Cryptography with Constant Input Locality*

(EXTENDED ABSTRACT)

Benny Applebaum, Yuval Ishai**, and Eyal Kushilevitz***

Computer Science Department, Technion, Haifa 32000, Israel
{abenny,yuvali,eyalk}@cs.technion.ac.il

Abstract. We study the following natural question: Which cryptographic primitives (if any) can be realized by functions with constant input locality, namely functions in which every bit of the *input* influences only a constant number of bits of the output? This continues the study of cryptography in low complexity classes. It was recently shown (Applebaum et al., FOCS 2004) that, under standard cryptographic assumptions, most cryptographic primitives can be realized by functions with constant *output* locality, namely ones in which every bit of the *output* is influenced by a constant number of bits from the input.

We (almost) characterize what cryptographic tasks can be performed with constant input locality. On the negative side, we show that primitives which require some form of non-malleability (such as digital signatures, message authentication, or non-malleable encryption) *cannot* be realized with constant input locality. On the positive side, assuming the intractability of certain problems from the domain of error correcting codes (namely, hardness of decoding a random linear code or the security of the McEliece cryptosystem), we obtain new constructions of one-way functions, pseudorandom generators, commitments, and semantically-secure public-key encryption schemes whose input locality is constant. Moreover, these constructions also enjoy constant *output locality*. Therefore, they give rise to cryptographic hardware that has constant-depth, constant fan-in and constant *fan-out*. As a byproduct, we obtain a pseudorandom generator whose output and input locality are both optimal (namely, 3).

1 Introduction

The question of minimizing the complexity of cryptographic primitives has been the subject of an extensive body of research (see [23, 3] and references therein). On one extreme, it is natural to ask whether one can implement cryptographic primitives in NC^0, i.e., by functions in which each output bit depends on a

* Research supported by grant 1310/06 from the Israel Science Foundation.
** Supported by grant 2004361 from the U.S.-Israel Binational Science Foundation.
*** Supported by grant 2002354 from the U.S.-Israel Binational Science Foundation.

constant number of input bits.[1] Few primitives, including pseudorandom *functions* [12], cannot even be realized in AC^0 [20]; no similar negative results are known for other primitives. However, it was shown recently [3, 2] that, under standard assumptions, most cryptographic primitives can be realized by functions with output locality 4, namely by NC^0 functions in which each bit of the output depends on at most 4 bits of the input.

Another possible extreme is the complementary question of implementing cryptographic primitives by functions in which each *input* bit affects only a constant number of output bits. This was not settled by [3], and was suggested as an open problem. This natural question can be motivated from several distinct perspectives:

– (Theoretical examination of a common practice) A well known design principle for practical cryptosystems asserts that each input bit must affect many output bits. This principle is sometimes referred to as Confusion/Diffusion or Avalanche property. It is easy to justify this principle in the context of block-ciphers (which are theoretically modeled as pseudorandom functions or permutations), but is it also necessary in other cryptographic applications (e.g., stream ciphers)?
– (Hardware perspective) Unlike NC^0 functions, functions with both constant input locality and constant output locality can be computed by constant depth circuits with bounded fan-in and *bounded fan-out*. Hence, the parallel time complexity of such functions is constant in a wider class of implementation scenarios.
– (Complexity theoretic perspective) One can state the existence of cryptography in NC^0 in terms of average-case hardness of Constraint Satisfaction Problems in which each constraint involves a constant number of variables (k-CSPs). The new question can therefore be formulated in terms of k-CSPs with bounded occurrences of each variable. It is known that NP hardness and inapproximability results can be carried from the CSP setting to this setting [24, 6], hence it is interesting to ask whether the same phenomenon occurs with respect to cryptographic hardness as well.

Motivated by the above, we would like to understand which cryptographic tasks (if any) can be realized with constant input *and* output locality, or even with constant input locality alone.

Another question considered in this work, which was also posed in [3], is that of closing the (small) gap between positive results for cryptography with locality 4 and the impossibility of cryptography with locality 2. It was shown in [3] that the existence of a OWF with locality 3 follows from the intractability of decoding a random linear code. The possibility of closing this gap for other primitives remained open.

[1] Equivalently, NC^0 is the class of functions computed by boolean circuits of polynomial size, constant depth, and bounded fan-in gates. We will also mention the classes AC^0 and NC^1 which extend this class. Specifically, in AC^0 we allow unbounded fan-in AND and OR gates, and in NC^1 the circuit depth is logarithmic.

1.1 Our Results

We provide an almost full characterization of the cryptographic tasks that can be realized by functions with constant input locality. On the negative side, we show that primitives which require some form of non-malleability (e.g., signatures, MACs, non-malleable encryption schemes) *cannot* be realized with constant (or, in some cases, even logarithmic) input locality.

On the positive side, assuming the intractability of some problems from the domain of error correcting codes, we obtain constructions of pseudorandom generators, commitments, and semantically-secure public-key encryption schemes with constant input locality and constant output locality. In particular, we obtain the following results:

- For PRGs, we answer simultaneously both of the above questions. Namely, we construct a collection[2] of PRGs whose output locality and input locality are both 3. We show that this is optimal in both output locality and input locality. Our construction is based on the intractability of decoding a random linear code. Previous constructions of PRGs (or even OWFs) [4, 9] which enjoyed constant input locality and constant output locality at the same time, were based on non-standard intractability assumptions.
- We construct a non-interactive commitment scheme, in the common reference string model, in which the output locality of the commitment function is 4, and its input locality is 3. The security of this scheme also follows from the intractability of decoding a random linear code. (We can also get a non-interactive commitment scheme in the standard model under the assumption that there exists an explicit binary linear code that has a large minimal distance but is hard to decode.)
- We construct a semantically secure public-key encryption scheme whose encryption algorithm has input locality 3. This scheme is based on the security of the McEliece cryptosystem [21], an assumption which is related to the intractability of decoding a random linear code, but is seemingly stronger. Our encryption function also has constant output locality, if the security of the McEliece cryptosystem holds when it is instantiated with some error correcting code whose relative distance is constant.
- We show that MACs, signatures and non-malleable symmetric or public-key encryption schemes cannot be realized by functions whose input locality is constant or, in some cases, even logarithmic in the input length. In fact, we prove that even the weakest versions of these primitives (e.g., one-time secure MACs) cannot be constructed in this model.

1.2 Our Techniques

Our constructions rely on the machinery of *randomized encoding*, which was was explicitly introduced in [16] (under the algebraic framework of *randomizing*

[2] All of our collections are indexed by a public random key. That is, $\{G_z\}_{z\in\{0,1\}^*}$ is a collection of PRGs if for every z the function G_z expands its input and the pair $(z, G_z(x))$ is pseudorandom for random x and z.

polynomials) and was implicitly used, in weaker forms, in the context of secure multiparty computation (e.g., [19, 8]). A randomized encoding of a function $f(x)$ is a randomized mapping $\hat{f}(x, r)$ whose output distribution depends only on the output of f. Specifically, it is required that: (1) there exists a decoder algorithm that recovers $f(x)$ from $\hat{f}(x, r)$, and (2) there exists a simulator algorithm that given $f(x)$ samples from the distribution $\hat{f}(x, r)$ induced by a uniform choice of r. That is, the distribution $\hat{f}(x, r)$ hides all the information about x except for the value $f(x)$.

In [3] it was shown that the security of most cryptographic primitives is inherited by their randomized encoding. Suppose that we want to construct some cryptographic primitive $\mathcal{P}$ in some low complexity class WEAK. Then, we can try to encode functions from a higher complexity class STRONG by functions from WEAK. Now, if we have an implementation f of the primitive $\mathcal{P}$ in STRONG, we can replace f by its encoding $\hat{f} \in \mathsf{WEAK}$ and obtain a low-complexity implementation of $\mathcal{P}$. This paradigm was used in [3, 2]. For example, it was shown that STRONG can be NC^1 and WEAK can be the class of functions whose output locality is 4.

However, it seems hard to adapt this approach to the current setting, since it is not clear whether there are non-trivial functions that can be encoded by functions with constant input locality. (In fact, we show that some very simple NC^0 functions cannot be encoded in this class.) We solve this problem by introducing a new construction of randomized encodings. Our construction shows that there exists a complexity class $\mathcal{C}$ of simple (but non-trivial) functions that can be encoded by functions with constant input locality. Roughly speaking, a function f is in $\mathcal{C}$ if each of its output bits can be written as a sum of terms over $\mathbb{F}_2$ such that each input variable of f participates in a constant number of *distinct* terms, ranging over all outputs of f. Moreover, if the algebraic degree of theses terms is constant, then f can be encoded by a function with constant input locality as well as constant output locality. (In particular, all linear functions over $\mathbb{F}_2$ admit such an encoding.)

By relying on the nice algebraic structure of intractability assumptions related to decoding random linear codes, and using techniques from [4], we construct PRGs, commitments and public-key encryption schemes in $\mathcal{C}$ whose algebraic degree is constant. Then, we use the new construction to encode these primitives, and obtain implementations whose input locality and output locality are both constant.

Interestingly, unlike previous constructions of randomized encodings, the new encoding does not have a universal simulator nor a universal decoder; that is, one should use different decoders and simulators for different functions in C. This phenomenon is inherent to the setting of constant input locality and is closely related to the fact that MACs cannot be realized in this model. See Section 6.2 for a discussion.

1.3 Previous Work

The existence of cryptographic primitives in NC^0 has been recently studied in [7, 22, 3]. Goldreich observed that a function whose output locality is 2 cannot even be one-way [9]. Cryan and Miltersen [7] proved that a PRG whose output locality is 3 cannot achieve a superlinear stretch; namely, it can only stretch n bits to $n + O(n)$ bits. Mossel et al. [22] extended this impossibility to functions whose output locality is 4.

On the positive side, Goldreich [9] suggested an approach for constructing OWFs based on expander graphs, an approach whose conjectured security does not follow from any well-known assumption. This general construction can be instantiated by functions with constant output locality and constant input locality. Mossel et al. [22] constructed (non-cryptographic) ε-biased generators with (non-optimal) constant input and output locality. Applebaum et al. [3, 2] subsequently showed that: (1) the existence of many cryptographic primitives (including OWFs, PRGs, encryptions, signatures and hash functions) in NC^1 implies their existence with output locality 4; and (2) the existence of these primitives in NC^1 is implied by most standard cryptographic assumptions such as the intractability of factoring, discrete logarithms and lattice problems. They also constructed a OWF with (optimal) output locality 3 based on the intractability of decoding a random linear code. However, all these constructions did not achieve constant input locality. The constructions in [3] were also limited to PRGs with small (sub-linear) stretch, namely, one that stretches a seed of length n to a pseudorandom string of length $n+o(n)$. This problem was addressed by [4], who gave a construction of a linear-stretch PRG with (large) constant output locality under a non-standard assumption taken from [1]. In fact, the construction of [4] can also give an NC^0 PRG with (large) constant input locality (under the same non-standard assumption).

2 Preliminaries

Notation. All logarithms in this paper are to the base 2. We use U_n to denote a random variable uniformly distributed over $\{0,1\}^n$. We let $\mathrm{H}_2(\cdot)$ denote the binary entropy function, i.e., for $0 < p < 1$, $\mathrm{H}_2(p) \stackrel{\mathrm{def}}{=} -p\log(p) - (1-p)\log(1-p)$. The *statistical distance* between discrete probability distributions Y and Y', denoted $\mathrm{SD}(Y, Y')$, is defined as the maximum, over all functions A, of the *distinguishing advantage* $|\Pr[A(Y) = 1] - \Pr[A(Y') = 1]|$.

A function $\varepsilon(\cdot)$ is said to be *negligible* if $\varepsilon(n) < n^{-c}$ for any constant $c > 0$ and sufficiently large n. We will sometimes use $\mathrm{neg}(\cdot)$ to denote an unspecified negligible function. For two distribution ensembles $\{X_n\}_{n\in\mathbb{N}}$ and $\{Y_n\}_{n\in\mathbb{N}}$, we write $X_n \equiv Y_n$ if X_n and Y_n are identically distributed, and $X_n \stackrel{\mathrm{s}}{\equiv} Y_n$ if the two ensembles are *statistically indistinguishable*; namely, $\mathrm{SD}(X_n, Y_n)$ is negligible in n. A weaker notion of closeness between distributions is that of *computational* indistinguishability: We write $X_n \stackrel{\mathrm{c}}{\equiv} Y_n$ if for every (non-uniform) polynomial-size circuit family $\{A_n\}$, the distinguishing advantage $|\Pr[A_n(X_n) = 1] - \Pr[A_n(Y_n) = 1]|$

is negligible. A distribution ensemble $\{X_n\}_{n\in\mathbb{N}}$ is said to be *pseudorandom* if $X_n \overset{c}{\equiv} U_{m(n)}$ where $m(n)$ is the length of strings over which X_n is distributed.

Locality. Let $f : \{0,1\}^n \to \{0,1\}^s$ be a function. The *output locality* of f is c if each of its output bits depends on at most c input bits. The *locality* of an input variable x_i in f is c if at most c output bits depend on x_i. The *input locality* of f is c if the input locality of all the input variables of f is bounded by c. The output locality (resp. input locality) of a function family $f : \{0,1\}^* \to \{0,1\}^*$ is c if for every n the restriction of f to n-bit inputs has output locality (resp. input locality) c. We envision circuits as having their inputs at the bottom and their outputs at the top. Hence, for functions $l(n), m(n)$, we let $\mathrm{Local}_{l(n)}^{m(n)}$ (resp. $\mathrm{Local}_{l(n)}$, $\mathrm{Local}^{m(n)}$) denote the non-uniform class which includes all functions $f : \{0,1\}^* \to \{0,1\}^*$ whose input locality is $l(n)$ and output locality is $m(n)$ (resp. whose input locality is $l(n)$, whose output locality is $m(n)$). The uniform versions of these classes contain only functions that can be computed in polynomial time. (All of our positive results are indeed uniform.) Note that $\mathrm{Local}^{O(1)}$ is equivalent to the class NC^0 which is the class of functions that can be computed by constant depth circuits with bounded fan-in. Also, the class $\mathrm{Local}_{O(1)}^{O(1)}$ is equivalent to the class of functions that can be computed by constant depth circuits with bounded fan-in and bounded fan-out.

2.1 Randomized Encoding

We review the notions of randomized encoding and randomizing polynomials from [16, 17, 3].

Definition 1. (Perfect randomized encoding [3]) *Let $f : \{0,1\}^n \to \{0,1\}^l$ be a function. We say that a function $\hat{f} : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^s$ is a perfect* randomized encoding *of f, if there exist an algorithm B, called a* decoder*, and a randomized algorithm S, called a* simulator*, for which the following hold:*

– **perfect correctness.** $B(\hat{f}(x,r)) = f(x)$ *for any input* $x \in \{0,1\}^n, r \in \{0,1\}^m$.
– **perfect privacy.** $S(f(x)) \equiv \hat{f}(x, U_m)$ *for any* $x \in \{0,1\}^n$.
– **balance.** $S(U_l) \equiv U_s$.
– **stretch preservation.** $s - (n+m) = l - n$, *or equivalently* $m = s - l$.

We refer to the second input of $\hat{f}$ as its *random input*, and to m and s as the *randomness complexity* and the *output complexity* of $\hat{f}$, respectively. The *overall complexity* (or complexity) of $\hat{f}$ is defined to be $m + s$.

Definition 1 naturally extends to infinite functions $f : \{0,1\}^* \to \{0,1\}^*$. In this case, the parameters l, m, s are all viewed as functions of the input length n, and the algorithms B, S receive 1^n as an additional input. By default, we require $\hat{f}$ to be computable in $\mathrm{poly}(n)$ time whenever f is. In particular, both $m(n)$ and $s(n)$ are polynomially bounded. We also require both the decoder and the simulator to be efficient.

We will rely on the following composition property of randomized encodings.

Lemma 1 (Lemma 4.6 in [3]). (Composition) *Let $g(x, r_g)$ be a perfect encoding of $f(x)$ and $h((x, r_g), r_h)$ be a perfect encoding of $g((x, r_g))$ (viewed as a single-argument function). Then, the function $\hat{f}(x, (r_g, r_h)) \stackrel{def}{=} h((x, r_g), r_h)$ is a perfect encoding of f.*

3 Randomized Encoding with Constant Input Locality

In this section we will show that functions with a "simple" algebraic structure (and in particular *linear* functions over $\mathbb{F}_2$) can be encoded by functions with constant input locality. We begin with the following construction that shows how to reduce the input locality of a function which is represented as a sum of functions.

Construction 1. (Basic input locality construction) *Let*

$$f(x) = (a(x) + b_1(x), a(x) + b_2(x), \ldots, a(x) + b_k(x), c_1(x), \ldots, c_l(x)),$$

where $f : \mathbb{F}_2^n \to \mathbb{F}_2^{k+l}$ and $a, b_1, \ldots, b_k, c_1, \ldots, c_l : \mathbb{F}_2^n \to \mathbb{F}_2$. The encoding $\hat{f} : \mathbb{F}_2^{n+k} \to \mathbb{F}_2^{2k+l}$ is defined by:

$$\hat{f}(x, (r_1, \ldots, r_k)) \stackrel{def}{=} (r_1 + b_1(x), r_2 + b_2(x), \ldots, r_k + b_k(x),$$
$$a(x) - r_1, r_1 - r_2, \ldots, r_{k-1} - r_k, c_1(x), \ldots, c_l(x)) \ .$$

Note that after the transformation the function $a(x)$ appears only once and therefore the locality of the input variables that appear in a is reduced. In addition, the locality of all the other original input variables does not increase.

Lemma 2. (Input locality lemma) *Let f and $\hat{f}$ be as in Construction 1. Then, $\hat{f}$ is a perfect randomized encoding of f.*

Proof. The encoding $\hat{f}$ is stretch-preserving since the number of random inputs equals the number of additional outputs (i.e., k). Moreover, given a string $\hat{y} = \hat{f}(x, r)$ we can decode the value of $f(x)$ as follows: To recover $a(x) + b_i(x)$, compute the sum $y_i + y_{k+1} + y_{k+2} + \ldots + y_{k+i}$; To compute $c_i(x)$, simply take y_{2k+i}. This decoder never errs.

Fix some $x \in \{0,1\}^n$. Let $y = f(x)$ and let $\hat{y}$ denote the distribution $\hat{f}(x, U_k)$. To prove perfect privacy, note that: (1) the last l bits of $\hat{y}$ are fixed and equal to $y_{[k+1\ldots k+l]}$; (2) the first k bits of $\hat{y}$ are independently uniformly distributed; (3) the remaining bits of $\hat{y}$ are uniquely determined by y and $\hat{y}_1, \ldots, \hat{y}_k$. To see (3), observe that, by the definition of $\hat{f}$, we have $\hat{y}_{k+1} = y_1 - \hat{y}_1$; and for every $1 < i \leq k$, we also have $\hat{y}_{k+i} = y_i - \hat{y}_i - \sum_{j=1}^{i-1} \hat{y}_{k+j}$.

Hence, define a perfect simulator as follows. Given $y \in \{0,1\}^{k+l}$, the simulator S chooses a random string r of length k, and outputs $(r, s, y_{[k+1\ldots k+l]})$, where $s_1 = y_1 - r_1$ and $s_i = y_i - r_i - \sum_{j=0}^{i-1} s_j$ for $1 < i \leq k$. This simulator is also balanced as each of its outputs is a linear function that contains a fresh random bit. (Namely, the output bit $S(y; r)_i$ depends on: (1) r_i if $1 \leq i \leq k$; or (2) y_{i-k} if $k + 1 \leq i \leq 2k + l$.) $\square$

An additive representation of a function $f : \mathbb{F}_2^n \to \mathbb{F}_2^l$ is a representation in which each output bit is written as as a sum (over $\mathbb{F}_2$) of functions of the input x. That is, each output bit f_i can be written as $f_i(x) = \sum_{a \in T_i} a(x)$, where T_i is a set of boolean functions over n variables. We specify such an additive representation by an l-tuple $(T_1, \ldots, T_l)$ where T_i is a set of boolean functions $a : \mathbb{F}_2^n \to \mathbb{F}_2$. We assume, without loss of generality, that none of the T_i's contains the constant functions 0 or 1. The following measures are defined with respect to a given additive representation of f. For a function $a : \mathbb{F}_2^n \to \mathbb{F}_2$, define the *multiplicity* of a to be the number of T_i's in which a appears, i.e., $\#a = |\{T_i \mid a \in T_i\}|$. For a variable x_j, we define the *rank* of x_j to be the number of different boolean functions a which depend on x_j and appear in some T_i. That is, $\mathrm{rank}(x_j) = |\{a : \mathbb{F}_2^n \to \mathbb{F}_2 \mid a \text{ depends on } x_j, a \in T_1 \bigcup \ldots \bigcup T_l\}|$.

Theorem 2. *Let $f : \mathbb{F}_2^n \to \mathbb{F}_2^l$ be a function, and fix some additive representation $(T_1, \ldots, T_l)$ for f. Then f can be perfectly encoded by a function $\hat{f} : \mathbb{F}_2^n \times \mathbb{F}_2^m \to \mathbb{F}_2^s$ such that the following hold:*

1. *The input locality of every x_j in $\hat{f}$ is at most $\mathrm{rank}(x_j)$, and the input locality of the random inputs r_i of $\hat{f}$ is at most 3.*
2. *If the output locality of f is i, then the output locality of $\hat{f}$ is $\max(i, 2)$.*
3. *The randomness complexity of $\hat{f}$ is $m = \sum_{a \in T} \#a$, where $T = \bigcup_{i=1}^{l} T_i$.*

Proof. We will use the following convention. The additive representation of a function $\hat{g}$ resulting from applying Construction 1 to a function g is the (natural) representation induced by the original additive representation of g. We construct $\hat{f}$ iteratively via the following process. (1) Let $f^{(0)} = f, i = 0$. (2) For $j = 1, \ldots, n$ do the following: (2a) while there exists a function a in $f^{(i)}$ that depends on x_j, whose multiplicity is greater than 1, apply Construction 1 to $f^{(i)}$, let $f^{(i+1)}$ be the resulting encoding and let $i = i + 1$. (3) Let $\hat{f} = f^{(i)}$. By Lemma 2, the function $f^{(i)}$ perfectly encodes the function $f^{(i-1)}$, hence by the composition property of randomized encodings (Lemma 1), the final function $\hat{f}$ perfectly encodes f. The first item of the theorem follows from the following observations: (1) In each iteration the input locality and the rank of each original variable x_j do not increase. (2) The multiplicity in $\hat{f}$ of every function a that depends on some original input variable x_j is 1. (3) The input locality of the random inputs which are introduced by the locality construction is at most 3. The last two items of the theorem follow directly from the definition of Construction 1 and the construction of $\hat{f}$. □

Remarks on Theorem 2.

1. By Theorem 2, every linear function admits an encoding of constant input locality, since each output bit can be written as a sum of degree 1 monomials. More generally, every function f whose canonic representation as a sum of monomials (i.e., each output bit is written as a sum of monomials) includes a constant number of monomials per input bit can be encoded by a function of constant input locality.

2. Interestingly, Construction 1 does not provide a universal encoding for any natural class of functions (e.g., the class of linear functions mapping n bits into l bits). This is contrasted with previous constructions of randomized encoding with constant output locality (cf. [16, 17, 3]). In fact, in Section 6.1 we prove that there is no universal encoding with constant input locality for the class of linear function $L : \mathbb{F}_2^n \to \mathbb{F}_2$.
3. When Theorem 2 is applied to a function family $f_n : \{0,1\}^n \to \{0,1\}^{l(n)}$ then the resulting encoding is uniform whenever the additive representation $(T_1, \ldots, T_l)$ is polynomial-time computable.
4. In Section 6.1, we show that Theorem 2 is tight in the sense that for each integer i we can construct a function f in which the rank of x_1 is i, and in every encoding $\hat{f}$ of f the input locality of x_1 is at least i.

In some cases we can combine Theorem 2 and the output-locality construction from [3, Construction 4.11] to derive an encoding which enjoys low input locality and output locality at the same time. In particular, we will use the following lemma which is implicit in [3].

Lemma 3 (implicit in [3]). *Let $f : \mathbb{F}_2^n \to \mathbb{F}_2^l$ be a function such that each of its output bits can be written as sum of monomials of degree d. Then, we can perfectly encode f by a function $\hat{f}$ such that: (1) The output locality of $\hat{f}$ is $d+1$; (2) The rank of every original variable x_i in $\hat{f}$ is equal to the rank of x_i in f; (3) The new variables introduced by $\hat{f}$ appear only in monomials of degree 1; hence their rank is 1.*

By combining Lemma 3 with Theorem 2 we get:

Corollary 1. *Let $f : \mathbb{F}_2^n \to \mathbb{F}_2^l$ be a function. Fix some additive representation for f in which each output bit is written as a sum of monomials of degree (at most) d and the* rank *of each variable is at most ρ. Then, f can be perfectly encoded by a function $\hat{f}$ of input locality $\max(\rho, 3)$ and output locality $d+1$. Moreover, the resulting encoding is uniform whenever the additive representation is polynomial-time computable.*

Proof. First, by Lemma 3, we can perfectly encode f by a function $f' \in \mathrm{Local}^{d+1}$ without increasing the rank of the input variables of f. Next, we apply Theorem 2 and perfectly encode f' by a function $\hat{f} \in \mathrm{Local}^{d+1}_{\max(\rho,3)}$. By the composition property of randomized encodings (Lemma 1), the resulting function $\hat{f}$ perfectly encodes f. Finally, the proofs of Theorem 2 and Lemma 3 both allow to efficiently transform an additive representation of the function f into an encoding $\hat{f}$ in $\mathrm{Local}^{d+1}_{\max(\rho,3)}$. Hence, the uniformity of f is inherited by $\hat{f}$. □

We remark that Theorem 2 as well as Lemma 3 generalize to any finite field $\mathbb{F}$. Hence, so does Corollary 1.

4 Primitives with Constant Input Locality and Output Locality

4.1 Main Assumption: Intractability of Decoding Random Linear Code

Our positive results are based on the intractability of decoding a random linear code. In the following we introduce and formalize this assumption.

An (m, n, δ) *binary linear code* is a n-dimensional linear subspace of $\mathbb{F}_2^m$ in which the Hamming distance between each two distinct vectors (codewords) is at least δm. We refer to the ratio n/m as the *rate* of the code and to δ as its (relative) *distance*. Such a code can be defined by an $m \times n$ *generator matrix* whose columns span the space of codewords. It follows from the Gilbert–Varshamov bound that whenever $n/m < 1 - \mathrm{H}_2(\delta) - \varepsilon$, almost all $m \times n$ generator matrices form (m, n, δ)-linear codes. Formally,

Fact 3 ([26]). *Let $0 < \delta < 1/2$ and $\varepsilon > 0$. Let $n/m \leq 1 - \mathrm{H}_2(\delta) - \varepsilon$. Then, a randomly chosen $m \times n$ generator matrix generates an (m, n, δ) code with probability $1 - 2^{-(\varepsilon/2)m}$.*

A proof of the above version of the Gilbert–Varshamov bound can be found in [25, Lecture 5].

Definition 2. *Let $m(n) \leq \mathrm{poly}(n)$ be a code length parameter, and $0 < \mu(n) < 1/2$ be a noise parameter. We say that* $\mathrm{CODE}(m, \mu)$ *is* intractable *if for every polynomial-time adversary A,*

$$\Pr[A(C, Cx + e) = x] \leq \mathrm{neg}(n),$$

where C is an $m(n) \times n$ random binary generator matrix, $x \leftarrow U_n$, and $e \in \{0,1\}^m$ is a random error vector in which each entry is chosen to be 1 with probability μ (independently of other entries), and arithmetic is over $\mathbb{F}_2$.

Typically, we let $m(n) = O(n)$ and μ be a constant such that $n/m(n) < 1 - \mathrm{H}_2(\mu + \varepsilon)$ where $\varepsilon > 0$ is a constant. Hence, by Fact 3, the random code C is, with overwhelming probability, an $(m, n, \mu + \varepsilon)$ code. Note that, except with negligible probability, the noise vector flips less than $\mu + \varepsilon$ of the bits of y. In this case, the fact that the noise is random (rather than adversarial) guarantees, by Shannon's coding theorem (for random linear codes), that x will be unique with overwhelming probability. That is, roughly speaking, we assume that it is intractable to correct μn *random* errors in a random linear code of relative distance $\mu + \varepsilon > \mu$. The plausibility of such an assumption is supported by the fact that a successful adversary would imply a major breakthrough in coding theory. Similar assumptions were put forward in [13, 5, 10].

We will rely on the following Lemma of [5].

Lemma 4. *Let $m(n)$ be a code length parameter, and $\mu(n)$ be a noise parameter. If $\mathrm{CODE}(m, \mu)$ is intractable then the distribution $(C, Cx+e)$ is pseudorandom, where C, x and e are as in Definition 2.*

4.2 Pseudorandom Generator in Local_3^3

A pseudorandom generator (PRG) is an efficiently computable function G which expands its input and its output distribution $G(U_n)$ is pseudorandom. An efficiently computable collection of functions $\{G_z\}_{z\in\{0,1\}^*}$ is a PRG collection if for every z, the function G_z expands its input and the pair $(z, G_z(x))$ is pseudorandom for random x and z. We show that pseudorandom generators (and therefore also one-way functions and one-time symmetric encryption schemes) can be realized by $\mathrm{Local}_{O(1)}^{O(1)}$ functions. Specifically, we get a PRG in Local_3^3. In the full version we also show that such a PRG has optimal output locality and optimal input locality. We rely on the following assumption.

Assumption 4. *The problem* $\mathrm{CODE}(13n, 1/4)$ *is intractable.*

Note that the code considered here is of rate $n/m = 1/13$ which is strictly smaller than $1-H_2(\frac{1}{3})$. Therefore, except with negligible probability, its relative distance is at least $\frac{1}{3}$. Hence the above assumption roughly says that it is intractable to correct $n/4$ random errors in a random linear code of relative distance $\frac{1}{3}$. (We did not attempt to optimize the constant 13 in the above.)

Let $m(n) = 13n$. Let $C \leftarrow U_{m(n)\times n}$, $x \leftarrow U_n$ and $e \in \{0,1\}^m$ be a random error vector of rate $1/4$, that is, each of the entries of e is 1 with probability $1/4$ (independently of the other entries). By Lemma 4, the distribution $(C, Cx+e)$ is pseudorandom under the above assumption. Since the noise rate is $1/4$, it is natural to sample the noise distribution e by using $2m$ random bits $r_1, \dots, r_{2m}$ and letting the i-th bit of e be the product of two fresh random bits, i.e., $e_i = r_{2i-1}\cdot r_{2i}$. We can now define the mapping $f(C, x, r) = (C, Cx + e(r))$ where $e(r) = (r_{2i-1}\cdot r_{2i})_{i=1}^m$. The output distribution of f is pseudorandom, however, f is not a PRG since it does not expand its input. In [4], it was shown how to bypass this problem by applying a randomness extractor. Namely, the following function was shown to be a PRG: $G(C, x, r, s) = (C, Cx + e(r), \mathrm{Ext}(r, s))$. Although the setting of parameters in [4] is different than ours, a similar solution works here as well. We rely on the leftover hashing lemma of [15] and base our extractor on a family of pairwise independent hash functions (which is realized by the mapping $x \mapsto Ax + b$ where A is a random matrix and b is a random vector).[3]

Construction 5. *Let* $m = 13n$ *and let* $t = \lceil 1.1\cdot m\rceil$. *Define the function*

$$G(x, C, r, A, b) \stackrel{def}{=} (C, Cx + e(r), Ar + b, A, b),$$

where $x \in \{0,1\}^n$, $C \in \{0,1\}^{m\times n}$, $r \in \{0,1\}^{2m}$, $A \in \{0,1\}^{t\times 2m}$, *and* $b \in \{0,1\}^t$.

Theorem 6. *Under Assumption 4, the function G defined in Construction 5 is a PRG.*

[3] We remark that in [4] one had to rely on a specially made extractor in order to maintain the large stretch of the PRG. In particular, the leftover hashing lemma could not be used there.

The proof of the above theorem is deferred to the full version of this paper. From now on, we fix the parameters m, t according to Construction 5. We can redefine the above construction as a collection of PRGs by letting C, A, b be the keys of the collection. Namely,

$$G_{C,A,b}(x, r) = (Cx + e(r), Ar + b).$$

We can now prove the main theorem of this section.

Theorem 7. *Under Assumption 4, there exists a collection of pseudorandom generators* $\{G_z\}_{z \in \{0,1\}^{p(n)}}$ *in* Local_3^3*. Namely, for every* $z \in \{0,1\}^{p(n)}$*, it holds that* $G_z \in \mathrm{Local}_3^3$*.*

Proof. Fix C, A, b and write each output bit of $G_{C,A,b}(x, r)$ as a sum of monomials. Note that in this case, each variable x_i appears only in degree 1 monomials, and each variable r_i appears only in the monomial $r_{2i-1}r_{2i}$ and also in degree 1 monomials. Hence, the rank of each variable is at most 2. Moreover, the (algebraic) degree of each output bit of $G_{C,A,b}$ is at most 2. Therefore, by Corollary 1, we can perfectly encode the function $G_{C,A,b}$ by a function $\hat{G}_{C,A,b}$ in Local_3^3. In [3, Lemma 6.1] it was shown that a uniform perfect encoding of a PRG is also a PRG. Thus, we get a collection of PRGs in Local_3^3. $\square$

We can rely on Theorem 7 to obtain a one-time semantically-secure symmetric encryption scheme (E, D) whose encryption algorithm is in Local_3^3 (see [2, Construction 4.3]). (This scheme allows to encrypt an arbitrary polynomially long message with a short key.) A similar approach can be also used to give multiple message security, at the price of requiring the encryption and decryption algorithms to maintain a synchronized *state*. The results of Section 4.4 give a direct construction of public-key encryption (hence also symmetric encryption) with constant input locality under the stronger assumption that the McEliece cryptosystem is one-way secure.

4.3 Commitment in Local_3^4

We will consider a non-interactive commitment scheme in the common reference string (CRS) model. In such a scheme, the sender and the receiver share a common public random key k (that can be selected once and be used in many invocations of the scheme). To commit to a bit b, the sender computes the commitment function $\mathrm{C{\scriptsize OM}}_k(b, r)$ that outputs a commitment c using the randomness r, and sends the output to the receiver. To open the commitment, the sender sends the randomness r and the committed bit b to the receiver who checks whether the opening is valid by computing the function $\mathrm{R{\scriptsize EC}}_k(c, b, r)$. The scheme should be both (computationally) hiding and (statistically) binding. Hiding requires that $c = \mathrm{C{\scriptsize OM}}_k(b, r)$ keep b computationally secret. Binding means that, except with negligible probability over the choice of the random public key, it is impossible for the sender to open its commitment in two different ways.

We construct a commitment scheme in Local_3^4, i.e., a commitment of input locality 3 and output locality 4. Let c be a constant that satisfies $c > \frac{1}{1-H_2(1/4)}$. Let $m = m(n) = \lceil cn \rceil$. Then, by Fact 3, a random $m \times n$ generator matrix generates, except with negligible probability (i.e., $2^{-\Omega(m)} = 2^{-\Omega(n)}$), a code whose relative distance is $1/4 + \varepsilon$, for some constant $\varepsilon > 0$. The public key of our scheme will be a random $m(n) \times n$ generator matrix C. To commit to a bit b, we first choose a random information word $x \in \{0,1\}^n$ and hide it by computing $Cx + e$, where $e \in \{0,1\}^m$ is a noise vector of rate $1/8$, and then take the exclusive-or of b with a hardcore bit $\beta(x)$ of the above function. That is, we send the receiver the value $(Cx + e, b + \beta(x))$. In particular, we can use the Goldreich-Levin [14] hardcore bit and get

$$\mathrm{Com}_C(b, (x, r, s)) = (Cx + e(r), s, b + \langle x, s \rangle),$$

where r is a random $3m$-bit string, $e(r) = (r_1 r_2 r_3, r_4 r_5 r_6, \ldots, r_{3m-2} r_{3m-1} r_{3m})$, s is a random n-bit string and $\langle \cdot, \cdot \rangle$ denotes inner product (over $\mathbb{F}_2$). Assuming that $\mathrm{CODE}(m, 1/8)$ is intractable, this commitment hides the committed bit b. (This is so because $\langle x, s \rangle$ is unpredictable given $(C, Cx + e, s)$, cf. [10, Construction 4.4.2].) Suppose that the relative distance of C is indeed $1/4 + \varepsilon$. Then, if e contains no more than $1/8 + \varepsilon/2$ ones, x is uniquely determined by $Cx + e$. Of course, the sender might try to cheat and open the commitment ambiguously by claiming that the weight of the error vector is larger than $1/8 + \varepsilon/2$. Hence, we let the receiver verify that the Hamming weight of the noise vector e given to him by the sender in the opening phase is indeed smaller than $1/8 + \varepsilon/2$. This way, the receiver will always catch a cheating sender (assuming that C is indeed a good code). Note that an honest sender will be rejected only if its randomly chosen noise vector is heavier than $1/8 + \varepsilon/2$, which, by a Chernoff bound, happens with negligible probability (i.e., $e^{-\Omega(m)} = e^{-\Omega(n)}$) as the noise rate is $1/8$. Hence, the pair $(\mathrm{Com}, \mathrm{Rec})$ defined above is indeed a commitment scheme. When C is fixed, the rank and algebraic degree of the function Com_C are 2 and 3 (with respect to the natural representation as a sum of monomials). Hence, by Corollary 1, we can encode Com_C by a function $\hat{\mathrm{Com}}_C \in \mathrm{Local}_3^4$. By [3], this encoding is also a commitment scheme. Summarizing, we have:

Theorem 8. *Let c be a constant that satisfies $c > \frac{1}{1-H_2(1/4)}$, and $m = m(n) = \lceil cn \rceil$. If $\mathrm{CODE}(m, 1/8)$ is intractable, then there exists a commitment scheme $(\mathrm{Com}, \mathrm{Rec})$ in Local_3^4; i.e., for every public key C, we have $\mathrm{Com}_C \in \mathrm{Local}_3^4$.*

We remark that we can eliminate the use of the CRS by letting C be a generator matrix of some fixed error correcting error whose relative distance is large (i.e., $1/4$ or any other constant) in which decoding is intractable. For example, one might use the dual of a BCH code.

4.4 Semantically Secure Public-Key Encryption in $\mathrm{Local}_3^{O(1)}$

We construct a semantically-secure public-key encryption scheme (PKE) whose encryption algorithm is in $\mathrm{Local}_{O(1)}^{O(1)}$. Our scheme is based on the McEliece cryptosystem [21]. We begin by reviewing the general scheme proposed by McEliece.

- **System parameters:** Let $m(n) : \mathbb{N} \to \mathbb{N}$, where $m(n) > n$, and $\mu(n) : \mathbb{N} \to (0, 1)$. For every $n \in \mathbb{N}$, let $\mathcal{C}_n$ be a set of generating matrices of $(m(n), n, 2(\mu(n) + \varepsilon))$ codes that have a (universal) efficient decoding algorithm D that, given a generating matrix from $\mathcal{C}_n$, can correct up to $(\mu(n) + \varepsilon) \cdot m(n)$ errors, where $\varepsilon > 0$ is some constant. We also assume that there exists an efficient sampling algorithm that samples a generator matrix of a random code from $\mathcal{C}_n$.
- **Key Generation:** Given a security parameter 1^n, use the sampling algorithm to choose a random code from $\mathcal{C}_n$ and let C be its generating matrix. Let $m = m(n)$ and $\mu = \mu(n)$. Choose a random $n \times n$ non-singular matrix S over $\mathbb{F}_2$, and a random $m \times m$ permutation matrix P. Let $C' = P \cdot C \cdot S$ be the public key and P, S, D_C be the private key where D_C is the efficient decoding algorithm of C.
- **Encryption:** To encrypt $x \in \{0, 1\}^n$ compute $c = C'x + e$ where $e \in \{0, 1\}^m$ is an error vector of noise rate μ.
- **Decryption:** To decrypt a ciphertext c, compute $P^{-1}y = P^{-1}(C'x + e) = CSx + P^{-1}e = CSx + e'$ where e' is a vector whose weight equals to the weight of e (since P^{-1} is also a permutation matrix). Now, use the decoding algorithm D to recover the information word Sx (i.e., $D(C, CSx + P^{-1}e) = Sx$). Finally, to get x multiply Sx on the left by S^{-1}.

By Chernoff bound, the weight of the error vector e is, except with negligible probability, smaller than $(\mu + \varepsilon) \cdot m$ and so the decryption algorithm almost never errs.[4] As for the security of the scheme, it is not hard to see that the scheme is *not* semantically secure. (For example, it is easy to verify that a ciphertext c is an encryption of a given plaintext x by checking whether the weight of $c - Cx$ is approximately μn.)

However, the scheme is conjectured to be a one-way cryptosystem; namely, it is widely believed that, for proper choice of parameters, any efficient adversary fails with probability $1 - \text{neg}(n)$ to recover x from $(c = C'x + e, C')$ where x is a random n-bit string.

Suppose that the scheme is indeed one-way with respect to the parameters $m(n), \mu(n)$ and $\mathcal{C}_n$. Then, we can convert it into a semantically secure public-key encryption scheme by extracting a hardcore predicate and xoring it with a 1-bit plaintext b (this transformation is similar to the one used for commitments in the previous section). That is, we encrypt the bit b by the ciphertext $(C'x + e, s, \langle s, x \rangle + b)$ where x, s are random n-bit strings, and e is a noise vector of rate μ. (Again, we use the Goldreich-Levin hardcore predicate [14].) To decrypt the message, we first compute x, by invoking the McEliece decryption algorithm, and then compute $\langle s, x \rangle$ and xor it with the last entry of the ciphertext. We refer to this scheme as the *modified* McEliece public-key encryption scheme. If the McEliece cryptosystem is indeed one-way, then $\langle s, x \rangle$ is pseudorandom given

[4] In fact, we may allow ε to decrease with n. In such case, we might get a non-negligible decryption error. This can be fixed (without increasing the rank or the degree of the encryption function) by repeating the encryption with independent fresh randomness. Details omitted.

$(C', C'x+e, s)$, and thus the modified McEliece public-key is semantically secure. Formally,

Lemma 5. *If the McEliece cryptosystem is one-way with respect to the parameters $m(n), \mu(n)$ and $\mathcal{C}_n$, then the* modified *McEliece PKE is semantically secure with respect to the same parameters.*

The proof of this lemma is essentially the same as the proof of [11, Prop. 5.3.14].

Let $\mu(n) = 2^{-t(n)}$. Then, we can sample the noise vector e by using the function $e(r) = \left(\prod_{j=1}^{t} r_{t\cdot(i-1)+j}\right)_{i=1}^{m(n)}$ where r is a $t(n) \cdot m(n)$ bit string. In this case, we can write the encryption function of the modified McEliece as $E_{C'}(b, x, r, s) = (C'x + e(r), s, \langle x, s\rangle + b)$.

The rank of each variable of this function is at most 2, and its algebraic degree is at most $t(n)$. Hence, by Corollary 1, we can encode it by a function $\hat{E} \in \mathrm{Local}_3^{t(n)+1}$, i.e., the output locality of $\hat{E}$ is $t(n)+1$ and its input locality is 3. In [3, Lem. 7.5] it was shown that randomized encoding preserves the security of PKE. Namely, if (G, E, D) is a semantically secure PKE then $(G, \hat{E}, \hat{D})$ is also an encryption scheme where $\hat{E}$ is an encoding of E, $\hat{D}(c) = D(B(c))$ and B is the decoder of the encoding. Hence we have,

Theorem 9. *If the McEliece cryptosystem is one-way with respect to to the parameters $m(n), \mu(n) = 2^{-t(n)}$ and $\mathcal{C}_n$, then there exists a semantically secure PKE whose encryption algorithm is in $\mathrm{Local}_3^{t(n)}$.*

The scheme we construct encrypts a single bit, however we can use concatenation to derive a PKE for messages of arbitrary (polynomial) length without increasing the input and output locality. Theorem 9 gives a PKE with constant output locality whenever the noise rate μ is constant. Unfortunately, the binary classical Goppa Codes, which are commonly used with the McEliece scheme [21], are known to have an efficient decoding only for *subconstant* noise rate. Hence, we cannot use them for the purpose of achieving constant output locality and constant input locality simultaneously. Instead, we suggest using algebraic-geometric (AG) codes which generalize the classical Goppa Codes and enjoy an efficient decoding algorithm for constant noise rate. It seems that the use of such codes does not decrease the security of the McEliece cryptosystem [18].

5 Negative Results for Cryptographic Primitives

In this section we show that cryptographic tasks which require some form of "non-malleability" cannot be performed by functions with low input locality. This includes MACs, signatures and non-malleable encryption schemes (e.g., CCA2 secure encryptions). We prove our results in the private-key setting (i.e., for MAC and symmetric encryption). This makes them stronger as any construction that gains security in the public-key setting is also secure in the private-key setting.

We will use the following simple observation.

Lemma 6. *Let $f : \{0,1\}^n \rightarrow \{0,1\}^{s(n)}$ be a function in* $\mathrm{Local}_{l(n)}$*. Then, there exist a (probabilistic) polynomial-size circuit family $\{A_n\}$ such that for every $x \in \{0,1\}^n$ and $i \in [n]$, the output of A_n on $(y = f(x), i, 1^n)$ equals, with probability $2^{-l(n)}$, to the string $y' = f(x')$ where x' differs from x only in the i-th bit. In particular, when $l(n) = O(\log(n))$, the success probability of A_n is* $1/\mathrm{poly}(n)$.

Proof. Since f is in $\mathrm{Local}_{l(n)}$, the input variable x_i affects at most $l(n)$ output bits. Hence, y and y' differ in at most $l(n)$ bits. Thus, we can randomly choose y' from a set of strings whose size is at most $2^{l(n)}$. (We assume that the set of output bits which are affected by the i-th input bit is hardwired into A_n.) □

In the full version we show how to get rid of the non-uniformity when f is polynomial-time computable. We now sketch the impossibility results.

5.1 MACs and Signatures

Let (S, V) be a MAC scheme, where the randomized signing function $S(k, \alpha, r)$ computes a signature β on the document α using the (random and secret) key k and randomness r, and the verification algorithm $V(k, \alpha, \beta)$ verifies that β is a valid signature on α using the key k. The scheme is secure (unforgeable) if it is infeasible to forge a signature in a chosen message attack. Namely, any efficient adversary that gets an oracle access to the signing process $S(s, \cdot)$ fails to produce a valid signature β on a document α (with respect to the corresponding key k) for which it has not requested a signature from the oracle.[5] The scheme is one-time secure if the adversary is allowed to query the signing oracle only once.

Suppose that the signature function $S(k, \alpha, r)$ has logarithmic input locality (i.e., $S(k, \alpha, r) \in \mathrm{Local}_{O(\log(|k|))}$). Then, by Lemma 6, we can break the scheme by transforming, with noticeable probability, a valid pair (α, β) of document and signature into a valid pair (α', β') for which α' and α differ in, say, their first bit. Since we used a single oracle call, such a scheme cannot be even one-time secure.

Now, suppose that for each *fixed* key k the signature function $S_k(\alpha, r) = S(k, \alpha, r)$ has input locality $\ell(n)$. In this case we cannot use Lemma 6 directly as we do not know which output bits are affected by the i-th input bit. When $\ell(n) = c$ is constant, we can easily overcome this problem. We guess which bits are affected by, say, the first input bit and then guess their value as in Lemma 6. This attack succeeds with probability $1/(m^c \cdot 2^c) = 1/\mathrm{poly}(n)$ where m is the length of the message (and so is polynomial in n). Again, this shows that the scheme is not even one-time secure. To summarize:

Theorem 10. *Let (S, V) be a MAC scheme. If $S(k, \alpha, r) \in \mathrm{Local}_{O(\log(|k|))}$ or $S_k(\alpha, r) \in \mathrm{Local}_{O(1)}$ for every k, then the scheme is not one-time secure.*

[5] When querying the signing oracle, the adversary chooses only the message and is not allowed to choose the randomness which the oracle uses to produce the signature.

5.2 Non-Malleable Encryption

Let (E, D) be a private-key encryption scheme, where the encryption function $E(k, m, r)$ computes a ciphertext c encrypting the message m using the (random and secret) key k and randomness r, and the decryption algorithm $D(k, c, r)$ decrypts the ciphertext c that was encrypted under the key k. Roughly speaking, non-malleability of an encryption scheme guarantees that it is infeasible to modify a ciphertext c into a ciphertext c' of a message related to the decryption of c. In the full version we prove the following theorem:

Theorem 11. *Let (E, D) be a private-key encryption scheme. If $E(k, m, r) \in \mathrm{Local}_{O(\log(|k|))}$ or $E_k(m, r) \in \mathrm{Local}_{O(1)}$ for every k, then the scheme is malleable with respect to an adversary that has no access to neither the encryption oracle nor the decryption oracle. If (G, E, D) is a public-key encryption scheme and $E_k(m, r) \in \mathrm{Local}_{O(\log(|k|))}$ for every k, then the scheme is malleable.*

6 Negative Results for Randomized Encodings

In the following, we prove some negative results regarding randomized encoding with low input locality. In Section 6.1, we provide a necessary condition for a function to have such an encoding. We use this condition to prove that some simple (NC^0) functions cannot be encoded by functions having sub-linear input locality (regardless of the complexity of the encoding). This is contrasted with the case of constant output locality, where it is known [17, 3] that *every* function f can be encoded by a function $\hat{f}$ whose output locality is 4 (and whose complexity is polynomial in the size of the branching program that computes f). In Section 6.2 we show that, although linear functions do admit efficient constant-input encoding, they do not admit an efficient *universal* constant-input encoding. That is, one should use different decoders and simulators for each linear function.

6.1 A Necessary Condition for Encoding with Low Input Locality

Let $f : \{0,1\}^n \to \{0,1\}^l$ be a function. For a string $x \in \{0,1\}^n$, let $x_{\oplus i}$ denote the string x with the i-th bit flipped. Define an undirected graph G_i over $\mathrm{Im}(f)$ such that there is an edge between the strings y and y' if there exists $x \in \{0,1\}^n$ such that $f(x) = y$ and $f(x_{\oplus i}) = y'$. Let $\hat{f} : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^s$ be a (perfectly correct and private) randomized encoding of f with decoder B and simulator S. Let t_i be the number of output bits in $\hat{f}$ which are affected by the input variable x_i. We rely on the following lemma whose proof is omitted and deferred to the full version of this paper.

Lemma 7. *The size of each connected component of G_i is at most 2^{t_i}.*

We conclude that a function $f : \{0,1\}^n \to \{0,1\}^l$ can be perfectly encoded by a function $\hat{f} : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^s$ in Local_t only if for every $1 \le i \le n$ the size of the connected components of G_i is at most 2^t. This shows that even some very simple functions do not admit an encoding with constant input locality. Consider, for example, the function

$$f(x_1, \dots, x_n) = x_1 \cdot (x_2, \dots, x_n) = (x_1 \cdot x_2, x_1 \cdot x_3, \dots, x_1 \cdot x_n).$$

For every $y \in \mathrm{Im}(f) = \{0,1\}^{n-1}$ it holds that $f(1,y) = y$ and $f(0,y) = 0^{n-1}$. Hence, every vertex in G_1 is a neighbor of 0^{n-1} and the size of the connected component of G_1 is 2^{n-1}. Thus, the input locality of x_1 in any perfect encoding of this function is $n-1$. (Note that this matches the results of Section 3 since $\mathrm{rank}(x_1) = n-1$.)

6.2 Impossibility of Universal Encoding for Linear Functions

For a class C of functions that map n-bits into l-bits, we say that C has a universal encoding in the class $\hat{C}$ if there exists a universal simulator S and a universal decoder B such that, for every function $f_z \in C$, there is an encoding $\hat{f}_z \in \hat{C}$ which is private and correct with respect to the simulator S and the decoder B.

We show that, although linear functions do admit encodings with constant input locality, they do not admit such a *universal* encoding. Suppose that the class of linear (equivalently affine) functions had a universal encoding with constant input locality. Then, by the results of [3], we would have a one-time secure MACs (S, V) whose signing algorithm has constant input locality for every fixed key; i.e., $S_k(\alpha, r) \in \mathrm{Local}_{O(1)}$ for every fixed key k. However, the results of Section 5.1 rule out the existence of such a scheme. In the full version of this paper, we give a more direct proof to the impossibility of obtaining a universal constant-input encoding for linear functions. This proof is based on the notions presented in Section 6.1.

Acknowledgments. We thank Ronny Roth for helpful discussions.

References

1. M. Alekhnovich. More on average case vs approximation complexity. In *Proc. 44th FOCS*, pages 298–307, 2003.
2. B. Applebaum, Y. Ishai, and E. Kushilevitz. Computationally private randomizing polynomials and their applications. *Computional Complexity*, 15(2):115–162, 2006.
3. B. Applebaum, Y. Ishai, and E. Kushilevitz. Cryptography in NC^0. *SIAM J. Comput.*, 36(4):845–888, 2006.
4. B. Applebaum, Y. Ishai, and E. Kushilevitz. On pseudorandom generators with linear stretch in NC^0. In *Proc. 10th Random.*, 2006.
5. A. Blum, M. Furst, M. Kearns, and R. J. Lipton. Cryptographic primitives based on hard learning problems. In *Advances in Cryptology: Proc. of CRYPTO '93*, volume 773 of *LNCS*, pages 278–291, 1994.
6. S. A. Cook. The complexity of theorem-proving procedures. In *STOC '71: Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158, New York, NY, USA, 1971. ACM Press.

7. M. Cryan and P. B. Miltersen. On pseudorandom generators in NC^0. In *Proc. 26th MFCS*, pages 272–284, 2001.
8. U. Feige, J. Killian, and M. Naor. A minimal model for secure computation (extended abstract). In *Proc. of the 26th STOC*, pages 554–563, 1994.
9. O. Goldreich. Candidate one-way functions based on expander graphs. *Electronic Colloquium on Computational Complexity (ECCC)*, 7(090), 2000.
10. O. Goldreich. *Foundations of Cryptography: Basic Tools.* Cambridge University Press, 2001.
11. O. Goldreich. *Foundations of Cryptography: Basic Applications.* Cambridge University Press, 2004.
12. O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *J. of the ACM.*, 33:792–807, 1986.
13. O. Goldreich, H. Krawczyk, and M. Luby. On the existence of pseudorandom generators. *SIAM J. Comput.*, 22(6):1163–1175, 1993.
14. O. Goldreich and L. Levin. A hard-core predicate for all one-way functions. In *Proc. 21st STOC*, pages 25–32, 1989.
15. R. Impagliazzo, L. A. Levin, and M. Luby. Pseudorandom generation from one-way functions. In *Proc. 21st STOC*, pages 12–24, 1989.
16. Y. Ishai and E. Kushilevitz. Randomizing polynomials: A new representation with applications to round-efficient secure computation. In *Proc. 41st FOCS*, pages 294–304, 2000.
17. Y. Ishai and E. Kushilevitz. Perfect constant-round secure computation via perfect randomizing polynomials. In *Proc. 29th ICALP*, pages 244–256, 2002.
18. H. Janwa and O. Moreno. Mceliece public key cryptosystems using algebraic-geometric codes. *Des. Codes Cryptography*, 8(3):293–307, 1996.
19. J. Kilian. Founding cryptography on oblivious transfer. In *Proc. 20th STOC*, pages 20–31, 1988.
20. N. Linial, Y. Mansour, and N. Nisan. Constant depth circuits, fourier transform, and learnability. *J. ACM*, 40(3):607–620, 1993.
21. R. J. McEliece. A public-key cryptosystem based on algebraic coding theory. Technical Report DSN PR 42-44, Jet Prop. Lab., 1978.
22. E. Mossel, A. Shpilka, and L. Trevisan. On ϵ-biased generators in NC^0. In *Proc. 44th FOCS*, pages 136–145, 2003.
23. M. Naor and O. Reingold. Synthesizers and their application to the parallel construction of pseudo-random functions. *J. of Computer and Systems Sciences*, 58(2):336–375, 1999.
24. C. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. *J. of Computer and Systems Sciences*, 43:425–440, 1991.
25. M. Sudan. Algorithmic introduction to coding theory - lecture notes, 2002. http://theory.csail.mit.edu/∼madhu/FT01/.
26. R. Varshamov. Estimate of the number of signals in error correcting codes. *Doklady Akademii Nauk SSSR*, 117:739–741, 1957.