J. Cryptol. (2009) 22: 429–469
DOI: 10.1007/s00145-009-9039-0

Journal of
CRYPTOLOGY

Cryptography with Constant Input Locality[*],[†]

Benny Applebaum[‡]
Department of Computer Science, Princeton University, Princeton, USA
benny.applebaum@gmail.com

Yuval Ishai[§]
Department of Computer Science, Technion, Haifa, Israel
yuvali@cs.technion.ac.il
and
UCLA, Los Angeles, CA, USA

Eyal Kushilevitz[¶]
Department of Computer Science, Technion, Haifa, Israel
eyalk@cs.technion.ac.il

Communicated by I. Damgaard

Received 27 November 2007 and revised 10 March 2009
Online publication 11 April 2009

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 by 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 binary 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

[*] This paper was solicted by the Editors-in-Chief as one of the best papers from EUROCRYPT 2008, based on the recommendation of the program committee.

[†] Research supported by grant 1310/06 from the Israel Science Foundation.

[‡] Supported by NSF grants CNS-0627526, CCF-0426582 and CCF-0832797. Most of this work done while studying in the Technion.

[§] Supported by BSF grant 2004361 and NSF grants 0205594, 0430254, 0456717, 0627781, 0716389.

[¶] Supported by grant 2002354 from the U.S.-Israel Binational Science Foundation.

enjoy constant *output locality* and thus 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).

Key words. Cryptography with low complexity, Input locality, NC^0, Hardness of decoding random linear code

1. Introduction

The question of minimizing the complexity of cryptographic primitives has been the subject of an extensive body of research (see [3,46] 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 constant number of input bits.[1] Some primitives, including pseudorandom *functions* [24], cannot even be realized in AC^0 [42]; no similar negative results are known for other primitives. However, it was shown recently [2,3] 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., probabilistic encryption)?
- (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) The possibility of cryptography in NC^0 is closely related to the intractability of Constraint Satisfaction Problems in which each constraint involves a constant number of variables (k-CSPs). (The k-CSP problem generalizes the well known k-SAT problem by allowing each clause to specify an arbitrary constraint on k input variables, as opposed to being restricted to a disjunction of literals in the case of k-SAT.) Constraint satisfaction problems are well studied in complexity theory and are known to be "hard" in several aspects. In particular, the Cook–Levin theorem [15,41] shows that it is NP-hard to exactly solve 3-CSP problems, while the PCP theorem [6,7] shows that it is NP-hard even

[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. In AC^0 circuits, we allow unbounded fan-in AND and OR gates, and in NC^1 circuits the depth is logarithmic.

to find an approximate solution. The existence of cryptography in NC^0 can be interpreted as a cryptographic version of these intractability results. Similarly, one can formulate the question of cryptography with constant input locality in terms of CSPs with bounded occurrences of each variable. It is known that NP hardness and inapproximability results can be carried from the general CSP setting to the setting of CSPs with bounded occurrences [15,47], 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 output locality 4 and the impossibility of cryptography with output locality 2. It was shown in [3] that the existence of a OWF with output locality 3 follows from the intractability of decoding a random binary linear code. The possibility of closing this gap for other primitives remained open.

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 (such as signatures, MACs, and non-malleable encryption schemes [17]) *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 conjectured intractability of decoding a random binary linear code. Previous constructions of PRGs (or even OWFs) which enjoyed constant input locality and constant output locality at the same time [4,20], were based on non-standard intractability assumptions.
- We construct a collection of non-interactive commitment schemes, 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 binary linear code. (We can also get a standard non-interactive commitment scheme under the assumption that there exists an explicit binary linear code that has a large minimal distance but is hard to decode.)

[2] A collection of PRGs is a PRG indexed by a public random key. A bit more precisely, $\{G_z\}_{z\in\{0,1\}^*}$ 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. We say that the (input or output) locality of the collection is c if for every z the function G_z has locality c. See [3], Appendix A for a more general and detailed discussion of collections of cryptographic primitives.

- 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 [44], an assumption which is related to the intractability of decoding a random binary 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.

Locality-Preserving Reductions We also present new locality-preserving reductions between different cryptographic primitives. (Unlike the results discussed above, here we consider *unconditional* reductions that do not rely on unproven assumptions.) Specifically, we get new locality-preserving constructions of one-time symmetric encryption scheme, non-interactive commitment, and (collection of) PRG from one-to-one OWF. (In fact, in the case of PRG the reduction holds even with more general types of one-way functions.) These reductions preserve both the input locality and the output locality of the underlying primitive up to an additive constant and extend the output locality preserving reductions of [2,3].

1.2. *Our Techniques*

Our constructions rely on the machinery of *randomized encoding*, which was explicitly introduced in [32] (under the algebraic framework of *randomizing polynomials*) and was implicitly used, in weaker forms, in the context of secure multiparty computation (e.g., [18,40]). 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 [2,3].[3] For example, it was shown that STRONG can be NC^1 and WEAK can be the class of functions whose output locality is 4.

[3] A different randomization approach, which was used for constructing parallel PRGs in [31,51], is to exploit the fact that generating random solved instances of a problem is sometimes easier than solving it on a given instance [8,14]. In contrast, randomized encodings are not sensitive to the input distribution and can be applied to any fixed instance.

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 by functions 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 these 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 binary 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 Sect. 6.2 for a discussion.

1.3. *Previous Work*

The existence of cryptographic primitives in NC^0 has been recently studied in [3,16,45]. Goldreich observed that a function whose output locality is 2 cannot even be one-way [20]. Cryan and Miltersen [16] 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. [45] extended this impossibility to functions whose output locality is 4.

On the positive side, Goldreich [20] 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. [45] constructed (non-cryptographic) ε-biased generators with (non-optimal) constant input and output locality. Applebaum et al. [3] subsequently showed that: (1) the existence of many cryptographic primitives (including OWFs, PRGs, encryption schemes, signatures and hash functions) in NC^1, or even in $\oplus\mathrm{L}/poly$, 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 binary 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).

Organization The rest of this paper is structured as follows. Section 2 contains some preliminaries including the definition and properties of randomized encoding (Sect. 2.1) as well as standard definitions of cryptographic primitives (Sect. 2.2). In Sect. 3, we construct a randomized encoding with constant input locality for functions with a "simple" algebraic structure. This construction is used in Sect. 4 to derive cryptographic primitives with low locality under coding related assumptions as well as unconditional locality-preserving cryptographic reductions between different primitives. In Sect. 5, we prove that MACs and non-malleable encryption schemes cannot be implemented with low input locality. Negative results for randomized encoding with low input locality are discussed in Sect. 6.

2. Preliminaries

Notation All logarithms in this paper are to the base 2. For a positive integer n, denote by $[n]$ the set $\{1, \dots, n\}$. For a string $x \in \{0,1\}^*$, let $|x|$ denote the length of x. For a string $x \in \{0,1\}^n$ and an integer $i \in [n]$, let x_i denote the ith bit of x. Similarly, for $S \subseteq [n]$, let x_S denote the restriction of x to the indices in S. We will write $x_{\oplus i}$ to denote the string x with the ith bit flipped. We will sometimes abuse notation and identify binary strings with vectors over $\mathbb{F}_2$. All vectors will be regarded by default as column vectors. Let $\langle \cdot, \cdot \rangle$ denote inner product over $\mathbb{F}_2$, i.e., for $x, y \in \mathbb{F}_2^n$, $\langle x, y \rangle = \sum_{i=1}^n x_i \cdot y_i$ where arithmetic is over $\mathbb{F}_2$. Let U_n denote a random variable uniformly distributed over $\{0,1\}^n$. If X is a probability distribution, or a random variable, we write $x \leftarrow X$ to indicate that x is a sample taken from X. Let $\mathrm{H}_2(\cdot)$ denote the binary entropy function, i.e., for $0 < p < 1$, $\mathrm{H}_2(p) \overset{\text{def}}{=} -p\log(p) - (1-p)\log(1-p)$. 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. A *distribution ensemble* $\{X_n\}_{n\in\mathbb{N}}$ is an infinite sequence of distributions, where the support of each distribution X_n consists of binary strings of some fixed length $m(n)$. 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 $X_n \overset{\mathrm{s}}{\equiv} Y_n$ 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 $X_n \overset{\mathrm{c}}{\equiv} Y_n$ 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. By definition, $X_n \equiv Y_n$ implies that $X_n \overset{\mathrm{s}}{\equiv} Y_n$ which in turn implies that $X_n \overset{\mathrm{c}}{\equiv} Y_n$. A distribution ensemble $\{X_n\}_{n\in\mathbb{N}}$ is said to be *pseudorandom* if $X_n \overset{\mathrm{c}}{\equiv} U_{m(n)}$ 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 $c(n)$ if for every n the restriction of f to n-bit inputs has output locality (resp., input locality) $c(n)$. We envision circuits as having their inputs at the bottom and their outputs at the top. Hence, for functions $\mathsf{in}(n), \mathsf{out}(n)$, we let $\mathrm{Local}_{\mathsf{in}(n)}^{\mathsf{out}(n)}$ (resp., $\mathrm{Local}_{\mathsf{in}(n)}$, $\mathrm{Local}^{\mathsf{out}(n)}$) denote the non-uniform class which includes all functions $f : \{0,1\}^* \to \{0,1\}^*$ whose input locality is $\mathsf{in}(n)$ and output locality is $\mathsf{out}(n)$ (resp., whose input locality is $\mathsf{in}(n)$, whose output locality is $\mathsf{out}(n)$). 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.

Locality-Preserving Reductions A *black-box reduction* from the function g to the function f is a polynomial-time algorithm G that computes the function g given oracle access to the function f. We say that such an algorithm G is a *reduction* from a cryptographic primitive $\mathcal{G}$ to a cryptographic primitive $\mathcal{F}$ (or, equivalently, a *construction* of $\mathcal{G}$ from $\mathcal{F}$, or a *transformation* from $\mathcal{F}$ to $\mathcal{G}$) if for any function f which implements $\mathcal{F}$ the function G^f implements the primitive $\mathcal{G}$. When defining $\mathrm{Local}_{\mathsf{in}(n)}^{\mathsf{out}(n)}$ reductions we restrict ourselves to very simple reductions which first query the oracle on some substrings of the original input (in a non-adaptive way) and then perform some $\mathrm{Local}_{\mathsf{in}(n)}^{\mathsf{out}(n)}$ computation. Formally, a reduction from a cryptographic primitive $\mathcal{G}$ to a cryptographic primitive $\mathcal{F}$ is said to be $\mathrm{Local}_{\mathsf{in}(n)}^{\mathsf{out}(n)}$ if it can be written as $G(x) = g(x, f(x^{(1)}), \dots, f(x^{(k)}))$ where the concatenation of $x^{(1)}, \dots, x^{(k)}$ form a prefix of x, and $g \in \mathrm{Local}_{\mathsf{in}(n)}^{\mathsf{out}(n)}$. A $\mathrm{Local}_{O(1)}^{O(1)}$ reduction G is also called a *locality-preserving reduction* as it preserves the input and output locality of f up to a constant factor.

2.1. *Randomized Encoding*

We review the notions of randomized encoding and randomizing polynomials from [3, 32,33].

Definition 2.1 (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 *complexity* of $\hat{f}$ is defined to be $m + s$.

Definition 2.1 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 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.

While our constructions yield perfect encodings, our negative results hold even when the notion of randomized encoding is relaxed as follows.

Definition 2.2 (Statistical Randomized Encoding). 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 δ-correct, ε-private randomized encoding of f, if there exist a (deterministic) decoder B, and a randomized simulator S, for which the following hold:

- *δ-correctness.* $\Pr[B(\hat{f}(x, U_m)) \neq f(x)] \leq \delta$ for any input $x \in \{0,1\}^n$.
- *ε-privacy.* $\mathrm{SD}(S(f(x)), \hat{f}(x, U_m)) \leq \varepsilon$ for any $x \in \{0,1\}^n$.

We can further relax privacy to the computational setting by replacing the second property with the requirement that for every string family $\{x_n\}$, the distribution ensembles $S(f(x_n))$ and $\hat{f}(x_n, U_{m(n)})$ cannot be distinguished by efficient adversaries with advantage larger than ε. Statistical and computational encoding preserve the security of some primitives as long as ε and δ are sufficiently small.

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

Lemma 2.3 (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 $\hat{f}(x, (r_g, r_h)) \stackrel{\text{def}}{=} h((x, r_g), r_h)$ is a perfect encoding of f.*

2.2. *Definitions of Primitives*

We will abuse notation and write $f : \{0,1\}^{m(n)} \to \{0,1\}^{s(n)}$ to denote the family $\{f_n : \{0,1\}^{m(n)} \to \{0,1\}^{s(n)}\}_{n \in \mathbb{N}}$.

2.2.1. *One Way Functions and Pseudorandom Generators*

We will use the following definition of one-way function.

Definition 2.4 (One-way Function). A polynomial-time computable function $f : \{0,1\}^n \to \{0,1\}^{s(n)}$ is called a *one-way function* (OWF) if for every (non-uniform) poly nomial-size circuit family $\{A_n\}$, we have

$$\Pr_{x \leftarrow U_n}\big[A_n\big(f(x)\big) \in f^{-1}\big(f(x)\big)\big] \leq \mathrm{neg}(n).$$

We will also mention the following variants of OWFs. A polynomial-time computable function f is *weakly one-way* if there exists a polynomial $p(\cdot)$ such that for any (non-uniform) polynomial size circuit family $\{A_n\}$, we have $\Pr[A_n(f(U_n)) \notin f^{-1}(f(U_n))] > 1/p(n)$. We can further relax the notion of one-wayness by requiring the adversary to find a *random* preimage of $y = f(U_n)$ (rather than some arbitrary preimage). Formally, the function f is *distributionally one-way* if there exists a polynomial $p(\cdot)$ such that any (non-uniform) polynomial size circuit family $\{A_n\}$, we have $\mathrm{SD}((A_n(f(U_n)), f(U_n)), (U_n, f(U_n))) > 1/p(n)$. We say that the function $f = \{f_n\}$ is *regular* if it maps the same (poly nomial-time computable) number of elements in $\{0,1\}^n$ to every element in $\mathrm{Im}(f_n)$. (This is the case, for instance, for any one-to-one function.) A collection of one-to-one functions $\mathcal{F} = \{f_z : D_z \to D_z\}_{z \in Z}$ is referred to as a trapdoor function if there exist probabilistic polynomial-time algorithms (I, D, F, F^{-1}) with the following properties. Algorithm I is an index selector algorithm that on input 1^n selects an index z from Z and a corresponding trapdoor for f_z; algorithm D is a domain sampler that on input z samples an element from the domain D_z; F is a function evaluator that given an index z and x returns $f_z(x)$; and F^{-1} is a trapdoor-inverter that given an index z, a corresponding trapdoor t and $y \in f_z(D_z)$ returns $f_z^{-1}(y)$. Additionally, the collection should be hard to invert, similarly to a standard one-way function. (For formal definition see [21, Definition 2.4.4].) We remark that the existence of OWFs is equivalent to the existence of weak OWFs and distributionally OWFs [30,52], but it is not known to imply the existence of trapdoor functions.

We move on to the definition of pseudorandom generators.

Definition 2.5 (Pseudorandom Generator). A pseudorandom generator (PRG) is a polynomial-time computable function, $G : \{0,1\}^n \to \{0,1\}^{s(n)}$, satisfying the following two conditions:

- *Expansion*: $s(n) > n$, for all $n \in \mathbb{N}$.
- *Pseudorandomness*: The ensemble $\{G(U_n)\}_{n \in \mathbb{N}}$ is pseudorandom.

It will sometimes be convenient to define a PRG (respectively, a OWF) by an infinite family of functions $G : \{0,1\}^{m(n)} \to \{0,1\}^{s(n)}$, where $m(\cdot)$ and $s(\cdot)$ are polynomials. Such a family can be transformed into a function that satisfies Definition 2.5 (resp., Definition 2.4) via padding. It will be also useful to consider collections of PRGs (resp., OWFs, weak OWFs, distributionally OWFs). Let $p(\cdot)$ be a polynomial, and let $\mathcal{G} = \{G_z\}_{z \in \{0,1\}^{p(n)}}$ be a polynomial-time computable collection of functions where $G_z : \{0,1\}^n \to \{0,1\}^{s(n)}$. Then $\mathcal{G}$ is a PRG collection (resp., OWF collection, weak OWF collection, distributional OWF collection), if $G'(x,z) = (G_z(x), z)$ is a PRG (resp., OWF, weak OWF, distributional OWF).

2.2.2. *Extractors*

We will also need the definition of extractors. The *min-entropy* of a random variable X is defined as $\mathrm{H}_\infty(X) \stackrel{\mathrm{def}}{=} \min_x \log(\frac{1}{\Pr[X=x]})$.

Definition 2.6 (Extractor). A function $\mathrm{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^t$ is a (k,ε)-extractor if for every distribution X on $\{0,1\}^n$ with $\mathrm{H}_\infty(X) \geq k$ the distribution $\mathrm{Ext}(X, U_d)$ is ε-close to the uniform distribution over $\{0,1\}^t$.

Extractors can be used to regain the entropy of sources that have high min-entropy with high probability. Formally,

Fact 2.7 (Implicit in [4]). *Let* $\mathrm{Ext}:\{0,1\}^n \times \{0,1\}^d \to \{0,1\}^t$ *be a* (k,ε) *extractor. Let* X *be a distribution over* $\{0,1\}^n$ *and* $A:\{0,1\}^n \to \{0,1\}^m$ *be a function. Let* $X|_{A(X)=a}$ *denote the conditional distribution of* X *given that* $A(X)=a$. *Suppose that*

$$\Pr_{a \leftarrow A(X')}\bigl[\mathrm{H}_\infty(X|_{A(X)=a}) \geq k\bigr] \geq 1-\delta,$$

where X' *is an independent copy of* X. *Then*,

$$\mathrm{SD}\bigl(\bigl(A(X), \mathrm{Ext}(X, U_d)\bigr), \bigl(A(X), U_t\bigr)\bigr) \leq \varepsilon + \delta.$$

An important construction of extractors is based on pairwise independent hashing.

Definition 2.8. A family of functions $H = \{h_z : \{0,1\}^n \to \{0,1\}^t\}$ is said to be a family of *pairwise independent hash functions* if for all distinct $x, x' \in \{0,1\}^n$, the outputs $(h_z(x), h_z(x'))$ induced by a random choice of z are uniformly and independently distributed over $\{0,1\}^t$.

We note that pairwise independent hash functions can be defined by the mapping $h_{M,v}(x) = Mx + v$ where M is a $t \times n$ binary matrix, v is a t-bit vector and arithmetic is over $\mathbb{F}_2$.

Lemma 2.9 (Leftover Hashing Lemma [27]). *Let* $H = \{h_z\}$ *be a family of pairwise independent hash functions that map* n*-bit strings to* t*-bit strings. Then, the function* $\mathrm{Ext}(x,z) = (h_z(x), z)$ *is a* (k,ε) *extractor where* $\varepsilon = 2^{-(k-t)/2}$.

2.2.3. *Commitments and Encryption Schemes*

We will consider a collection of non-interactive commitment schemes. In such a scheme, the sender and the receiver share a common public random key z (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 $\textsc{Com}_z(b; r)$ that outputs a commitment σ 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 $\textsc{Rec}_z(\sigma, b, r)$. The scheme should be both (computationally) hiding and (statistically) binding. Hiding requires that $\sigma = \textsc{Com}_z(b; r)$ keep b computationally secret. Binding means that, except with negligible probability over the choice of the random public key, there is no commitment string that can be opened in two different ways.

Definition 2.10 (Commitment Scheme). A commitment scheme is a pair $(\textsc{Com}, \textsc{Rec})$ where $\textsc{Com}$ is a probabilistic polynomial-time algorithm and $\textsc{Rec}$ is a deterministic polynomial-time algorithm. The scheme should satisfy the following conditions:

- *Viability*: For every bit $b \in \{0,1\}$, it holds $\Pr_{z,r}[\textsc{Rec}_z(\textsc{Com}_z(b; r), b, r) = \mathsf{reject})] \leq \mathrm{neg}(|z|)$.

- *Hiding*: $\{(z, \text{COM}_z(0; r))\}_n \overset{c}{\equiv} \{(z, \text{COM}_z(1; r))\}_n$ where $z \leftarrow U_n, r \leftarrow U_{p(n)}$, and the polynomial $p(\cdot)$ is the randomness complexity of COM.
- *Binding*: $\Pr_z[\exists \sigma,\ r_0,\ r_1$ such that $\text{REC}_z(\sigma, 0, r_0) = \text{REC}_z(\sigma, 1, r_1) = \mathsf{accept}] < \text{neg}(|z|)$.

We will use the following definition of semantically secure public-key encryption scheme [26]:

Definition 2.11 (Public-key Encryption). A *secure public-key encryption scheme* (PKE) is a triple (G, E, D) of probabilistic polynomial-time algorithms satisfying the following conditions:

- *Viability*: On input 1^n the randomized key generation algorithm, G, outputs a pair of keys (e, d). For every pair (e, d) such that $(e, d) \in G(1^n)$, and for every plaintext $x \in \{0, 1\}^*$, the algorithms E, D satisfy

$$\Pr\bigl[D\bigl(d, E(e, x)\bigr) \neq x\bigr] \leq \text{neg}(n).$$

- *Security*: For every polynomial $\ell(\cdot)$, and every families of plaintexts $\{x_n\}_{n\in\mathbb{N}}$ and $\{x'_n\}_{n\in\mathbb{N}}$ where $x_n, x'_n \in \{0, 1\}^{\ell(n)}$, it holds that

$$\bigl(e \leftarrow G_1\bigl(1^n\bigr), E(e, x_n)\bigr) \overset{c}{\equiv} \bigl(e \leftarrow G_1\bigl(1^n\bigr), E(e, x'_n)\bigr), \tag{1}$$

where $G_1(1^n)$ denotes the first element in the pair $G(1^n)$.

The following definition of Non-Malleable private-key encryption [17] is based on the definition of [37]. Since this definition is used here for negative results, we allow ourselves to use a simplified version which is weaker than the original definition. Informally, an encryption scheme is said to be non-malleable if it is impossible, given a ciphertext c encrypting a message x, to efficiently generate an encryption c' of a "related" message x' except by copying c.

Definition 2.12 (Non-malleable Private-key Encryption). A *non-malleable private-key encryption scheme* is a triple (G, E, D) of probabilistic polynomial-time algorithms satisfying the following conditions:

- *Viability*: On input 1^n the randomized key generation algorithm, G, outputs a key k. For every $k \in \text{support}(G(1^n))$ and every plaintext x, the algorithms E, D satisfy

$$\Pr\bigl[D\bigl(k, E(k, x)\bigr) \neq x\bigr] \leq \text{neg}(n),$$

where the probability is taken over the internal randomness of E and D.
- *Non-malleability*: For an adversary A, consider the following experiment which is indexed by n, the size of the key. First a random n-bit key k is selected. Then, A outputs a message space distribution $\mathcal{M}$ which consists of strings of equal length (and is represented by a probabilistic polynomial-sized circuit), and a binary relation R (which is also represented by a polynomial-sized circuit). Next, two random

strings $x, \tilde{x}$ are chosen from $\mathcal{M}$, and the ciphertext $c = E(k, x)$ is given to A. Finally, A outputs a ciphertext $c' \neq c$. The advantage of A is defined to be

$$\varepsilon_A(n) = \big|\Pr\big[\big(D(k, c'), x\big) \in R\big] - \Pr\big[\big(D(k, c'), \tilde{x}\big) \in R\big]\big|.$$

The scheme is *non-malleable* if for every (non-uniform) efficient adversary A the advantage $\varepsilon_A(n)$ of A is negligible in n.

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.

3.1. *Key Lemmas*

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 3.1 (Basic Input Locality Construction). Let

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

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

$$\begin{aligned}\hat{f}\big(x, (r_1, \dots, r_k)\big) \overset{\text{def}}{=} \big(&r_1 + b_1(x), r_2 + b_2(x), \dots, r_k + b_k(x),\\ &a(x) - r_1, r_1 - r_2, \dots, r_{k-1} - r_k,\\ &c_1(x), \dots, c_l(x)\big).\end{aligned}$$

We refer to a as the *pivot* of the construction.

Note that after the transformation the pivot 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. For example, applying the locality construction to the function $f(x) = (x_1x_2 + x_2, x_1x_2 + x_2x_3, x_1x_2 + x_3, x_3)$ with x_1x_2 as a pivot, results in the encoding $\hat{f}(x, r) = (r_1 + x_2, r_2 + x_2x_3, r_3 + x_3, x_1x_2 - r_1, r_1 - r_2, r_2 - r_3, x_3)$. Hence, in this case it reduces the locality of x_1 from 3 to 1.

Lemma 3.2 (Input Locality Lemma). *Let f and $\hat{f}$ be as in Construction 3.1. 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 $y_i + y_{k+1} + y_{k+2} + \cdots + y_{k+i}$; 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 $y_{[k+1...k+l]}$; (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 $1 < i \le k$, 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 $(r, s, y_{[k+1...k+l]})$, where $s_1 = y_1 - r_1$ and $s_i = y_i - r_i - \sum_{j=0}^{i-1} s_j$ for $1 < i \le k$. 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 $1 \le i \le k$; or (2) y_{i-k} if $k + 1 \le i \le 2k + l$.) □

We will also need the following simple transformation.

Lemma 3.3. *Let $f(x) = (a(x), a(x) + b_1(x), a(x) + b_2(x), \ldots, a(x) + b_k(x), c_1(x), \ldots, c_l(x))$, where $f : \mathbb{F}_2^n \to \mathbb{F}_2^{k+l+1}$ and $a, b_1, \ldots, b_k, c_1, \ldots, c_l : \mathbb{F}_2^n \to \mathbb{F}_2$. Then the function*

$$\hat{f}(x) = \big(a(x), b_1(x), b_2(x), \ldots, b_k(x), c_1(x), \ldots, c_l(x)\big)$$

is a perfect (deterministic) encoding of f. We refer to a as the pivot *of this construction.*

Proof. The encoding $\hat{f}$ is stretch-preserving since we did not add any additional outputs and did not use randomness at all. Moreover, there exists a fixed matrix $M \in \mathbb{F}_2^{(k+l+1)\times(k+l+1)}$ of full rank such that $f(x) = M \cdot \hat{f}(x)$ for every x. Hence, the encoding is perfectly private, perfectly correct and balanced. □

Again, after the transformation the locality of the input variables that appear in the pivot a is reduced, while the locality of all the other original input variables does not increase.

3.2. *Main Results*

In the following, it will often be useful to take an algebraic view of functions over bit-strings, specifying such functions using an *additive representation* over $\mathbb{F}_2$.

Definition 3.4 (Additive Representation). 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 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.

For example, any function f whose algebraic degree over $\mathbb{F}_2$ is d admits an additive representation in which each a is a product of at most d input variables.

The following measures are defined with respect to a given additive representation of f.

Definition 3.5 (Multiplicity and Rank). 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., $\#a = |\{T_i \mid a \in T_i\}|$. Given an additive representation of f, we define the *rank* of a variable x_j to be the number of different boolean functions a which depend on x_j and appear in some T_i. That is,

$$\mathrm{rank}(x_j) = \left|\left\{a : \mathbb{F}_2^n \to \mathbb{F}_2 \mid a \text{ depends on } x_j, a \in T_1 \cup \cdots \cup T_l\right\}\right|.$$

Theorem 3.6. *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. *The output locality of $\hat{f}$ is bounded from above by the output locality of f.*
3. *The randomness complexity of $\hat{f}$ is at most $\sum_{a\in T} \#a$, 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 3.1 or Lemma 3.3 to a function g is the (natural) representation induced by the original additive representation of g. Let $T = \bigcup_{i=1}^{l} T_i$ where $(T_1, \ldots, T_l)$ is the original additive representation of f. We construct $\hat{f}$ iteratively via the following process:

- Let $f^{(0)} \leftarrow f$ and $i \leftarrow 0$.
- For all $a \in T$
  - **if** one of the output bits of $f^{(i)}$ is equal to a **then** apply Lemma 3.3 to $f^{(i)}$ with a as a pivot.
  - **elseif** the multiplicity of a in $f^{(i)}$ is greater than 1 **then** apply Construction 3.1 to $f^{(i)}$ with a as a pivot.
  - record the resulting encoding in $f^{(i+1)}$ and let $i \leftarrow i + 1$.
- Output $\hat{f} \leftarrow f^{(i)}$.

By Lemmas 3.2 and 3.3, the function $f^{(i)}$ perfectly encodes the function $f^{(i-1)}$; hence, by the composition property of randomized encodings (Lemma 2.3), 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.

To prove the second item of the theorem, it suffices to show that in each iteration the output locality is not increased. Indeed, Construction 3.1 does not increase the output locality as long as the pivot does not appear as an output bit. Moreover, in the latter case instead of using Construction 3.1 we apply Lemma 3.3 which does not increase the output locality at all.

Finally, the last item follows by noting that the randomness complexity of Construction 3.1 is equal to the multiplicity of the pivot a. □

Remarks on Theorem 3.6.

1. By Theorem 3.6, 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 3.1 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. [3,32,33]). In fact, in Sect. 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 3.6 is applied to a function family $f : \{0,1\}^n \to \{0,1\}^{l(n)}$ then the resulting encoding is uniform whenever the additive representation $(T_1, \dots, T_l)$ is polynomial-time computable.
4. In Sect. 6.1, we show that Theorem 3.6 is tight in the sense that for each integer $i > 0$ 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 3.6 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 3.7 (Implicit in [3]). *Let $f : \mathbb{F}_2^n \to \mathbb{F}_2^l$. Fix some additive representation T for f in which each output bit is written as a sum of monomials of degree (at most) d. Then, we can perfectly encode f by a function $\hat{f}$ with an additive representation $\hat{T}$ such that:*

- $\hat{f} \in \mathrm{Local}^{\max(d+1,3)}$.
- *The rank of every original variable x_i in $\hat{f}$ (with respect to $\hat{T}$) is equal to the rank of x_i in f (with respect to T).*
- *The new variables introduced by $\hat{f}$ appear only in monomials of degree* 1; *hence, their rank is* 1.

By combining Lemma 3.7 with Theorem 3.6 we get:

Corollary 3.8. *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* $\max(d+1, 3)$. *Moreover, the resulting encoding is uniform whenever the additive representation is polynomial-time computable.*

Proof. First, by Lemma 3.7, we can perfectly encode f by a function $f' \in \mathrm{Local}^{\max(d+1,3)}$ without increasing the rank of the input variables of f. Next, we apply Theorem 3.6 and perfectly encode f' by a function $\hat{f} \in \mathrm{Local}^{\max(d+1,3)}_{\max(\rho,3)}$. By the composition property of randomized encodings (Lemma 2.3), the resulting function $\hat{f}$ perfectly encodes f. Finally, the proofs of Theorem 3.6 and Lemma 3.7 both allow efficiently transforming an additive representation of the function f into an encoding $\hat{f}$ in $\mathrm{Local}^{\max(d+1,3)}_{\max(\rho,3)}$. Hence, the uniformity of f is inherited by $\hat{f}$. □

It can be shown that if each output bit of f can be written as the sum of at most t degree-d monomials, then the randomness and output complexity of the above encoding is at most $O(tl)$. We also remark that Theorem 3.6, Lemma 3.7, and Corollary 3.8 generalize to any finite field $\mathbb{F}$.

Remark 3.9. By Corollary 3.8 any linear (or affine) function $L : \mathbb{F}_2^n \to \mathbb{F}_2^l$ can be encoded by a function $\hat{L}(x, r) \in \mathrm{Local}^3_3$. Moreover, a closer look at Theorem 3.6 and [3, Construction 4.11] shows that $\hat{L}$ has the following additional properties: (1) the input locality of the x's is 1; (2) the outputs that depend on the x's have (output) locality 2; and (3) the complexity of $\hat{L}$ is $O(n \cdot \min(n, l))$.

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 binary 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 4.1 ([50]). *Let* $0 < \delta < 1/2$ *and* $\varepsilon > 0$. *Let* $n/m \le 1 - \mathrm{H}_2(\delta) - \varepsilon$. *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 [49, Lecture 5]. For code length parameter $m = m(n)$, and noise parameter $\mu = \mu(n)$, we will consider the following "decoding game". Pick a random $m \times n$ matrix C representing a linear code, and a random information word x. Encode x with C and transmit the resulting codeword $y = Cx$ over a binary symmetric channel in which every bit is flipped with probability μ. Output the noisy codeword $\tilde{y}$ along with the code's description C. The adversary's task is to find the information word x. We say that the above game is intractable if every polynomial-time adversary wins in the above game with no more than negligible probability in n.

Definition 4.2. Let $m(n) \leq \text{poly}(n)$ be a code length parameter, and $0 < \mu(n) < 1/2$ be a noise parameter. The problem $\text{CODE}(m, \mu)$ is defined as follows:

- *Input:* $(C, Cx + e)$, 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 $\mathbb{F}_2$.
- *Output:* x.

We say that $\text{CODE}(m, \mu)$ is *intractable* if every (non-uniform) polynomial-time adversary A solves the problem with probability negligible in n.

We note that $\text{CODE}(m, \mu)$ becomes harder when m is decreased and μ is increased, as we can always add noise or ignore the suffix of the noisy codeword. Formally,

Proposition 4.3. *Let $m'(n) \leq m(n)$ and $0 < \mu(n) \leq \mu'(n) < 1/2$ for every n. Then, if $\text{CODE}(m, \mu)$ is intractable, so is $\text{CODE}(m', \mu')$.*

Proof. Fix n and let $m' = m'(n), m = m(n), \mu = \mu(n)$ and $\mu' = \mu'(n)$. We reduce the problem $\text{CODE}(m, \mu)$ to $\text{CODE}(m', \mu')$ as follows. Given an input (C, y) for $\text{CODE}(m, \mu)$ (i.e., C is an $m \times n$ binary matrix and y is an m-bit vector), we construct the pair (C', y') by letting C' denote the $m' \times n$ binary matrix that contains the first m' rows of C, and $y' \in \{0,1\}^{m'}$ be the vector resulting by taking the first m' entries of y and adding (over $\mathbb{F}_2$) a random vector $r \in \{0,1\}^{m'}$ in which each entry is chosen to be 1 (independently of other entries) with probability $(\mu' - \mu)/(1 - 2\mu)$.

Suppose that (C, y) is drawn from the input distribution of $\text{CODE}(m, \mu)$, that is, C is random matrix and $y = Cx + e$ where $x \leftarrow U_n$, and $e \in \{0,1\}^m$ is a random error vector of noise rate μ. Then, the pair (C', y') can be written as $(C', C'x + e')$ where $e' = e + r$ is a random noise vector of rate

$$\mu \cdot \left(1 - \frac{\mu' - \mu}{1 - 2\mu}\right) + (1 - \mu)\frac{\mu' - \mu}{1 - 2\mu} = \mu + \frac{(1 - 2\mu)(\mu' - \mu)}{1 - 2\mu} = \mu'.$$

Hence, given an algorithm A that solves $\text{CODE}(m', \mu')$, we can find the information word x by running A on (C', y'). □

The hardness of $\text{CODE}(m, \mu)$ is well-studied [12,13,19,29,35,39,43]. It can be also formulated as the problem of learning parity with noise, and it is known to be NP-complete in the worst-case [9]. It is widely believed that the problem is hard for every fixed μ and every $m(n) \in O(n)$, or even $m(n) \in \text{poly}(n)$. Similar assumptions were put forward in [12,21,25,29,35,36,38]. The plausibility of such an assumption is supported by the fact that a successful attack would imply a major breakthrough in coding theory. We mention that the best known algorithm for $\text{CODE}(m, \mu)$, due to Blum et al. [13], runs in time $2^{O(n/\log n)}$ and requires m to be $2^{O(n/\log n)}$. Lyubashevsky [43] showed how to reduce m to be only super-linear, i.e., $n^{1+\alpha}$, at the cost of increasing the running time to $2^{O(n/\log\log n)}$. When $m = O(n)$ (and μ is constant), the problem is only known to be solved in *exponential* time.

Our Parameters Typically, we let $m(n) \in O(n)$ 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 4.1, 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 $\mu + \varepsilon > \mu$ and some (fixed) constant rate.

Pseudorandomness We now show that distinguishing the distribution $(C, Cx + e)$ from the uniform distribution reduces to decoding x. A similar lemma was proved by Blum et al. [12, Theorem 13]. However, their version (as well as the version that appears in [48]) does not preserve the length of the codewords. Namely, they show that the hardness of decoding random linear code with codewords of length $m(n)$ implies the pseudorandomness of the distribution $(C, Cx + e)$ in which the length of the codewords is *polynomially smaller* than $m(n)$.

Lemma 4.4. *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 $C \leftarrow U_{m(n)\times n}$, $x \leftarrow U_n$, and $e \in \{0, 1\}^{m(n)}$ is a random error vector of noise rate μ.*

Proof. Assume that $\mathrm{CODE}(m, \mu)$ is intractable. Then, by the Goldreich–Levin hardcore bit theorem [23], given $(C, Cx + e)$ and a random n-bit vector r, an efficient adversary cannot compute $\langle r, x\rangle$ with probability greater than $\frac{1}{2} + \mathrm{neg}(n)$. Assume, towards a contradiction, that there exists an efficient distinguisher $A = \{A_n\}$ and a polynomial $p(\cdot)$ such that

$$\Pr\bigl[A_n(C, Cx + e) = 0\bigr] - \Pr\bigl[A_n(U_{m(n)\times n}, U_m) = 0\bigr] > 1/p(n),$$

for infinitely many n's. We will use A_n to construct an efficient adversary A'_n that breaks the security of the Goldreich–Levin hardcore bit. Given $(C, y = Cx + e)$ and a random n-bit vector r, the adversary A'_n chooses a random m-bit vector s and computes a new $m(n) \times n$ binary matrix $C' \overset{\mathrm{def}}{=} C - s \cdot r^T$, where r^T denotes the transpose of r. Now A'_n applies the distinguisher A_n to (C', y) and outputs his answer. Before we analyze the success probability of A'_n we need two observations: (1) the matrix C' is a random $m(n) \times n$ binary matrix; and (2) $y = Cx + e = C'x + s \cdot r^T \cdot x + e = C'x + s \cdot \langle r, x\rangle + e$. Hence, when $\langle r, x\rangle = 0$ it holds that $(C', y) = (C', C'x + e)$, and when $\langle r, x\rangle = 1$ we have $(C', y) = (C', C'x + e + s) \equiv (C', U_m)$, where U_m is independent of C'. Therefore, we have

$$\begin{aligned}
&\Pr\bigl[A'_n(C, Cx + e, r) = \langle x, r\rangle\bigr]\\
&\quad= \Pr\bigl[A'_n(C, Cx + e, r) = 0 \mid \langle x, r\rangle = 0\bigr] \cdot \Pr\bigl[\langle x, r\rangle = 0\bigr]\\
&\qquad+ \Pr\bigl[A'_n(C, Cx + e, r) = 1 \mid \langle x, r\rangle = 1\bigr] \cdot \Pr\bigl[\langle x, r\rangle = 1\bigr]
\end{aligned}$$

$$= \frac{1}{2} \cdot \bigl(\Pr\bigl[A_n(C', C'x + e) = 0\bigr] + 1 - \Pr\bigl[A_n(C', U_m) = 0\bigr]\bigr)$$
$$\geq \frac{1}{2} + \frac{1}{2p(n)},$$

where the last inequality holds for infinitely many n's. Thus, we derive a contradiction to the security of the GL-hardcore bit. □

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. (See Sect. 2.2 for formal definitions.) 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. Recall that, by the tractability of 2-SAT, it is impossible to construct a PRG (and even OWF) in Local^2 [16,20]. In Appendix A, we also prove that there is no PRG in Local_2. Hence, our PRG has optimal input locality as well as optimal output locality.

We rely on the following assumption.

Assumption 4.5. The problem $\mathrm{CODE}(6n, 1/4)$ is intractable.

Note that the code considered here is of rate $n/m = 1/6$ which is strictly smaller than $1 - H_2(1/4)$. Therefore, except with negligible probability, its relative distance is larger than $1/4$. Hence, the above assumption roughly says that it is intractable to correct $n/4$ random errors in a random linear code of relative distance $1/4 + \varepsilon$, for some constant $\varepsilon > 0$.

Let $m(n) = 6n$. 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 4.4, 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 $r_1, \ldots, r_{2m}$ and letting the ith 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. Indeed, we have "wasted" too much randomness on creating the error vector e. In [4], it was shown how to bypass this problem by applying a randomness extractor (see Definition 2.6). 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 (Lemma 2.9) and base our extractor on a family of pairwise independent hash functions (which is realized by the mapping $x \mapsto Mx + v$ where M is a random matrix and v is a random vector).[4]

[4] 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.

Construction 4.6. Let $m = 6n$ and let $t = \lceil 7.1 \cdot n \rceil$. Define the function

$$G(x, C, r, M, v) \overset{\text{def}}{=} \big(C, Cx + e(r), Mr + v, M, v\big),$$

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

Theorem 4.7. *Under Assumption 4.5, the function G defined in Construction 4.6 is a PRG.*

The proof of the above theorem is deferred to Appendix B. From now on, we fix the parameters m, t according to Construction 4.6. We can redefine the above construction as a collection of PRGs by letting C, M, v be the keys of the collection. Namely,

$$G_{C,M,v}(x, r) = \big(Cx + e(r), Mr + v\big). \tag{2}$$

We can now prove the main theorem of this section.

Theorem 4.8. *Under Assumption 4.5, 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, M, v and write each output bit of $G_{C,M,v}(x, r)$ 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 $r_{2\cdot\lceil i/2\rceil-1} \cdot r_{2\cdot\lceil i/2\rceil}$ 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 $G_{C,M,v}$ is at most 2. Therefore, by Corollary 3.8, we can perfectly encode the function $G_{C,M,v}$ by a function $\hat{G}_{C,M,v}$ 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. □

Since the encoding $\hat{G}_{C,M,v}$ has $N = \Theta(n^2)$ inputs and $N + \Theta(n)$ outputs, we get a pseudorandom generator whose stretch is only *sub-linear* in the input length. We mention that, by relying on the results of [4], one can obtain a PRG with linear stretch and (large) constant input and output locality. However, the security of this construction is based on a non-standard intractability assumption taken from [1].

Remark 4.9 (Single PRG in Local^3). Theorem 4.7 gives a PRG G of degree-2. By applying the output locality reduction of [3] (see also Lemma 3.7), we can encode G by a function $\hat{G}$ in Local^3 and get a *single* PRG (rather than a collection of PRGs) with optimal output locality. In a subsequent work [5], it is shown how to improve this result and obtain a single PRG in Local_3^3 under the same intractability assumption.

4.3. *Symmetric Encryption*

We can rely on Theorem 4.8 to obtain a (collection of) one-time semantically-secure symmetric encryption scheme (E_z, D_z) with low input and output locality (whose key is shorter than the message). Specifically, for a (public) collection key z, a private key k, a plaintext x, and a ciphertext c, we define the scheme $(E_z(k,x) = G_z(k) + x, D_z(k,c) = G_z(k) + c)$. It is not hard to prove the security of the scheme assuming that G_z is a collection of PRGs. We can instantiate this scheme with the PRG of Theorem 4.8 and obtain an encryption scheme whose both encryption algorithm and decryption algorithm are in Local_3^4. However this scheme is restricted to encrypt messages whose length is only slightly larger than the key length (as the stretch of G is only sub-linear).

We can remove this limitation and also obtain an encryption algorithm in Local_3^3, at the expense of increasing the complexity of the decryption algorithm. The idea is to use a variant of the aforementioned construction. In particular, Construction 4.3 of [2] uses a PRG (with a one-bit stretch) to obtain a one-time semantically-secure symmetric encryption (E, D) that allows to encrypt an arbitrary polynomially long message with a short key. Their encryption algorithm is defined as follows: $E(k, x, (s_1, \ldots, s_{\ell-1})) \overset{\text{def}}{=} (G(k) + s_1, G(s_1) + s_2, \ldots, G(s_{\ell-2}) + s_{l-1}, G(s_{\ell-1}) + x)$, where $k \leftarrow U_n$ is the private key, x is a $(k+\ell)$-bit plaintext and $s_i \leftarrow U_{n+i}$ serve as the coin tosses of E. If we instantiate this scheme with the PRG collection G_z of (2), we get a collection of encryption function E_z in which both the rank and the degree of E_z are at most 2. Hence, by employing Corollary 3.8, we can encode E_z by a function $\hat{E}_z$ in Local_3^3. In [3], it was shown that in this case $\hat{E}_z$ forms a one-time semantically-secure encryption scheme (together with the decryption function $\hat{D}_z(k, \hat{c}) = D_z(k, B(\hat{c}))$, where B is the decoding algorithm of the encoding). Hence, we get a one-time semantically-secure symmetric encryption in Local_3^3. However, the decryption is no longer in $\mathrm{Local}_{O(1)}^{O(1)}$.

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 Sect. 4.5 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.

4.4. *Commitment in* Local_3^4

We construct a collection of commitment schemes in Local_3^4 (i.e., a commitment of input locality 3 and output locality 4) under the following assumption.

Assumption 4.10. There exists a constant c that satisfies $c > \frac{1}{1-H_2(1/4)}$, for which the problem $\mathrm{CODE}(\lceil cn \rceil, 1/8)$ is intractable.

We begin by constructing a commitment scheme COM_z with low rank and low algebraic degree. Suppose that Assumption 4.10 holds with respect to c (for concreteness we may think of $c = 5.3$). Let $m = m(n) = \lceil cn \rceil$. The public key of our scheme will be a random $m \times n$ generator matrix C. To commit to a bit b, we first choose a random information word $x \in \{0,1\}^n$, 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. Assuming that $\mathrm{CODE}(m, 1/8)$ is intractable, this commitment hides the committed bit b. To see that the scheme is binding, recall that by Fact 4.1, the matrix C almost always generates a code whose relative distance is $1/4 + \varepsilon$, for some constant $\varepsilon > 0$. 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 $(1/8+\varepsilon/2)\cdot m$. 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 $(1/8+\varepsilon/2)\cdot m$. This way, the receiver will always catch a cheating sender (assuming that C is indeed a good code).

Construction 4.11. Given a constant c that satisfies $c > \frac{1}{1-H_2(1/4)}$, let $\varepsilon > 0$ be a constant for which $c > \frac{1}{1-H_2(1/4+\varepsilon)}$, and let $m = m(n) = \lceil cn \rceil$. We define the following scheme:

- Common random key: a random $m \times n$ generator matrix C.
- Commitment algorithm: $\textsc{Com}_C(b;(x,r,s)) = (Cx + e(r), s, b + \langle x,s\rangle)$, where $x,s \leftarrow U_n, r \leftarrow U_{3m}$, and $e(r) = (r_1r_2r_3, r_4r_5r_6, \ldots, r_{3m-2}r_{3m-1}r_{3m})$.
- Receiver algorithm: $\textsc{Rec}_C(\sigma, b, (x,r,s)) = \mathsf{accept}$ if and only if $\textsc{Com}_C(b;(x,r,s)) = \sigma$ and the Hamming weight of the noise vector $e(r)$ is smaller than $(1/8 + \varepsilon/2)\cdot m$.

Theorem 4.12. *Suppose that Assumption 4.10 holds with respect to the constant c. Then, the scheme defined in Construction 4.11 (instantiated with c) forms a collection of non-interactive commitment schemes.*

Proof. (1) Viability: An honest sender will be rejected only if its randomly chosen noise vector $e(r)$ is heavier than $(1/8+\varepsilon/2)\cdot m$, which, by a Chernoff bound, happens with negligible probability (i.e., $2^{-\Omega(n)}$) as the noise rate is $1/8$.

(2) Hiding: Let C,x,s,r distributed as in Construction 4.11. Then, by Assumption 4.10 and the Goldreich–Levin theorem [23], we have

$$\begin{aligned}\big(C, \textsc{Com}_C\big(0;(x,r,s)\big)\big) &\equiv \big(C, \big(Cx+e(r), s, \langle x,s\rangle\big)\big)\\ &\overset{c}{\equiv} \big(C, \big(Cx+e(r), s, U_1\big)\big)\\ &\equiv \big(C, \big(Cx+e(r), s, 1+U_1\big)\big)\\ &\overset{c}{\equiv} \big(C, \big(Cx+e(r), s, 1+\langle x,s\rangle\big)\big)\\ &\equiv \big(C, \textsc{Com}_C\big(1;(x,r,s)\big)\big).\end{aligned}$$

(3) Binding: Suppose that σ is an ambiguous commitment string. Namely, $\textsc{Rec}_C(\sigma, 0, (x,r,s)) = \textsc{Rec}_C(\sigma, 1, (x',r',s)) = \mathsf{accept}$ for some x,r,s,x',r' of appropriate length. Then, there are two distinct code words Cx and Cx' for which $Cx+e(r) = Cx'+e(r')$. Since $e(r)$ and $e(r')$ are of weight smaller than $(1/8+\varepsilon/2)\cdot m$, we conclude that the relative distance of the code C is smaller than $2\cdot(1/8+\varepsilon/2) = 1/4+\varepsilon$. However, by Fact 4.1, this event happens only with negligible probability (i.e., $2^{-\Omega(m)} = 2^{-\Omega(n)}$) over the choice of C. □

When C is fixed, the rank and algebraic degree of the function $\textsc{Com}_C$ are 2 and 3 (with respect to the natural representation as a sum of monomials). Hence, by Corollary 3.8, we can encode $\textsc{Com}_C$ by a function $\hat{\textsc{Com}}_C \in \mathrm{Local}_3^4$. By [3], this encoding is also a commitment scheme. Summarizing, we have:

Theorem 4.13. *Under Assumption 4.10, there exists a collection of commitment schemes* $(\textsc{Com}, \textsc{Rec})$ *in* Local_3^4; *i.e., for every public key C, we have* $\textsc{Com}_C \in \mathrm{Local}_3^4$.

We remark that we can get a standard non-interactive commitment (rather than collection of commitment schemes) 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.

4.5. *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 $\mathrm{Local}_{O(1)}^{O(1)}$ (for definition see Sect. 2.2). Our scheme is based on the McEliece cryptosystem [44]. 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.[5] 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 M over $\mathbb{F}_2$, and a random $m \times m$ permutation matrix P. Let $C' = P \cdot C \cdot M$ be the public key and P, M, 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 $P^{-1}c = P^{-1}(C'x + e) = CMx + P^{-1}e = CMx + e'$ 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 Mx (i.e., $D(C, CMx + P^{-1}e) = Mx$). Finally, to get x multiply Mx on the left by M^{-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. 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 $c - C'x$ 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 $1 - \mathrm{neg}(n)$ to recover x from $(c = C'x + e, C')$ where x is a random n-bit

[5] In fact, we may allow ε to decrease with n. See Remark 4.16.

string. (In other words, the McEliece cryptosystem is considered to be a collection of trapdoor one-way functions which is almost one-to-one with respect to its first argument; i.e., x.)

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 taking the exclusive-or of a hardcore predicate and a one-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 [23].) To decrypt the message, we first compute x, by invoking the McEliece decryption algorithm, and then compute the exclusive-or of $\langle s,x\rangle$ and 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 $(C', C'x+e, s)$, and thus the modified McEliece public-key is semantically secure. Formally,

Lemma 4.14. *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 [22, Proposition 5.3.14].

Let $\mu(n) = 2^{-d(n)}$. Then, we can sample the noise vector e by using the function $e(r) = (\prod_{j=1}^{\text{out}} r_{d\cdot(i-1)+j})_{i=1}^{m(n)}$ where r is a $d(n)\cdot m(n)$ 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 $d(n)$. Hence, by Corollary 3.8, we can encode it by a function $\hat{E} \in \mathrm{Local}_3^{d(n)+1}$, i.e., the output locality of $\hat{E}$ is $d(n)+1$ and its input locality is 3. In [3, Lemma 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 4.15. *If the McEliece cryptosystem is one-way with respect to the parameters $m(n)$, $\mu(n) = 2^{-d(n)}$ and $\mathcal{C}_n$, then there exists a semantically secure PKE whose encryption algorithm is in* $\mathrm{Local}_3^{d(n)+1}$.

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 4.15 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 [44], 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 [34].

Remark 4.16. Our description of the McEliece cryptosystem assumes that the error-correcting code being used is efficiently decodable from a *constant* noise rate. Specifically, we assume that codes from $\mathcal{C}_n$ can correct up to $(\mu(n)+\varepsilon)\cdot m(n)$ errors, for some *constant* $\varepsilon > 0$. This requirement can be waived. In particular, we may allow $\varepsilon = \varepsilon(n)$ to decrease with n as long as it is larger than, say, $1/\sqrt{m(n)}$. In this case, by Chernoff bound, the decryption algorithm errs with probability at most $\exp(-2\varepsilon^2 m) < 1/2$. This error can be decreased to negligible by repeating the encryption (of the modified McEliece scheme) $\Omega(n)$ times with independent fresh randomness, and by taking the majority while decrypting. Note that this transformation does not increase the rank or the degree of the encryption function.

Remark 4.17. Recall that in the standard definition of semantic security the adversary's goal is to find two messages x and x', whose encryptions can be distinguished. In particular, x and x' should be chosen before the adversary sees the public-key. One may consider a stronger variant of semantic security in which the choice of the pair x and x' may depend on the public-key e. It is not hard to show that Lemma 4.14 holds also with respect to this notion of security. (See [22, Sect. 5.4.2].) Furthermore, randomized encoding preserves semantic-security under key-dependent attacks [3], and therefore Theorem 4.15 extend to this setting as well.

4.6. *Locality Preserving Reductions between Different Primitives*

In this section, we show that, in some cases, our machinery can be used to get locality-preserving reductions between different primitives. That is, we can transform a primitive $\mathcal{F}$ (say one-to-one one-way function) into a different primitive $\mathcal{G}$ (say pseudorandom generator) while preserving the input and output locality of $\mathcal{F}$. Given such a reduction and an implementation of $\mathcal{F}$ with input locality $\mathsf{in}(n)$ and output locality $\mathsf{out}(n)$ we get an implementation of $\mathcal{G}$ with input locality $\mathsf{in}(n) + O(1)$ and output locality $\mathsf{out}(n) + O(1)$. In particular, if $\mathcal{F}$ can be implemented with constant input locality and constant output locality then so is $\mathcal{G}$.

The general idea is to encode the known construction from $\mathcal{F}$ to $\mathcal{G}$ into a corresponding $\mathrm{Local}^{O(1)}_{O(1)}$ construction. Consider, for example, the Blum–Micali–Yao construction [11,52] of PRG collection G from one-way permutation f which is defined by $G_z(x) = (f(x), \langle x, z\rangle)$, where $x, z \in \{0,1\}^n$ (recall that the collection key z is public). Then, for any fixed collection key z the term $\langle x, z\rangle$ is just a fixed function $L_z(x)$ which is linear in x. Hence, by Remark 3.9, it can be encoded by a function $\hat{L}_z(x, r) \in \mathrm{Local}^3_3$. Therefore, the function $\hat{G}_z(x, r) = (f(x), \hat{L}_z(x, r))$ is a (perfect) encoding of G_z and so it forms a reduction from a collection of PRGs to a one-way permutation. This reduction preserves the locality of f. In particular, when $f \in \mathrm{Local}^{\mathsf{out}}_{\mathsf{in}}$ we get a collection of PRGs in $\mathrm{Local}^{\mathsf{out}}_{\mathsf{in}+1}$, assuming that $\mathsf{in} \geq 2, \mathsf{out} \geq 3$.[6]

[6] The input locality is $\mathsf{in}+1$ since the input locality of the x's in $\hat{L}_z$ is only 1. See Remark 3.9.

More generally, let $\mathcal{G}$ be a cryptographic primitive whose security is respected by perfect encoding. Suppose that $G(x) = g(x, f(x^{(1)})), \dots, f(x^{(k)}))$ defines a black-box construction of $\mathcal{G}$ from an instance f of a primitive $\mathcal{F}$, where g can be encoded in $\mathrm{Local}_{O(1)}^{O(1)}$ and the concatenation of $x^{(1)}, \dots, x^{(k)}$ forms a prefix of the input x, i.e., $x = (x^{(1)}, \dots, x^{(k)}, x^{(k+1)})$. (The function g is fixed by the reduction and do not depend on f.) Then, letting $\hat{g}((x, y_1, \dots, y_k), r)$ be a perfect $\mathrm{Local}_{O(1)}^{O(1)}$ encoding of g, the function $\hat{G}(x, r) = \hat{g}((x, f((x^{(1)})), \dots, f(x^{(k)})), r)$ perfectly encodes G, and hence, defines a black-box locality-preserving reduction from a $\mathcal{G}$ to $\mathcal{F}$.

It turns out that several known cryptographic reductions are of the above form where the function g is a random public linear function (e.g., g is used to extract hard-core bits using Goldreich–Levin [23], or as a pairwise independent hash function). Since linear functions can be encoded by Local_3^3 functions (Remark 3.9) we get a locality-preserving reduction for every fixed choice of the linear function. This results in a locality-preserving transformation from $\mathcal{F}$ to collection of $\mathcal{G}$. In the following lemma, we instantiate this approach with several cryptographic constructions.

Lemma 4.18. *Let* $\mathsf{in}(n) \geq 2, \mathsf{out}(n) \geq 3$ *be locality parameters. Let* $f \in \mathrm{Local}_{\mathsf{in}(n)}^{\mathsf{out}(n)}$. *Then*,

1. f *is distributionally one-way* $\Rightarrow \exists$ *collection of OWFs in* $\mathrm{Local}_{\mathsf{in}(n)+1}^{\mathsf{out}(n)}$.
2. f *is a regular OWF* $\Rightarrow \exists$ *collection of PRGs in* $\mathrm{Local}_{\mathsf{in}(n)+1}^{\mathsf{out}(n)+1}$.
3. f *is a one-to-one OWF* $\Rightarrow \exists$ *non-interactive commitment scheme such that the sender's computation is in* $\mathrm{Local}_{\mathsf{in}(n)+1}^{\mathsf{out}(n)}$.
4. f *is a one-to-one trapdoor function* $\Rightarrow \exists$ *public-key encryption scheme* (G, E, D) *such that the encryption algorithm* E *is in* $\mathrm{Local}_{\mathsf{in}(n)+1}^{\mathsf{out}(n)}$.
5. f *is a PRG* $\Rightarrow \exists$ *one-time symmetric-key encryption* (E, D) (*with a short key*) *such that the encryption algorithm* E *is in* $\mathrm{Local}_{\mathsf{in}(n)+1}^{\mathsf{out}(n)+1}$.

Proof. (1) Let f be a distributional OWF. We will employ the reduction of [30] which can be written as $F_z(x) = (f(x), L_z(x))$ where L_z is a linear function (originally, L_z is a projection of a pairwise independent hash function). In [30], it was shown that the collection F is weakly one-way (see also [21, p. 96]). By Remark 3.9, L_z can be encoded by a function $\hat{L}_z(x, r) \in \mathrm{Local}_3^3$. Hence, the encoding $\hat{F}_z(x, r) = (f(x), \hat{L}_z(x, r))$ forms a collection of weak OWF. Since the input locality of the original variables (the x's) in $\hat{L}_z$ is only 1, the input locality of $\hat{F}_z$ is only $\mathsf{in}(n) + 1$. Finally, we can transform $\hat{F}$ to a strong (i.e., standard) OWF by applying $\hat{F}$ to polynomially many independent inputs (cf. [52], [21, Theorem 2.3.2]). This step does not increase the input locality nor the output locality of the reduction. Hence, we get a collection of OWF in $\mathrm{Local}_{\mathsf{in}(n)+1}^{\mathsf{out}(n)}$.

(2) We rely on the PRG construction from [27]. When this construction is applied to a regular OWF f, it involves only the computation of universal hash functions and hard-core bits, and therefore can be written as $G_z(x) = L_z(x, f(x^{(1)})), \dots, f(x^{(k)}))$, where L_z is a linear function [27, Theorem 5.1.4].[7] Hence, by Remark 3.9, we can encode the

[7] In more detail, suppose that we have a OWF $f : \{0, 1\}^n \to \{0, 1\}^m$ which is t-to-one, where $t = t(n)$ is computable in polynomial-time. Then, the construction of [27] (see also [28]) can be described in two steps:

reduction by a function $\hat{L}_z(x, y_1, \ldots, y_k, r)$ in Local_3^3. If f is in $\mathrm{Local}_{\mathsf{in}(n)}^{\mathsf{out}(n)}$ then the input locality of the x's in G_z is $\mathsf{in}(n) + 1$, and the output locality of the outputs that involve x is $\mathsf{out}(n) + 1$. (The random inputs r affect 3 outputs and participate in outputs that depend on 3 inputs.)

(3, 4) We rely on the construction of [10] (instantiated with the Goldreich–Levin hardcore predicate [23]): to commit to b using the randomness s, z we compute $\textsc{Com}(b; s, z) = (f(s), \langle s, z\rangle + b, z)$. Since the degree and rank of the function $g(b, s, z) = (\langle s, z\rangle + b, z)$ are both 2, we can apply the same argument used in (1). The same construction results in a public-key encryption scheme when f is a one-to-one trapdoor function. (See [22, Proposition 5.3.14].) Note that in both cases, encrypting (resp., committing to) a long message can be done by applying the basic construction to every bit separately (with independent fresh randomness). This extension preserves the locality of the 1-bit schemes.

(5) In [2], it was shown how to transform a PRG with minimal (1-bit) stretch into a one-time semantically-secure private-key encryption that allows to encrypt messages whose length is polynomially longer than the key length (for any arbitrary polynomial). Specifically, the encryption algorithm was defined as follows: $E_k(x, (s_1, \ldots, s_{\ell-1})) \overset{\text{def}}{=} (G(k) \oplus s_1, G(s_1) \oplus s_2, \ldots, G(s_{\ell-2}) \oplus s_{l-1}, G(s_{\ell-1}) \oplus x)$, where $k \leftarrow U_n$ is the private key, x is a $(k + \ell)$-bit plaintext and $s_i \leftarrow U_{n+i}$ serve as the coin tosses of E. When the PRG is in $\mathrm{Local}_{\mathsf{in}(n)}^{\mathsf{out}(n)}$ we get an encryption in $\mathrm{Local}_{\mathsf{in}(n)+1}^{\mathsf{out}(n)+1}$. □

We note that the theorem holds even when f is a collection.

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 achieves security in the public-key setting is also secure in the private-key setting.

5.1. *Basic Observations*

Let $f : \{0, 1\}^n \to \{0, 1\}^{s(n)}$ be a function and let $s = s(n)$. For $i \in [n]$ and $x \in \{0, 1\}^n$, we let $Q_i(x) \subseteq [s]$ be the set of indices in which $f(x)$ and $f(x_{\oplus i})$ differ. (Recall that $x_{\oplus i}$ denote the string x with the ith bit flipped.) We let $Q_i^n \overset{\text{def}}{=} \bigcup_{x \in \{0,1\}^n} Q_i(x)$, equivalently, Q_i^n is the set of output bits which are affected by the ith input bit. From now on, we

First we define a collection $g_{y,r}(x) = (f(x), h_y(x), \langle x, r\rangle)$ where $\{h_y\}$ is a collection of pairwise independent hash functions that map n bits to $\log(t) + 2$ bits. This function is shown to have large "pseudoentropy", that is, when y, r and x are randomly chosen, the bit $\langle x, r\rangle$ has low entropy given the values of $g(x), h_y(x), y$ and r, but it is computationally unpredictable. Then, in the second step we use many instances of g to construct a PRG. That is, we define the function $G_{w,\vec{y},\vec{r}}(\vec{x}) = h'_w(g_{y^{(1)},r^{(1)}}(x^{(1)}), \ldots, g_{y^{(k)},r^{(k)}}(x^{(k)}))$, where $\{h'_w\}$ is a collection of pairwise independent hash functions of output length ℓ, and k, ℓ are some explicit functions of n, m and t. Hence, when $z = (w, \vec{y}, \vec{r})$ is fixed, and the hash functions are implemented via affine transformations, $G_z(\vec{x})$ can be written as $L_z(\vec{x}, f(x^{(1)}), \ldots, f(x^k))$ where L_z is an affine function.

omit the superscript n whenever the input length is clear from the context. We show that, given oracle access to f, we can efficiently approximate the set Q_i for every i.

Lemma 5.1. *There exists a probabilistic algorithm A that, given oracle access to $f: \{0,1\}^n \to \{0,1\}^s$, an index $i \in [n]$ and an accuracy parameter ε, outputs a set $Q \subseteq Q_i$ such that $\Pr_x[Q_i(x) \nsubseteq Q] \le \varepsilon$, where the probability is taken over the coin tosses of A and the choice of x (which is independent of A). Moreover, when $\{f : \{0,1\}^n \to \{0,1\}^{s(n)}\}$ is an infinite collection of functions the time complexity of A is polynomial in $n, s(n)$ and $1/\varepsilon(n)$. In particular, if $s(n) = \mathrm{poly}(n)$, then for every constant c, one can reduce the error to n^{-c} in time $\mathrm{poly}(n)$.*

Proof. Let $t = \ln(2s/\varepsilon)$ and $\alpha = \varepsilon/(2s)$. The algorithm A constructs the set Q iteratively, starting from an empty set. In each iteration, A chooses uniformly and independently at random a string $x \in \{0,1\}^n$ and adds to Q the indices for which $f(x)$ and $f(x_{\oplus i})$ differ. After t/α iterations A halts and outputs Q. Clearly, $Q \subseteq Q_i$. Let $p_j \stackrel{\text{def}}{=} \Pr_x[j \in Q_i(x)]$. We say that j is common if $p_j > \alpha$. Then, if j is common we have

$$\Pr[j \notin Q] \le (1-p_j)^{t/\alpha} \le (1-\alpha)^{t/\alpha} \le \exp(-t) = \varepsilon/(2s).$$

Since $|Q_i| \le s$, there are at most s common j's and thus, by a union bound, the probability that A misses a common j is at most $\varepsilon/2$. On the other hand, for a random x, the probability that $Q_i(x)$ contains an uncommon index is at most $s \cdot \alpha = \varepsilon/2$. Hence, we have $\Pr_x[Q_i(x) \nsubseteq Q] \le \varepsilon$, which completes the proof. □

Our negative results are based on the following simple observation.

Lemma 5.2. *Let $f : \{0,1\}^n \to \{0,1\}^{s(n)}$ be a function in $\mathrm{Local}_{\mathsf{in}(n)}$. Then, there exist a probabilistic polynomial-time algorithm A such that for every $x \in \{0,1\}^n$ and $i \in [n]$, the output of A on $(y = f(x), i, Q_i^n, 1^n)$ equals, with probability at least $2^{-\mathsf{in}(n)}$, to the string $y' = f(x_{\oplus i})$. In particular, when $\mathsf{in}(n) = O(\log(n))$, the success probability of A is $1/\mathrm{poly}(n)$.*

Proof. Fix n and let $s = s(n)$ and $Q_i = Q_i^n$. By definition, y and y' may differ only in the indices of Q_i. Hence, we may randomly choose y' from a set of size $2^{|Q_i|} \le 2^{\mathsf{in}(n)}$, and the lemma follows. □

Note that the above lemma generalizes to the case in which, instead of getting the set Q_i^n, the algorithm A gets a set Q_i' that satisfies $Q_i(x) \subseteq Q_i' \subseteq Q_i^n$.

By combining the above lemmas we get the following corollary:

Corollary 5.3. *Let $f : \{0,1\}^n \to \{0,1\}^{s(n)}$ be a function in $\mathrm{Local}_{\mathsf{in}(n)}$, where $s(n) = \mathrm{poly}(n)$. Then, there exist a probabilistic polynomial-time algorithm A that, given oracle access to f, converts, with probability $(1-1/n) \cdot 2^{-\mathsf{in}(n)}$, an image $y = f(x)$ of a randomly chosen string $x \leftarrow U_n$ into an image $y' = f(x_{\oplus 1})$. Namely,*

$$\Pr_x\left[A^f\left(f(x), 1^n\right) = f(x_{\oplus 1})\right] \ge (1-1/n) \cdot 2^{-\mathsf{in}(n)},$$

where the probability is taken over the choice of x and the coin tosses of A. In particular, when $l(n) = O(\log(n))$ the algorithm A succeeds with probability $1/\text{poly}(n)$,

Proof. First, we use algorithm A_1 of Lemma 5.1 to learn, with accuracy $\varepsilon = 1/n$, an approximation Q'_1 of the set Q^n_1. Then, we invoke the algorithm A_2 of Lemma 5.2 on $(f(x), 1, Q'_1, 1^n)$ where $f(x)$ is the challenge given to us, and output the result. Let E_1 be the event where $Q_1(x) \subseteq Q'_1 \subseteq Q_1$. Since x is uniformly chosen, Lemma 5.1 implies that this event happens with probability larger than $1 - 1/n$. Let E_2 denote the event that A_2 succeeds and outputs $f(x_{\oplus 1})$. Conditioning on the event E_1, the probability of E_2 is at least $2^{-\mathsf{in}(n)}$ (by Lemma 5.2). It follows that our overall success probability is at least $(1 - 1/n) \cdot 2^{-\mathsf{in}(n)}$, and the corollary follows. □

Clearly, one can choose to flip any input bit and not just the first one. Also, we can increase the success probability to $(1 - n^{-c}) \cdot 2^{-\mathsf{in}(n)}$ for any constant c.

We now prove the impossibility results.

5.2. *MACs and Signatures*

Let (G, S, V) be a MAC scheme, where G is a key generation algorithm, the randomized signing function $S(k, \alpha, r)$ computes a signature β on the document α using the 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 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.[8] The scheme is one-time secure if the adversary is allowed to query the signing oracle only once. One-time secure MACs are known to exist even in an information-theoretic setting. Such schemes do not require any assumption and are secure even against computationally unlimited adversaries.

Suppose that the signature function $S(k, \alpha, r)$ has logarithmic input locality (i.e., $S(k, \alpha, r) \in \text{Local}_{O(\log(|k|))}$). Then, we can use Corollary 5.3 to break the scheme with a single oracle call. First, ask the signing oracle $S(k, \cdot)$ to sign on a randomly chosen document α. Then, use the algorithm of Corollary 5.3 to transform, with probability $1/\text{poly}(n)$, the valid pair (α, β) we received from the signing oracle into a valid pair $(\alpha_{\oplus 1}, \beta')$. (Note that when applying Corollary 5.3 we let $S(\cdot, \cdot, \cdot)$ play the role of f.)

Now, suppose that for each *fixed* key $k \in \{0, 1\}^n$ the signature function $S_k(\alpha, r) = S(k, \alpha, r)$ has input locality $\mathsf{in}(n)$. In this case, we cannot use Corollary 5.3 directly. The problem is that we cannot apply Lemma 5.1 to learn the set Q_i (i.e., the set of output bits which are affected by the ith input bit of $f = S_k(\cdot, \cdot)$) since we do not have a full oracle access to S_k. (In particular, we do not see or control the randomness used in each invocation of S_k.) However, we can guess the set Q_i and then apply Lemma 5.2. This attack succeeds with probability $\left(1/\binom{s(n)}{\mathsf{in}(n)}\right) \cdot 2^{-\mathsf{in}(n)}$ where $s(n)$ is the length of the

[8] 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.

signature (and so is polynomial in n). When $\mathsf{in}(n) = c$ is constant, the success probability is $1/\Theta(s(n)^c) = 1/\text{poly}(n)$ and therefore, in this case, we break the scheme.[9] To summarize:

Theorem 5.4. *Let (G, S, V) be a MAC scheme. If $S(k, \alpha, r) \in \text{Local}_{O(\log(|k|))}$ or $S_k(\alpha, r) \in \text{Local}_{O(1)}$ for every k, then the scheme is not one-time secure.*

Remarks on Theorem 5.4.

1. Theorem 5.4 is true even if *some* bit of α has low input locality. This observation also holds in the case of non-malleable encryption scheme.
2. If we have access to the verification oracle (for example, in the public-key setting where (G, S, V) is a digital-signature scheme), we can even break the scheme in a stronger sense. Specifically, we can forge a signature to *any* target document given a single signature to, say, 0^n. To see this, note that, given a signature β of the document α, we can *deterministically* find a signature β' of the document $\alpha_{\oplus i}$ by checking all the polynomially many candidates. Hence, we can apply this procedure (at most) n times and gradually transform a given signature of some arbitrary document into a signature of any target document. Therefore, such a scheme is universally forgeable.
3. A *weaker* version of Theorem 5.4 still holds even when the input locality of the signing algorithm is *logarithmic* with respect to any *fixed* key (i.e., when $S_k(\alpha, r) \in \text{Local}_{O(\log(|k|))}$ for every k). In particular, we can break such MAC schemes assuming that we are allowed to ask for several signatures that were produced with some fixed (possibly unknown) randomness. In such a case, we use Lemma 5.1 to (approximately) learn the output bits affected by, say, the first input bit, and then apply Lemma 5.2 to break the scheme. This attack rules out the existence of a *deterministic* MAC scheme for which $S_k(\alpha) \in \text{Local}_{O(\log(|k|))}$ for every k.

Theorem 5.4 is tight since if the signing algorithm is allowed to use super-constant input locality (for every fixed key), then there exists a one-time secure MAC. Formally,

Lemma 5.5. *Let $\mathsf{in}(n)$ be a locality function and $s(n)$ be a signature length function. Then, there exists a MAC scheme (G, S, V) which cannot be broken by any (computationally unlimited) adversary via a one-time attack with probability larger than $1/\binom{s(n)}{\mathsf{in}(n)}$. Moreover, $S_k(\alpha, r) \in \text{Local}_{\mathsf{in}(n)}$ for every fixed key k. In particular, by setting $s(n) = \Omega(n)$, we get super-polynomial security for any super-constant locality.*

The proof of this lemma is deferred to Appendix C.

5.3. *Non-malleable Encryption*

Let (G, E, D) be a private-key encryption scheme, where G is a key generation algorithm, the encryption function $E(k, m, r)$ computes a ciphertext c encrypting the message m using the key k and randomness r, and the decryption algorithm $D(k, c, r)$

[9] When the locality $\mathsf{in}(n)$ of S_k is logarithmic (for every fixed key k), this approach yields an attack that succeeds with probability $1/n^{\Theta(\log(n))}$.

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.

Theorem 5.6. *Let (G, 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 either the encryption oracle or 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.*

Proof. The proof is similar to the proof of Theorem 5.4. Let n be the length of the key k, $p = p(n), m = m(n)$, and $s = s(n)$ be the lengths of the message x, randomness r, and ciphertext length c, respectively; i.e., $E : \{0,1\}^n \times \{0,1\}^p \times \{0,1\}^m \to \{0,1\}^s$. Our attacks will use the message space $\mathcal{M} = \{0,1\}^p$ and the relation R for which $(x, x') \in R$ if and only if x and x' differ only in their first bit.

Suppose that the encryption function $E(k, x, r)$ has logarithmic input locality (i.e., $E(k, x, r) \in \mathrm{Local}_{O(\log(|k|))}$). Then, by Corollary 5.3, we can break the scheme by transforming, with noticeable probability, the challenge ciphertext c into a ciphertext c' such that the corresponding plaintexts differ only in their first bit. Clearly, the probability for this relation to hold with respect to $\tilde{c}$ which is a ciphertext of a random plaintext is negligible. Hence, we break the scheme.

Now, suppose that for each fixed key $k \in \{0,1\}^n$ the encryption function $E_k(x, r) = E(k, x, r)$ has input locality $\mathsf{in}(n)$. In this case, we guess the set Q_1 and then apply Lemma 5.2. This attack succeeds with probability $\left(1/\binom{s(n)}{\mathsf{in}(n)}\right) \cdot 2^{-\mathsf{in}(n)}$. When $\mathsf{in}(n)$ is constant, the success probability is $1/\mathrm{poly}(n)$, and therefore, in this case, the scheme is broken.

We move on to the case in which the input locality of E_k is logarithmic. The previous attack succeeds in this case with probability $1/n^{\Theta(\log(n))}$. However, we can improve this to $1/\mathrm{poly}(n)$ if we have *stronger* access to the encryption oracle. In particular, we should be able to get several ciphertexts that were produced with some fixed (possibly unknown) randomness. In such a case, we use Lemma 5.1 to (approximately) learn the output bits affected by, say, the first input bit, and then apply Lemma 5.2 to break the scheme. The public-key setting is a special case in which this attack is feasible as we get a full access to the randomness of the encryption oracle. □

6. Negative Results for Randomized Encodings

In the following, we prove some negative results regarding randomized encoding with low input locality. In Sect. 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 [3,33] 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 a branching program that computes f). In Sect. 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 different linear functions. This is contrasted with previous constructions of randomized encoding with constant output locality (cf. [3,32,33]) which gives a (non-efficient) universal encoding for the class of all functions $f : \{0,1\}^n \to \{0,1\}^l$ as well as an efficient universal encoding for classes such as all linear functions or all size-s BPs (where s is polynomial in n).

These results hold in the case of perfect encoding as well as in the more liberal setting of statistically correct and statistically (or even computationally) private encodings in which the simulator and decoder are allowed to err.

6.1. *A Necessary Condition for Encoding with Low Input Locality*

Let $f : \{0,1\}^n \to \{0,1\}^l$ be a function. Define an undirected graph $G_i(f)$ 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 $f(x_{\oplus i}) = y'$. Note that two vertices which lie in the same connected component of $G_i(f)$ differ only in the indices which are affected by the ith input. Hence, when f has low input locality in, the size of each component of $G_i(f)$ is at most $2^{|\mathsf{in}|}$. It turns out that a similar restriction also holds when f is encoded by a function $\hat{f}$ with low input locality, even when f itself has large input locality. Specifically, in Sects. 6.1.1 and 6.1.2 we will prove the following theorem:

Theorem 6.1. *Let $f : \{0,1\}^n \to \{0,1\}^l$ be a function which is encoded by a function $\hat{f} : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^s$ in* $\mathrm{Local}_{\mathsf{in}}$. *Then,*

1. *If $\hat{f}$ is a perfect encoding then for every $1 \le i \le n$ the size of the connected components of $G_i(f)$ is at most 2^{in}.*
2. *If $\hat{f}$ is a δ-correct and ε-private encoding then for every $1 \le i \le n$ the degree of each vertex of $G_i(f)$ is at most $\frac{n}{-\log(\delta+\varepsilon)} \cdot 2^{\mathsf{in}}$. In particular, if $\varepsilon + \delta < 0.9$ then the degree is bounded by $7n2^{\mathsf{in}}$.*

We will actually show that the second conclusion holds even when the privacy is relaxed to be $\varepsilon(n)$-computational as long as $\mathsf{in}(n) \le O(\log n)$. Also note that the theorem is meaningful as long as the sum of ε and δ is upper bounded away from 1. This limitation is rather weak since one typically requires ε and δ to be negligible in n.

Theorem 6.1 shows that even some very simple functions do not admit an encoding with constant input locality. Consider, for example, the function

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

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 (respectively, computational encoding) of this function is at least $n-1$ (respectively, $n-2-\log n$ for sufficiently large n). Note that this matches the results of Sect. 3 since $\mathrm{rank}(x_1) = n-1$.

6.1.1. *Proof of Theorem 6.1 for Perfect Encoding*

Fix $i \in [n]$ and let $G = G_i(f)$. Let $\hat{f} : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^s$ be a perfectly correct and private randomized encoding of $f : \{0,1\}^n \to \{0,1\}^l$ with decoder B and simulator S. Let $Q \subseteq \{1, \ldots, s\}$ be the set of output bits in $\hat{f}$ which are affected by the input variable x_i. Namely, $j \in Q$ iff $\exists x \in \{0,1\}^n, r \in \{0,1\}^m$ such that the strings $\hat{f}(x,r)$ and $\hat{f}(x_{\oplus i}, r)$ differ on the jth bit.

We begin with the following claims.

Claim 6.2. *Let $y, y' \in \mathrm{Im}(f)$ be adjacent vertices in G_i. Then, for every $\hat{y} \in \mathrm{support}(S(y))$ there exists $\hat{y}' \in \mathrm{support}(S(y'))$ which differs from $\hat{y}$ only in indices which are in Q.*

Proof. Let $x \in \{0,1\}^n$ be an input string for which $f(x) = y$ and $f(x_{\oplus i}) = y'$. Fix some $\hat{y} \in \mathrm{support}(S(y))$. Then, by perfect privacy, there exists some $r \in \{0,1\}^m$ for which $\hat{y} = \hat{f}(x, r)$. Let $\hat{y}' = \hat{f}(x_{\oplus i}, r)$. By the definition of Q, the strings $\hat{y}$ and $\hat{y}'$ differ only in indices which are in Q. Also, by the perfect privacy of $\hat{f}$, we have that $\hat{y}' \in \mathrm{support}(S(y'))$ and the claim follows. □

Claim 6.3. *Let $y \in \mathrm{Im}(f)$ and let $\hat{y} \in \mathrm{Im}(\hat{f})$. Then, $y = B(\hat{y})$ if and only if $\hat{y} \in \mathrm{support}(S(y))$.*

Proof. Let $x \in f^{-1}(y)$. By perfect correctness, $y = B(\hat{y})$ iff $\hat{y} \in \mathrm{support}(\hat{f}(x, U_m))$. By perfect privacy, $\mathrm{support}(\hat{f}(x, U_m)) = \mathrm{support}(S(f(x))) = \mathrm{support}(S(y))$, and the claim follows. □

We can now prove the first part of Theorem 6.1. The idea is to label each vertex y of G by a distinct string $\hat{y} \in \mathrm{Im}(\hat{f})$ and to show that the vertices of each connected component are labeled by a small set of strings. Specifically, we show that if u and v are in the same connected component then their labels differ only in the indices which are in Q. It follows that the size of such component is bounded by $2^{|Q|}$.

Proof of Theorem 6.1 part 1. Fix $u \in \mathrm{Im}(f)$ and let $\hat{u} \in \{0,1\}^s$ be some arbitrary element of $\mathrm{support}(S(u))$. Let $Z \overset{\mathrm{def}}{=} \{z \in \{0,1\}^s \mid z_i = \hat{u}_i, \forall i \in [s] \setminus Q\}$. That is, $z \in Z$ if it differs from $\hat{u}$ only in indices which are in Q. To prove the claim, we define an onto mapping from Z (whose cardinality is $2^{|Q|}$) to the members of the connected component of u. The mapping is defined by applying the decoder B of $\hat{f}$, namely $z \to B(z)$. (Assume, wlog, that if the decoder is invoked on a string z which is not in $\mathrm{Im}(\hat{f})$ then it outputs $\perp$.) Let $v \in \mathrm{Im}(f)$ be a member of the connected component of u. We prove that there exists $z \in Z$ such that $v = B(z)$.

The proof is by induction on the distance (in edges) of v from u in the graph G. In the base case when the distance is 0, we let $z = \hat{u}$ and, by perfect correctness, get that $B(\hat{u}) = u$. For the induction step, suppose that the distance is $i > 1$. Then, let w be the last vertex in a shortest path from u to v. By the induction hypothesis, there exists a string $\hat{w} \in Z$ for which $w = B(\hat{w})$. Hence, by Claim 6.3, $\hat{w} \in \mathrm{support}(S(w))$. Since v and w are neighbors, we can apply Claim 6.2 and conclude that there exists

$\hat{v} \in \text{support}(S(v))$ which differs from $\hat{w}$ only in indices which are in Q. Since $\hat{w} \in Z$ it follows that $\hat{v}$ is also in Z. Finally, by Claim 6.3, we have that $B(\hat{v}) = v$ which completes the proof. □

6.1.2. *Proof of Theorem 6.1 for Statistical and Computational Encoding*

In the following, we will keep the notation of the previous section but relax $\hat{f} : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^s$ to be a δ-correct and ε-private randomized encoding of f.

We say that a string $\hat{y} \in \{0,1\}^s$ is *good* for x if there exists a string $\hat{u}$ such that: (1) $\hat{u}$ differs from $\hat{y}$ only in indices which are in Q; and (2) $B(\hat{u}) = f(x_{\oplus i})$, where B is the decoder of $\hat{f}$.

Claim 6.4. *For every $x \in \{0,1\}^n$ a string $\hat{y}$ which is chosen from the distribution $S(f(x))$ will be good with probability $1-\delta-\varepsilon$, where S is a simulator for the encoding.*

Proof. Fix x. Consider the imaginary experiment where $\hat{y}$ is chosen from the distribution $\hat{f}(x,r)$ where $r \leftarrow U_m$. Let $\hat{y}' = \hat{f}(x_{\oplus i}, r)$. Clearly, $\hat{u}$ differs from $\hat{y}$ only in indices which are in Q. Furthermore, by the correctness of the encoding $\hat{u}$ decodes to $f(x_{\oplus i})$ with probability at least $1-\delta$ (since r is uniformly distributed). Hence, in our imaginary experiment $\hat{y}$ is good for x with probability $1-\delta$. Finally, privacy guarantees that $\text{SD}(\hat{f}(x, U_m), S(y)) \leq \varepsilon$ and therefore the probability that $\hat{y}$ is good for x in the real experiment is at least $1-\delta-\varepsilon$. □

Lemma 6.5. *For every $y \in \text{Im}(f)$ there exists a set $T_y \subseteq \text{Im}(\hat{f})$ of size at most $\frac{n}{-\log(\delta+\varepsilon)}$ such that for every $x \in f^{-1}(y)$ there exists a good $\hat{y} \in T_y$.*

Proof. Fix y and let $X = f^{-1}(y)$. We will construct the set T_y iteratively. We begin with an empty set T_y and with $X_0 = X$. In the ith iteration, we will choose a string $\hat{y}$ which is good for at least $1-\delta-\varepsilon$ fraction of the entries in X_i and put the remaining x's in X_{i+1}. Since the initial size of X_0 is bounded by 2^n we will need at most $\frac{n}{-\log(\delta+\varepsilon)}$ iterations.

It is left to argue that in each iteration there exists such a good $\hat{y}$. Fix some $X_i \subseteq X$. By Claim 6.4 and the linearity of expectation, a random $\hat{y}$ which is chosen from the distribution $S(y)$ is expected to be good for at least $1-\delta-\varepsilon$ fraction of the x's of X_i. The existence of such fixed $\hat{y}$ is therefore guaranteed by an averaging argument. □

We can now prove the second part of Theorem 6.1. The proof is similar to the proof of the first part. However, now we will label each vertex y of G by a small *collection* of strings $\hat{y}_1, \ldots, \hat{y}_k \in \text{Im}(\hat{f})$. We will show that if u and v are neighbors then there exits a corresponding label $\hat{u}_i$ (in the collection of u) and a string $\hat{v}$ such that: (1) $\hat{u}_i$ and $\hat{v}$ differ only in the locations which are indexed by Q; and (2) $\hat{v}$ decodes to v. It follows that the *degree* of G is bounded by $k \cdot 2^{|Q|}$.

Lemma 6.6. *The degree of each vertex of G is at most $\frac{n}{-\log(\delta+\varepsilon)} \cdot 2^{|Q|}$.*

Proof. Fix $u \in \text{Im}(f)$ and let $T_u \subseteq \{0,1\}^s$ be a set of size at most $\frac{n}{-\log(\delta+\varepsilon)}$ which satisfies Claim 6.5. Let $Z \stackrel{\text{def}}{=} \{z \in \{0,1\}^s \mid z_{[s]\setminus Q} = \hat{u}_{[s]\setminus Q}, \exists \hat{u} \in T_u\}$. That is, $z \in Z$ if it differs from some $\hat{u} \in T_u$ only in indices which are in Q. To prove the claim, we define an onto mapping from Z to the neighbors of u. The mapping is defined by applying the decoder B of $\hat{f}$, namely $z \to B(z)$.

Let $v \in \mathrm{Im}(f)$ be a neighbor of u. We prove that there exists $z \in Z$ such that $v = B(z)$. Indeed, let x be a preimage of u for which $v = f(x_{\oplus i})$ and let $\hat{u} \in T_u$ be a good string for x. It follows that there exists a string $\hat{v}$ which decodes to v and agrees with $\hat{u}$ on the coordinates $[s] \setminus Q$, which completes the proof. □

The Computational Setting The above lemma extends to the case where the encoding is only ε-computational private (and δ-correct) as long as Q is of logarithmic size and the decoder is efficient. To see this, note that Claim 6.4 (which is the only place where privacy was used) still holds in the computational setting. Indeed, if the claim does not hold for some infinite family of $\{x_n\}$ then one can efficiently distinguish between the ensembles $S(f(x_n))$ and $\hat{f}(x_n, U_{m(n)})$ with advantage bigger than ε by checking whether a sample $\hat{y}$ is good for x_n. This test is efficiently computable (by a polynomial-size circuit family) as long as Q is sufficiently small and B is efficient.

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 Sect. 5.2 rule out the existence of such a scheme. We now give a more direct proof to the impossibility of obtaining a universal encoding with constant input locality for linear functions. The proof is similar to the proofs in Sect. 6.1.

Let C be a class of functions that map n bits into l bits. For each input bit $1 \le i \le n$, we define a graph G_i over $\bigcup_{f \in C} \mathrm{Im}(f)$ such that there is an edge between the strings y and y' if there exists $x \in \{0,1\}^n$ and $f \in C$ such that $f(x) = y$ and $f(x_{\oplus i}) = y'$. Namely, $G_i = \bigcup_{f \in C} G_i(f)$, where $G_i(f)$ is the graph defined in Sect. 6.1. Suppose that C has universal encoding in $\mathrm{Local}_{\mathsf{in}}$ with decoder B and simulator S. That is, for every $f_z \in C$ there exists a perfect randomized encoding $\hat{f} : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^s$ in $\mathrm{Local}_{\mathsf{in}}$ whose correctness and privacy hold with respect to B and S.

Lemma 6.7. *The degree of every vertex in G_i is bounded by $\binom{s}{\mathsf{in}} \cdot 2^{\mathsf{in}}$.*

Proof. Let y be a vertex of G_i and fix some $\hat{y} \in S(y)$. Let y' be a neighbor of y with respect to $f \in C$. Let $Q = Q_i \subseteq \{1, \ldots, s\}$ be the set of output bits in $\hat{f}$ which are affected by the input variable x_i. Then, by the proof of Theorem 6.1 part 1, there exists a set $Z_Q \subseteq \{0,1\}^s$ of size 2^{in} such that $y' \in \mathrm{Im}(B(Z_Q))$. Hence, we have an onto mapping from $Q \times Z_Q$ to the neighbors of y. Thus, the number of neighbors is at most $\binom{s}{\mathsf{in}} \cdot 2^{\mathsf{in}}$. □

Lemma 6.8. *Let C be the class of linear functions $L : \{0,1\}^n \to \{0,1\}^l$ where $l \le n$. Then, for every $1 \le i \le n$, the graph G_i is a complete graph over $\{0,1\}^l$.*

Proof. Consider, for example, the graph $G = G_1$ and fix some $y, y' \in \{0,1\}^l$. Then, for $\sigma = (0,1,\ldots,1)$ and $\sigma_{\oplus 1} = (1,1,\ldots,1)$, there exists a linear function $L : \{0,1\}^n \to \{0,1\}^l$ for which $y = L(\sigma)$ and $y' = L(\sigma_{\oplus 1})$. To see this, write L as a matrix $M \in \{0,1\}^{l \times n}$ such that $L(x) = Mx$. Let $M = (M_1, M')$, that is, M_1 denotes the leftmost column of M, and M' denotes the matrix M without M_1. Now, we can first solve the linear system $M \cdot \sigma = y$ which is equivalent to $M' \cdot \sigma = y$ and then solve the linear system $M \cdot \sigma_{\oplus 1} = y'$ which is now equivalent to $M_1 = y' - y$. □

Let $l \le n$. By combining the above claims, we conclude that the output complexity s of any universal encoding in $\mathrm{Local}_{\mathsf{in}}$ for linear functions $L : \{0,1\}^n \to \{0,1\}^l$ must satisfy $\binom{s}{\mathsf{in}} \cdot 2^{\mathsf{in}} \ge 2^l$. In particular, when in is constant, the output complexity of the encoding must be exponential in l. A similar bound also holds when the encoding is only 0.45-correct and 0.45-computationally private as long as l is super-logarithmic in n. This can be proven by a straightforward extension of Lemma 6.7 which uses the second part of Theorem 6.1.

Acknowledgements

We thank Ronny Roth for helpful discussions.

Appendix A. The Impossibility of Implementing a PRG in Local_2

We prove that there is no PRG in Local_2, and thus the PRG constructed in Sect. 4.2 has optimal input locality as well as optimal output locality.

Lemma A.1. *Let $G : \{0,1\}^n \to \{0,1\}^{s(n)}$ be a polynomial-time computable function in Local_2 where $s(n) > n$. Then, there exists polynomial-size circuit family $\{A_n\}$ that given $z \in \mathrm{Im}(G)$ reads some subset $S \subset [s(n)]$ of z's bits and outputs (S, k, z_k) for some $k \notin S$.*

Proof. Fix n and let $s = s(n)$. Define $H_G = ((\mathrm{Out} = [s], \mathrm{In} = [n]), E)$ to be the bipartite graph whose edges correspond to the input-output dependencies in G; that is, (i, j) is an edge if and only if the ith output bit of G depends on the jth input bit. Since G is in Local_2 the average degree of output vertices is $2n/s$ which is smaller than 2 (as $s > n$). The circuit A_n implements the following procedure:

1. Initialize S and T to be empty sets, let $H \leftarrow H_G$ and let $x = 0^n$.
2. If there exists an output $k \in \mathrm{Out}$ which is not connected to any input in the graph H, then halt and predict z_k by computing $G(x)_k$.

3. Otherwise, there exists an output $j \in \mathrm{Out}$ which depends on a single input bit $i \in \mathrm{In}$ in the graph H (since the average out degree is smaller than 2). Add j to S and add i to T. Let $x_i = 0$ if $G(x) = z_j$, and 1 otherwise. Remove i and j from the graph H.
4. Goto 2.

The procedure stops after at most n steps since there are only $n < s$ inputs. The correctness follows by noting that: (1) in each iteration $x_T = x^\star_T$ where $x^\star$ is the preimage of z under G; and (2) the kth output bit depends only on the input bits which are indexed by T. To see (1), observe that in each iteration all the output bits which are indexed by S depend only on the input bits which are indexed by T. □

We can now conclude that there is no PRG in Local_2.

Corollary A.2. *There is no PRG in* Local_2.

Proof. Assume, towards a contradiction, that $G : \{0,1\}^n \to \{0,1\}^{s(n)}$ is a PRG in Local_2. Fix n and let $s = s(n)$. Let A_n be the adversary defined in Lemma A.1. Then, we define an adversary B_n that given $z \in \{0,1\}^s$ invokes A_n and checks whether A_n predicts z_k correctly. By Lemma A.1, when $z \in \mathrm{Im}(G)$ the adversary B_n always outputs 1. However, when z is a random string the probability that B_n outputs 1 is at most $1/2$. Indeed, if $B_n(z) = 1$ for some $z \in \{0,1\}^s$ then $B_n(z_{\oplus k}) = 0$, where k is the bit that A_n predicts when reading z (and $z_{\oplus i}$ denote the string z with the ith bit flipped). Hence, A_n errs on at least half of the strings in $\{0,1\}^s$, and so it distinguishes $G(U_n)$ from U_s with advantage $1/2$. □

The above corollary can be extended to rule out the existence of a *collection* of PRGs with input locality 2. Note that when G is chosen from a collection, the graph H_G might not be available to the adversary constructed in Lemma A.1. However, a closer look at this lemma shows that, in fact, the adversary does not need an explicit description of H_G; rather, it suffices to find an approximation of H_G (in the sense of Lemma 5.1). As shown in Lemma 5.1, such an approximation can be found efficiently (with, say, $1/n$ error probability) given an (oracle) access to G. This modification also shows that such a PRG can be broken by a *uniform* adversary.

The above negative result does not rule out the existence of a OWF in Local_2, which is left as an open problem.

Appendix B. Proof of Theorem 4.7

Let r be a uniformly chosen $2m$-bit string. Let $r_{|e(r)=z}$ denote the distribution of r conditioned on the event that $e(r) \stackrel{\mathrm{def}}{=} (r_{2i-1} \cdot r_{2i})_{i=1}^m$ is equal to z. Then the following claim is implicit in [4].

Claim B.1 (Similar to Lemma 3 of [4]).

$$\Pr_{z \leftarrow e(U_{2m})}\bigl[\mathrm{H}_\infty(r_{|e(r)=z}) \geq 1.17m\bigr] \geq 1 - \exp\bigl(-\Omega(m)\bigr).$$

Proof. We view $e(r)$ as a sequence of m independent Bernoulli trials, each with a probability 0.25 of success. Recall that r is composed of m pairs of bits, and that the ith bit of $e(r)$ is 1 if and only if r_{2i-1} and r_{2i} are both equal to 1. Hence, whenever $e(r)_i = 0$, the pair (r_{2i-1}, r_{2i}) is uniformly distributed over the set $\{00, 01, 10\}$. Let $z \leftarrow e(U_{2m})$. Consider the case in which at most $0.26m$ components of z are ones. By a Chernoff bound, the probability of this event is at least $1 - \exp(-\Omega(m))$. In this case, $r_{|e(r)=z}$ is uniformly distributed over a set of size at least $3^{0.74m}$. Hence, conditioning on the event that at most $0.26m$ components of z are ones, the min-entropy of $r_{|e(r)=z}$ is at least $0.74m \log(3) > 1.17m$. □

We can now prove Theorem 4.7. Let $m = 6n$ and $t = 7.01n$. First we show that G expands its input. Indeed, the difference between the output length and the input length is: $m + t - (n + 2m) = 0.01n > 0$.

Let $x \leftarrow U_n$, $C \leftarrow U_{m\cdot n}$, $r \leftarrow U_{2m}$, $M \leftarrow U_{t\cdot 2m}$ and $v \leftarrow U_t$. We prove that the distribution $G(x, C, r, M, v)$ is pseudorandom. Define

$$\Delta(n) \overset{\text{def}}{=} \mathrm{SD}\big(\big(C, Cx + e(r), Mr + v, M, v\big), \big(C, Cx + e(r), U_{t+2tm+m}\big)\big).$$

First observe that

$$\Delta(n) \le \mathrm{SD}\big(\big(e(r), Mr + v, M, v\big), \big(e(r), U_{t+2tm+m}\big)\big),$$

as for every distributions X and Y and every randomized process A, it holds that $\mathrm{SD}(A(X), A(Y)) \le \mathrm{SD}(X, Y)$. Now, by Lemma 2.9, Fact 2.7 and Claim B.1, we have that

$$\begin{aligned}
&\mathrm{SD}\big(\big(e(r), Mr + v, M, v\big), \big(e(r), U_{t+2tm+m}\big)\big)\\
&\quad\le 2^{-(1.17m-t)/2} + \exp\big(-\Omega(m)\big)\\
&\quad= 2^{-0.005n} + \exp\big(-\Omega(n)\big) \le \exp\big(-\Omega(n)\big).
\end{aligned}$$

Hence, under Assumption 4.5, we have

$$\big(C, Cx + e(r), Mr + v, M, v\big) \overset{\mathrm{s}}{\equiv} \big(C, Cx + e(r), U_{t+2tm+m}\big) \overset{\mathrm{c}}{\equiv} U_{mn+m+t+2tm+m},$$

which completes the proof.

Appendix C. Proof of Lemma 5.5

Given the length of the signature $s(n)$ and the locality parameter $\mathsf{in}(n)$, we construct the following scheme:

- *Key Generation:* Choose a random $s(n) \times n$ binary matrix M by selecting each of the n columns uniformly and independently from the set of all $s(n)$-bit vectors whose Hamming weight is exactly $\mathsf{in}(n)$. In addition, uniformly choose an $s(n)$ bit vector v.
- *Signature:* To sign compute $S_{M,v}(\alpha) = M \cdot \alpha + v$.
- *Verification:* To verify that (α, β) is valid check if $M \cdot \alpha + v = \beta$.

First, note that each input variable affects at most $\mathsf{in}(n)$ output bits as any column of M has at most $\mathsf{in}(n)$ ones. We move on to prove that the scheme is secure. Let $s = s(n)$. We begin by showing that if the adversary does not query the signing algorithm at all, then he cannot forge a signature. Fix $\alpha \in \{0, 1\}^n$ and $\beta \in \{0, 1\}^s$. Then we can write

$$\Pr_{M,v}[M\alpha + v = \beta] = \Pr_{M,v}[v = \beta - M\alpha] = 2^{-s},$$

where the last equality holds since for every fixed M we have $\Pr_v[v = \beta - M\alpha] = 2^{-s}$. Suppose that the adversary used his single oracle query to learn the signature $\beta^{(1)}$ to some document $\alpha^{(1)}$. We show that the probability that the adversary finds a signature on any other document $\alpha^{(2)} \neq \alpha^{(1)}$ is at most $1/\binom{s(n)}{\mathsf{in}(n)}$. Indeed, fix some $\alpha^{(1)} \neq \alpha^{(2)} \in \{0, 1\}^n$ and $\beta^{(1)}, \beta^{(2)} \in \{0, 1\}^s$. We will prove that

$$\Pr_{M,v}\left[M\alpha^{(2)} + v = \beta^{(2)} | M\alpha^{(1)} + v = \beta^{(1)}\right] \leq \frac{1}{\binom{s(n)}{\mathsf{in}(n)}}.$$

First note that $M\alpha^{(1)} + v = \beta^{(1)}$ if and only if $v = \beta^{(1)} - M\alpha^{(1)}$. Hence, by letting $\alpha = \alpha^{(2)} - \alpha^{(1)}$ and $\beta = \beta^{(2)} - \beta^{(1)}$, we have

$$\Pr_{M,v}\left[M\alpha^{(2)} + v = \beta^{(2)} | M\alpha^{(1)} + v = \beta^{(1)}\right] = \Pr_M[M\alpha = \beta].$$

Observe that $\alpha \neq 0^n$ (since $\alpha^{(1)} \neq \alpha^{(2)}$). Assume, without loss of generality, that α_1, the first bit of α is 1 (otherwise, permute α and the columns of M). Let M_i denote the ith column of M. Then,

$$\Pr_M[M\alpha = \beta] = \Pr_M\left[M_1 = \beta - \sum_{i=2}^{n} \alpha_i \cdot M_i\right].$$

We complete the proof by noting that M_1 is distributed uniformly over a set of size $\binom{s(n)}{\mathsf{in}(n)}$ independently of the other columns, and thus the above term is bounded by $1/\binom{s(n)}{\mathsf{in}(n)}$.

References

[1] M. Alekhnovich, More on average case vs approximation complexity, in *Proc. 44th FOCS*, 2003, pp. 298–307

[2] B. Applebaum, Y. Ishai, E. Kushilevitz, Computationally private randomizing polynomials and their applications. *Comput. Complex.* **15**(2), 115–162 (2006). Preliminary version in Proc. 20th CCC, 2005

[3] B. Applebaum, Y. Ishai, E. Kushilevitz, Cryptography in NC^0. *SIAM J. Comput.* **36**(4), 845–888 (2006). Preliminary version in Proc. 45th FOCS, 2004

[4] B. Applebaum, Y. Ishai, E. Kushilevitz, On pseudorandom generators with linear stretch in NC^0, in *Proc. 10th Random*, 2006

[5] B. Applebaum, Y. Ishai, E. Kushilevitz, Cryptography with constant latency. Manuscript, 2009

[6] S. Arora, S. Safra, Probabilistic checking of proofs: A new characterization of np. *J. ACM* **45**(1), 70–122 (1998). Preliminary version in Proc. 33rd FOCS, 1992

[7] S. Arora, C. Lund, R. Motwani, M. Sudan, M. Szegedy, Proof verification and hardness of approximation problems. *J. ACM* **45**(3), 501–555 (1998). Preliminary version in Proc. 33rd FOCS, 1992

[8] L. Babai, Random oracles separate PSPACE from the polynomial-time hierarchy. *Inf. Process. Lett.* **26**(1), 51–53 (1987)

[9] E.R. Berlekamp, R.J. McEliece, H.C. van Tilborg, On the inherent intractability of certain coding problems. *IEEE Trans. Inf. Theory* **24**(3), 384–386 (1978)

[10] M. Blum, Coin flipping by telephone: a protocol for solving impossible problems. *SIGACT News* **15**(1), 23–27 (1983)

[11] M. Blum, S. Micali, How to generate cryptographically strong sequences of pseudo-random bits. *SIAM J. Comput.* **13**, 850–864 (1984). Preliminary version in Proc. 23rd FOCS, 1982

[12] A. Blum, M. Furst, M. Kearns, R.J. Lipton, Cryptographic primitives based on hard learning problems, in *Advances in Cryptology: Proc. of CRYPTO '93*, LNCS, vol. 773 (1994), pp. 278–291

[13] A. Blum, A. Kalai, H. Wasserman, Noise-tolerant learning, the parity problem, and the statistical query model. *J. ACM* **50**(4), 506–519 (2003). Preliminary version in Proc. 32nd STOC, 2000

[14] R.B. Boppana, J.C. Lagarias, One-way functions and circuit complexity. *Inf. Comput.* **74**(3), 226–240 (1987)

[15] S.A. Cook, The complexity of theorem-proving procedures, in *STOC '71: Proceedings of the Third Annual ACM Symposium on Theory of Computing*, New York, NY, USA (ACM Press, New York, 1971), pp. 151–158

[16] M. Cryan, P.B. Miltersen, On pseudorandom generators in NC^0, in *Proc. 26th MFCS*, 2001, pp. 272–284

[17] D. Dolev, C. Dwork, M. Naor, Non-malleable cryptography. *SIAM J. Comput.* **30**(2), 391–437 (2000)

[18] U. Feige, J. Killian, M. Naor, A minimal model for secure computation (extended abstract), in *Proc. of the 26th STOC*, 1994, pp. 554–563

[19] V. Feldman, P. Gopalan, S. Khot, A.K. Ponnuswami, New results for learning noisy parities and halfspaces, in *Proc. 47th FOCS*, 2006, pp. 563–574

[20] O. Goldreich, Candidate one-way functions based on expander graphs. *Electron. Colloq. Comput. Complex. (ECCC)* **7**(090) (2000)

[21] O. Goldreich, *Foundations of Cryptography: Basic Tools* (Cambridge University Press, Cambridge, 2001)

[22] O. Goldreich, *Foundations of Cryptography: Basic Applications* (Cambridge University Press, Cambridge, 2004)

[23] O. Goldreich, L. Levin, A hard-core predicate for all one-way functions, in *Proc. 21st STOC*, 1989, pp. 25–32

[24] O. Goldreich, S. Goldwasser, S. Micali, How to construct random functions. *J. ACM* **33**, 792–807 (1986)

[25] O. Goldreich, H. Krawczyk, M. Luby, On the existence of pseudorandom generators. *SIAM J. Comput.* **22**(6), 1163–1175 (1993). Preliminary version in Proc. 29th FOCS, 1988

[26] S. Goldwasser, S. Micali, Probabilistic encryption. *J. Comput. Syst. Sci.* **28**(2), 270–299 (1984). Preliminary version in Proc. STOC '82

[27] J. Håstad, R. Impagliazzo, L.A. Levin, M. Luby, A pseudorandom generator from any one-way function. *SIAM J. Comput.* **28**(4), 1364–1396 (1999)

[28] T. Holenstein, Pseudorandom generators from one-way functions: A simple construction for any hardness, in *Proc. 3rd TCC*, 2006, pp. 443–461

[29] N.J. Hopper, M. Blum, Secure human identification protocols, in *Advances in Cryptology: Proc. of ASIACRYPT '01*, LNCS, vol. 2248 (2001), pp. 52–66

[30] R. Impagliazzo, M. Luby, One-way functions are essential for complexity based cryptography, in *Proc. of the 30th FOCS*, 1989, pp. 230–235

[31] R. Impagliazzo, M. Naor, Efficient cryptographic schemes provably as secure as subset sum. *J. Cryptol.* **9**(4), 199–216 (1996). Preliminary version in FOCS '89

[32] Y. Ishai, E. Kushilevitz, Randomizing polynomials: A new representation with applications to round-efficient secure computation, in *Proc. 41st FOCS*, 2000, pp. 294–304

[33] Y. Ishai, E. Kushilevitz, Perfect constant-round secure computation via perfect randomizing polynomials, in *Proc. 29th ICALP*, 2002, pp. 244–256

[34] H. Janwa, O. Moreno, Mceliece public key cryptosystems using algebraic-geometric codes. *Des. Codes Cryptogr.* **8**(3), 293–307 (1996)

[35] A. Juels, S. Weis, Authenticating pervasive devices with human protocols, in *Advances in Cryptology: Proc. of CRYPTO '05*, LNCS, vol. 3621 (2005), pp. 293–308

[36] J. Katz, J.-S. Shin, Parallel and concurrent security of the hb and hb+ protocols, in *Advances in Cryptology: Proc. of Eurocrypt 06'*, LNCS, vol. 4004 (2006), pp. 73–87

[37] J. Katz, M. Yung, Complete characterization of security notions for probabilistic private-key encryption, in *Proc. 32nd STOC*, 2000, pp. 245–254

[38] M. Kearns, Y. Mansour, D. Ron, R. Rubinfeld, R.E. Schapire, L. Sellie, On the learnability of discrete distributions, in *Proc. 26th STOC*, 1994, pp. 273–282

[39] M.J. Kearns, Efficient noise-tolerant learning from statistical queries. *J. ACM* **45**(6), 983–1006 (1998)

[40] J. Kilian, Founding cryptography on oblivious transfer, in *Proc. 20th STOC*, 1988, pp. 20–31

[41] L.A. Levin, Universal sequential search problems. *PINFTRANS: Probl. Inf. Transm.* Translated from Problemy Peredachi Informatsii (Russian) **9** (1973)

[42] N. Linial, Y. Mansour, N. Nisan, Constant depth circuits, Fourier transform, and learnability. *J. ACM* **40**(3), 607–620 (1993). Preliminary version in Proc. 30th FOCS, 1989

[43] V. Lyubashevsky, The parity problem in the presence of noise, decoding random linear codes, and the subset sum problem, in *Proc. 9th Random*, 2005

[44] R.J. McEliece, A public-key cryptosystem based on algebraic coding theory. Technical Report DSN PR 42-44, Jet Prop. Lab., 1978

[45] E. Mossel, A. Shpilka, L. Trevisan, On ϵ-biased generators in NC^0, in *Proc. 44th FOCS*, 2003, pp. 136–145

[46] M. Naor, O. Reingold, Synthesizers and their application to the parallel construction of pseudo-random functions. *J. Comput. Syst. Sci.* **58**(2), 336–375 (1999). Preliminary version in Proc. 36th FOCS, 1995

[47] C. Papadimitriou, M. Yannakakis, Optimization, approximation, and complexity classes. *J. Comput. Syst. Sci.* **43**, 425–440 (1991). Preliminary version in Proc. 20th STOC, 1988

[48] O. Regev, On lattices, learning with errors, random linear codes, and cryptography, in *Proc. 37th STOC*, 2005, pp. 84–93

[49] M. Sudan, Algorithmic introduction to coding theory—lecture notes, 2002. http://theory.csail.mit.edu/~madhu/FT01/

[50] R. Varshamov, Estimate of the number of signals in error correcting codes. *Dokl. Akad. Nauk SSSR* **117**, 739–741 (1957)

[51] E. Viola, On constructing parallel pseudorandom generators from one-way functions, in *Proc. IEEE Conference on Computational Complexity 2005*, pp. 183–197

[52] A.C. Yao, Theory and application of trapdoor functions, in *Proc. 23rd FOCS*, 1982, pp. 80–91