costs over the best codes in the information-theoretic setting. Yekhanin's Codes, described in [30], which are currently the shortest known locally decodable codes in the information-theoretic setting, still have codewords which are almost exponential in the message size, while our codewords are only a constant times larger than the message.

Our system has 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 [24], 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 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, our code still presents a very significant improvement in locality over standard error correcting codes. Third, our decryption algorithm is not particularly efficient. While it is true that decryption is fairly computationally intensive, it remains polynomial in the security parameter k. Throughout this paper we have focused presenting our algorithms clearly, and have not made an effort to optimize them where we have felt that it might be in conflict with the clarity of the exposition. We stress, however, that while the algorithms presented could undoubtedly be improved somewhat in efficiency, they are all computationally feasible as presented.

2 Preliminaries

2.1 Notation

This paper brings together a number of different cryptographic building blocks, which, unfortunately, have conflicting names and notations. To minimize confusion we use the following naming conventions for our variables.

- x will denote a plaintext message, which will usually be n bits in length.
- $\pi_i = p_i^{c_i}$ will denote a prime-power modulus used for Chinese Remaindering.
- m will denote an RSA modulus, which, i.e. a product of two approximately equal size primes, and m will usually be k bits in length.
- k will denote our security parameter.
- $\nu(k)$ will denote a function which is negligible in k.
- G_m will denote the largest cyclic subgroup of $(\mathbb{Z}/m\mathbb{Z})^*$. We will assume that $m = pq$ will always be an RSA modulus, and $\gcd((p-1)/2, (q-1)/2)) = 1$, this gives $|G_m| = \varphi(m)/2$.

We will use the notation $\in_R$, to denote an element drawn uniformly at random from a set.

2.2 The Small Primes Φ-Hiding Assumption

The Φ-Hiding Assumption is a relatively new computational hardness assumption, which relates to the difficulty of finding small prime factors of $\varphi(m)$, where φ is the Euler Totient Function. If a prime p divides $\varphi(m)$, we say that m Φ-hides p. The Φ-Hiding assumption was first proposed by Cachin, Micali and Stadler in [5], and a variant was proposed by Gentry and Ramzan in [9]. Our constructions require only the security of the Gentry-Ramzan PIR scheme, and so we make the following variant of the Φ-Hiding Assumption

Let $\mathcal{P}_k$ denote the set of primes of bit-length $\frac{k}{2}$, $\mathcal{H}_k$ be the set of products of two primes in $\mathcal{P}_k$ with $\gcd(p-1, q-1) = 2$, and let $\mathcal{H}_k^\pi \subset \mathcal{H}_k$ denote the set of composite moduli that Φ-hide π, i.e.

$$\mathcal{H}_k^\pi = \{m : m = pq,\ \{p, q\} \subset \mathcal{P}_k,\ \gcd(p-1, q-1) = 2,\ p \equiv 1 \mod \pi\}.$$

Assumption 1. The *Small Primes Φ-Hiding Assumption*

For all small prime powers, π_0, π_1 such that $3 < \pi_0 < \pi_1 < 2^{\frac{k}{4}-1}$, given $b \in_R \{0, 1\}$ and $m \in_R \mathcal{H}_k^{\pi_b}$, for all probabilistic polynomial-time algorithms A, we have

$$\Pr[A(\pi_0, \pi_1, m) = b] \leq \frac{1}{2} + \nu(k),$$

for some negligible function $\nu(k)$, where the probability is taken over all $m \in \mathcal{H}_k^{\pi_b}$, $b \in \{0, 1\}$, and the internal randomness of A.

Thus we are asserting that no probabilistic polynomial-time adversary can determine which prime power a given modulus Φ-hides. We will sometimes find it convenient to use a slightly different form. Specifically, we assert that given two moduli m_0, m_1 which Φ-hide two prime powers π_0, π_1, no probabilistic polynomial-time adversary can tell whether $\pi_0 = \pi_1$ with probability better than one half.

Lemma 1. Under the Small Primes Φ-Hiding Assumption, if $\pi_0 \in_R \{5, \ldots, \lfloor 2^{\frac{k}{4}-1} \rfloor\}$, $S_0 = \{\pi_0\}$, $S_1 = \{5, \ldots, \lfloor 2^{\frac{k}{4}-1} \rfloor\} \setminus \{\pi_0\}$, $b^* \in_R \{0,1\}$, $\pi_1 \in_R S_{b^*}$, $b \in_R \{0,1\}$ and $m_0 \in_R \mathcal{H}_k^{\pi_b}$ and $m_1 \in_R \mathcal{H}_k^{\pi_{1-b}}$. Then for all probabilistic polynomial-time adversaries A,

$$\Pr[A(m_0, m_1) = 0 \text{ and } \pi_0 = \pi_1] + \Pr[A(m_0, m_1) = 1 \text{ and } \pi_0 \neq \pi_1] \leq \frac{1}{2} + \nu(k),$$

for some negligible function $\nu(k)$, where the probability is taken over the internal randomness of A, the choice of π_0, π_1, m_0, m_1, and the choice of b^* and b.

Proof. Assume there exists a polynomial-time adversary A which can correctly determine whether $\pi_0 = \pi_1$ with probability $\frac{1}{2} + \epsilon(k)$ for some non-negligible function $\epsilon(k)$. Given π_0, π_1 and m such that $\pi_b \mid \varphi(m)$, we wish to construct an algorithm A' that guesses b, as follows: Pick a random $b' \in \{0,1\}$, and generate $m' \in \mathcal{H}_k^{\pi_{b'}}$. Then run A on (m, m'). If A returns 0 then A' returns b', otherwise A' returns $1 - b'$. Since A succeeds with probability $\frac{1}{2} + \epsilon(k)$, A' succeeds with probability $\frac{1}{2} + \epsilon(k)$ which is still non-negligible in k, and thus a violation of the Φ-Hiding assumption ■

In particular, we are asserting that there is no efficient algorithm which can match the π_i to the moduli m_i significantly better than by guessing randomly. Notice that in the small primes Φ-hiding assumption we have excluded $\pi = 2$ or 3, this is because every odd number Φ-hides 2, and $m \equiv 2 \mod 3$, only if m Φ-hides 3. Notice also that we restrict the π_i to be smaller than $\sqrt[4]{m_i}$, this is to prevent the lattice based attack described in [7], [6]. When the p_i's and the π_i's are chosen subject to these restrictions, there are no efficient algorithms known for breaking the Φ-Hiding assumption.

3 Computationally Locally Decodable Codes

3.1 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 [29]. 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. Most error correcting and locally-decodable codes were designed for Hamming's model. Lipton [20] observed in 1994 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. In fact, this is implied by our hardness assumption, since we show that any channel which introduces noise in a strictly worst-case fashion could be used to break the Φ-hiding assumption. Throughout this paper, we use Lipton's model.

3.2 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 *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'.[2] Furthermore, it is computationally, locally-decodable. That is, for all probabilistic polynomial-time adversaries A'' and A''',

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

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.

4 Construction

4.1 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.

[2] As is standard practice, to 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 $E(PK, x^b, r)$.

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 $p, q \in_R \mathcal{P}_k$, subject to $\pi \mid p-1$. The public key 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. $\pi_1 = 5^{c_1}, \pi_2 = 7^{c_2}, \pi_3 = 11^{c_3}$. 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 [11], for larger π it was described in [4] and [3]. An extension of this scheme is described in [25].

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 reduction is simple, for suppose there exists an adversary A which can determine whether a group element $h \in G_m$ is a πth power. Noticing that if $\pi \mid \varphi(m)$ exactly 1 in π elements will be πth powers, while if $\gcd(\pi, \varphi(m)) = 1$, then *every* element is a πth power, by simply sending random group elements h_i to A, and measuring the probability which A says that h_i is a πth power, we can distinguish whether $\pi \mid \varphi(m)$.

The proof that this protocol is semantically secure under the Φ-hiding assumption is left to Appendix B.

4.2 Binary Error Correction

A drawback of many error-correcting codes, and locally-decodable codes is that they are defined over large alphabets. 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 CRT ECC or Reed-Solomon codes can tolerate a constant fraction of errors, we mean a constant fraction of errors in their natural alphabet. In the CRT ECC, if one bit of each remainder 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

- $|\operatorname{ECC}(x)| = c|x|$ for some constant c,
- ECC can recover from an error-rate of $\frac{1}{2} - \delta$ in the *bits* of $\operatorname{ECC}(x)$.

Such codes exist, for $\delta > \frac{1}{4}$ in the unbounded adversarial channel model, and $\delta > 0$ in the computationally bounded channel model. See Appendix D for a more in-depth discussion.

4.3 High Level Outline of Our 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 $\pi_1 = 5^{c_1}$, $\pi_2 = 7^{c_2}$, ... as in §4.1. 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 $\{0 \ldots 2^{\ell k}\}$. 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 $\lceil n/\ell k \rceil$, 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 you can 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 (see Appendix E). 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 the encryption of the whole message is now of $\frac{n}{\ell k}\frac{\rho\ell}{d} = \frac{\rho n}{dk}$ encryptions of size k. Finally, we encode each of the $\frac{\rho n}{kd}$ encryptions independently using the error correcting code in §4.2. So our final encryption is of size $\frac{\rho c n}{d}$ 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.

4.4 Error Correcting Public Key Encryption

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

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

- Let $p_1, \ldots, p_t$ be primes with $5 \leq p_1 < p_2 < \cdots < p_t$, and choose $c_j = \left\lfloor \frac{k}{4 \log p_j} \right\rfloor$, thus c_j is the largest integer such that $\log\left(p_j^{c_j}\right) < dk$, for some $d < \frac{1}{4}$. Set $\pi_j = p_j^{c_j}$. 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

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

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 $\{0 \ldots 2^{\ell k}\}$.
- For block X_i, we will use the s prime powers $\pi_{(i-1)s+1}, \ldots, \pi_{is}$ to encode X_i. Since the moduli $m_{\sigma^{-1}((i-1)s+1)}, \ldots, m_{\sigma^{-1}(is)}$ 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 $\tilde{X}_i \in [1, \ldots, (\pi_1 \cdots \pi_t)]$, such that

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

  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 $j \in [1, \ldots, t]$, generate a random $r_j \in \{0, \ldots, \pi_1 \cdots \pi_t\}$.
- Then calculate $h_j = g_j^{\tilde{X} + r_j \pi_1 \cdots \pi_t} \mod m_j$ for each $j \in \{1, \ldots, t\}$. 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 §4.1. 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 $(\mathrm{ECC}(h_1), \mathrm{ECC}(h_2), \ldots, \mathrm{ECC}(h_t))$.

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

- Select the s encryptions that encode X_i, $\{\mathrm{ECC}(h_{\sigma^{-1}((i-1)s+1)}), \ldots, \mathrm{ECC}(h_{\sigma^{-1}(is)})\}$.
- Decode each $\mathrm{ECC}(h_j)$ to find obtain $\{h_{\sigma^{-1}((i-1)s+1)}, \ldots, h_{\sigma^{-1}(is)}\}$.
- Decrypt each of the s encryptions using the decryption algorithm from §4.1. This gives $a_1, \ldots, a_s$ 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 $a_1, \ldots, a_s$. Note that if there are no errors introduced, this step can be replaced by simple Chinese Remaindering.

We have introduced many parameters in the definition of this scheme, and their roles can be summarized as follows

n	the number of bits in the message X.
k	the security parameter.
α	the relation between n and k, $n = k^\alpha$.
t	the total number of moduli used. We will set $t = \frac{\rho n}{dk}$.
d	the fraction of bits that can be Φ-hidden, d is a fixed constant $d < \frac{1}{4}$.
ℓ	a parameter which affects the "spread" of the code. We will choose $\ell \approx k$.
s	the number of π_i encoding each block, for us $s = \frac{\rho \ell}{d}$.
c	is the expansion factor of the error correcting code ECC.
ρ	is the expansion factor of the CRT ECC we set $\rho = \frac{1}{2\delta}$, to recover from error-rate $\frac{1}{2} - \delta$.

4.5 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 $(\mathrm{ECC}(h_1), \ldots, \mathrm{ECC}(h_t))$ 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 s blocks of the ciphertext $\{\mathrm{ECC}(h_{\sigma^{-1}((i-1)s+1)}), \ldots, \mathrm{ECC}(h_{\sigma^{-1}(is)})\}$. Since $|h_j| = k$ and $|\,\mathrm{ECC}(h_j)| = ck$, we must read a total of $sck = \frac{\rho c \ell k}{d}$ 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 $5c\rho k^2$ 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 $n = \mathcal{O}(k^{2+\epsilon})$. For $n \approx k^3$, our system already offers a significant improvement over standard error-correcting codes. It should also be noted, that for any semantically-secure cryptosystem, to recover one bit of the plaintext, you must read at least $\omega(\log k)$ bits of the ciphertext. It is an interesting question whether the locality of such a scheme can be improved from $\mathcal{O}(k^2)$ to $\mathcal{O}(k)$.

4.6 Running Time

We now show that the three algorithms G, E, D probabilistic polynomial-time algorithms in the security parameter k.

We begin by analyzing G. Choosing the π_i and the permutation σ, can clearly be done in polynomial-time, so it only remains to show that generating $m_i \in \mathcal{H}_k^{\pi_i}$ can be done in polynomial-time. We can find primes $p \in \mathcal{P}_k$ in polynomial-time, by simply choosing random numbers and testing them for primality using an algorithm like the Rabin-Miller primality test [27] or the deterministic algorithm presented in [1]. The Prime Number Theorem tells us that the density of primes of length $\frac{k}{2}$ is asymptotic to $\frac{2}{k}$, so we expect to find a prime after only a polynomial number of guesses. A similar theorem, also proven by de la Valée Poussin, gives the density of primes of length $\frac{k}{2}$ in the arithmetic progression $1 + \pi n$ to be asymptotic to $\frac{2}{\varphi(\pi)k}$, so, in either case we expect to find a prime p such that $p \equiv 1 \mod \pi$ after $\mathcal{O}(k)$ guesses. This allows us to generate $m \in_R \mathcal{H}_k^\pi$ in polynomial time.

The encryption algorithm E will be polynomial-time in k, $\tilde{X}$ can be computed in polynomial-time using the Chinese Remainder Theorem, and $\log \tilde{X} \approx 2 \log X$. Thus the encryptions can $g^{\tilde{X} + r\pi_1 \cdots \pi_t}$ can be done in polynomial time using the square-and-multiply algorithm.

Finally, the decryption will be polynomial-time because decrypting each h_i to $\tilde{X} \mod \pi_{\sigma(i)}$ involves a single exponentiation, which can be done in polynomial-time via the square-and-multiply algorithm. Then recovering the block X_i is done via the polynomial-time CRT-ECC algorithm described in [15].

4.7 Proof of Security

The semantic security of our scheme follows immediately from the semantic security of the underlying encryption BasicEncrypt (see Appendix B.) The full proof of the correctness (i.e. local decodability) of our scheme requires some care. The formal proof can be found in Appendix C. Here, we outline only the high-level ideas of the proof. The structure of the proof is as follows. Given an encryption $(\mathrm{ECC}(h_1), \ldots, \mathrm{ECC}(h_t))$. The outer ECC forces an adversary to concentrate their errors among only a few h_i. Thus, we may assume that the adversary is only allowed to introduce errors into a constant fraction of the h_i. Then, we note that any polynomial-time adversary cannot tell which remainders h_i encode which block X_j by the Φ-hiding assumption. Thus any errors introduced in the h_i will be essentially uniform among the underlying CRT moduli π_i. Next, we note show that our code has sufficient "spread" so that errors introduced uniformly among the h_i will cluster on the h_i encoding a given block X_j with only negligible probability. Finally, if the errors are not clustered among the h_i that encode a given block, we show that the CRT code will correctly recover that block.

4.8 Extensions

First, it should be clear that choosing to encode each block by twice as many moduli as necessary was fairly arbitrary. By increasing the redundancy of the CRT ECC we would increase the error-tolerance, and also the ciphertext expansion, for different applications different parameters may be more desirable.

Second, 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 [24]. As in §4.4, we break our message X into ℓk-bit blocks $X_1, \ldots, X_t$. Then, before applying the CRT ECC to each block, we sign each block using any Public Key Signature Scheme which is existentially unforgeable under the chosen message attack. The existence of such a scheme is implied by the existence of a one-way function [28], which in turn is implied by the Φ-Hiding assumption. We can improve the efficiency of the digital signature by using the standard trick of first hashing the message, the signing the hash. Since every PIR protocol is a collision-resistant hash function [17], we can use the Gentry Ramzan PIR to first "hash" the ℓk bits in a block down to k bits, then sign the resulting k-bit block. Now, we proceed as before, encoding each signed block using the hidden CRT ECC, and finally each of these blocks is further encoded by a binary ECC. As mentioned above, the rate of the binary ECC can also be improved via this method. Again, we note that this construction requires the receiver to know the public key for the sender's signature scheme, in addition to the sender knowing the public key to the receiver's encryption scheme.

To decode in this situation, we first decode the binary ECC, then we *list-decode* the hidden CRT ECC using one of the list-decoding methods described in [10] or [15]. Then, with all but negligible probability, only one of the possible decodings will be a valid *signed* block. This has the effect of improving the information rate of the CRT ECC. It should be noted that our scheme has constant codeword expansion, and can recover from constant error-rate even without these improvements. The use of digital signatures before applying the CRT ECC or the binary ECC has the effect of increasing the maximum tolerable error-rate, and decreasing the codeword expansion.

References

[1] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. PRIMES is in P. *Annals of Mathematics*, 160:781–793, 2004.

[2] Paul T. Bateman and Harold G. Diamond. *Analytic Number Theory: An Introductory Course*. World Scientific, 2004.

[3] Josh Cohen Benaloh. Dense probabilistic encryption. In *Proceedings of the Workshop on Selected Areas in Cryptography*, pages 120–128, 1994.

[4] Josh Daniel Cohen Benaloh. *Verifiable secret-ballot elections*. PhD thesis, Yale University, 1987.

[5] Christian Cachin, Silvio Micali, and Markus Stadler. Computationally private information retrieval with polylogarithmic communication. In *Advances in Cryptology: EUROCRYPT '99*, volume 1592 of *Lecture Notes in Computer Science*, pages 402–414. Springer Verlag, 1999.

[6] Don Coppersmith. Finding a small root of a bivariate integer equation; factoring with high bits known. In *EUROCRYPT*, pages 178–189, 1996.

[7] Don Coppersmith. Finding a small root of a univariate modular equation. In *EUROCRYPT*, pages 155–165, 1996.

[8] George David Forney. *Concatenated Codes*. PhD thesis, MIT, 1966.

[9] Craig Gentry and Zulfikar Ramzan. Single-database private information retrieval with constant communication rate. In *Automata, Languages and Programming*, volume 3580 of *Lecture Notes in Computer Science*, pages 803–815. Springer Berlin / Heidelberg, 2005.

[10] Oded Goldreich, Dana Ron, and Madhu Sudan. Chinese remaindering with errors. In *STOC '99: Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pages 225–234, New York, NY, USA, 1999. ACM Press.

[11] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[12] Parikshit Gopalan, Richard J. Lipton, and Yan Z. Ding. Error correction against computationally bounded adversaries. Manuscript, 2004.

[13] Venkatesan Guruswami, Johan Håstad, Madhu Sudan, and David Zuckerman. Combinatorial bounds for list decoding. *IEEE Transactions on Information Theory*, 48:1021–1034, 2000.

[14] Venkatesan Guruswami and Piotr Indyk. Efficiently decodable codes meeting gilbert-varshamov bound for low rates. In *SODA '04: Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 756–757, Philadelphia, PA, USA, 2004. Society for Industrial and Applied Mathematics.

[15] Venkatesan Guruswami, Amit Sahai, and Madhu Sudan. "soft-decision" decoding of chinese remainder codes. In *FOCS '00: Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, page 159, Washington, DC, USA, 2000. IEEE Computer Society.

[16] Don Hush and Clint Scovel. Concentration of the hypergeometric distribution. *Statistics and Probability Letters*, 75:127–132, 2005.

[17] Yuval Ishai, Eyal Kushilevitz, and Rafail Ostrovsky. Sufficient conditions for collision resistant hashing. In *Theory of Cryptography*, volume 3378, pages 445–456. Springer Berlin / Heidelberg, 2005.

[18] Jonathan Katz and Luca Trevisan. 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*, pages 80–86, 2000.

[19] Eyal Kushilevitz and Rafail Ostrovsky. Replication is NOT needed: SINGLE database, computationally-private information retrieval. In *IEEE Symposium on Foundations of Computer Science*, pages 364–373, 1997.

[20] Richard J. Lipton. A new approach to information theory. In *STACS '94: Proceedings of the 11th Annual Symposium on Theoretical Aspects of Computer Science*, pages 699–708, London, UK, 1994. Springer-Verlag.

[21] David Mandelbaum. Error correction in residue arithmetic. In *IEEE Transactions on Computers*, volume C-21, pages 538–545. IEEE Computer Society, 1972.

[22] David Mandelbaum. On a class of arithmetic codes and a decoding algorithm. In *IEEE Transactions on Information Theory*, volume 22, pages 85–88. IEEE Transactions on Information Theory Society, 1976.

[23] David Mandelbaum. Further results on decoding arithmetic residue codes. In *IEEE Transactions on Information Theory*, volume 24, pages 643–644. IEEE Transactions on Information Theory Society, 1978.

[24] Silvio Micali, Chris Peikert, Madhu Sudan, and David A. Wilson. Optimal error correction against computationally bounded noise. In Joe Kilian, editor, *TCC*, volume 3378 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2005.

[25] David Naccache and Jacques Stern. A new public key cryptosystem based on higher residues. In *CCS '98: Proceedings of the 5th ACM conference on Computer and communications security*, pages 59–66, New York, NY, USA, 1998. ACM Press.

[26] Rafail Ostrovsky, Omkant Pandey, and Amit Sahai. 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*, pages 387–298. Springer, 2007.

[27] Michael Rabin. Probabilistic algorithm for testing primality. *Journal of Number Theory*, 12:128–138, 1980.

[28] John Rompel. One-way functions are necessary and sufficient for secure signatures. In *STOC '90: Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 387–394, New York, NY, USA, 1990. ACM Press.

[29] Claude E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–343, 623–656, 1948.

[30] Sergey Yekhanin. Towards 3-query locally decodable codes of subexponential length. In *Proceedings of the 39th ACM Symposiom on the Theory of Computinng (STOC)*, 2007.

Appendix

A Semantic Security

By a *Public Key Cryptosystem*, we mean a a triple of probabilistic polynomial time algorithms G, E, D, such that $(PK, SK) \leftarrow G(1^k)$, $c \leftarrow E(PK, x, r)$ $x' \leftarrow D(SK, c)$ Where PK, SK denote the public and secret keys and $x' = x$ w.h.p for the same message. A public key encryption system is semantically secure if, given two messages x^0 and x^1, $b \in_R \{0, 1\}$, and an encryption of one of the messages, $E(PK, x^b)$, no polynomial time adversary can determine b with probability significantly greater than one half. That is:

Definition 2. A Public Key Cryptosystem, G, E, D, with security parameter k is called *semantically secure* (in the sense of indistinguishability) if for all message pairs $\{x^0, x^1\}$ and for all probabilistic polynomial time adversaries A, and for all $b \in_R \{0, 1\}$,

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

Where x^0 and x^1 must be of equal length, and the probability is taken over the key generation algorithm's randomness, choice of b, randomness r used in the encryption algorithm E and the internal randomness of A.

B Semantic Security of BasicEncrypt

We now prove the semantic security of the simple encryption protocol given in §4.1 under the Φ-hiding assumption, we prove this as a sequence of lemmas, lemma 2 through lemma 4.

Lemma 2. Under the Small Primes Φ-Hiding Assumption, if we define

$$H_0 = \{g \in G_m : \langle g \rangle = G_m, \text{ i.e. } g \text{ generates } G_m \},$$

and $H_1 = G_m \setminus H_0$, then, if $b \in_R \{0, 1\}$, given $m \in_R \mathcal{H}_k$, $g \in_R H_b$, no probabilistic polynomial time distinguisher D can correctly distinguish whether $g \in H_0$ with probability noticeably greater than $\frac{1}{2}$.

Proof. Suppose D correctly guesses whether g generates G_m with probability $\frac{1}{2} + \epsilon$ for some noticeable function ϵ. We will use D to break the Φ-hiding assumption. Our adversary A is given m,π according to the distributions given in Assumption 1, and will use D as a subroutine to determine whether $m \in \mathcal{H}_k^\pi$.

First notice that G_m is a cyclic group, so $|H_0| = \varphi(|G_m|) = \varphi(\varphi(m)/2)$. A well-known consequence of the Prime Number Theorem is the lower bound

$$\varphi(n) > \frac{cn}{\log\log(e^2 n)},$$

for some constant c and all n (see for example [2]). Thus

$$\begin{aligned} |H_0| &= \varphi(|G_m|) \\ &> \frac{c|G_m|}{\log\log(e^2|G_m|)} \\ &> \frac{c|G_m|}{\log\log(e^2 m)}. \end{aligned}$$

In particular $\frac{|H_0|}{|G_m|} > \frac{c}{\log\log(e^2 m)}$ which is noticeable in k, since $k \approx \log m$. Thus an element drawn uniformly at random from G_m will be a generator with noticeable probability, we call this probability ι. Then to determine if $m \in \mathcal{H}_k^\pi$, we generate a random $g \in G_m$, and send g^π and m to D. If D says $g \in H_0$, A replies that $m \notin \mathcal{H}_k^\pi$, i.e. m does not Φ-hide π.

To show that we succeed with noticeable probability, we note that if $m \notin \mathcal{H}_k^\pi$, then $g^\pi \in H_0$ iff $g \in H_0$, so $g^\pi \in H_0$ with probability ι. If $m \in \mathcal{H}_k^\pi$, then g^π cannot generate G_m, so $g^\pi \notin H_0$.

Thus

	$\pi \mid \varphi(m)$	$\pi \nmid \varphi(m)$
A says $g^\pi \in H_0$	$\frac{1}{2} - \epsilon$	$\frac{1}{2} - \epsilon + 2\iota\epsilon$
A says $g^\pi \notin H_0$	$\frac{1}{2} + \epsilon$	$\frac{1}{2} + \epsilon - 2\iota\epsilon$

so A is correct with probability

$$\begin{aligned}
\frac{1}{2}\left(\frac{\frac{1}{2}+\epsilon+2\iota\epsilon}{(\frac{1}{2}-\epsilon)+(1-\epsilon+2i\epsilon)} + \frac{\frac{1}{2}+\epsilon}{(\frac{1}{2}+\epsilon)+(\frac{1}{2}+\epsilon-2\iota\epsilon)}\right) &= \frac{1}{2}\left(\frac{\frac{1}{2}-\epsilon+2\iota\epsilon}{1-2\epsilon+2\iota\epsilon} + \frac{\frac{1}{2}+\epsilon}{1+2\epsilon-2\iota\epsilon}\right) \\
&> \frac{1}{2}\left(\frac{1}{2} + \frac{\iota\epsilon}{1-2\epsilon+2\iota\epsilon} + \frac{1}{2} + \frac{\iota\epsilon}{1+2\epsilon-2\iota\epsilon}\right) \\
&> \frac{1}{2}\left(\frac{1}{2} + \iota\epsilon + \frac{1}{2}\right) \\
&= \frac{1}{2} + \frac{\iota\epsilon}{2}.
\end{aligned}$$

Which is non-negligible since both ι and ϵ are non-negligible. ■

Next, we prove a straightforward fact about the distribution $r \mod \varphi(m)$, where $r \in_R \mathbb{Z}/m\mathbb{Z}$.

Lemma 3. If r is selected uniformly at random in $\mathbb{Z}/m\mathbb{Z}$, and r' is selected uniformly at random in $\mathbb{Z}/|G_m|\mathbb{Z}$, then the distributions of $r \mod |G_m|$ and r' are statistically close, i.e.

$$\frac{1}{2}\sum_{x \in \mathbb{Z}/|G_m|\mathbb{Z}} |\Pr[r = x] - \Pr[r' = x]|$$

is negligible in k.

Proof. Since $|G_m| = \frac{\varphi(m)}{2} = \frac{pq-p-q+1}{2}$, the distribution for $r \mod |G_m|$ becomes

$$P(r = x) = \begin{cases} \frac{2}{m} \text{ for } |G_m| - p - q + 1 \text{ elements} \\ \\ \frac{3}{m} \text{ for } p + q - 1 \text{ elements} \end{cases}$$

Thus

$$\frac{1}{2}\sum_{x \in \mathbb{Z}/|G_m|\mathbb{Z}} |\Pr[r = x] - \Pr[r' = x]| = \frac{1}{2}(|G_m| - p - q + 1)\left(\frac{1}{|G_m|} - \frac{2}{m}\right) + \frac{1}{2}(p+q-1)\left(\frac{3}{m} - \frac{1}{|G_m|}\right).$$

Now,

$$\frac{1}{|G_m|} - \frac{2}{m} = 2\left(\frac{1}{pq-p-q+1} - \frac{1}{pq}\right) = \frac{p+q-1}{m|G_m|},$$

so

$$\frac{1}{2}(|G_m| - p - q + 1)\left(\frac{1}{|G_m|} - \frac{2}{m}\right) \leq \frac{p+q}{2m}.$$

Similarly, we have

$$\frac{3}{m} - \frac{1}{|G_m|} = \frac{3(pq-p-q+1)-2pq}{m(pq-p-q+1)} = \frac{pq-3(p+q-1)}{2m|G_m|} < \frac{pq-p-q+1}{2m|G_m|} = \frac{1}{m}.$$

so

$$\frac{1}{2}(p+q-1)\left(\frac{3}{m} - \frac{1}{|G_m|}\right) \leq \frac{p+q}{2m}.$$

Thus the statistical distance is less than

$$\frac{(p+q)}{m}$$

which is negligible in k since $\log m \approx k$, and $\log p \approx \log q \approx \frac{k}{2}$. ■

Now we are ready to prove the semantic security of our cryptosystem.

Lemma 4. The encryption in §4.1 is semantically secure under the small primes Φ-hiding assumption.

Proof. Given any distinguisher D for the encryption protocol that succeeds with non-negligible probability, we construct an adversary A which violates the Φ-hiding assumption with non-negligible probability. Given m and π where $m \in_R \mathcal{H}_k^\pi$ with probability $\frac{1}{2}$ and $m \in_R \mathcal{H}_k \setminus \mathcal{H}_k^\pi$ with probability $\frac{1}{2}$, the adversary A picks a $g \in G_m$ and sends g, m to the distinguisher D, and D responds with two messages x^0, x^1. Then A chooses $b \in_R \{0,1\}$, and $r \in_R \mathbb{Z}/m\mathbb{Z}$ and computes

$$c = g^{x^b + \pi r} \mod m$$

and sends c to the distinguisher D. D responds with a bit b^*. If $b^* = b$ the adversary responds that m Φ-hides π, otherwise the adversary responds m does not Φ-hide π.

Now we must show that this adversary breaks the Φ-hiding assumption with non-negligible probability.

First, assume g generates G_m. If m Φ-hides π, then c is a valid encryption of x^b, and so by the definition of D, we must have that $b^* = b$ with probability $\frac{1}{2} + \epsilon$ for some non-negligible function ϵ.

On the other hand, if m does not Φ-hide π, then $\pi \in (\mathbb{Z}/|G_m|\mathbb{Z})^*$. Now, notice that if r' were chosen uniformly in $\mathbb{Z}/|G_m|\mathbb{Z}$ instead of $\mathbb{Z}/m\mathbb{Z}$, we would have $x^b + \pi r' \mod |G_m|$ is also uniformly distributed in $\mathbb{Z}/|G_m|\mathbb{Z}$. Thus $g^{x^b+\pi r'}$ would be uniformly distributed in G_m, and hence any distinguisher D could guess b from $g^{x^b+\pi r'}$ with probability at most one half. By lemma 3, the statistical distance between $g^{x^b+\pi r}$ and $g^{x^b+\pi r'}$ is negligible, thus any distinguisher D succeeds in guessing b with probability $\frac{1}{2} + \nu$ for some negligible function ν.

Then, following this scheme, if g generates G_m with probability a, our algorithm succeeds in breaking the Φ-hiding assumption with probability $\frac{1}{2} + \frac{\epsilon - \nu}{2}$ which is a non-negligible since ϵ is non-negligible and ν is negligible.

If, instead, g does not generate G_m, then by lemma 2, D's output distribution must be negligibly different from when g generates G_m. Thus in this case as well, A correctly guesses whether $\pi | \varphi(m)$ with probability noticeably greater than $\frac{1}{2}$. ■

C Proof of Local-Decodability

Here, we show correctness, i.e. that our system is computationally locally-decodable up to a constant fraction of errors. By an encryption of a message X, we mean a t-tuple $(\mathrm{ECC}(h_1), \ldots, \mathrm{ECC}(h_t))$ where $t = \frac{2n}{dk}$, and each h_i is an element of G_{m_i}. We show that our decoding algorithm decodes correctly with all but negligible probability, at most a $\frac{1}{4} - \delta$ fraction of the bits of the encryption

have been corrupted by a polynomial-time adversary A. Notice that our algorithm will decode a block X_i correctly whenever no more than $\frac{1}{4}\frac{2\ell}{d}$ of the h_i that encode it are corrupted. Thus we will show that any polynomial-time adversary that corrupts a $\frac{1}{4}-\delta$ fraction of the bits, only corrupts more than $\frac{1}{4}$ of the h_i that encode a given block of the message with negligible probability. We prove this through a series of lemmas.

We begin by noticing that any adversary A that corrupts at most $\frac{1}{4}-\delta$ fraction of the *bits* of the message, can only corrupt at most a $\frac{1}{2}-\delta-\delta^2$ fraction of the h_i.

Lemma 5. Given $(\mathrm{ECC}(h_1),\ldots,\mathrm{ECC}(h_t))$, where ECC recovers from a binary error-rate of $\frac{1}{2}-\delta$, any adversary A that corrupts at most $\frac{1}{4}-\delta$ bits of the entire codeword, can corrupt no more than $\frac{1}{2}-\delta+\delta^2$ of the h_i

Proof. This is simply counting. A can corrupt a total of $(\frac{1}{4}-\delta)ct|h_i|$ bits, and to corrupt one h_i A needs to spend $(\frac{1}{2}-\delta)c|h_i|$, thus A can corrupt at most $(\frac{1}{2}-\delta-\delta^2)t$ of the h_i since

$$\begin{aligned}\frac{\left(\frac{1}{4}-\delta\right)ct|h_i|}{\left(\frac{1}{2}-\delta\right)c|h_i|} &\leq \frac{\frac{1}{4}-\frac{\delta}{2}-\frac{\delta}{2}+\delta^2-\frac{\delta^2}{2}+\delta^3}{\frac{1}{2}-\delta}t \\ &= \frac{\left(\frac{1}{2}-\delta-\delta^2\right)\left(\frac{1}{2}-\delta\right)}{\frac{1}{2}-\delta}t \\ &= \left(\frac{1}{2}-\delta-\delta^2\right)t.\end{aligned}$$

■

For the rest of the proof of correctness, we assume that A is restricted to corrupting a $\frac{1}{2}-\delta-\delta^2$ fraction of the h_i, rather than $\frac{1}{4}-\delta$ *bits* of the message.

Now, we show that any such corrupting adversary cannot detect whether inputs are "well-formed", i.e. A behaves in an indistinguishable manner whether the t-tuple $(h_1,\ldots,h_t)$ is a valid encryption or not.

Lemma 6. For all probabilistic polynomial-time adversaries A, such that A introduces errors in t-tuples $(h_1,\ldots,h_t)$, where each $h_i \in G_{m_i}$ and each m_i Φ-hides a distinct prime-power π_i, then A will also introduce errors in t-tuples $(h_1,\ldots,h_t)$ where each m_i Φ-hides *the same* prime-power π_i.

Proof. Instead of running A on a t-tuple where each modulus m_i Φ-hides a distinct π_i, we provide A with t-tuple in which each modulus m_i Φ hides *the same* π. Assume A fails to introduce errors on this malformed input with non-negligible probability ϵ. Now we proceed via hybrid argument. Since the probability the A fails on t moduli Φ-hiding the same π, is ϵ greater than when each m_i Φ-hides a different prime, then triangle inequality tells us that there must be some $t^* < t$ such that,

$$|\Pr[A \text{ fails when } t^* \ m_i \ \Phi\text{-hide the same } \pi\ ] - \Pr[A \text{ fails when } t^*+1 \ m_i \ \Phi\text{-hide the same } \pi\ ]| > \frac{\epsilon}{t}.$$

We can now use A to break the Φ-hiding assumption. Given a modulus m^* such that m^* Φ-hides π^* where π^* equals π_0 or π_1, we construct t^* moduli $m_1,\ldots,m_{t^*}$ that Φ-hide π_0, and $t-t^*-1$ moduli $m_{t^*+1},\ldots,m_{t-1}$ that Φ-hide primes other than π_0,π_1. We then run A, on the t-tuple $(h_1,\ldots,h_{t-1},h^*)$ where $h_i \in G_{m_i}$ for $1 \leq i < t$, and $h^* \in G_{m^*}$. If A fails to introduce errors on this t-tuple, we say that m^* Φ-hides π_0. This algorithm correctly distinguishes whether m^* Φ-hides π_0 or π_1 with probability at least $\frac{1}{2}+\frac{\epsilon}{2t}$. Although, we do not know the exact value of t^*, we can we can guess it with probability $\frac{1}{t}$, to obtain an algorithm which decides whether m^* Φ-hides π_0 with advantage $\frac{\epsilon}{2t^2}$ which is a violation of the Φ-hiding assumption. ■

If each m_i Φ-hides the same π, then A must distribute errors randomly among the blocks, since the notion of blocks in this case is completely arbitrary. Since A must behave identically when each m_i Φ-hides the same modulus as when they all Φ-hide different moduli, we notice that an adversary cannot focus the errors on the remainders encoding a specific block. To make this formal, recall that the message X was divided into blocks $\frac{n}{\ell k}$ blocks X_i, and each block X_i was encoded s remainders, and t was the total number of remainders $t = s\frac{n}{\ell k}$. If we define $S_i \subset \{h_1, \ldots, h_t\}$ to be the set of remainders encoding block X_i, $S_i = h_{\sigma^{-1}((i-1)s+1,\ldots,h_{\sigma^{-1}(is)})}$, then $|S_i| = s$, and we obtain the following lemma.

Lemma 7. If A is a probabilistic polynomial-time machine which introduces errors in $\{h_1, \ldots, h_t\}$, the distribution of the errors in the set $\{S_1, \ldots, S_{\frac{n}{\ell k}}\}$ is computationally indistinguishable from the uniform random distribution on $\{S_1, \ldots, S_{\frac{n}{\ell k}}\}$.

Proof. Suppose there exists a distinguisher D that can distinguish the corruptions A introduces among the S_i from uniform random with advantage ϵ. Then we run D on A's output when A is given moduli that Φ-hide between one and t distinct π's. When we run A on a t-tuple $(h_1, \ldots, h_t)$ where each m_i Φ-hides the same π, in this situation A must distribute errors uniformly, since A has *no* information about the underlying subsets S_i. Thus in this situation, D cannot distinguish A's corruptions from random with probability greater than $\frac{1}{2}$, since in this case A's corruptions *are* random. Now we proceed via a hybrid argument. When A is run on moduli that Φ-hide t distinct π_i, then D can distinguish A's corruptions from random with advantage ϵ, thus by the triangle inequality, there exists a $t^* < t$ such that D can distinguish A's output when A is run on moduli, t^* of which are then same, from A's output when A is run on moduli, $t^* + 1$ of which are the same, with advantage $\frac{\epsilon}{t}$. This allows us to break the Φ-hiding assumption in exactly the manner described before. Given a modulus m^* that Φ-hides either π_0 or π_1, we construct t^* moduli $m_1, \ldots, m_{t^*}$ which all Φ-hide π, and $t - t^* - 1$ moduli $m_{t^*+1}, \ldots, m_{t-1}$ which all Φ-hide different moduli. Then we run D on A's output, when A is given $(m_1, \ldots, m_{t-1}, m^*)$. By the definition of t^* D succeeds in distinguishing whether m^* Φ-hides π_0 with advantage $\frac{\epsilon}{2t}$. Thus by guessing a random value in $\{1, \ldots, t-1\}$ for t^*, we break the Φ-hiding assumption with advantage $\frac{\epsilon}{2t^2}$, a contradiction. ■

Lemma 8. If A distributes $(\frac{1}{4} - \delta)t$ errors uniformly among the t remainders, the probability that A destroys any given block X_i is negligible in ℓ.

Proof. If A distributes errors at random, then we can view A as selecting remainders to corrupt uniformly at random . The adversary A destroys a block X_i exactly when A corrupts more than $\frac{1}{2} - \delta$ of the remainders that encode that block, the probability that A destroys a block is exactly the probability that more than $(\frac{1}{2} - \delta)\frac{\rho\ell}{d}$ moduli that encode X_i are corrupted. This distribution is then the Hypergeometric Distribution, where $\frac{\rho\ell}{d}$ items are selected and $(\frac{1}{2} - \delta - \delta^2)t$ of which are corrupted. In [16], Hush and Scovel give the bound

$$\Pr\left[\ \#\text{ of errors in encoding of block } X_i\ > \left(\frac{1}{2} - \delta\right)\frac{\rho\ell}{d}\right] < e^{-2\left(\frac{d}{\rho\ell+d}\right)\left(\frac{\delta^4\rho^2\ell^2}{d^2}-1\right)},$$

where the probability is taken over the uniform distribution on the t remainders, and this probability is clearly negligible in ℓ. ■

Lemma 9. If at most $(\frac{1}{4} - \delta)t$ of the t encryptions are corrupted by a probabilistic polynomial-time adversary A, then the probability that any bit of the message fails to decode properly is negligible in k.

Proof. For a given block X_i the probability that that block is damaged under the corruptions created by A is negligibly different in k than if A produced the corruptions at random, which itself would damage X_i with only negligible probability in ℓ. Taking $\ell \approx k$, we have that the block X_i is damaged with at most negligible probability in k. The union bound then gives that the probability that *any* block X_i is damaged is at most t times the probability that a specific block is damaged, which remains negligible in k. ■

D Constant Rate Binary Error Correcting Codes

For our scheme to have constant information rate, we need to find a *binary* error-correcting code which can tolerate an error-rate of $\frac{1}{2} - \delta$.

One method for creating such a code, uses the notion of Concatenated Codes, originally described by Forney in [8]. By combining a Reed-Solomon Code and a Random Linear Code as described in [14], it is possible to obtain a binary error correcting code which recovers from $\frac{1}{4} - \delta$ error-rate, but the information-rate of the resulting code is very low, about 10^{-4} for their construction.

Since we are working in the computationally bounded channel model, we can take advantage of the constructions described in [24], to create a binary code with error-rate $\frac{1}{2} - \delta$, and significantly better information rates than in the unbounded channel model. Applying Micali et al's construction to the binary codes with list-decoding rate $\frac{1}{2}$ and information rate δ^4 described in [13], we obtain a code which uniquely decodes from error-rate $\frac{1}{2} - \delta$, and has information rate about $\frac{1}{\delta^4}$.

E CRT-Based Error Correction

It was observed in the 1970s [21], [22], [23], that the Chinese Remainder Theorem could be used to make efficient Error Correcting Codes. If $\pi_1, \ldots, \pi_n, \pi_{n+1}, \ldots, \pi_{n+t}$ an increasing sequence of pairwise coprime integers, i.e. $\pi_1 < \pi_2 < \cdots < \pi_{n+t}$, and $\gcd(\pi_i, \pi_j) = 1$ whenever $i \neq j$. Then for any integer x with $x < \prod_{i=1}^{n} \pi_i$, we encode x as the $(n+t)$-tuple $\{x \mod p_1, \ldots, x \mod p_{n+t}\}$. If x and x' are distinct integers less than $\prod_{i=1}^{n}$, then the two vectors $E(x) = \{x \mod \pi_1, \ldots, x \mod \pi_{n+t}\}$ and $E(x') = \{x' \mod \pi_1, \ldots, x' \mod \pi_{n+t}\}$ must differ in at least $t+1$ coordinates since the residue of x modulo *any* n of the moduli π_i uniquely determines x. Thus the minimum distance in this code is t, and so it can correct $\lfloor \frac{t}{2} \rfloor$ errors. Thus if we take $n + t = \rho n$, this code can recover from error-rate $\frac{1}{2} - \frac{1}{2\rho}$, in the digits of the code.

This code differs significantly from most other error correcting codes in that each "digit", i.e. each remainder, of the codeword carries a different amount of information. Thus the Hamming distance between two codewords, measured as the number of remainders in which they differ is not the natural distance to consider for this code. This fact made finding an efficient decoding algorithm a nontrivial task. In his original paper in 1972, Mandelbaum proposed an algorithm that ran in expected polynomial-time. Since then, many variants of that algorithm have appeared, but it was not until 2001 [15] that the first polynomial-time decoding algorithm was found. Since the Chinese Remainder Codes are efficiently list decodable [10], [15], we can apply the technique in [24] of combining list-decoding with digital signatures to our protocol to further improve the information-rate.

F Gentry-Ramzan PIR

While our scheme does not explicitly rely on the Gentry-Ramzan PIR scheme, our protocol was inspired by their use of the Φ-hiding assumption to do "hidden" Chinese Remaindering. In the interest both of giving some context for our scheme, and of showing what else can be achieved by

hidden Chinese Remaindering, we briefly sketch the Gentry-Ramzan Private Information Retrieval scheme [9]. This scheme allows computationally private single database PIR with constant communication rate under the Φ-hiding assumption. Here "constant" means proportional to the security parameter. The scheme allows retrieval of entire blocks at once, and the scheme we describe will retrieve an ℓ-bit block from an n-bit database.

The scheme assumes some initial set-up. First, sequence of small primes $p_1, \ldots, p_t$ are fixed in advance. Then we set $\ell = \lceil n/t \rceil$, and $c_i = \lceil \log_{p_i} \ell \rceil$. Setting $\pi_i = p_i^{c_i}$, we have that $\pi_i > 2^\ell$ for all i, and the integers $\pi_1, \ldots, \pi_t$ are pairwise relatively prime. This initial set-up is assumed to be known to both the user and the database, and is not included in the communication complexity of the scheme.

To begin the scheme, the database must do some pre-processing. Instead of viewing the database as a single n-bit string, we instead view it as a concatenation of t ℓ-bit integers $a_1, \ldots, a_t$. Recall that we have chosen our π_i such that $a_i < \pi_i$ for each i. Using the Chinese Remainder Theorem, the database can find an integer $e < \prod_{i=1}^{t} \pi_i$, such that $e \mod \pi_i = a_i$.

To retrieve the jth block of the database, a_j, the user then chooses an RSA modulus $m = pq$ that Φ-hides π_j, and a g for cyclic the group G_m, i.e. g has order $\frac{\varphi(m)}{2}$ in $(\mathbb{Z}/m\mathbb{Z})^*$. Since $\pi_j | \varphi(m)$, we have that G_m has a subgroup of order π_j. Letting $q = \frac{\varphi(m)}{2\pi_j}$, this subgroup is generated by g^q. The user then sends both m, and g to the database. The database calculates $g^e \mod m$ and returns the result.

Given $g^e \mod m$, the user then calculates $(g^e)^q = (g^q)^e = g^{e \mod \pi_j} \mod m$ since g^q has order π_j in G_m. Then by performing (a tractable) discrete-log computation in the subgroup of order π_j generated by g^q the user recovers $e \mod \pi_j = a_j$. Using Pohlig-Hellman algorithm this discrete-log computation can be calculated in $\mathcal{O}(c_j \sqrt{p_j})$ time.

If $\log_2(m) = k$, then the user sends $2k$ bits to the database, and the database replies with k bits, so the total communication complexity is $3k$ bits. To avoid the lattice-based attacks described in [7] and [6], we must choose m such that $\pi_i < m^{\frac{1}{4}}$ for all i, i.e. $\ell < 4k$.