Innovations in Computer Science 2010

Cryptography by Cellular Automata or How Fast Can Complexity Emerge in Nature?

Benny Applebaum[1] Yuval Ishai[2] Eyal Kushilevitz[2]
[1]Computer Science Department, Princeton University
[2]Computer Science Department, Technion and UCLA
benny.applebaum@gmail.com yuvali@cs.technion.ac.il eyalk@cs.technion.ac.il

Abstract: Computation in the physical world is restricted by the following spatial locality constraint: In a single unit of time, information can only travel a bounded distance in space. A simple computational model which captures this constraint is a *cellular automaton*: A discrete dynamical system in which cells are placed on a grid and the state of each cell is updated via a local deterministic rule that depends only on the few cells within its close neighborhood. Cellular automata are commonly used to model real world systems in nature and society.

Cellular automata were shown to be capable of a highly complex behavior. However, it is not clear *how fast* this complexity can evolve and how common it is with respect to all possible initial configurations. We examine this question from a computational perspective, identifying "complexity" with computational intractability. More concretely, we consider an n-cell automaton with a random initial configuration, and study the minimal number of computation steps $t = t(n)$ after which the following problems can become computationally hard:

- **The inversion problem.** Given the configuration y at time t, find an initial configuration x which leads to y in t steps.
- **The prediction problem.** Given an arbitrary sequence of $\ell > n$ intermediate values of cells in the computation, predict some value in the sequence based on the previous values with a significant advantage over guessing.

These two problems capture the natural goals of inferring the past from the present and predicting the future based on partial observations of the past. Our main results show that, under widely believed conjectures, there are cellular automata for which both problems become hard even after a *single* computation step. This is done by constructing cryptographic one-way functions and pseudorandom generators which are computed by a single step of a cellular automaton. Our results support the view that computational forms of complexity can emerge from simple local interactions in a very common and immediate way.

Our results build on and strengthen previous results of Applebaum et al. (FOCS 2004, CRYPTO 2007) on the parallel complexity of cryptography. These previous works implement cryptographic primitives by circuits with constant depth, constant fan-in and constant fan-out, but inherently fail to satisfy the strong spatial locality requirement.

Keywords: Cryptography; Cellular Automata; Natural Computation; Constant Parallel Time.

1 Introduction

Computation in the physical world is restricted by the following spatial locality constraint: In a single unit of time, information can only travel a bounded distance in space. A simple computational model which captures this constraint is a *cellular automaton* (CA): A discrete dynamical system in which cells are placed on a grid and the state of each cell is updated via a local deterministic rule that depends only on the few cells within its close neighborhood. Conway's Game of Life [1] is a famous instance of CA in which each cell is initialized to be either "alive" or "dead," and in each time step each cell interacts with the eight neighboring cells surrounding it and determines whether to live or die based on the number of its living neighbors.

CAs were introduced by von Neumann and Ulam in an attempt to model natural physical and biological systems [2]. Despite their simple structure, CAs were shown to exhibit complex computational and dynamical phenomena such as self-replication, universal computation, synchronization, fractality, and chaos [2–6].

The ability to generate complex behaviors by using simple basic rules makes CAs a powerful tool: Scientists use them to simulate and analyze real world systems in nature and society, and engineers view them as massive parallel processing machines that can efficiently solve complicated algorithmic tasks. (See [7] for a survey on CAs and their applications.) Indeed, in the last few decades CAs have become the focus of extensive research efforts with specialized conferences and journals devoted to their study.

Being a universal computational model, CAs are capable of a highly complex behavior. However, it is not clear *how fast* this complexity can evolve and how common it is with respect to all possible initial configurations. We examine this question from a computational perspective, identifying "complexity" with computational intractability. We focus on two fundamental notions of computational intractability: "one-wayness" [8], which captures the intractability of inferring the past from the present; and "pseudorandomness" [9, 10] which roughly corresponds to the intractability of making informed predictions about the future based on *partial* observations of the past. Our main results show that, under widely accepted conjectures, even a single step of a CA is capable of generating one-wayness and pseudorandomness with respect to almost all initial configurations. This is formally captured by the ability to compute cryptographically strong functions by a single step of CA. Our results show that, at least from a computational point of view, the evolution of complexity can be both *common* and *fast*.

1.1 The Inversion Problem

Can the present be used to make inferences about the past? Consider a two-dimensional CA in which each of the n cells takes a binary value of 0 or 1, and the local update rules need not be identical for all grid cells [11, 12]. We assume that the full description of the CA, including all the local rules, is known. Initialize the CA with a *random* configuration $x = (x_1, \ldots, x_n)$ by assigning a randomly chosen value to each cell, and let the CA evolve for t steps to a configuration $y = (y_1, \ldots, y_n)$. Is it possible to efficiently recover the initial configuration x from the observed configuration y, or alternatively some other initial configuration x' which leads to y in t steps?

While it is easy to efficiently emulate the computation in the forward direction (that is, compute y given x), it is not clear how to do the same in the backward direction, and it seems plausible that, if we wait long enough (i.e., let t be sufficiently large), the inversion problem becomes computationally hard. Since we would like to know how fast intractability evolves, let us consider the extreme case where $t = 1$. As each cell only affects and depends on its close neighborhood, finding x boils down to solving a system of equations $y_i = f_i(x)$ of a very simple form: each equation involves a small number (say 9) of unknown variables, and each unknown variable x_j participates in a small number of equations. Furthermore, these dependencies are dictated by a simple geometrical structure implied by the spatial locality. One can show that when the CA is one-dimensional (embedded on a line) the inversion problem is easy – a consistent initial configuration can always be found in time polynomial in n [13]. (Note that a Turing Machine can be viewed as an instance of a one-dimensional CA over a large alphabet.) Moreover, even in the case of two-dimensional CAs, one can exploit the geometrical structure and obtain non-trivial algorithmic shortcuts. Specifically, by using the techniques of [14, 15], one can find a consistent initial configuration in subexponential time of $2^{O(\sqrt{n})}$, or alternatively, find in polynomial time an approximate solution: an initial configuration x' that leads to a configuration y' which differs from y only in an arbitrary small constant fraction of the cells. (See Proposition 3.2 for formal details.)

Despite this evidence for simplicity, the proof of Cook's theorem [16] shows that, even for a single computation step, the inversion problem of a two-dimensional CA is NP-hard. While NP-hardness suggests that there are *some* hard instances, it says nothing about their frequency. It might still be the case that only a tiny fraction of the instances are hard, and there exists an efficient algorithm that solves the inversion problem with probability which is extremely close to 1. A more satisfactory notion of hardness will be to show that for some CAs the problem is almost always intractable – namely, the question is whether CAs are capable of computing one-way functions in a small number of steps.

Assuming the intractability of decoding a random linear code (DRLC), we give an affirmative answer to this question and construct a two-dimensional CA that computes a one-way function in a single computation step. We conclude that, barring a major breakthrough in coding theory, there are CAs for which the inversion problem is hard with respect to almost all initial configurations even when $t = 1$.

1.2 Pseudorandomness and Predictability

One may try to capture irregularity and chaotic behavior via the cryptographic notion of pseudorandomness. Let us consider the following experiment: a CA

is initialized with an unknown random configuration $x \in \{0,1\}^n$ and runs for t computation steps. Suppose that we have only a partial view of the computation, that is, we are given only a sequence of ℓ intermediate values $y = y_1, \ldots, y_\ell$ collected from several sites during the computation. How random can this sequence look like? Can we identify regular patterns or predict some values of the sequence based on previous values? Equivalently, is it possible to distinguish y from a truly random binary sequence z of length ℓ?

To make the question meaningful, let us assume that the number of observations ℓ is larger than n, the amount of randomness that was used to initialize the system. In this case, y is statistically far from being truly random and with no computational limitations it is easy to distinguish y from a uniformly chosen ℓ-bit string. This also implies that it is possible in principle to predict some values in the sequence based on previous values with a significant advantage over guessing [10]. However, it might be the case that no such *efficient* procedures exist. In cryptographic terms, we ask whether the distribution of y can be pseudorandom.

Wolfram [17] suggested a heuristic construction of a CA which can generate a long pseudorandom sequence from a random initial configuration. Concretely, he suggested to let y_i be the value of the middle cell in a simple one-dimensional CA at the beginning of the i-th computation step.[1] In this case, a non-trivial pseudorandom sequence of length $\ell = n+1$ may be generated in $t = n$ steps. Following Wolfram many other candidates were suggested [18–22]. However, to the best of our knowledge, all these constructions require t to be relatively large, at least linear in n. In addition, none of these constructions is proven to be secure under a well studied intractability assumption. We show that unpredictable behavior can evolve very fast — even a single computation step of a CA is capable of generating a pseudorandom sequence of length $\ell > n$. The sequence includes some cells from the initial configuration together with all cells of the next configuration. Again, this construction is based on the conjectured intractability of decoding random linear codes.

1.3 Other Cryptographic Primitives

We also study the possibility of performing other useful cryptographic computations by a single step of a CA. In addition to one-way functions and pseudorandom generators, we show that commitment schemes, symmetric encryption schemes and private-key identification schemes can all be computed in this model under the DRLC assumption.

We also show that, assuming the security of the McEliece cryptosystem [23], it is possible to construct a semantically-secure public-key cryptosystem [24] whose encryption algorithm can be computed by a single step of a CA. The possibility of basing a public-key cryptosystem on the hardness of inverting different models of automata (including CAs) was suggested in the past [18, 25–27]. However, previous attempts resulted in ad-hoc constructions (with no proof of security) that were not computable in a constant number of steps. Our construction is the first to avoid these drawbacks.

On the negative side, we describe a $2^{O(\sqrt{n})}$ attack against CA-based primitives which tightly matches the security of our constructions. We also show that functions which are computable by CAs in a constant number of steps cannot realize several other primitives such as pseudorandom generators with linear stretch and encryption schemes of a *constant rate*. (See Corollary 3.3.) This should be contrasted with the previous positive results from [28] – see Section 1.5 below for discussion. We also note that other negative results that were proved for weaker notions of parallel cryptography [29–31] apply to our setting as well. These negative results imply that cryptographic tasks such as decryption, signing, and verifying signatures (in both the private key and the public key world) cannot be implemented by a CA in a constant number of steps. The combination of our positive results and these negative results provides a fairly complete picture as to which major cryptographic primitives can be implemented by CAs in a constant number of steps.

1.4 Application: Cryptography with Constant Physical Latency

Traditional theoretical measures of circuit complexity treat circuits as topological rather than physical objects. As such they do not take into account the size of gates and the distances traveled by signals in an actual embedding of a circuit in physical space. In particular, parallel time complexity is measured by counting the maximal number of gates, or equivalently wires, traversed by an input signal on its way to an output, without considering the *physical length* of the wires.

In reality, however, chips with long wires are slower and harder to power than chips with short wires [32]. On a more fundamental level, barriers on the speed of light and minimal size of devices impose inherent limitations on parallel time complexity in the physical world. This discrepancy between theoretical measures of parallel complexity and physical limitations has

[1] Interestingly, his construction is being used as the pseudorandom generator of the software Mathematica.

drawn a considerable amount of attention in the areas of VLSI design and parallel computing (see [33, 34] and references therein). It has also led to criticism against the meaningfulness of the complexity class NC (as a model for "efficient" parallel computation).

The possibility of fast cryptographic computations by CAs allows to address this limitation. Our CA constructions yield implementations of several primitives (i.e., one-way functions, pseudorandom generators, symmetric/public-key encryption schemes, commitment schemes and symmetric identification schemes) with *constant physical latency* – i.e., the circuit families that compute these primitives can be embedded on a two-dimensional grid such that the maximal distance traveled by an input signal on its way to an output gate is bounded by an absolute constant that does not grow with the input length or the level of security. This shows that, at least in principle, useful cryptographic computations can be carried out in constant parallel time even when the geometry of the physical world is taken into account. (An additional discussion of the physical latency model appears in Section 7.)

1.5 Previous Notions of Parallelism

It is instructive to compare the models considered in this paper to previous notions of highly parallel cryptography. For this, it will be useful to view the input-output dependencies of a function f as a bipartite graph G in which left nodes correspond to input variables, right nodes correspond to output variables, and an edge connects an input node to an output node if and only if the corresponding input variable affects the corresponding output variable. The function has constant *output locality* (resp., constant *input locality*) if the degree of the output nodes (resp., input nodes) is bounded by a constant.

In [30] it was shown that, under standard intractability assumptions, many useful cryptographic tasks can be implemented by functions with constant output locality. This positive result was later strengthened for a subclass of these tasks to allow implementations with both constant input locality and constant output locality [31]. In contrast, the current work mainly deals with constant *spatial locality*: it should be possible to embed the graph G (via an injective mapping) on the two dimensional grid $\mathbb{N} \times \mathbb{N}$ such that the Euclidean distance between each pair of adjacent nodes is bounded by some constant d. It can be shown that this notion of locality is strictly stronger than the previous ones (see Proposition 3.1). More importantly, as we argue below, this distinction makes a qualitative difference even from a cryptographic point of view.

The spatial locality property prevents G from having good connectivity properties. In particular, G is a poor expander which has relatively small separators. This limitation does not apply to graphs with low input and output locality, as constant-degree bipartite graphs can have a very good expansion. From a cryptographic perspective, good expansion makes cryptanalysis harder and, indeed, several previous cryptographic constructions of low input/output locality were based on such expander graphs [28, 35, 36]. The good expansion of these constructions prevents them from having spatially local embedding. To the best of our knowledge, *all* previous implementations of nontrivial cryptographic primitives in the literature (even parallel constructions that do not rely on expanders) can be proved to have super-constant spatial locality.

In fact, the existence of small separators for spatially local graphs leads to some actual attacks. As already mentioned, spatially local functions can be inverted in sub-exponential time, or alternatively, approximately inverted in polynomial time. This vulnerability can be used to show that variants of some primitives (e.g., pseudorandom generators with linear stretch) that can be computed with constant input and output locality [28], cannot be realized by spatially local functions. (See Section 3.4.) All in all, one may suspect that spatially local functions are too weak to perform any cryptographic computation. Still, our results show that many cryptographic primitives can be implemented in this class.

Finally, we mention that, as a byproduct of our techniques, we also obtain some improvement over previous results on the input locality of cryptographic primitives. Previous constructions from [31] only yield a *collection* of primitives (e.g., one-way functions and pseudorandom generators) with constant input locality, where a random function from the collection is secure with high probability. In this work, we obtain the first *explicit* one-way functions and pseudorandom generators in which each bit of the input influences only a constant number of output bits. (As in [31], our constructions achieve optimal input-locality and output-locality simultaneously.) Similarly, our symmetric encryption scheme is the first encryption scheme in which each bit of the input, secret key, or randomness influences only a constant number of output bits. A previous construction from [31] only has this property for any *fixed* key (i.e., by allowing the topology of the encryption circuit to depend on the key).

1.6 Organization

We begin with an overview of our techniques (Section 2). Then, we formally present the notions of cellular automata and spatially local functions, and make some initial observations regarding these computational models (Section 3). Our main results regarding the average-case hardness of CAs are proven in three steps (Section 4– 6): First we show that spatially local primitives can be constructed based on "algebraically simple" cryptographic functions (Section 4), then we argue that, under the DRLC assumption, "algebraically simple" functions can achieve cryptographic hardness (Section 5), and finally we show how to convert spatially local cryptographic primitives into primitives that can be implemented by a single computation step of a CA (Section 6). We end the paper by showing that our results give rise to cryptographic hardware with constant physical latency (Section 7).

2 Overview of Techniques

At the heart of our results is a new randomization method for "algebraically simple" functions. Before describing our construction and its relation to previous works, let us briefly recall the notion of a *randomized encoding* of functions from [30, 37].

We say that a function $f(x)$ is encoded by a randomized mapping $\hat{f}(x,r)$ if for every x the value $f(x)$ is "information-theoretically equivalent" to the distribution $\hat{f}(x,r)$ induced by a random choice of r. Specifically, one should be able to decode $f(x)$ given a sample from $\hat{f}(x,r)$, and vice versa: given $f(x)$ it should be possible to sample the distribution $\hat{f}(x,r)$. We mention that, syntactically, the input length and output length of $\hat{f}$ are typically larger than those of f. We will later elaborate more on the notion of randomized encoding and its usefulness for low complexity cryptography. For now, let us just mention that, by the results of [30], in order to implement a cryptographic primitive by a function with constant spatial locality (CSL), it suffices to show that functions with CSL can encode some non-trivial class of functions STRONG in which cryptographic functions exist.

To illustrate our main technique, consider the following problem. We are given a linear function $L : \mathbb{F}_2^n \to \mathbb{F}_2^\ell$ which maps a column vector x to the vector Mx where M is some matrix in $\mathbb{F}_2^{\ell \times n}$. Our goal is to encode the function L by a function $\hat{L}$ which can be computed with CSL. Recall that the spatial locality of $\hat{L}$ is d if the dependencies graph $\hat{G}$ of $\widehat{\mathcal{L}}$ can be embedded (via an injective mapping) on a two dimensional grid $\mathbb{N} \times \mathbb{N}$ such that the Euclidean distance between each pair of adjacent nodes is at most d. Note that constant spatial locality implies, in particular, that the degrees of all output nodes and the degrees of all input nodes of $\hat{G}$ are bounded by a constant (i.e., $O(d^2)$).

1) The encoding of [31].

An encoding that achieves the low-degree properties was suggested in [31]. This encoding relies on two "gadgets": an output-reduction gadget which replaces an output node of large degree with a (larger) subgraph of low degree, and an input-reduction gadget which does the same for a high-degree input node. (In both cases, new inputs and outputs are introduced.) These gadgets can be used to reduce the degree of some specific node without increasing the degree of other nodes. Hence, they allow to gradually improve the encoding by applying a sequence of local operations. In the end of this process, one gets an encoding whose dependency graph $\hat{G}$ has a constant degree. It is natural to try to obtain a dense embedding of $\hat{G}$ in which the edges are short (e.g., by "squeezing" the graph). However, it can be shown that this is, in general, impossible — we will always have some adjacent nodes u and v which are placed far from each other.[2]

2) Shortening edges via randomization.

We cope with the above problem by making an additional use of randomization: Instead of moving the adjacent nodes, u and v, one towards the other, we will bridge the distance by adding new intermediate (random) inputs and outputs that will fill the gap. Indeed, this is possible (in the case of linear functions) by splitting a single output $u = v + (w_1 + \ldots + w_k)$ into a "chain" of $k+1$ outputs of the form $(u_1 = v + r_1, u_2 = r_1 + r_2, \ldots, u_k = r_{k-1} + r_k, u_{k+1} = r_k + (w_1 + \ldots + w_k))$ where $r = (r_1, \ldots, r_k)$ are new random inputs. However, these new intermediate nodes occupy area and therefore this transformation is possible only if the area between the two problematic nodes is free.

3) A problematic local approach.

We can try to take care of each "long" edge $e = (u, v)$ locally by pushing away the nodes which are placed along e. However, this may lengthen other edges that we already took care of. Hence, we need a global strategy that keeps the area "under" the edges free. Furthermore, we should avoid having a long edge e which crosses a large number of other long edges, as

[2] This can be proven by letting M be the adjacency matrix of a good (bipartite) expander of fixed degree. In this case, the encoding of [31] has linear number of variables and low (logarithmic) diameter. Hence, any planar, or even 3-dimensional, embedding of its dependency graph suffers from long (non-constant) edges. See also Proposition 3.1.

in this case our edge shortening procedure will place many vertices along e. In general, it seems hard to satisfy these requirements. Nevertheless, it turns out that the special structure of the encoding of [31] gives rise to such a nice arrangement.

4) Aligning the graph to the matrix M.

The important observation here is that both the output-reduction and the input-reduction gadgets result in a "chain" or "comb"-like graph structures. Hence, we can place them along the rows and columns of the grid. Specifically, we will associate each row of the grid with one original output of L, and each column with an original input variable — similarly to the structure of the matrix M which defines the linear function. Our encoding is now constructed as follows: (1) Apply the output gadget to the i-th output and place the resulting chain-like graph along the i-th row. Do this while keeping the adjacent inputs aligned to their columns. This leads to long horizontal edges whenever the i-th row of M has a large number of consecutive zeroes; (2) Apply the input gadget to the i-th input and place the resulting comb-like graph along the i-th column. Do this while keeping the adjacent outputs aligned to their rows. This leads to long vertical edges whenever the i-th column of M has a large number of consecutive zeroes; (3) Since the long edges (and in fact all edges) are stretched over free area, we can apply our transformation and replace a long edge with a sequence of new random inputs and new outputs.

5) Encoding the universal linear function.

While the encoding of [31] works for every *fixed* linear function, it falls short of encoding the universal linear function $\mathcal{L}(M, x) = (M, Mx)$, which takes the matrix M as part of its input. As a result, the cryptographic constructions of [31] allow fast parallel implementation only after some additional *preprocessing*. Due to the correspondence between the embedding of our current encoding and the structure of the matrix M, we are capable of dealing with $\mathcal{L}$, and as a result get new implementations of primitives that do not require any preprocessing.

3 CAs and Spatial Locality

In this section we formally define the notions of cellular automata and spatial locality. We also provide some basic observations regarding these notions.

3.1 Cellular Automata

We will consider two-dimensional CAs over the binary alphabet with inhomogeneous updating rules [11, 12].[3] In such a CA, cells are ordered on an $\ell \times n$ grid and they can take a binary value of 0 or 1. A configuration of the CA consists of binary assignments to each of its cells. At the beginning, an initial configuration $x = (x_{i,j})_{1 \le i \le \ell, 1 \le j \le n}$ is selected by assigning a value for each cell. At the $t+1$ time step, the state of each cell is updated by applying some fixed function $f_{i,j}$ to the current state of the cell and the states of the cells in its neighborhood. We will assume that the neighborhood consists of the ρ-th closest cells for some small fixed integer ρ which is called the radius of the CA. Different cells are allowed to use different updating functions f_i as long as the function remains fixed over time. Every CA naturally defines a next configuration function $\delta : \{0,1\}^{\ell \cdot n} \to \{0,1\}^{\ell \cdot n}$ which maps a configuration to its successor. We will sometimes abuse notation and refer to δ as mapping N-bit strings to N-bit strings where $N = \ell \cdot n$. We say that a function family $g = \left\{g_n : \{0,1\}^{N(n)} \to \{0,1\}^{N(n)}\right\}_{n \in \mathbb{N}}$ is *computable by a CA in $t(n)$ steps* if there exists a sequence of CAs $\{C_n\}_{n \in \mathbb{N}}$ for which the following conditions hold. For each n and initial configuration $x \in \{0,1\}^{N(n)}$, the CA C_n reaches the configuration $g_n(x)$ after $t(n)$ computation steps. Furthermore, we require the existence of a fixed integer ρ which upper-bounds the radius of all the C_n's. In this work, we construct cryptographic functions that can be computed by a CA in a *single* step.

6) Uniformity.

We will always assume that CA families (resp., circuit families) are *polynomial-time uniform*: there exists a probabilistic polynomial time Turing machine T which given 1^n outputs the description of n-th CA (resp., circuit) in the family. In fact, most of our constructions satisfy a stronger notion of DLOGTIME uniformity [38, 39] which means that given $(1^n, 1^i)$ the machine T outputs the description of the i-th cell (resp. gate) of the n-th CA (resp., circuit) in logarithmic time.

3.2 Spatial Locality

Instead of proving our results by directly constructing suitable families of CAs, it will be more convenient to use the following notion of *spatial locality*. We view the input-output dependencies of a function $g : \{0,1\}^N \to \{0,1\}^{N'}$ as a bipartite directed graph $G_g = ((\text{In}, \text{Out}), E)$ where the set of input nodes In correspond to input variables, the set of output nodes Out correspond to output variables, and

[3] The CAs we construct are inhomogeneous in that they apply different update rules to different cells. However, the pattern of the update rules is highly regular.

the i-th input node is connected to the j-th output node if and only if the i-th input influences the j-th output. We say that the *spatial locality* of g is d if its dependencies graph G_g can be embedded (via an injective mapping) on a two dimensional grid such that the Euclidean distance between each pair of adjacent nodes is at most d. We say that a function family $g = \left\{g_n : \{0,1\}^{N(n)} \to \{0,1\}^{N'(n)}\right\}_{n\in\mathbb{N}}$ is *spatially local* if it has constant spatial locality, i.e., there exists a constant d which bounds the spatial locality of g_n for all $n \in \mathbb{N}$.

Spatial locality is closely related to the CA model: if the function family g is computable by a CA in $O(1)$ steps then it is also spatially local. Our construction will first be formulated as spatially local functions, and then implemented via a CA.

3.3 Relation to Other Notions of Locality

Cryptographic functions which satisfy other notions of locality were studied in [30, 31]. In particular, the function family $f = f_n$ has *output locality* (resp. *input locality*) d if the outdegrees (resp. indegrees) of the graph family G_{f_n} is bounded by d. It is not hard to see that spatial locality of d implies an $O(d^2)$ upper bound on both input and output locality. Hence, the class of spatially local functions is a sub-class of the class of functions for which both the input locality and the output locality are constant. The converse is however not true — there are functions with constant input and output locality whose dependency graphs cannot be embedded on the grid with constant spatial locality. For instance, this is the case with functions whose dependency graph is a good (constant-degree) expander with low diameter. As shown in the next proposition, such functions do not have spatial locality.

Proposition 3.1. *Let G_f be the dependency graph of a function $f : \{0,1\}^n \to \{0,1\}^m$. If G_f has diameter d then the spatial locality of f is no smaller than $\sqrt{n+m}/d$.*

Proof. Fix some embedding of G_f. Assume, without loss of generality, that the leftmost bottom node u is placed in position $(1,1)$. Since there are $m+n$ nodes, there exists at least a single node v which lies outside of the square $[1..\sqrt{n+m}-1]\times[1..\sqrt{n+m}-1]$. Hence, v and u are $\sqrt{n+m}$ far from each other. Since u and v are connected in G_f via a path of length d, it follows that at least one of the edges of the path connects two nodes which are at least $\sqrt{n+m}/d$-far from each other.

Proposition 3.1 implies that previous constructions of highly parallel cryptographic functions from [28, 30, 31] are not spatially local, and hence cannot be computed by a CA in a constant number of steps.

3.4 Inverting a CA in the Worst-case

Consider the problem InvertCA$_1$ (resp. InvertCA$_2$) whose input consists of a one-dimensional CA (resp. two-dimensional CA) which computes the function $f : \{0,1\}^N \to \{0,1\}^N$ in a single step and a string $y \in \{0,1\}^N$, and the output should be a preimage x of y under f or a $\perp$ symbol if such a preimage does not exist.

Proposition 3.2. *The worst-case intractability of* InvertCA *is summarized as follows:*

1. InvertCA$_1$ *can be solved in polynomial time. Moreover, we can efficiently output a random preimage which is uniformly distributed over the set $f^{-1}(y)$.*
2. InvertCA$_2$ *is* NP*-hard. In fact, it is even* NP*-hard to decide whether y is in the image of f.*
3. InvertCA$_2$ *has a PTAS. That is, given an arbitrary constant proximity parameter $\epsilon > 0$ and a string $y \in \{0,1\}^N$ in the image of f, it is possible to find in polynomial time a string $x' \in \{0,1\}^N$ for which $\Delta(y, f(x')) \le \epsilon$, where $\Delta(\cdot,\cdot)$ denotes the relative Hamming distance.*
4. InvertCA$_2$ *can be solved in $2^{\sqrt{N}}$ time.*

Proof. (1) The proof is similar to the algorithm of [13] for inverting functions that can be computed by a ring-protocol of low communication complexity. Assume that the inputs $x = (x_1, \ldots, x_N)$ and outputs $(f_1, \ldots, f_N)$ of f are ordered on a line from left to right. Let ρ be the radius of the CA. We partition the inputs to distinct blocks of length 2ρ where the i-th block B_i contains the inputs $x_{2\rho(i-1)+1}, x_{2\rho(i-1)+2}, \ldots, x_{2\rho(i-1)+2\rho}$. (We pad the inputs with additional dummy variables to make n a multiple of 2ρ.) The important observation is that each bit of the string y impose a constraint on inputs that lie in (at most) two *consecutive* input blocks. Let Y_i be the set of constraints which are imposed by y on the blocks B_i and B_{i+1}. We reduce the inversion problem to a connectivity problem. We construct a layered graph G with $k = N/2\rho$ layers, where each layer contains $2^{2\rho}$ nodes labeled by strings in $\{0,1\}^{2\rho}$. We connect a node $v \in \{0,1\}^{2\rho}$ of the i-th layer to a node $u \in \{0,1\}^{2\rho}$ in the $i+1$-th layer if the assignment (u,v) is consistent with all the constraints in Y_i. It is not hard to verify that $v_1, \ldots, v_k$ is a path from the first to the last layer if and only if $v = (v_1, \ldots, v_k) \in \{0,1\}^n$ is a preimage of y under f. Clearly, we can find such a path, or even a randomly chosen path, in polynomial-time and thus the first part of the proposition follows.

(2) The Cook-Levin Theorem [16, 40] shows that it is NP-hard to decide whether the following constraints satisfaction problem (CSP) is satisfiable.

- **Variables**: $(X_{i,j})_{1\leq i\leq t, 1\leq j\leq m}$, where $t = t(n)$ and $m = m(n)$ are polynomials and the domain of $X_{i,j}$ is some constant-size alphabet Σ.
- **Constraints**: $(Y_{i,j})_{1\leq i\leq t, 1\leq j\leq m}$ where $Y_{i,j}$ involves the variables $X_{i,j}, X_{i-1,j-1}, X_{i-1,j}$ and $X_{i-1,j+1}$ (where $X_{0,i}$ and $X_{i,0}$ are set to be dummy variables with some fixed arbitrary value).

We reduce this CSP to InvertCA_2. We define $f' : \Sigma^{tm} \to \{0,1\}^{tm}$ as follows. For each $X_{i,j}$ we put an input $x_{i,j}$ and for each $Y_{i,j}$ we put an output $f'_{i,j}$ which is 1 if the value of $x_{i,j}, x_{i-1,j-1}, x_{i-1,j}$ and $x_{i-1,j+1}$ satisfies the constraint $Y_{i,j}$, and 0 otherwise. The function f' can be computed by a CA in a single computation step over the alphabet Σ. Since the size of Σ is constant, f' can be also implemented by single step of a binary CA f (by replacing each $x_{i,j}$ with $\lceil \log |\Sigma| \rceil$ new boolean variables). We complete the proof by noting that $y = 1^{tm}$ is in the image of f if and only if the CSP problem is satisfiable.

(3, 4) To prove the last items of the theorem, note that the inversion problem can be described as a CSP problem whose constraint graph is "almost planar" and, it trivially has a relatively small $(O(\sqrt{N}))$ separator S which partition the graph into two roughly equal parts A and B. (Just "cut" the grid horizontally or vertically into two halves by removing the middle ρ rows/columns.) Such a CSP can be approximated in polynomial time or solved in sub-exponential time by using standard divide-and-conquer techniques, e.g. [14, 15]. For completeness, let us describe the $2^{O(\sqrt{N})}$-time algorithm. Try each of all possible $2^{O(\sqrt{N})}$ assignments for the variables in S, and for each such partial assignment recursively solve the two sub-problems which correspond to the variables in A and B (these instances are induced by the partial assignment for S). The time complexity of this attack is given by the recursion: $T(N) = \text{poly}(N) + 2^{O(\sqrt{N})}T(N/2)$ which solves to $2^{O(\sqrt{N})}$.

We will later show that if a random linear code of constant rate cannot be decoded in sub-exponential time (an assumption which is consistent with the current state of knowledge), then there exists a $2^{\Omega(\sqrt{N})}$-secure one-way function which can be computed by a single computation step of a two-dimensional CA. This matches the upper-bound of item 4.

Proposition 3.2 can be easily generalized to the case of spatially local functions. By combining item 3 of (this generalized version of) Proposition 3.2 with the results of [28] we derive the following corollary.

Corollary 3.3. *The following primitives cannot be computed by a single step of CA (or by a spatially local function): (1) pseudorandom generator with linear stretch (i.e., n bits of inputs are expanded into $n + \Omega(n)$ pseudorandom bits); (2) an encryption algorithm of an error-free public-key cryptosystem with constant expansion factor (i.e., plaintexts of size n are encrypted by $O(n)$ size ciphertexts); and (3) a non-interactive string commitment with constant expansion factor (i.e., commitments to n-bit strings are of length $O(n)$).*

4 Encoding Functions with CSL

4.1 Overview

We say that a function $f(x)$ is encoded by a randomized mapping $\hat{f}(x, r)$ if, for every x, the value $f(x)$ is "information-theoretically equivalent" to the distribution $\hat{f}(x, r)$, induced by a random choice of r (a formal definition is given below). In [30], it was shown that the security of most cryptographic primitives is inherited by their randomized encoding. Therefore, in order to implement a cryptographic primitive by a function with constant spatial locality (CSL), it suffices to show that functions with CSL can encode some non-trivial class of functions STRONG in which cryptographic functions exist. If this is indeed the case, then one can convert a cryptographic function f in STRONG into a cryptographic function $\hat{f}$ with CSL. A similar approach was used in [30, 31, 41].

A useful observation of [31] is that, assuming intractability assumptions related to error-correcting codes, one can implement cryptographic primitives by "algebraically simple" functions. Hence, we can let STRONG be the class of such functions. We will follow this approach and construct a CSL encoding for a class of "semi-linear" functions, which is slightly weaker than the class of functions that was considered in [31], but is still sufficiently strong to achieve cryptographic hardness. In Section 4.2, we describe the encoding for the case of linear functions, and then, in Section 4.3, extend this encoding to handle a larger class of functions. Before that, let us formally define the notion of randomized encoding.

Definition 4.1. (Perfect randomized encoding [30]) *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,* $s = l + m$,

where U_n *denotes a random variable uniformly distributed over* $\{0,1\}^n$, *and the notation* $X \equiv Y$ *indicates that the random variables* X *and* Y *are identically distributed.*

The first input of $\hat{f}$ is referred to as the *(deterministic) input* and the second input of $\hat{f}$ is referred to as the *random input*. We refer 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 4.1 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.

4.2 Encoding Linear Functions

Let $M = (M_{i,j})$ be a matrix in $\mathbb{F}_2^{\ell \times n}$ and $v = (v_1, \ldots, v_n)$ be a vector in $\mathbb{F}_2^n$. The universal linear function $\mathcal{L}_{n,\ell} : \mathbb{F}_2^{\ell n + n} \to \mathbb{F}_2^{\ell n + \ell}$ takes input (M, v) and outputs the pair (M, Mv). We construct a CSL encoding for this function. It will be convenient to embed the encoding on a "virtual" grid in which each point can hold six nodes (a 2×3 grid). A spatially local encoding over a standard grid can then be obtained by scaling up the virtual grid. In the following, let $[n]$ denotes the set $\{1, \ldots, n\}$. We will sometimes identify binary strings with vectors over $\mathbb{F}_2$.

Construction 4.2. (spatially local encoding for linear functions) *Let* n, ℓ *be positive integers. The encoding* $\widehat{\mathcal{L}}_{n,\ell} : \mathbb{F}_2^{\ell n + n} \times \mathbb{F}_2^{\ell \cdot n + \ell \cdot (n-1)} \to \mathbb{F}_2^{3\ell n}$ *is defined as follows:*

- *Deterministic inputs:* $(M_{i,j})_{i \in [\ell], j \in [n]}$ *and* $(v_j)_{j \in [n]}$ *(we will also refer to* v_j *as* $w_{0,j}$*).*
- *Random inputs:* $(w_{i,j})_{i \in [\ell], j \in [n]}$, *as well as* $(s_{i,j})_{i \in [\ell], j \in [n-1]}$.
- *Embedding: We use an* $(\ell + 1) \times n$ *grid indexed by* $(i,j) \in \{0, \ldots, \ell\} \times [n]$. *For each* $j \in [n]$, *we will place the input* v_j *in the point* $(0, j)$. *Additionally, in the point* $(i, j) \in [\ell] \times [n]$ *we'll have the inputs* $(M_{i,j}, w_{i,j}, s_{i,j})$ *and the outputs:*

$$\begin{aligned} m_{i,j} &= M_{i,j}, \\ c_{i,j} &= w_{i-1,j} - w_{i,j}, \\ r_{i,j} &= s_{i,j} - s_{i,j-1} + M_{i,j} \cdot w_{i,j}, \end{aligned}$$

where $s_{i,0} = s_{i,n} = 0$ *for all* i*'s. We refer to* $m_{i,j}, c_{i,j}$ *and* $r_{i,j}$ *as the* matrix-entry, column-entry *and* row-entry, *respectively.*

The construction is outlined in Figure 1. Clearly, the spatial locality of the encoding $\widehat{\mathcal{L}}_{n,\ell}$ is constant ($d = \sqrt{10}$ over a "standard" grid), and it has input-locality of 3 and output-locality of 4. Also note that the inputs $v = (v_1, \ldots, v_n)$ lie on the boundary of the grid, have input-locality 1 and participate in outputs of locality 2.

Lemma 4.3. *For every* $n, \ell \in \mathbb{N}$, *the function* $\widehat{\mathcal{L}}_{n,\ell}$ *defined in Construction 4.2 perfectly encodes the universal-linear function* $\mathcal{L}_{n,\ell}$.

Proof. Fix n and ℓ and let $\mathcal{L} = \mathcal{L}_{n,\ell}$ and $\widehat{\mathcal{L}} = \widehat{\mathcal{L}}_{n,\ell}$. The encoding $\widehat{\mathcal{L}}$ is stretch-preserving since the number of random inputs equals the number of additional outputs (i.e., $2\ell n - \ell$). Moreover, given a string $(m_{i,j}, c_{i,j}, r_{i,j})_{i,j} = \widehat{\mathcal{L}}(M, v, w, s)$, we can decode the value of $\widehat{\mathcal{L}}(v)$ as follows: to compute $M_{i,j}$, simply take $m_{i,j}$ and, to recover the i-th entry of Mv, sum-up the row-entries of the i-th row together with the column-entries in locations (k, j) for which $k \le i$ and $M_{i,j} = 1$. Indeed, this results in

$$\begin{aligned} &\sum_{j \in [n]} M_{i,j} \cdot w_{i,j} + \sum_{\substack{k \in [i] \\ j : M_{i,j} = 1}} w_{k-1,j} - w_{k,j} \\ &= \sum_{j : M_{i,j} = 1} w_{i,j} + \sum_{\substack{k \in [i] \\ j : M_{i,j} = 1}} w_{k-1,j} - w_{k,j} \\ &= \sum_{j : M_{i,j} = 1} w_{0,j} = \sum_{j : M_{i,j} = 1} v_j, \end{aligned}$$

as required. Hence, the decoder never errs.

Fix some $v \in \mathbb{F}_2^n$ and $M \in \mathbb{F}_2^{\ell \times n}$. Let $(M, y) = \mathcal{L}(M, v)$ and let $(m_{i,j}, c_{i,j}, r_{i,j})_{i,j}$ denote the distribution $\widehat{\mathcal{L}}(M, v, U_{\ell n + \ell(n-1)})$. To prove perfect privacy, we will need the following simple claim.

Claim 4.4. *(1) the entries* $(m_{i,j})_{i \in [\ell], j \in [n]}$ *are fixed and equal to* $(M_{i,j})_{i \in [\ell], j \in [n]}$; *(2) the entries* $(c_{i,j})_{i \in [\ell], j \in [n]}$ *and* $(r_{i,j})_{i \in [\ell], j \in [n-1]}$ *are independently uniformly distributed; (3) the remaining entries* $(r_{i,n})_{i \in [\ell]}$ *are uniquely determined by* M, y *and the previous outputs. In particular, for each* $i \in \ell$, *we have* $r_{i,n} = y_i - (\sum_{j=1}^{n-1} r_{i,j} + \sum_{1 \le k \le i, j : M_{i,j} = 1} c_{k,j})$.

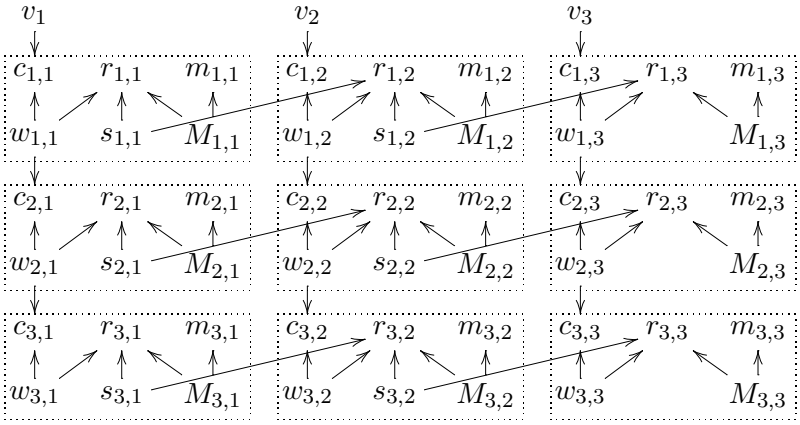

Figure 1: Randomized Encoding for the Universal-Linear Function $\mathcal{L}_{3,3}(M, v) = (M, Mv)$. Arrows connect inputs to outputs. Rows are ordered from top to bottom.

of Claim. The first item is trivial. To see (2), observe that each of these outputs is a linear function that contains a fresh random bit. Namely, the output $c_{i,j}$ (respectively, $r_{i,j}$) depends on $w_{i,j}$ (respectively, $s_{i,j}$). Item (3) follows from the perfect correctness of the decoder as for every $i \in [\ell]$, we have $y_i = \sum_{j=1}^{n} r_{i,j} + \sum_{1 \le k \le i, j: M_{i,j}=1} c_{k,j}$.

Hence, define a perfect simulator as follows. Given $(M, y) \in \{0,1\}^{\ell n + \ell}$, the simulator S chooses two random strings $(\sigma_{i,j})_{i \in [\ell], j \in [n]}$ and $(\rho_{i,j})_{i \in [\ell], j \in [n-1]}$, and outputs:

$$m_{i,j} = M_{i,j}, \qquad c_{i,j} = \sigma_{i,j}, \qquad r_{i,j} = \rho_{i,j},$$

where $(i, j) \in [\ell] \times [n-1]$, and, for $i \in [\ell]$

$$m_{i,n} = M_{i,n}, \qquad c_{i,n} = \sigma_{i,n},$$

and

$$r_{i,n} = y_i - \left(\sum_{j=1}^{n-1} r_{i,j} + \sum_{1 \le k \le i, M_{i,j}=1} c_{k,j} \right).$$

This simulator is also balanced as each of its outputs is a linear function that contains a fresh random bit. Namely, the output bit $m_{i,j}$ depends on $M_{i,j}$, the output $c_{i,j}$ depends on $\sigma_{i,j}$, and $r_{i,j}$ depends on $\rho_{i,j}$ if $1 \le j \le n-1$ and on y_i if $j = n$.

The above proof also shows that when $\ell(\cdot)$ is polynomial, the family $\widehat{\mathcal{L}}_{n,\ell(n)}$ forms a *uniform* perfect encoding for the family $\mathcal{L}_{n,\ell(n)}$.

Remark 4.5. *The output locality of the above construction can be reduced to 3 by adding additional $\ell \cdot n$ random inputs $(p_{i,j})$ and $\ell \cdot n$ additional outputs $(q_{i,j})$. The new encoding g is similar to $\widehat{\mathcal{L}}$, except that we let $q_{i,j} = M_{i,j} \cdot w_{i,j} - p_{i,j}$ and redefine $r_{i,j}$ to be $s_{i,j} - s_{i,j-1} + p_{i,j}$. It is not hard to adopt Lemma 4.3 to show that g perfectly encodes $\widehat{\mathcal{L}}$. (Alternatively, one can show that g encodes $\widehat{\mathcal{L}}$ and use the Composition Lemma of [30, Lemma 4.6] to conclude that g encodes $\mathcal{L}$.)*

Remark 4.6. *It is instructive to note that the proof of Lemma 4.3 crucially relies on the fact that the entries of the matrix M are given in the output. Indeed, it was shown in [31] that when M is hidden one cannot achieve constant input locality and, in particular, the function $f(M, v) = Mv$ does not have such a randomized encoding.*

4.3 Encoding Semi-Linear Functions

We show how to use the basic construction to derive spatially local encoding for other functions. Consider, for example, the case of a fixed linear function $f(v) = Av$ where A is a fixed matrix. Our construction can be trivially modified to encode this function (with CSL) by fixing the inputs $M_{i,j}$ to be $A_{i,j}$ and omitting them from the output. More generally, some entries of the matrix M can be fixed and some can be given as part of the input. Construction 4.2 will work as long as the mapping from the input to the entries of M is injective.

Another useful extension is obtained by replacing the vector v with the result of some simple function. For example, consider the function $f(M, x) = (M, Mv)$ where $v_i = x_i \cdot x_{i+1}$. Again, it is not hard to obtain an encoding with CSL by modifying the basic

construction. The following definition captures these variations.

Definition 4.7. *A function $f : \mathbb{F}_2^{t+k} \to \mathbb{F}_2^{t+\ell}$ is* semi-linear *if it maps the pair $(z, x) \in \mathbb{F}_2^t \times \mathbb{F}_2^k$ to the pair $(z, M(z) \cdot v(x))$, where:*

- *The mapping $M : \mathbb{F}_2^t \to \mathbb{F}_2^{\ell \times n}$ takes a t-length vector z and outputs a matrix $M(z)$ whose entries are either fixed to some constant or equal to some entry of z. Moreover, each entry of z appears in a single entry of M.*
- *The mapping $v : \mathbb{F}_2^k \to \mathbb{F}_2^n$ is a function that can be computed with CSL with respect to a 1-dimensional embedding of its dependencies graph.*

Theorem 4.8. *A semi-linear function $f(z, x) = (z, M(z) \cdot v(x))$ can be perfectly encoded with CSL on a grid in which the input nodes x are located in the first row. Moreover, if the input locality of the function v is a and its output locality is b, then the input and output locality of f are $\max(3, a)$ and $\max(3, b + 1)$, respectively.*

To prove the theorem, we will need the following simple properties of randomized encodings.

Lemma 4.9. *Let $f : \{0,1\}^n \to \{0,1\}^l$ be a function which is perfectly encoded by the function $\hat{f}(x, r)$. Consider the function $h(w) \stackrel{def}{=} f(g(w))$ where $w \in \{0,1\}^k$ and $g : \{0,1\}^k \to \{0,1\}^n$. Then, the function $\hat{h}(w, r) \stackrel{def}{=} \hat{f}(g(w), r)$ perfectly encodes h.*

Proof. It is not hard to verify that both the original decoder and the simulator of $\hat{f}$ allow perfect decoding and simulation for $\hat{h}$. Indeed, given $\hat{y} = \hat{h}(w, r) = \hat{f}(g(w), r)$, by the perfect correctness of $\hat{f}$, the decoder returns $y = f(g(w)) = h(w)$. Similarly, on an input $y = h(w) = f(g(w))$, the simulator samples the distribution $\hat{f}(g(w), U_m) \equiv \hat{h}(w, U_m)$. Finally, $\hat{h}$ is stretch-preserving since: (1) $\hat{f}$ is stretch-preserving; (2) f and h have the same output length; and (3) $\hat{h}$ and $\hat{f}$ have the same randomness complexity and output complexity.

Lemma 4.10. *Let $f : \{0,1\}^n \to \{0,1\}^{l+k}$ be a function of the form $f(x) = (g(x), \sigma)$ where $g : \{0,1\}^n \to \{0,1\}^l$ is a function and $\sigma \in \{0,1\}^k$ is a fixed string. Suppose that f is perfectly encoded by a function $\hat{f} : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^{s+k}$ of the form $\hat{f}(x, r) = (\hat{g}(x, r), \sigma)$ where $\hat{g} : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^s$. Then, $\hat{g}$ perfectly encodes g.*

Proof. Let B and S be the decoder and simulator of $\hat{f}$. We define a new decoder B' (respectively, new simulator S') which, given $\hat{y} \in \{0,1\}^s$ (respectively, $y \in \{0,1\}^l$), pads it with the string σ, hands the padded string to the original decoder (respectively, simulator) and outputs the result with the last k bits omitted. It is not hard to verify perfect correctness and privacy as well as stretch preservation. We prove that the new simulator S' is balanced. Assume, towards a contradiction, that S' is not balanced. Then, since S' is distributed over $\{0,1\}^s$, there exists a string $\alpha \in \{0,1\}^s$ for which $\Pr[S'(U_l) = \alpha] > 2^{-s}$. By definition, $S'(U_l)$ is equivalent to the s-length prefix of $S(U_l, \sigma)$. Also, by the perfect privacy of $\hat{f}$, the k-length suffix of $S(U_l, \sigma)$ is fixed to σ. It follows that $\Pr[S(U_{l+k}) = (\alpha, \sigma)] > 2^{-(s+k)}$, which contradicts the fact that S is balanced.

of Thm.4.8. By Lemma 4.9, f' is perfectly encoded by the function

$$\hat{f}'((z, x), (w, s)) \stackrel{\text{def}}{=} \widehat{\mathcal{L}}_{n,\ell}((M(z), v(x)), (w, s)).$$

Observe that f is derived from f' by omitting from the output the fixed entries of $M(z)$. Hence, by Lemma 4.10, the encoding $\hat{f}$ which results from $\hat{f}'$ by omitting the fixed entries of $M(z)$ from the output, perfectly encodes f. The CSL implementation of $\hat{f}$ is evident from the definition of $\hat{f}$ and the spatial locality of $\widehat{\mathcal{L}}$. The "Moreover" part follows by using the modification of Remark 4.5, and by noting that the entries of $v(x)$ participate at a single output entry of the form $v_j - w_{1,j}$ and so they contribute output locality of $b + 1$ and input locality of a.

Theorem 4.8 can be extended to work for more general definitions of semi-linear functions (e.g., where M is computed by a 1-dimensional CPL circuit). However, the current definition suffices for our applications.

5 Primitives with CSL

In this section, we provide a CSL implementation for several primitives. We will mainly focus on one way functions and pseudorandom generators (Section 5.2), and only briefly mention extensions to other primitives (i.e. commitment schemes, symmetric and public-key encryption schemes and identification schemes) in Section 5.3. Most of our constructions are based on the intractability of decoding a random binary linear code. In the following subsection we formalize this assumption. The reader is referred to [31] for more background on the problem.

5.1 Main Assumption: Intractability of Decoding Random Linear Code

An (m, n, δ) *binary linear code* is a n-dimensional linear subspace of $\mathbb{F}_2^m$ in which the Hamming distance between any 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. Let $H_2(\cdot)$ denote the binary entropy function, i.e., for $0 < p < 1$, $H_2(p) \stackrel{\text{def}}{=} -p\log(p) - (1-p)\log(1-p)$. It follows from the Gilbert–Varshamov bound [42] that whenever $n/m < 1 - H_2(\delta) - \epsilon$, all but a $2^{-(\epsilon/2)m}$ fraction of the $m \times n$ generator matrices form (m, n, δ)-linear codes (cf. [43, Lecture 5]).

In the following, we follow the standard cryptographic convention and refer to a function $\epsilon(\cdot)$ as *negligible* if $\epsilon(n) < n^{-c}$ for any constant $c > 0$ and sufficiently large n.

Definition 5.1. *Let $m(n) \leq \text{poly}(n)$ be a code-length parameter, and $0 < \mu(n) < 1/2$ be a noise parameter. The problem* DECODE(m, μ) *is defined as follows:*

- **Input:** *$(C, Cx + e)$, where arithmetic is over $\mathbb{F}_2$ and C is an $m(n) \times n$ random binary generator matrix, $x \stackrel{R}{\leftarrow} U_n$, and $e \in \{0,1\}^m$ is a random vector of error rate μ, i.e., each of its entries is chosen to be 1 with probability μ independently of other entries.*
- **Output:** *x.*

We say that DECODE(m, μ) *is* intractable *if every polynomial-size circuit family $\mathcal{A} = \{\mathcal{A}_n\}$ solves the problem with no more than negligible probability in n. We say that* DECODE(μ) *is intractable if* DECODE(m, μ) *is intractable for every polynomial $m(\cdot)$.*

The hardness of DECODE(m, μ) is well studied [44–50]. 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 [51]. It is widely believed that the problem is hard for every fixed μ and every $m(n) = \Theta(n)$, or even $m(n) = \text{poly}(n)$. Similar assumptions were put forward in [44, 46, 49, 52–55]. 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 DECODE(m, μ), due to Blum et al. [47], runs in time $2^{O(n/\log n)}$ and requires m to be $2^{O(n/\log n)}$. Lyubashevsky [48] 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 $m = \Theta(n)$ (and μ is constant), the problem is only known to be solved in *exponential* time.

5.2 One-way Functions and Pseudorandom Generators

A *one-way function* (OWF) is an efficiently computable function $f : \{0,1\}^n \to \{0,1\}^{m(n)}$ which cannot be efficiently inverted with non-negligible success probability. Formally, for any efficient adversary (modeled as a polynomial-size circuit family) $\mathcal{A} = \{\mathcal{A}_n\}$, the inversion probability with respect to a uniformly chosen input

$$\epsilon_{\mathcal{A}}(n) = \Pr_{x \stackrel{R}{\leftarrow} U_n, y = f(x)}[\mathcal{A}_n(y) \in f^{-1}(y)]$$

should be negligible in n.

A *pseudorandom generator* (PRG) is an efficiently computable function $G : \{0,1\}^n \to \{0,1\}^{m(n)}$ which expands its input (i.e., $m(n) > n$) and its output distribution $G(U_n)$ is pseudorandom, that is, for every (non-uniform) polynomial-size circuit family $\{\mathcal{A}_n\}$, the distinguishing advantage

$$|\Pr[\mathcal{A}_n(G(U_n)) = 1] - \Pr[A_n(U_{m(n)}) = 1]|$$

is negligible in n.

We show that, under the appropriate assumptions, pseudorandom generators can be realized by CSL functions. This also gives a one-way function with CSL, as any PRG is also one-way. We rely on the following construction of [31].

Construction 5.2. *Let $m = 13n$ and let $t = \lceil 1.1 \cdot m \rceil$. Define the function*

$$G(A, C, x, \rho) \stackrel{def}{=} (A, C, Cx + e(\rho), A \cdot \rho)$$

where $A \in \mathbb{F}_2^{t \times 2m}$, $C \in \mathbb{F}_2^{m \times n}$, $x \in \mathbb{F}_2^n$, $\rho \in \mathbb{F}_2^{2m}$, and $e(\rho) = (\rho_{2i-1} \cdot \rho_{2i})_{i=1}^m$.

The function G was shown to be a PRG, assuming that DECODE$(6n, 1/4)$ is intractable.[4] The idea was to first argue that, under assumption DECODE$(6n, 1/4)$, the distribution $(C, Cx + e(\rho))$ is pseudorandom but not expanding (as ρ is too long); and then use the mapping $(A, \rho) \mapsto (A, A\rho)$ to extract more random outputs from ρ via the leftover hashing lemma of [56]. We use G to prove the following theorem:

Theorem 5.3. *Suppose that the problem* DECODE$(6n, 1/4)$ *is intractable, then there exist a PRG and a OWF with CSL.*

[4] This assumption roughly says that it is intractable to correct $n/4$ random errors in a random linear code of relative distance $1/4 + \epsilon$, for some constant $\epsilon > 0$. (The rate $n/m = 1/6$ is strictly smaller than $1 - H_2(1/4)$, and therefore, except with negligible probability, the relative distance is larger than $1/4$. This assumption is very conservative as coding theory does not have any explicit constructions that achieves such parameters.)

Proof. We show that the function G from Construction 5.2 is a semi-linear function. Let $v(x,\rho)$ be the $n+3m$ column vector

$$(x, (g(\rho_1,\rho_2), g(\rho_3,\rho_4), \cdots, g(\rho_{2m-1},\rho_{2m})),$$

where $g(\rho_i,\rho_{i+1})$ denotes the triplet $(\rho_i, \rho_i\cdot\rho_{i+1}, \rho_{i+1})$. Also, let $M(A,C)$ be the $(m+t)\times(n+3m)$ matrix whose first n leftmost columns are $\binom{C}{\mathbf{0}_{t\times n}}$ and the remaining $3m$ columns are

$$\begin{pmatrix} \mathbf{0}_m\ \mathbf{e}_1\ \mathbf{0}_m & \mathbf{0}_m\ \mathbf{e}_2\ \mathbf{0}_m & \cdots & \mathbf{0}_m\ \mathbf{e}_m\ \mathbf{0}_m \\ A_1\ \mathbf{0}_t\ A_2 & A_3\ \mathbf{0}_t\ A_4 & \cdots & A_{2m-1}\ \mathbf{0}_t\ A_{2m} \end{pmatrix},$$

where $\mathbf{0}_k$ is the all zeroes vector of length k, $\mathbf{e}_i$ is the i-th unit vector of length m, and A_i is the i-th column of A. Then, G can be written as

$$G(A,x,C,\rho) = ((A,C), M(A,C)\cdot v(x,\rho)).$$

It is not hard to see that this formulation satisfies Definition 4.7. It follows that G is a semi-linear function and therefore, by Theorem 4.8, can be perfectly encoded by a CPL circuit $\hat{G}$. By [31, Thm. 6], the function G is a PRG and therefore, by [30, Lemma 6.1], the encoding $\hat{G}$ is also a PRG. Since any PRG is also one-way, $\hat{G}$ is also OWF.

Remark 5.4. ***(Optimal Locality)*** *Since the input locality and output locality of v are 3 and 2 (respectively), we can use the modification described in Remark 4.5 and, by Theorem 4.8, get a PRG whose input-locality and output-locality are both 3. These parameters are optimal as shown in [31]. This slightly improves the results of [31] which construct a PRG* collection *with similar parameters.*[5]

Remark 5.5. ***(Optimal Stretch)*** *Our PRG has only sublinear stretch as it maps N input bits to $N+o(N)$ pseudorandom bits. This is essentially optimal as, by Corollary 3.3, spatially local PRGs cannot have linear stretch (i.e., cannot map N inputs to $N+\Omega(N)$ outputs). Interestingly, this limitation does not apply to functions with constant input and output locality as it is shown in [28] that, under some intractability assumptions, such functions can implement PRGs with linear stretch.*

Remark 5.6. ***(Optimal Security)*** *If* DECODE$(6n, 1/4)$ *is exponentially hard — a plausible assumption which is consistent with the current state of knowledge — then, by the analysis of [31], Construction 5.2 achieves exponential security. Since the overhead which is involved in Theorem 5.3 is quadratic, we derive a $2^{\Omega(\sqrt{n})}$-secure PRG with constant spatial locality, and similarly, as shown in Section 6, a $2^{\Omega(\sqrt{n})}$-secure PRG which can be computed by a single step of CA. By Proposition 3.2 this security is optimal.*

[5] That is, [31] construct a collection $\{G_z\}_{z\in\{0,1\}^*}$ such that: (1) for every z the function G_z expands its input; (2) the distribution $(z, G_z(x))$ is pseudorandom for random x and z; and (3) for every z the input-locality and output-locality of G_z are 3.

5.3 More Cryptographic Primitives

By relying on the works of [31, 46, 57] it is possible to construct semi-linear implementations of several other primitives (symmetric encryption, commitment scheme, and private-key identification scheme) under the DRLC assumption. Furthermore, we can also construct public-key encryption scheme with a semi-linear encryption function. For this, we rely on the well known McEliece assumption [23]. That is, we assume the existence of a family $\mathcal{C}$ of efficiently decodable linear codes in which decoding is infeasible given a "scrambled" version of a code C in $\mathcal{C}$. The scrambling is obtained by randomly permuting the coordinates of C and representing it by a random generating matrix (uniformly chosen from all the matrices that generates C).

Theorem 5.7 (informal statement)**.** *Under the intractability of* DECODE$(1/8)$*, there exist:*

1. *semantically-secure symmetric encryption scheme with semi-linear encryption function;*
2. *non-interactive commitment scheme whose sender computes a semi-linear function;*
3. *one-round symmetric identification scheme whose prover computes a semi-linear function.*

Moreover, assuming the McEliece assumption with respect to some family of codes with constant relative distance[6]*, we can also get a semantically-secure public-key encryption scheme with semi-linear encryption function.*

Proof sketch. The proof of the first three items follows by a close inspection of the constructions of [31, 46, 57, 60]. (Details omitted.)

To prove the last item, recall that McEliece provides a cryptosystem with one-way security whose encryption function has the following form: To encrypt a vector $u\in\mathbb{F}_2^n$ under the key $C'\in\mathbb{F}_2^{m\times n}$, which is a garbled generating matrix of a good code C, output

[6] The McEliece scheme is usually instantiated with binary classical Goppa Codes (cf. [58], for analysis and suggestions of concrete parameters). Unfortunately, these codes are known to have an efficient decoding only for *subconstant* noise rate and therefore we cannot use them in Theorem 5.7. 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 [59].

the ciphertext $c = C'u + e$ where $e \in \mathbb{F}_2^m$ is a random error vector of fractional weight 2^{-k} for a constant k. It is not hard to see that this function has a semi-linear implementation by writing $f_{C'}(u, \rho) = (\mathbf{I}_m) \cdot (u \circ e(\rho))^{\mathrm{T}}$ where $\rho \in \mathbb{F}_2^{m \times k}$, and $e(\rho) = (\rho_{i,1} \cdot \rho_{i,2} \cdot \ldots \cdot \rho_{i,k})_{i \in [m]}$. Hence, we get a one-way cryptosystem (or a trapdoor function) with semi-linear implementation. To obtain a semantically-secure public-key encryption scheme (for a single bit), we take the exclusive-or of a hardcore predicate of f_C and a one-bit plaintext b. (This transformation is standard, cf. [61, Construction 5.3.13].) By using the Goldreich-Levin hardcore bit [62], we get the following encryption function: $E_{C'}(b; u, \rho, s) = (C'u + e(\rho), s, \langle s, u \rangle + b)$ where $u, s \stackrel{R}{\leftarrow} U_n, \rho \stackrel{R}{\leftarrow} U_{m \times k}$, $\langle \cdot, \cdot \rangle$ denotes inner product (over $\mathbb{F}_2$), and $e(\cdot)$ is defined as before. (Decryption can be done by relying on the decryption function of McEliece.) A standard argument shows that the scheme is semantically-secure. Again, it is not hard to observe that $E_{C'}$ is semi-linear as we can rewrite it as: $E_{C'}(b; u, \rho, s) = (s, M(C', s) \cdot v(u, \rho, b))$, where $M(C', s)$ is the $(m + 1) \times (n + m + 1)$ matrix

$$\begin{pmatrix} C' & \mathbf{I}_m & \mathbf{0}_{m,1} \\ s^T & \mathbf{0}_{1,m} & 1 \end{pmatrix},$$

and $v(u, \rho, b) = (u \circ e(\rho) \circ b)^{\mathrm{T}}$. In order to support polynomially long plaintexts one can use standard concatenation.

We can apply the CSL encoding of Theorem 4.8 to the semi-linear primitives of Theorem 5.7 and, since randomized encoding preserves the security of all the above primitives [30], we get implementations with constant spatial locality.[7]

6 From Spatial Locality to CA

6.1 One-wayness of CAs

We will show how to construct a CA which computes a one-way function in a single step from a spatially local OWF. In particular, it is possible to convert a function family with spatial locality to a function family f' which is computable by a single step of a CA by adding "dummy" inputs (which do not affect the output) as well as "dummy" constant outputs. The security of most cryptographic primitives are not affected by such a padding.

Proposition 6.1. *Let $f : \{0,1\}^n \to \{0,1\}^s$ be a function with spatial locality d. Then, there exists a CA with radius d which computes a function $f' : \{0,1\}^k \to \{0,1\}^k$ in a single step, such that the function f' is a padded version of f in the following sense: for every $x \in \{0,1\}^n$ and $y \in \{0,1\}^{k-n}$ it holds that $f'(x, y) = (f(x), 0^{k-s})$.*

Proof. Let H be the dependencies graph of f. Consider an optimal embedding of H which achieves spatial locality of d. In this embedding each connected component of size t lies in a grid of size at most $td \times td$, hence (by possibly rearranging disconnected components), we may assume that the whole embedding is placed on a $k \times k$ grid where $k \leq d(n + s)$. We define a CA over a $k \times k$ grid as follows: if there exists an output node v of H in the position (i, j) then set the local updating function of this cell to be the function that computes the v-th entry of f; otherwise, if this position is not occupied or contains an input node, set the corresponding local updating function to be the constant zero function. By definition, the radius of this CA is d. Also, it is not hard to verify that it computes the padded version f' of f in a single step.

A standard argument shows that the above padding preserves the security of OWFs. Hence, by Theorem 5.3, we have:

Corollary 6.2. *Suppose that the problem* DECODE$(6n, 1/4)$ *is intractable. Then there exist a CA which computes a OWF in a single step.*

It follows that, under the DECODE$(6n, 1/4)$ assumption, one cannot invert a single step of CA with non-negligible success probability with respect to a uniformly chosen initial configuration.

Remark 6.3. ***(Other primitives)*** *The security of all the primitives listed in Section 5.3 (commitments, identification, symmetric and public-key encryption schemes) is preserved under the above padding scheme. Hence, by Proposition 6.1 and Theorem 5.7, these primitives can be computed in a single step of CA.*

6.2 Pseudorandomness of CAs

Defining what it means for a CA to compute a PRG is a bit more delicate. Syntactically, we defined CAs to always compute functions of the form

[7] For the case of encryption, commitment, identification and trapdoor function, Theorem 5.7 yields semi-linear implementations only for the encryption function, the sender's computation, the prover's computation, and the evaluation algorithm respectively. Accordingly, spatial locality holds only with respect to the sender (i.e., the commitment, encryption and prover's functions), while the receiver's computation (i.e., the verification and decryption functions) is not computed with CSL. However, this limitation is inherent even if one aims only for constant output locality [30].

$g : \{0,1\}^N \to \{0,1\}^N$ while PRGs, by definition, extend their input. To meaningfully define the computation of PRGs by CAs we adopt the following convention. Suppose that a CA computes a function $g : \{0,1\}^N \to \{0,1\}^N$ which ignores some of its inputs, i.e, there exists a function $g' : \{0,1\}^{N-k} \to \{0,1\}^N$ for which $g(x,y) = g'(x)$ for any $x \in \{0,1\}^{N-k}$ and $y \in \{0,1\}^k$. Then, we may say that the CA computes g'.

Under this convention, a CA with N cells computes a PRG in a single step if the function $G : \{0,1\}^N \to \{0,1\}^N$ associated with the CA satisfies the following: (1) the distribution $G(U_N)$ is pseudorandom; and (2) there exists at least a single cell whose state never affects the state of any cell (including itself) in the next time step — that is, this cell can be treated as an "output" cell whose input is completely ignored. We remark that the existence of such a PRG implies that the prediction problem described in the abstract and in Section 1.2 can be hard. If we initialize the CA to a random configuration and observe the initial values of the "output" cells together with the values of all the cells (including the output cells) after a single step, then we get a pseudorandom sequence y of length larger than N.

In the following we show that such a PRG can be constructed from any spatially local PRG $G = \{G_n : \{0,1\}^n \to \{0,1\}^m\}$ that satisfies an additional property called *strong spatial locality*. A function G satisfies this property if the dependencies graph H_n of G_n can be embedded with CSL such that there exists an injection $\Gamma : [n] \to [m]$ which maps each input node v of H_n to an output node u. Moreover, the distance between v and u should be bounded by a constant (independent of n).

Proposition 6.4. *Let $G : \{0,1\}^n \to \{0,1\}^m$ be a PRG with strong spatial locality. Then, there exists a CA which computes a PRG in a single step.*

Proof. Let H be the dependencies graph of G. Consider the embedding of H which achieves spatial locality of d and strong spatial locality via the local injection Γ which maps each input to an output of distance at most e. As in Proposition 6.1, we may assume that the embedding is placed on a $k \times k$ grid for $k \leq d(n+s)$. We define a CA over a $k \times k$ grid as follows: if there exists an input node v of H in the position (i,j) then set the local updating function of this cell to be the function that computes the $\Gamma(v)$-th entry of G. (Call such a cell *regular*.) If there exists an output node u in the position (i,j) that is not in the image of Γ then set the local updating function of this cell to be the function that computes the u-th entry of G. (Call such a cell *output*.) Finally, if the position (i,j) is occupied with an output node which appears in the image of Γ, or if the position is empty, then let the (i,j) cell compute the identity function that sets the value of the cell to be the same value it had at the previous time step. (Call such a cell *pad*.)

It is not hard to verify that each cell depends on a neighborhood of distance at most $e+d$. We prove that the CA computes a PRG. Indeed, if we let x, y and z denote the content of regular, output and pad cells, respectively, then the resulting function $G'(x,y,z)$ can be written as $(G(x), z)$ which is pseudorandom according to our assumption (padding by z does not violate this). Hence, G' satisfies our definition.

We will show that the PRG constructed in Theorem 5.3 has strong spatial locality by relying on the following useful claim.

Claim 6.5. *Suppose we have $a(a+1)$ red tokens $\{r_{i,j}\}_{i\in[a+1],j\in[a]}$ and $a(a+1)$ yellow tokens $\{y_{i,j}\}_{i\in[a],j\in[a+1]}$ placed on the plane where $r_{i,j}$ (respectively, $y_{i,j}$) is placed in the position (i,j). Then, there is a bijection between red and yellow tokens such that each yellow token is mapped to a red token which lies in a neighboring position.*

Proof. Consider the main diagonal which begins at $(1,1)$ and ends at (a,a). A yellow token $y_{i,j}$ which is placed on the right side of the diagonal will be mapped to its left neighbor $r_{i,j-1}$. A yellow token $y_{i,j}$ which is placed on the diagonal or left to the diagonal will be mapped to the token $r_{i+1,j}$ which is located below him.

Theorem 6.6. *The PRG constructed in Theorem 5.3 has strong spatial locality.*

Proof. Let $\hat{G}$ be the PRG constructed in Theorem 5.3. We describe an injective mapping Γ from the inputs of $\hat{G}$ to its outputs such that the distance between an input node v and the output node $\Gamma(v)$ is constant. Recall that $\hat{G}$ is obtained by applying the encoding of Theorem 4.8 to the semi-linear function $G(A,x,C,\rho) = ((A,C), M(A,C)\cdot v(x,\rho))$ where v and M are as defined in the proof of Theorem 5.3.

Let us take a close look at the structure of $\hat{G}$. Recall that the encoding $\hat{G}$ allocates (at most) 3 inputs and 3 outputs to each entry of the matrix M. For simplicity, let us assume that each of these 6-tuples is placed on a single point on a "virtual" grid. (We can later scale up this grid to a standard grid.) Each (non-fixed) entry (i,j) of the matrix $M(A,C)$ appears both as an input node and as an output node at location (i,j). Naturally, we will match these entries to each

other. Also, we match each column entry $c_{i,j}$ of $\hat{G}$ to the random-input $w_{i,j}$, as $c_{i,j}$ and $w_{i,j}$ are located next to each other at the (virtual) point (i,j). For all columns of the form $\binom{\mathbf{e}_i}{\mathbf{0}_t}$, we also match the row entry $r_{i,j}$ to the random input $s_{i,j}$. Let us "remove" these columns of the grid.

We are left with an $(m+t) \times (n+2m-1)$ grid whose (i,j) point contains an input entry $s_{i,j}$ and an output entry $r_{i,j}$. We also have an additional column (the $n+2m$-th rightmost column) which contains only output entries $(r_{i,n+2m})_{i\in[m+t]}$ with no $s_{i,j}$'s, and an additional row (the topmost 0-th row) that contains only $n+2m$ inputs $x = (x_1, \ldots, n), \rho = (\rho_1, \ldots, \rho_{2m})$. Since $n+2m < m+t$ we can apply Claim 6.5 to the top $(n+2m) \times (n+2m)$ square, and get a short-distance mapping from all these inputs to the outputs. Finally, for the entries below the top square, we match each row entry $r_{i,j}$ to the random input $s_{i,j}$.

By combining Theorem 6.6 and Proposition 6.4 we derive the following corollary:

Corollary 6.7. *Suppose that the problem* DECODE$(6n, 1/4)$ *is intractable. Then there exists a CA which computes a PRG in a single step.*

7 Circuits with Constant Latency

A circuit of *physical latency* d is a standard Boolean circuit of bounded fan-in and bounded fan-out which can be embedded in the plane (e.g., in $\mathbb{N}^2$) such that the length of the longest directed path in the graph (measured with respect to the Euclidean metric) is bounded by d. We assume that gates have area and wires have width and therefore, in each location $(i,j) \in \mathbb{N}^2$ there exists at most a single gate and a constant number of wire crossovers. (A similar notion was studied in [33].) A circuit family $\left\{C_n : \{0,1\}^n \to \{0,1\}^{m(n)}\right\}_{n\in\mathbb{N}}$ has *constant physical latency* (CPL in short) if there exists a constant, d such that every C_n has latency d.

By simple manipulations (such as scaling up) one can show that a function can be computed by a CPL circuit if and only if it is spatially local.[8] Hence, the results of Section 5 implies that many cryptographic primitives can be implemented with CPL.

Theorem 7.1 (informal statement)**.** *Suppose that* DECODE$(1/8)$ *is intractable. Then there exist CPL implementations of: (1) one-way function; (2) pseudorandom generator; (3) semantically-secure symmetric encryption scheme; (4) non-interactive commitment scheme; and (5) one-round symmetric identification scheme. Moreover, under the McEliece assumption we can also get a CPL semantically-secure public-key encryption scheme and CPL trapdoor function.*[9]

It should be mentioned that, in CPL circuits, inputs and outputs do not have to be placed on the boundary of the circuit. This may be justified by the possibility of using the third dimension to "stack" circuits on top of each other.[10]

7.1 Discussion

Our main goal here is to explore the *theoretical* limits of parallelism achievable in cryptography. We do not attempt (nor do we feel qualified) to address the multitude of challenges and considerations that arise when implementing cryptographic hardware in the real world. Still, it is interesting to discuss some conceptual and practical questions that relate to the constant latency model.

For concreteness, let us focus on cryptographic primitives such as encryption or identification which involve two parties: a "sender", who performs the encryption or proof of identity, and a "receiver" who performs the decryption or identity verification. In this case, one may criticize our model by claiming that we only optimize the "internal latency" of the sender's device, while the overall latency is dominated by other parameters such as the "external" distance between the sender and the receiver, and the "internal latency" of the receiver. (Recall that our constant latency implementations apply only to the sender's side.)

We argue that despite these valid concerns, the notion of constant physical latency is still conceptually meaningful, at least in some scenarios. For instance,

[8] Spatial locality does not specify any restriction on the edge/wire layout. Nevertheless, edges can be laid out without violating the crossover limitation. For example, stretching the edges along the shortest paths ensures that in each point the number of crossovers is bounded by $(\pi \cdot d^2)^2$. (In this case, a node u is a member of an edge that crosses a point p only if u is d-close to p. Hence, since nodes are placed on integer coordinates, the number of potential nodes with an edge over p is bounded by the area of a d-radius circle, and the number of possible edges crossing p is at most quadratic in this number.)

[9] Again, in the cases of encryption, commitment and identification, we refer only to the efficiency of the "sender". See Footnote 7.

[10] This convention is essential for cryptographic constructions as, by Proposition 3.2, when inputs (or outputs) are placed on the boundary the resulting function can be easily inverted. However, in the case of randomized primitives (such as randomized encryption), whose input consists of deterministic input and random input, our constructions allow to position all deterministic inputs on the boundary, placing only output gates, logical AND/OR gates and randomness gates in the interior.

consider the case of an RFID tag authentication, or a smart card that communicates encrypted data to a card reader. In these cases, the physical distance between the sender's output gates and the receiver output gates is small and independent of the input length or the security parameter (imagine the parties as two large planar devices aligned in parallel to each other). Furthermore, in these cases the receiver is much more powerful than the sender. One can also imagine other meaningful scenarios in which the above concerns do not apply (e.g., secure devices whose physical medium requires information to travel very slowly, or cases in which the receiver's computation is applied only to a small sample of the messages computed by the sender). In such scenarios, the sender's latency may become more significant than (and can be measured independently of) the additional latency corresponding to the receiver and to the physical distance between the parties.

Taking a more pragmatic point of view, even in the typical case where the above best-case scenarios do not fully apply, the type of constructions we present might provide useful efficiency tradeoffs. For instance, our private-key identification scheme is based on the HB scheme [46] which was suggested as suitable for implementation on weak devices such as RFID tags. Comparing the concrete cost of our constant latency implementation to the original construction, our implementation achieves a much better physical latency at the expense of a moderate cost in hardware size and communication. Whether this type of tradeoffs can be useful in practice remains for further study.

Acknowledgment

We thank Sanjeev Arora for useful discussions. The first author was supported by NSF grants CNS-0627526, CCF-0426582 and CCF-0832797. The second and third authors were supported in part by ISF grant 1310/06, BSF grants 2004361, 2008411. The second author was also supported by NSF grants 0830803, 0716835, 0456717, 0627781.

References

[1] ER Berlekamp, JH Conway, and RK Guy. *Winning Ways for Your Mathematical Plays*. Academic Press, New York, 1983.

[2] John Von Neumann. *Theory of Self-Reproducing Automata*. University of Illinois Press, Champaign, IL, USA, 1966.

[3] Edward F. Moore, editor. *Sequential Machines: Selected Papers*. Addison-Wesley Longman Ltd., Essex, UK, UK, 1964. ISBN B0000CMBI0.

[4] Stephen Wolfram. Universality and complexity in cellular automata. *Physica D: Nonlinear Phenomena*, 10 (1-2):1–35, 1984.

[5] Gerard Y. Vichniac. Simulating physics with cellular automata. *Physica D: Nonlinear Phenomena*, 10(1-2):96–115, 1984.

[6] C. Bennett and C. Grinstein. Role of irreversibility in stabilizing complex and nonenergodic behavior in locally interacting discrete systems. *Physical Review Letters*, 55:657–660, 1985.

[7] N. Ganguly, B.K. Sikdar, A. Deutsch, G. Canright, and P.P. Chaudhuri. A survey on cellular automata. Technical Report 30, Centre for High Performance Computing, Dresden University of Technology, 2003. www.cs.unibo.it/bison/publications/CAsurvey.pdf.

[8] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.

[9] M. Blum and 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.

[10] A. C. Yao. Theory and application of trapdoor functions. In *Proc. 23rd FOCS*, pages 80–91, 1982.

[11] G. Y. Vichniac, P. Tamayo, and H. Hartman. Annealed and quenched inhomogeneous cellular automata (INCA). *Journal of Statistical Physics*, 45: 875–883, December 1986. doi: 10.1007/BF01020578.

[12] M. Sipper. Non-uniform cellular automata: Evolution in rule space and formation of complex structures. In R. A. Brooks P. Maes, editor, *Artificial Life IV*, pages 394–399. The MIT Press, 1994.

[13] Shang-Hua Teng. Functional inversion and communication complexity. *J. Cryptology*, 7(3):153–170, 1994.

[14] R. J. Lipton and R. E. Tarjan. Applications of a planar separator theorem. *SIAM J. Comput.*, 9(3): 615–627, 1980, August.

[15] Baker. Approximation algorithms for NP-complete problems on planar graphs. *JACM: Journal of the ACM*, 41, 1994.

[16] Stephen A. Cook. The complexity of theorem-proving procedures. In *STOC '71: Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158, New York, NY, USA, 1971. ACM Press. doi: http://doi.acm.org/10.1145/800157.805047.

[17] Stephen Wolfram. Random sequence generation by cellular automata. *Adv. Appl. Math.*, 7(2):123–169, 1986.

[18] P. Guan. Cellular automaton public-key cryptosystems. *Complex Systems*, 1, 1987.

[19] Toshiki Habutsu, Yoshifumi Nishio, Iwao Sasase, and Shinsaku Mori. A secret key cryptosystem by iterating a chaotic map. In *Proceedings of Eurocrypt '91*, number 127–140, 1991.

[20] W. Meier and O. Staffelbach. Analysis of pseudo random sequences generated by cellular automata. *Proceedings of Eurocrypt '91*, pages 186–199, 1991.

[21] Howard Gutowitz. Cryptography with dynamical systems. In N. Boccara, E. Goles, S. Martinez, and P. Picco, editors, *Cellular Automata and Cooperative Phenomena*, pages 237–274. Kluwer Academic Publishers, 1993.

[22] S. Nandi, B. K. Kar, and P. Pal Chaudhuri. Theory and applications of cellular automata in cryptography. *IEEE Trans. Comput.*, 43(12):1346–1357, 1994.

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

[24] S. Goldwasser and S. Micali. Probabilistic encryption. *JCSS*, 28(2):270–299, 1984. Preliminary version in Proc. STOC, 1982.

[25] R. Tao and S. Chen. Two varieties of finite automata public key cryptosystem and digital signature. *J. of Computer Science and Technology*, 1(1):9–18, 1986.

[26] J. Kari. Cryptosystems based on reversible cellular automata. *preprint*, April 1992.

[27] Feng Bao and Yoshihide Igarashi. A randomized algorithm to finite automata public key cryptosystem. In *ISAAC '94: Proceedings of the 5th International Symposium on Algorithms and Computation*, pages 678–686, London, UK, 1994. Springer-Verlag. ISBN 3-540-58325-4.

[28] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. On pseudorandom generators with linear stretch in NC^0. *Computional Complexity*, 17(1):38–69, 2008. ISSN 1016-3328. doi: http://dx.doi.org/10.1007/s00037-007-0237-6. Preliminary version in Proc. 10th Random, 2006.

[29] Nathan Linial, Yishay Mansour, and Noam Nisan. Constant depth circuits, Fourier transform, and learnability. *J. ACM*, 40(3):607–620, 1993. ISSN 0004-5411. doi: http://doi.acm.org/10.1145/174130.174138. Preliminary version in Proc. 30th FOCS, 1989.

[30] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography in NC^0. *SIAM J. Comput.*, 36 (4):845–888, 2006. doi: 10.1137/S0097539705446950. URL http://link.aip.org/link/?SMJ/36/845/1. Preliminary version in Proc. 45th FOCS, 2004.

[31] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography with constant input locality. *Journal of Cryptology*, 22(4):429–469, 2009. Preliminary version in CRYPTO '07.

[32] T. F. Leighton. *Complexity Issues in VLSI*. MIT Press, 1983.

[33] Bernard Chazelle and Louis Monier. A model of computation for VLSI with related complexity results. *J. ACM*, 32(3):573–588, 1985.

[34] Gianfranco Bilardi and Franco P. Preparata. Horizons of parallel computation. *Journal of Parallel and Distributed Computing*, 27:172–182, 1995.

[35] Oded Goldreich. Candidate one-way functions based on expander graphs. *Electronic Colloquium on Computational Complexity (ECCC)*, 7(090), 2000. URL citeseer.nj.nec.com/382413.html.

[36] Elchanan Mossel, Amir Shpilka, and Luca Trevisan. On ϵ-biased generators in NC^0. In *Proc. 44th FOCS*, pages 136–145, 2003.

[37] Yuval Ishai and Eyal Kushilevitz. Randomizing polynomials: A new representation with applications to round-efficient secure computation. In *Proc. 41st FOCS*, pages 294–304, 2000. URL citeseer.nj.nec.com/ishai00randomizing.html.

[38] David A. Mix Barrington, Neil Immerman, and Howard Straubing. On uniformity within ncź. *JCSS*, 41(3):274–306, 1990.

[39] Heribert Vollmer. *Introduction to Circuit Complexity: A Uniform Approach*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1999. ISBN 3540643109.

[40] Leonid A. Levin. Universal sequential search problems. *PINFTRANS: Problems of Information Transmission (translated from Problemy Peredachi Informatsii (Russian))*, 9, 1973.

[41] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Computationally private randomizing polynomials and their applications. *Computional Complexity*, 15(2):115–162, 2006. ISSN 1016-3328. doi: http://dx.doi.org/10.1007/s00037-006-0211-8. Preliminary version in Proc. 20th CCC, 2005.

[42] R.R. Varshamov. Estimate of the number of signals in error correcting codes. *Doklady Akademii Nauk SSSR*, 117:739–741, 1957.

[43] Madhu Sudan. Algorithmic introduction to coding theory - lecture notes, 2002. http://theory.csail.mit.edu/∼madhu/FT01/.

[44] Avrim Blum, Merrick Furst, Michael Kearns, and Richard J. Lipton. Cryptographic primitives based on hard learning problems. In *Advances in Cryptology: Proc. of CRYPTO '93*, volume 773 of *LNCS*, pages 278–291, 1994. URL citeseer.nj.nec.com/blum94cryptographic.html.

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

[46] Nicholas J. Hopper and Manuel Blum. Secure human identification protocols. In *Advances in Cryptology: Proc. of ASIACRYPT '01*, volume 2248 of *LNCS*, pages 52–66, 2001.

[47] Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *Journal of the ACM*, 50(4): 506–519, 2003. Preliminary version in Proc. 32nd STOC, 2000.

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

[49] A. Juels and S. Weis. Authenticating pervasive devices with human protocols. In *Advances in Cryptology: Proc. of CRYPTO '05*, volume 3621 of *LNCS*, pages 293–308, 2005.

[50] Vitaly Feldman, Parikshit Gopalan, Subhash Khot, and Ashok Kumar Ponnuswami. New results for learning noisy parities and halfspaces. In *Proc. 47th*

FOCS, pages 563–574, 2006.

[51] Elwyn R. Berlekamp, Robert J. McEliece, and Henk C. van Tilborg. On the inherent intractability of certain coding problems. *IEEE Transactions on Information Theory*, 24(3):384–386, 1978.

[52] Oded Goldreich, Hugo Krawczyk, and Michael Luby. On the existence of pseudorandom generators. *SIAM J. Comput.*, 22(6):1163–1175, 1993. ISSN 0097-5397. Preliminary version in Proc. 29th FOCS, 1988.

[53] Michael Kearns, Yishay Mansour, Dana Ron, Ronitt Rubinfeld, Robert E. Schapire, and Linda Sellie. On the learnability of discrete distributions. In *Proc. 26th STOC*, pages 273–282, 1994.

[54] Oded Goldreich. *Foundations of Cryptography: Basic Tools*. Cambridge University Press, 2001. ISBN 0521791723.

[55] J. Katz and J.-S. Shin. Parallel and concurrent security of the hb and hb+ protocols. In *Advances in Cryptology: Proc. of Eurocrypt 06'*, volume 4004 of *LNCS*, pages 73–87, 2006.

[56] Russell Impagliazzo, Leonid A. Levin, and Michael Luby. Pseudorandom generation from one-way functions. In *Proc. 21st STOC*, pages 12–24, 1989.

[57] Henri Gilbert, Matthew J. B. Robshaw, and Yannick Seurin. How to encrypt with the LPN problem. In *ICALP (2)*, pages 679–690, 2008.

[58] Daniel J. Bernstein, Tanja Lange, and Christiane Peters. Attacking and defending the mceliece cryptosystem. Cryptology ePrint Archive, Report 2008/318, 2008. http://eprint.iacr.org/.

[59] Heeralal Janwa and Oscar Moreno. Mceliece public key cryptosystems using algebraic-geometric codes. *Des. Codes Cryptography*, 8(3):293–307, 1996.

[60] Benny Applebaum. Fast cryptographic primitives based on the hardness of decoding random linear code. Technical Report 845, Princeton University, 2008. http://www.cs.princeton.edu/research/techreps/TR-845-08.

[61] Oded Goldreich. *Foundations of Cryptography: Basic Applications*. Cambridge University Press, 2004. ISBN 0521791723.

[62] O. Goldreich and L. Levin. A hard-core predicate for all one-way functions. In *Proc. 21st STOC*, pages 25–32, 1989.