Fine-Grained Cryptography

Citation: Degwekar, Akshay et al. "Fine-Grained Cryptography." Advances in Cryptology – CRYPTO 2016. Lecture Notes in Computer Science 9816 (2016): 533–562. © 2016 International Association for Cryptologic Research

As Published: http://dx.doi.org/10.1007/978-3-662-53015-3_19

Publisher: Springer

Persistent URL: http://hdl.handle.net/1721.1/111069

Version: Author's final manuscript: final author's manuscript post peer review, without publisher's formatting or copy editing

Fine-grained Cryptography*

Akshay Degwekar Vinod Vaikuntanathan Prashant Nalini Vasudevan

Abstract

Fine-grained cryptographic primitives are ones that are secure against adversaries with an a-priori bounded polynomial amount of resources (time, space or parallel-time), where the honest algorithms use less resources than the adversaries they are designed to fool. Such primitives were previously studied in the context of time-bounded adversaries (Merkle, CACM 1978), space-bounded adversaries (Cachin and Maurer, CRYPTO 1997) and parallel-time-bounded adversaries (Håstad, IPL 1987). Our goal is come up with fine-grained primitives (in the setting of parallel-time-bounded adversaries) and to show unconditional security of these constructions when possible, or base security on widely believed separation of worst-case complexity classes. We show:

1. NC^1-cryptography: Under the assumption that $\mathsf{NC}^1 \neq \oplus\mathsf{L}/\mathsf{poly}$, we construct one-way functions, pseudo-random generators (with sub-linear stretch), collision-resistant hash functions and most importantly, *public-key encryption schemes*, all computable in NC^1 and secure against all NC^1 circuits. Our results rely heavily on the notion of randomized encodings pioneered by Applebaum, Ishai and Kushilevitz, and crucially, make *non-black-box* use of randomized encodings for logspace classes.

2. AC^0-cryptography: We construct (unconditionally secure) pseudo-random generators with arbitrary polynomial stretch, weak pseudo-random functions, secret-key encryption and perhaps most interestingly, *collision-resistant hash functions*, computable in AC^0 and secure against all AC^0 circuits. Previously, one-way permutations and pseudo-random generators (with linear stretch) computable in AC^0 and secure against AC^0 circuits were known from the works of Håstad and Braverman.

*MIT. E-mail: {akshayd,vinodv,prashvas}@mit.edu. Research supported in part by NSF Grants CNS-1350619 and CNS-1414119, Alfred P. Sloan Research Fellowship, Microsoft Faculty Fellowship, the NEC Corporation, a Steven and Renee Finn Career Development Chair from MIT. This work was also sponsored in part by the Defense Advanced Research Projects Agency (DARPA) and the U.S. Army Research Office under contracts W911NF-15-C-0226.

Contents

1 Introduction

The last four decades of research in the theory of cryptography has produced a host of fantastic notions, from public-key encryption [DH76, RSA78, GM82] and zero-knowledge proofs [GMR85] in the 1980s, to fully homomorphic encryption [RAD78, Gen09, BV11] and program obfuscation [BGI+01, GGH+13, SW14] in the modern day. Complexity theory is at the heart of these developments, playing a key role in coming up with precise mathematical definitions as well as constructions whose security can be reduced to precisely stated computational hardness assumptions.

However, the uncomfortable fact remains that a vast majority of cryptographic constructions rely on *unproven assumptions*. At the very least, one requires that $\mathsf{NP} \not\subseteq \mathsf{BPP}$ [IL89], but that is hardly ever enough — when designing advanced cryptographic objects, cryptographers assume the existence of one-way functions as a given, move up a notch to assuming the hardness of specific problems such as factoring, discrete logarithms and the approximate shortest vector problem for lattices, and, more recently, even more exotic assumptions. While there are some generic transformations between primitives, such as from one-way functions to pseudo-random generators and symmetric encryption (e.g., [HILL99]), there are large gaps in our understanding of relationships between most others. In particular, it is a wide open question whether $\mathsf{NP} \not\subseteq \mathsf{BPP}$ suffices to construct even the most basic cryptographic objects such as one-way functions, or whether it is possible to construct public-key encryption assuming only the existence of one-way functions (for some partial negative results, see [BT03, AGGM06, BB15, IR88]).

In this work, we ask if a weaker version of these cryptographic primitives can be constructed, with security against a *bounded* class of adversaries, based on either *mild complexity-theoretic assumptions* or *no assumptions* at all. Indeed, this question has been asked by several researchers in the past.

1. Merkle [Mer78] constructed a non-interactive key exchange protocol (and thus, a public-key encryption scheme) where the honest parties run in linear time $O(n)$ and security is shown against adversaries that run in time $o(n^2)$. His assumption was the existence of a random function that both the honest parties and the adversary can access (essentially, the random oracle model [BR93]). Later, the assumption was improved to exponentially strong one-way functions [BGI08]. This work is timeless, not only because it jump-started public-key cryptography, but also because it showed how to obtain a primitive with much structure (trapdoors) from one that apparently has none (namely, random oracles and exponentially strong one-way functions).

2. Maurer [Mau92] introduced the bounded storage model, which considers adversaries that have an a-priori bounded amount of space but unbounded computation time. Cachin and Maurer constructed symmetric-key encryption and key-exchange protocols that are *unconditionally secure* in this model [CM97] assuming that the honest parties have storage $O(s)$ and the adversary has storage $o(s^2)$ for some parameter s. There has been a rich line of work on this model [CM97, AR99, DM04] following [Mau92].

3. Implicit in the work of Håstad [Has87] is a beautiful construction of a one-way permutation that can be computed in NC^0 (constant-depth circuits with AND and OR gates of unbounded fan-in and NOT gates), but inverting which is hard for any AC^0 circuit. Here is the function:

$$f(x_1, x_2, \ldots, x_n) = \big(x_1, x_1 \oplus x_2, x_2 \oplus x_3, \ldots, x_{n-1} \oplus x_n\big)$$

Clearly, each output bit of this function depends on at most two input bits. Inverting the function implies in particular the ability to compute x_n, which is the parity of all the output bits. This is hard for AC^0 circuits as per [FSS84, Ajt83, Hås86].

All these works share two common features. First, security is achieved against a class of adversaries with bounded resources (time, space and parallel time, respectively, in the three works above). Secondly, the honest algorithms use less resources than the class of adversaries they are trying to fool. We propose to call the broad study of such cryptographic constructions *fine-grained cryptography*, and construct several fine-grained cryptographic primitives secure against parallel-time-bounded adversaries.

We study two classes of low-depth circuits (as adversaries). The first is AC^0, which is the class of functions computable by *constant-depth* polynomial-sized circuits consisting of AND, OR, and NOT gates of *unbounded fan-in*, and the second is NC^1, the class of functions computable by *logarithmic-depth* polynomial-sized circuits consisting of AND, OR, and NOT gates of *fan-in 2*. In both cases, we mean the non-uniform versions of these classes. Note that this also covers the case of adversaries that are randomized circuits with these respective restrictions. This is because for any such randomized adversary $\mathcal{A}$ there is a non-uniform adversary $\mathcal{A}'$ that performs as well as $\mathcal{A}$ – $\mathcal{A}'$ is simply $\mathcal{A}$ hard-coded with the randomness that worked best for it.

Early developments in circuit lower bounds [FSS84, Ajt83, Hås86] showed progressively better and *average-case* and *exponential* lower bounds for the PARITY function against AC^0 circuits. This has recently been sharpened to an average-case depth hierarchy theorem [RST15]. We already saw how these lower bounds translate to meaningful cryptography, namely one-way permutations against AC^0 adversaries. Extending this a little further, a reader familiar with Braverman's breakthrough result [Bra10] (regarding the pseudorandomness of n^ϵ-wise independent distributions against AC^0) will notice that his result can be used to construct large-stretch pseudo-random generators that are computable by fixed-depth AC^0 circuits and are pseudo-random against arbitrary constant-depth AC^0 circuits. Can we do more? *Can we construct secret-key encryption, collision-resistant hash functions, and even trapdoor functions, starting from known lower bounds against* AC^0 *circuits*? Our first contribution is a positive answer to some of these questions.

Our second contribution is to study adversaries that live in NC^1. In this setting, as we do not know any lower bounds against NC^1, we are forced to rely on an unproven complexity-theoretic assumption; however, we aim to limit this to a worst-case, widely believed, separation of complexity classes. Here, we construct several cryptographic primitives from the *worst-case* hardness assumption that $\oplus\mathsf{L}/\mathsf{poly} \not\subseteq \mathsf{NC}^1$, the most notable being an additively-homomorphic public-key encryption scheme where the key generation, encryption and decryption algorithms are all computable in $\mathsf{AC}^0[2]$ (constant-depth circuits with MOD2 gates; note that $\mathsf{AC}^0[2] \subsetneq \mathsf{NC}^1$ [Raz87, Smo87]), and the scheme is semantically secure against NC^1 adversaries. ($\oplus\mathsf{L}/\mathsf{poly}$ can be thought of as the class of languages with polynomial-sized branching programs. Note that by Barrington's Theorem [Bar86], all languages in NC^1 have polynomial-sized branching programs of constant width.)

Apart from theoretical interest stemming from the fact that these are rather natural objects, one possible application of such constructions (that was suggested to us independently by Ron Rothblum and Yuval Ishai) is in using them in conjunction with other constructions that are secure against polynomial-time adversaries under stronger assumptions. This could be done to get hybrids that are secure against polynomial-time adversaries under these stronger assumptions while also being secure against bounded adversaries unconditionally (or under minimal assumptions). For instance, consider an encryption scheme where the message is first encrypted using the AC^0-encryption scheme

from Section 5.3, and the resultant ciphertext is then encrypted using a scheme that works in AC^0 and is secure against polynomial-time adversaries under some standard assumptions (see [AIK04] for such schemes). This resultant scheme can be shown to be secure against polynomial-time adversaries under the same assumptions while being unconditionally secure against AC^0 adversaries.

We now briefly describe the relation between our results and the related work on randomized encodings [IK00, AIK04], and move on to describing the results in detail.

Relation to Randomized Encodings and Cryptography in NC^0. Randomized encodings of Ishai and Kushilevitz [IK00, AIK04] are a key tool in our results against NC^1 adversaries. Using randomized encodings, Applebaum, Ishai and Kushilevitz [AIK04] showed how to convert several cryptographic primitives computable in logspace classes into ones that are computable in NC^0. The difference between their work and ours is two-fold: (1) Their starting points are cryptographic schemes secure against arbitrary polynomial-time adversaries, which rely on average-case hardness assumptions, whereas in our work, the focus is on achieving security either with no unproven assumptions or only worst-case assumptions; of course, our schemes are not secure against polynomial-time adversaries, but rather, limited adversarial classes; (2) In the case of public-key encryption, they manage to construct key generation and encryption algorithms that run in NC^0, but the decryption algorithm retains its high complexity. In contrast, in this work, we can construct public key encryption (against NC^1 adversaries) where the encryption algorithm can be computed in NC^0 and the key generation and decryption in $\mathsf{AC}^0[2]$.

A Remark on Cryptographic vs. Non-Cryptographic Constructions An important *desideratum* for us is that the (honest) algorithms in our constructions can be implemented with fewer resources than the adversary that they are trying to fool. We call such constructions *cryptographic* in contrast to *non-cryptographic* constructions where this is not necessarily the case. Perhaps the clearest and the most well-known example of this distinction is the case of pseudo-random generators (PRGs) [BM84, Yao82, NW94]. Cryptographic PRGs, pioneered in the works of Blum, Micali and Yao [BM84, Yao82] are functions computable in a *fixed* polynomial time that produce outputs that are indistinguishable from random against *any* polynomial-time machine. The designer of the PRG does not know the precise power of the adversary: he knows that the adversary is polynomial-time, but not *which* polynomial. On the other hand, non-cryptographic ("Nisan-Wigderson type") PRGs [NW94] take more time to compute than the adversaries they are designed to fool.

Our constructions will be exclusively in the *cryptographic* regime. For example, our one-way functions, pseudo-random generators and collision-resistant hash functions against AC^0 are computable by circuits of fixed polynomial size $q(\lambda)$ and fixed (constant) depth d, and maintain security (in the appropriate sense) against adversarial circuits of size $p'(\lambda)$ and depth d' for any polynomial function p' and any constant d'.

1.1 Our Results and Techniques

Our results are grouped into two classes — primitives secure against NC^1 circuits based on minimal worst-case assumptions, and those that are unconditionally secure against AC^0 circuits. In the description below and throughout the rest of the paper, all algebra is over $\mathbb{F}_2$.

1.1.1 Constructions against NC^1 Adversaries

We construct one-way functions (OWFs), pseudo-random generators (PRGs), additively homomorphic public-key encryption (PKE), and collision-resistant hash functions (CRHFs) that are computable in NC^1 and are secure against NC^1 adversaries, based on the worst-case assumption that $\oplus\mathsf{L}/\mathsf{poly} \not\subseteq \mathsf{NC}^1$. An important tool we use for these constructions is the notion of *randomized encodings* of functions introduced in [IK00].

A randomized encoding of a function f is a randomized function $\hat{f}$ that is such that for any input x, the distribution of $\hat{f}(x)$ reveals $f(x)$, but nothing more about x. We know through the work of [IK00, AIK04] that there are randomized encodings for the class $\oplus\mathsf{L}/\mathsf{poly}$ that can be computed in (randomized, uniform) NC^0. Randomized encodings naturally offer a flavor of worst-to-average case reductions as they reduce the problem of evaluating a function on a given input to deciding some property of the distribution of its encoding. Our starting point is the observation, implicit in [AIK04, AR15], that they can be used to generically construct infinitely-often one-way functions and pseudo-random generators with additive stretch, computable in NC^0 and secure against NC^1 adversaries (assuming, again, that $\oplus\mathsf{L}/\mathsf{poly} \not\subseteq \mathsf{NC}^1$). We start with the following general theorem.

Theorem 1.1 (Informal)**.** *Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be two classes such that $\mathcal{C}_2 \not\subseteq \mathcal{C}_1$ and $\mathcal{C}_2$ has perfect randomized encodings computable in $\mathcal{C}_1$. Then, there are OWFs and PRGs that are computable in $\mathcal{C}_1$ and are secure against arbitrary adversarial functions in $\mathcal{C}_1$.*

Informally, the argument for Theorem 1.1 is the following: Let L be the language in $\mathcal{C}_2$ but not $\mathcal{C}_1$. The PRG is a function that takes an input r and outputs the randomized encoding of the indicator function for membership in L on the input 0^λ, using r as the randomness (where λ is a security parameter). Any adversary that can distinguish the output of this function from random can be used to decide if a given x is in the language L by computing the randomized encoding of x and feeding it to the adversary. This gives us a PRG with a non-zero additive stretch (and also a OWF) if the randomized encoding has certain properties (they need to be *perfect*) — see Section 3 for details.

While we have one way functions and pseudorandom generators, a black-box construction of public key cryptosystems from randomized encodings seems elusive. Our first contribution in this work is to use the algebraic structure of the randomized encodings for $\oplus\mathsf{L}/\mathsf{poly}$ to construct an additively homomorphic public key encryption scheme secure against NC^1 circuits (assuming that $\oplus\mathsf{L}/\mathsf{poly} \not\subseteq \mathsf{NC}^1$).

Additively Homomorphic Public-Key Encryption. The key attribute of the randomized encodings of [IK00, AIK04] for $\oplus\mathsf{L}/\mathsf{poly}$ is that the encoding is not a structureless string. Rather, the randomized encodings of computations are matrices whose rank corresponds to the result of the computation. Our public-key encryption construction uses two observations:

- Under the assumption $\oplus\mathsf{L}/\mathsf{poly} \not\subseteq \mathsf{NC}^1$, there exist, for an infinite number of values of n, distributions D_0^n and D_1^n over $n \times n$ matrices of rank $(n-1)$ and n, respectively, that are indistinguishable to NC^1 circuits.
- It is possible to sample a matrix $\mathbf{M}$ from D_0^n along with the non-zero vector $\mathbf{k}$ in its kernel. The sampling can be accomplished in NC^1 or even $\mathsf{AC}^0[2]$.

The public key in our scheme is such a matrix $\mathbf{M}$, and the secret key is the corresponding $\mathbf{k}$. Encryption of a bit b is a vector $\mathbf{r}^T\mathbf{M} + b\mathbf{t}^T$, where $\mathbf{r}$ is a random vector[1] and $\mathbf{t}$ is a vector such that $\langle \mathbf{t}, \mathbf{k} \rangle = 1$. In effect, the encryption of 0 is a random vector in the row-span of $\mathbf{M}$ while the encryption of 1 is a random vector outside. Decryption of a ciphertext $\mathbf{c}$ is simply the inner product $\langle \mathbf{c}, \mathbf{k} \rangle$. Semantic security against NC^1 adversaries follows from the fact that D_0^n and D_1^n are indistinguishable to NC^1 circuits. In particular, (1) We can indistinguishably replace the public key by a random full rank matrix $\mathbf{M}'$ chosen from D_n^1; and (2) with $\mathbf{M}'$ as the public key, encryptions of 0 are identically distributed to the encryptions of 1. The following is an informal restatement of Theorem 4.1.

Theorem 1.2 (Informal)**.** *If $\oplus\mathsf{L}/\mathsf{poly} \neq \mathsf{NC}^1$, there is a public-key encryption scheme secure against NC^1 adversaries where key generation, encryption and decryption are all computable in (randomized) $\mathsf{AC}^0[2]$.*

The scheme above is additively homomorphic, and thus, Collision-Resistant Hash Functions (CRHF) against NC^1 follow immediately from the known generic transformations [IKO05] which work in NC^1.

Theorem 1.3 (Informal)**.** *If $\oplus\mathsf{L}/\mathsf{poly} \neq \mathsf{NC}^1$, there is a family of collision-resistant hash functions that is secure against NC^1 adversaries where both sampling hash functions and evaluating them can be performed in (randomized) $\mathsf{AC}^0[2]$.*

We remark that in a recent work, Applebaum and Raykov [AR15] construct CRHFs against polynomial-time adversaries under the assumption that there are average-case hard functions with perfect randomized encodings. Their techniques also carry over to our setting and imply, for instance, the existence of CRHFs against NC^1 under the assumption that there is a language that is average-case hard for NC^1 that has perfect randomized encodings that can be computed in NC^1. This does not require any additional structure on the encodings apart from perfectness, but does require average-case hardness in place of our worst-case assumptions.

1.1.2 Constructions against AC^0 Adversaries

We construct one-way functions (OWFs), pseudo-random generators (PRGs), weak pseudo-random functions (weak PRFs), symmetric-key encryption (SKE) and collision-resistant hash functions (CRHFs) that are computable in AC^0 and are unconditionally secure against arbitrary AC^0 circuits. While some constructions for OWFs and PRGs against AC^0 were already known [Hås86, Bra10], and the existence of weak PRFs and SKE, being minicrypt primitives, is not that surprising, the possibility of unconditionally secure CRHFs against AC^0 is somewhat surprising, and we consider this to be our primary contribution in this section. We also present a candidate construction for public-key encryption, but we are unable to prove its unconditional security against AC^0 circuits.

As we saw earlier, Håstad [Has87] constructed one-way permutations secure against AC^0 circuits based on the hardness of computing PARITY. When allowed polynomial running time, we have black-box constructions of pseudorandom generators [HILL99] and pseudorandom functions [GGM86] from one-way functions. But because these reductions are not implementable in AC^0, getting primitives computable in AC^0 requires more effort.

[1] We maintain the convention that all vectors are by default column vectors. For a vector $\mathbf{r}$, the notation $\mathbf{r}^T$ denotes the row vector that is the transpose of $\mathbf{r}$.

Our constructions against AC^0 adversaries are primarily based on the theorem of Braverman [Bra10] (and its recent sharpening by Tal [Tal14]) regarding the pseudo-randomness of polylog-wise independent distributions against constant depth circuits. We use this to show that AC^0 circuits cannot distinguish between the distribution $(\mathbf{A}, \mathbf{Ak})$, where $\mathbf{A}$ is a random "sparse" matrix of dimension $\mathsf{poly}(n) \times n$ and $\mathbf{k}$ is a uniformly random secret vector, from the distribution $(\mathbf{A}, \mathbf{r})$, where $\mathbf{r}$ is a uniformly random vector. Sparse here means that each row of $\mathbf{A}$ has at most $\mathsf{polylog}(n)$ many ones.

(This is shown as follows. It turns out that with high probability, a matrix chosen in this manner is such that any set of $\mathsf{polylog}(n)$ rows is linearly independent (Lemma 2.9). Note that when a set of rows of $\mathbf{A}$ is linearly independent, the corresponding set of bits in $\mathbf{Ak}$ are uniformly distributed. This implies that if all $\mathsf{polylog}(n)$-sized sets of rows of $\mathbf{A}$ are linearly independent, then $\mathbf{Ak}$ is $\mathsf{polylog}(n)$-wise independent. This fact, along with the theorems regarding pseudo-randomness mentioned above prove the indistinguishability by AC^0.)

We also crucially use the fact, from [AB84], that the inner product of an arbitrary vector with a sparse vector can be computed in constant depth.

OWFs and PRGs This enables us to construct PRGs in NC^0 with constant multiplicative stretch and in AC^0 with polynomial multiplicative stretch. The construction is to fix a sparse matrix $\mathbf{A}$ with the linear independence properties mentioned above, and the PRG output on seed $\mathbf{k}$ is $\mathbf{Ak}$. Pseudo-randomness follows from the indistinguishability arguments above. This is stated in the following informal restatement of Theorem 5.1(along with Remark 5.1). We need to show that there exist such matrices $\mathbf{A}$ in which any $\mathsf{polylog}$-sized set of rows are linearly independent, and yet are sparse. As we show in Section 2.3, there are indeed matrices that have these properties.

Theorem 1.4 (Informal)**.** *For any constant c, there is a family of circuits $\left\{C_n : \{0,1\}^n \to \{0,1\}^{n^c}\right\}$ such that for any n, each output bit of C_n depends on at most $O(c)$ input bits. Further, for large enough n, AC^0 circuits cannot distinguish the output distribution $C_n(U_n)$ from U_{n^c}.*

We note that similar techniques have been used in the past to construct PRGs that fool circuit families of a fixed constant depth - see, for instance, [Vio12].

Weak PRFs against AC^0. A Pseudo-Random Function family (PRF) is a collection of functions such that a function chosen at random from this collection is indistinguishable from a function chosen at random from the set of all functions (with the appropriate domain and range), based on just a polynomial number of evaluations of the respective functions. In a *Strong* PRF, the distinguisher is allowed to specify (even adaptively) the input points at which it wants the function to be evaluated. In a *Weak* PRF, the distinguisher is given function evaluations at input points chosen uniformly at random.

We construct Weak PRFs against AC^0 that are unconditionally secure. In our construction, each function in the family is described by a vector $\mathbf{k}$. The computation of the pseudo-random function proceeds by mapping its input $\mathbf{x}$ to a sparse vector $\mathbf{a}$ and computing the inner product $\langle \mathbf{a}, \mathbf{k} \rangle$ over $\mathbb{F}_2$. Given polynomially many samples of the form $(\mathbf{a}, \langle \mathbf{a}, \mathbf{k} \rangle)$, one can write this as $(\mathbf{A}, \mathbf{Ak})$, where $\mathbf{A}$ is a matrix with random sparse rows. Our mapping of $\mathbf{x}$'s to $\mathbf{a}$'s is such that $(\mathbf{A}, \mathbf{Ak})$ is in some sense the only useful information contained in a set of random function evaluations. This is now indistinguishable from $(\mathbf{A}, \mathbf{r})$ where $\mathbf{r}$ is uniformly random, via the arguments mentioned earlier in this section. The following is an informal restatement of Theorem 5.2.

Theorem 1.5 (Informal)**.** *There is a Weak Pseudo-Random Function family secure against* AC^0 *adversaries and is such that both sampling a function at random and evaluating it can be performed in* AC^0*.*

We note that while our construction only gives us quasi-polynomial security (that is, an adversary might be able to achieve an inverse quasi-polynomial advantage in telling our functions from random) as opposed to exponential security, we show that this is an inherent limitation of weak PRFs computable in AC^0. Roughly speaking, due to the work of [LMN93], we know that a constant fraction of the Fourier mass of any function on n inputs computable in AC^0 is concentrated on Fourier coefficients upto some $\mathsf{polylog}(n)$. So there is at least one coefficient in the case of such a function that is at least $\Omega\left(\frac{1}{2^{\mathsf{polylog}(n)}}\right)$ in absolute value, whereas in a random function any coefficient would be exponentially small. So, by guessing and estimating a Fourier coefficient of degree at most $\mathsf{polylog}(n)$ (which can be done in AC^0), one can distinguish functions computed in AC^0 from a random function with some $\Omega\left(\frac{1}{2^{\mathsf{polylog}(n)}}\right)$ advantage. See Observation 5.4 for more details.

Symmetric Key Encryption against AC^0**.** In the case of polynomial-time adversaries and constructions, weak PRFs generically yield symmetric key encryption schemes, and this continues to hold in our setting. However, we present an alternative construction that has certain properties that make it useful in the construction of collision-resistant hash functions later on. The key in our scheme is again a random vector $\mathbf{k}$. The encryption of a bit b is a random sparse vector $\mathbf{c}$ such that $\langle \mathbf{c}, \mathbf{k} \rangle = b$ over $\mathbb{F}_2$. (Similar schemes, albeit without the sparsity, have been employed in the past in the leakage-resilience literature — see [GR12] and references therein.)

Encryption is performed by rejection sampling to find such a $\mathbf{c}$, and decryption is performed by computing $\langle \mathbf{c}, \mathbf{k} \rangle$, which can be done in constant depth owing to the sparsity of $\mathbf{c}$. We reduce the semantic security of this construction to the indistinguishability of the distributions $(\mathbf{A}, \mathbf{Ak})$ and $(\mathbf{A}, \mathbf{r})$ mentioned earlier. Note that this scheme is additively homomorphic, a property that will be useful later. The following is an informal restatement of Theorem 5.5.

Theorem 1.6 (Informal)**.** *There is a Symmetric Key Encryption scheme that is secure against* AC^0 *adversaries and is such that key generation, encryption and decryption are all computable in (randomized)* AC^0*.*

Collision Resistance against AC^0**.** Our most important construction against AC^0, which is what our encryption scheme was designed for, is that of Collision Resistant Hash Functions. Note that while there are generic transformations from additively homomorphic encryption schemes to CRHFs ([IKO05]), these transformations do not work in AC^0 and hence do not yield the construction we desire.

Our hash functions are described by matrices where each column is the encryption of a random bit under the above symmetric encryption scheme. Given such a matrix $\mathbf{M}$ that consists of encryptions of the bits of a string m, the evaluation of the function on input $\mathbf{x}$ is $\mathbf{Mx}$. Note that we wish to perform this computation in constant depth, and this turns out to be possible to do correctly for most keys because of the sparsity of our ciphertexts.

Finding a collision for a given key $\mathbf{M}$ is equivalent to finding a vector $\mathbf{u}$ such that $\mathbf{Mu} = 0$. By the additive homomorphism of the encryption scheme, and the fact that $\mathbf{0}$ is a valid encryption

of 0, this implies that $\langle m, \mathbf{u} \rangle = 0$. But this is non-trivial information about m, and so should violate semantic security. However showing that this is indeed the case turns out to be somewhat non-trivial.

In order to do so, given an AC^0 adversary A that finds collisions for the hash function with some non-negligible probability, we will need to construct another AC^0 adversary, B, that breaks semantic security of the encryption scheme. The most straightforward attempt at this would be as follows. B selects messages $\mathbf{m}_0$ and $\mathbf{m}_1$ at random and sends them to the challenger who responds with $\mathbf{M} = \mathsf{Enc}(\mathbf{m}_b)$. B then forwards this to A who would then return, with non-negligible probability, a vector $\mathbf{u}$ such that $\langle \mathbf{u}, \mathbf{m}_b \rangle = 0$. If B could compute $\langle \mathbf{u}, \mathbf{m}_0 \rangle$ and $\langle \mathbf{u}, \mathbf{m}_1 \rangle$, B would then be able to guess b correctly with non-negligible advantage. The problem with this approach is that $\mathbf{u}$, $\mathbf{m}_0$ and $\mathbf{m}_1$ might all be of high Hamming weight, and this being the case, B would not be able to compute the above inner products.

The solution to this is to choose $\mathbf{m}_0$ to be a sparse vector and $\mathbf{m}_1$ to be a random vector and repeat the same procedure. This way, B can compute $\langle \mathbf{u}, \mathbf{m}_0 \rangle$, and while it still cannot check whether $\langle \mathbf{u}, \mathbf{m}_1 \rangle = 0$, it can instead check whether $\mathbf{Mu} = \mathbf{0}$ and use this information. If it turns out that $\mathbf{Mu} = \mathbf{0}$ and $\langle \mathbf{u}, \mathbf{m}_0 \rangle \neq 0$, then B knows that $b = 1$, due to the additive homomorphism of the encryption scheme. Also, when $b = 1$, since $\mathbf{m}_0$ is independent of $\mathbf{m}_1$, the probability that A outputs $\mathbf{u}$ such that $\langle \mathbf{u}, \mathbf{m}_0 \rangle \neq 0$ is non-negligible. Hence, by guessing $b = 1$ when this happens and by guessing b at random otherwise, B can gain non-negligible advantage against semantic security. This achieves the desired contradiction and demonstrates the collision resistance of our construction. The following is an informal restatement of Theorem 5.11.

Theorem 1.7 (Informal)**.** *There is a family of Collision Resistant Hash Functions that is secure against* AC^0 *adversaries and is such that both sampling a hash function at random and evaluating it can be performed in (randomized)* AC^0*.*

Candidate Public Key Encryption against AC^0 We also propose a candidate Public Key Encryption scheme whose security we cannot show. It is similar to the LPN-based cryptosystem in [Ale03]. The public key is a matrix of the form $\mathbf{M} = (\mathbf{A}, \mathbf{Ak})$ where $\mathbf{A}$ is a random $n \times n$ matrix and $\mathbf{k}$, which is also the secret key, is a random sparse vector of length n. To encrypt 0, we choose a random sparse vector $\mathbf{r}$ and output $\mathbf{c}^T = \mathbf{r}^T\mathbf{M}$, and to encrypt 1 we just output a random vector $\mathbf{c}^T$ of length $(n + 1)$. Decryption is simply the inner product of $\mathbf{c}$ and the vector $(\mathbf{k}\ 1)^T$, and can be done in AC^0 because $\mathbf{k}$ is sparse.

1.2 Other Related Work: Cryptography against Bounded Adversaries

The big bang of public-key cryptography was the result of Merkle [Mer78] who constructed a key exchange protocol where the honest parties run in linear time $O(n)$ and security is obtained against adversaries that run in time $o(n^2)$. His assumption was the existence of a random function that both the honest parties and the adversary can access. Later, the assumption was improved to strong one-way functions [BGI08]. This is, indeed, a fine-grained cryptographic protocol in our sense.

Ajtai and Wigderson [AW85] construct PRGs secure against AC^0 and computable in AC^0 with arbitrary polynomial stretch. The construction in Section 5.1 achieves the same stretch and is much simpler owing to improvements in the understanding of pseudo-randomness against constant-depth circuits over the past 30 years.

The study of ϵ-biased generators [AGHP93, MST06] is related to this work. In particular, ϵ-biased generators with exponentially small ϵ give us almost k-wise independent generators for large k, which in turn fool AC^0 circuits by a result of Braverman [Bra10]. This and other techniques have been used in the past to construct PRGs that fool circuits of a fixed constant depth, with the focus generally being on optimising the seed length [Vio12, TX13].

The notion of precise cryptography introduced by Micali and Pass [MP06] studies reductions between cryptographic primitives that can be computed in linear time. That is, they show constructions of primitive B from primitive A such that if there is a $\mathsf{TIME}(f(n))$ algorithm that breaks primitive B, there is a $\mathsf{TIME}(O(f(n)))$ algorithm that breaks A.

Maurer [Mau92] introduced the bounded storage model, which considers adversaries that have a bounded amount of space and unbounded computation time. There are many results known here [DM04, Vad04, AR99, CM97] and in particular, it is possible to construct Symmetric Key Encryption and Key Agreement protocols unconditionally in this model[CM97].

2 Preliminaries

In this section we establish notation that shall be used throughout the rest of the presentation and recall the notion of randomized encodings of functions. We state and prove some results about certain kinds of random matrices that turn out to be useful in Section 5. In Sections 2.4 and 2.5, we present formal definitions of a general notion of adversaries with restricted computational power and also of several standard cryptographic primitives against such restricted adversaries (as opposed to the usual definitions, which are specific to probabilistic polynomial time adversaries).

2.1 Notation

For a distribution D, by $x \leftarrow D$ we denote x being sampled according to D. Abusing notation, we denote by $D(x)$ the probability mass of D on the element x. For a set S, by $x \leftarrow S$, we denote x being sampled uniformly from S. We also denote the uniform distribution over S by U_S, and the uniform distribution over $\{0,1\}^\lambda$ by U_λ. We use the notion of total variational distance between distributions, given by:

$$\Delta(D_1, D_2) = \frac{1}{2} \sum_x |D_1(x) - D_2(x)|$$

For distributions D_1 and D_2 over the same domain, by $D_1 \equiv D_2$ we mean that the distributions are the same, and by $D_1 \approx D_2$, we mean that $\Delta(D_1, D_2)$ is a negligible function of some parameter that will be clear from the context. Abusing notation, we also sometimes use random variables instead of their distributions in the above expressions.

For any $n \in \mathbb{N}$, we denote by $\lfloor n \rfloor_2$ the greatest power of 2 that is not more than n. For any n, k, and $d \leq k$, we denote by $SpR_{k,d}$ the uniform distribution over the set of vectors in $\{0,1\}^k$ with exactly d non-zero entries, and by $SpM_{n,k,d}$ the distribution over the set of matrices in $\{0,1\}^{n \times k}$ where each row is distributed independently according to $SpR_{k,d}$.

We identify strings in $\{0,1\}^n$ with vectors in $\mathbb{F}_2^n$ in the natural manner. For a string (vector) x, $\|x\|$ denotes its Hamming weight. Finally, we note that all arithmetic computations (such as inner products, matrix products, etc.) in this work will be over $\mathbb{F}_2$, unless specified otherwise.

2.2 Constant-Depth Circuits

Here we state a few known results on the computational power of constant depth circuits that shall be useful in our constructions against AC^0 adversaries.

Theorem 2.1 (Hardness of Parity, [Hås14])**.** *For any circuit C with n inputs, size s and depth d,*

$$\Pr_{x \leftarrow \{0,1\}^n}[C(x) = \mathsf{PARITY}(x)] \leq \frac{1}{2} + 2^{-\Omega(n/(\log s)^{d-1})}$$

Theorem 2.2 (Partial Independence, [Bra10, Tal14])**.** *Let D be a k-wise independent distribution over $\{0,1\}^n$. For any circuit C with n inputs, size s and depth d,*

$$\left| \Pr_{x \leftarrow D}[C(x) = 1] - \Pr_{x \leftarrow \{0,1\}^n}[C(x) = 1] \right| \leq \frac{s}{2^{\Omega(k^{1/(3d+3)})}}$$

Theorem 2.3 (Polylog Hamming Weight, [AB84, RW91])**.** *For any constant c and for any function $t : \mathbb{N} \to \mathbb{N}$ such that $t(\lambda) = O(\log^c \lambda)$, the family $\mathcal{H}^t = \left\{h^t_\lambda\right\}$ is in AC^0, where h^t_λ takes inputs from $\{0,1\}^\lambda$ and is defined as:*

$$h^t_\lambda(x) = 1 \Leftrightarrow \|x\| = t(\lambda)$$

Lemma 2.4 (Polylog Parities)**.** *For any constant c and for any function $t : \mathbb{N} \to \mathbb{N}$ such that $t(\lambda) = O(\log^c \lambda)$, there is an AC^0 family $\mathcal{G}^t = \left\{g^t_\lambda\right\}$ such that for any λ,*

- *g^t_λ takes inputs from $\{0,1\}^\lambda$.*
- *For any $x \in \{0,1\}^\lambda$ such that $\|x\| \leq t(\lambda)$, $g^t_\lambda(x) = \mathsf{PARITY}(x)$.*

Proof. Denote the family promised by Theorem 2.3 for function t' by $\mathcal{H}^{t'} = \left\{h^{t'}_\lambda\right\}$. Then, for any t satisfying the hypothesis for Lemma 2.4, a family $\mathcal{G}^t = \left\{g^t_\lambda\right\}$ that proves the lemma can be computed as:

$$g^t_\lambda(x) = \left\{ \bigwedge_{odd\ i \leq t(\lambda)} h^i_\lambda(x) \right\}$$

□

Lemma 2.5 (Polylog Inner Products)**.** *For any constant c and for any function $t : \mathbb{N} \to \mathbb{N}$ such that $t(\lambda) = O(\log^c \lambda)$, there is an AC^0 family $\mathcal{I}^t = \left\{ip^t_\lambda\right\}$ such that for any λ,*

- *ip^t_λ takes inputs from $\{0,1\}^\lambda \times \{0,1\}^\lambda$.*
- *For any $x, y \in \{0,1\}^\lambda$ such that $\min(\|x\|, \|y\|) \leq t(\lambda)$, $ip^t_\lambda(x,y) = \langle x, y \rangle$.*

Proof. This follows from Lemma 2.4 and the fact that $\langle x, y \rangle = \mathsf{PARITY}(x_1 \wedge y_1, \ldots, x_\lambda \wedge y_\lambda)$. □

2.3 Graphs and Linear Codes

In this section we describe and prove some properties of a sampling procedure for random matrices. While it is not necessary to do so, the proofs in this section are most easily presented in terms of properties of random bipartite graphs. Most of the analysis is as suggested in Gallager's early work ([Gal62]) on Low-Density Parity Check codes, but we repeat it here to make dependencies on certain parameters explicit and to be able to easily derive certain lemmas that we need.

Notation. We denote a bipartite (undirected) graph with vertex sets L and R and set of edges E between L and R by $G(L \cup R, E)$. The *adjacency matrix* of this graph, denoted A_G, is a $\{0,1\}$-matrix of dimension $|L| \times |R|$. Its rows are labeled by vertices in L and columns by vertices in R such that $\mathsf{A}_G[u,v]$ is 1 if and only if $(u,v) \in E$. Given a $\{0,1\}$-matrix M, G_M denotes the bipartite graph whose adjacency matrix is M.

Given a bipartite graph $G(L \cup R, E)$, for any vertex u, $N(u)$ denotes the set of neighbors of u. For a set $S \subseteq L$ (or $S \subseteq R$), $N(S)$ denotes the set of neighbors of vertices in S, that is, $N(S) = \bigcup_{u \in S} N(u)$. And $U(S)$ denotes the set of vertices that are neighbors of a unique vertex in S, that is, $U(S) = \{v \mid |N(v) \cap S| = 1\}$.

Definition 2.1 (Bipartite Expander)**.** An (n,k,d,γ,α)-bipartite expander is a bipartite graph $G(L \cup R, E)$ where:

- $|L| = n$ and $|R| = k$.
- The degree of any vertex in L is d.
- For every $S \subseteq L$ with $|S| \leq \gamma n$, $|N(S)| \geq \alpha |S|$.

We describe the following two sampling procedures that we shall use later. SRSamp and SMSamp abbreviate *Sparse Row Sampler* and *Sparse Matrix Sampler*, respectively. $\mathsf{SRSamp}(k,d,r)$ samples unformly at random a vector from $\{0,1\}^k$ with exactly d non-zero entries, using r for randomness – it chooses a set of d distinct indices between 0 to $k-1$ (via rejection sampling) and outputs the vector in which the entries at those indices are 1 and the rest are 0. When we don't specifically need to argue about the randomness, we drop the explicitly written r. $\mathsf{SMSamp}(n,k,d)$ samples an $n \times k$ matrix whose rows are samples from $\mathsf{SRSamp}(k,d,r)$ using randomly and independently chosen r's.

For any fixed k and $d < k$, note that the function $S_{k,d} : \{0,1\}^{d^2 \lceil \log(k) \rceil} \to \{0,1\}^k$ given by $S_{k,d}(x) = \mathsf{SRSamp}(k,d,x)$ can be easily seen to be computed by a circuit of size $O((d^3 + kd^2)\log(k))$ and depth 8. And so the family $\mathcal{S} = \left\{S_{\lambda,d(\lambda)}\right\}$ is in AC^0. When, in our constructions, we require computing $\mathsf{SRSamp}(k,d,x)$, this is to be understood as being performed by the circuit for $S_{k,d}$ that is given as input the prefix of x of length $d^2 \lceil \log(k) \rceil$. So if the rest of the construction is computed by polynomial-sized constant depth circuits, the calls to SRSamp do not break this property.

Recall that we denote by $SpR_{k,d}$ the uniform distribution over the set of vectors in $\{0,1\}^k$ with exactly d non-zero entries, and by $SpM_{n,k,d}$ the distribution over the set of matrices in $\{0,1\}^{n \times k}$ where each row is sampled independently according to $SpR_{k,d}$. The following lemma states that the above sampling procedures produce something close to these distributions.

Lemma 2.6 (Uniform Sparse Sampling)**.** *For any n, and $d = \log^2(k)$, there is a negligible function ν such that for any k that is a power of two, when $r \leftarrow \{0,1\}^{\log^5(k)}$,*

1. $\Delta(\mathsf{SRSamp}(k,d,r), SpR_{k,d}) \leq \nu(k)$

Construction 2.1 Sparse row and matrix sampling.

$\mathsf{SRSamp}(k, d, r)$: Samples a vector with exactly d non-zero entries.

1. If r is not specified or $|r| < d^2 \lceil \log(k) \rceil$, sample $r \leftarrow \{0,1\}^{d^2 \lceil \log(k) \rceil}$ anew.
2. For $l \in [d]$ and $j \in [d]$, set $u_j^l = r_{((l-1)d+j-1)\lceil \log(k) \rceil + 1} \cdots r_{((l-1)d+j)\lceil \log(k) \rceil}$.
3. If there is no l such that for all distinct $j_1, j_2 \in [d]$, $u_{j_1}^l \neq u_{j_2}^l$, output 0^k.
4. Else, let l_0 be the least such l.
5. For $i \in [k]$, set $v_i = 1$ if there is a $j \in [d]$ such that $u_j^{l_0} = i$ (when interpreted in binary), or $v_i = 0$ otherwise.
6. Output $v = (v_1, \ldots, v_k)$.

$\mathsf{SMSamp}(n, k, d)$: Samples a matrix where each row has d non-zero entries.

1. For $i \in [n]$, sample $r_i \leftarrow \{0,1\}^{d^2 \lceil \log(k) \rceil}$ and $a_i \leftarrow \mathsf{SRSamp}(k, d, r_i)$.
2. Output the $n \times k$ matrix whose i-th row is a_i.

2. $\Delta(\mathsf{SMSamp}(n, k, d), SpM_{n,k,d}) \leq n\nu(k)$

Proof. (1) implies (2) because $\mathsf{SMSamp}(n, k, d)$ and $SpM_{n,k,d}$ simply consist of n independent samples from $\mathsf{SRSamp}(k, d, r)$ and $SpR_{k,d}$, respectively.

$\mathsf{SRSamp}(k, d, r)$ parses r into d sets of d elements from $[k]$ and outputs 0^k when all of these have at least one collision. The probability that any one set has a pair which collide is at most $\frac{d^2}{k}$, by the union bound. So the probability that all sets have at least one pair which collide is at most $\left(\frac{d^2}{k}\right)^d$, which is a negligible function of k when $d = \log^2(k)$. Further, conditioned on this not happening, the output of $\mathsf{SRSamp}(k, d, r)$ is distributed according to $SpR_{k,d}$. So its distance from $SpR_{k,d}$ is exactly the probability that it outputs 0^k. □

The following lemma says that if we sample matrices using SMSamp, they will be adjacency matrices of bipartite expanders with very high probability. It will be used later to argue about certain linear algebraic properties of such matrices that find use in our contructions.

Lemma 2.7 (Sampling expanders)**.** *For any constant $c > 0$, any $n \leq k^c$, $d = \log^2(k)$, $\alpha = \frac{3d}{4}$, and $\gamma = \frac{k}{\log^3(k) n}$, there is a negligible function ν such that for any k that is a power of two,*

$$\Pr_{\mathsf{A}_G \leftarrow \mathsf{SMSamp}(n,k,d)} [G \text{ is not an } (n, k, d, \gamma, \alpha)\text{-expander}] \leq \nu(k)$$

Proof. The proof of this Lemma is a probabilistic argument. By Lemma 2.6, the output of $\mathsf{SMSamp}(n, k, d)$ for our parameters is negligibly close to $SpM_{n,k,d}$. So it is sufficient to prove the theorem when A_G is sampled according to $SpM_{n,k,d}$. This corresponds to a distribution over graphs with vertex sets L and R such that $|L| = n$, $|R| = k$, and for each vertex in L, a set of d of its k possible edges are chosen uniformly at random to be added to the graph. So each vertex in L has degree exactly d.

By definition, a bipartite graph $G(L \cup R, E)$ where the degree of any vertex in L is d is *not* an $(n, k, d, \gamma, \alpha)$-expander if and only if there exist sets $S \subseteq L$ and $T \subseteq R$ such that $|S| \leq \gamma n$,

$|T| = \alpha\,|S|$ such that $N(S) \subseteq T$. We shall now estimate the probability that there exists such an S, T in a graph whose adjacency matrix is randomly generated according to $SpM_{n,k,d}$, given that k is a power of 2 (so that $\lceil \log(k) \rceil = \log(k)$).

Given vertex sets L and R, pick any pair of sets $S \subseteq L$ and $T \subseteq R$. Let $|S| = s$ and $|T| = t$. The probability that all neighbours of S are in T is given by:

$$\begin{aligned}
\Pr\left[N(S) \subseteq T\right] &= \Pr\left[\forall u \in S : \text{all } d \text{ edges chosen connect to vertices in } T\right] \\
&\leq \Pr\left[\forall u \in S : d \text{ independently chosen vertices of } R \text{ are all in } T\right] \\
&\leq \left(\frac{t}{k}\right)^{ds}
\end{aligned}$$

The probability that there exist such an S and T such that $|S| \leq \gamma n$ and $|T| = \alpha|S|$, is bounded as follows:

$$\begin{aligned}
\Pr\left[\exists S, T : |S| \leq \gamma n, |T| = \alpha|S|, N(S) \subseteq T\right] &\leq \sum_{s=1}^{\gamma n} \binom{n}{s}\binom{k}{\alpha s} \Pr\left[N(S) \subseteq T\right] \\
&\leq \sum_{s=1}^{\gamma n} \binom{n}{s}\binom{k}{\alpha s} \left(\frac{\alpha s}{k}\right)^{ds} \\
&\leq \sum_{s=1}^{\gamma n} \left(\frac{ne}{s}\left(\frac{ke}{\alpha s}\right)^{\alpha}\left(\frac{\alpha s}{k}\right)^{d}\right)^{s}
\end{aligned}$$

where the last inequality follows from the fact that $\binom{n}{k} \leq \left(\frac{ne}{k}\right)^k$.

Now we consolidate the terms in the above expression, and use the fact that $1 \leq s \leq \gamma n$. Further, as all terms are positive, extending the sum to infinity provides an upper bound.

$$\begin{aligned}
\sum_{s=1}^{\gamma n} \left(\frac{ne}{s}\left(\frac{ke}{\alpha s}\right)^{\alpha}\left(\frac{\alpha s}{k}\right)^{d}\right)^{s} &= \sum_{s=1}^{\gamma n} \left(\frac{ne^{\alpha+1}}{s}\left(\frac{\alpha s}{k}\right)^{d-\alpha}\right)^{s} \\
&\leq \sum_{s=1}^{\infty} \left(ne^{\alpha+1}\left(\frac{\alpha\gamma n}{k}\right)^{d-\alpha}\right)^{s}
\end{aligned}$$

We now set the parameters as specified in the hypothesis: $d = \log^2(k)$, $\gamma n = \frac{k}{\log^3(k)}$, $\alpha = \frac{3d}{4}$, and $n \leq k^c$ for some constant c to get:

$$\sum_{s=1}^{\infty} \left(ne^{\alpha+1}\left(\frac{\alpha\gamma n}{k}\right)^{d-\alpha}\right)^{s} \leq \sum_{s=1}^{\infty} \left(k^c e\left(\frac{3e^3\log^2(k)}{4}\frac{k}{k\log^3(k)}\right)^{\frac{1}{4}\log^2(k)}\right)^{s}$$

For large enough k, the term inside the paranthesis is smaller than $\frac{1}{2}$. For such k, the following holds:

$$\sum_{s=1}^{\infty} \left(k^c e\left(\frac{3e^3}{4\log(k)}\right)^{\frac{1}{4}\log^2(k)}\right)^{s} \leq 2k^c e\left(\frac{3e^3}{4\log(k)}\right)^{\frac{1}{4}\log^2(k)}$$

Asymptotically, the following relation may be seen by moving all terms to the exponent:

$$\begin{aligned}
&2k^c e \left(\frac{3e^3}{4\log(k)}\right)^{\frac{1}{4}\log^2(k)} \\
&= exp\left[1 + c\log(k) + \left(\frac{3}{4}\log^2(k) + 1\right)\log(e) - \frac{1}{4}\log^2(k)\log\left(\frac{4}{3}\log(k)\right)\right] \\
&= 2^{-\Omega(\log^2(k)\log\log(k))}
\end{aligned}$$

As noted at the beginning, this is also an upper bound on the probability that a graph sampled with $\mathsf{SMSamp}(n,k,d)$ conditioned on none of its calls to SRSamp failing is not an (n,k,d,γ,α)-expander for these parameters. As the probability of any call to SRSamp failing was also seen to be negligible, this shows the existence of a negigible function ν as required. □

Expander codes are linear codes that are constructed by taking the adjacency matrix of a bipartite expander as the parity check matrix of the linear code. We describe the following property of such codes.

Lemma 2.8. *Let G be an (n,k,d,γ,α) expander. If $\alpha > d/2$, the $[n,(n-k)]_2$ code whose parity check matrix is A_G has minimum distance greater than γn.*

Proof. Recall that for any $S \subseteq L$, $U(S)$ is the set of vertices in R that have exactly one neighbour in S. We first show that in an (n,k,d,γ,α)-expander with $\alpha > d/2$, $U(S)$ is non-empty for all S of size at most γn.

This is because for any $S \subseteq L$ of size at most γn, G being an expander implies that $|N(S)| \geq \alpha|S| > d|S|/2$. We know that all vertices in S have degree at most d, and so the number of outgoing edges from S is at most $d|S|$. If $U(S)$ is empty, this implies that all vertices in $N(S)$ have at least 2 edges from S, implying that the number of edges from S to $N(S)$ is at least $2|N(S)| > d|S|$, which is a contradiction.

Consider any non-zero codeword $\mathbf{x}$ in the $[n,(n-k)]_2$ code that has A_G as its parity check matrix. The fact that $\mathbf{x}$ belongs to the code implies that the rows indexed by non-zero positions in $\mathbf{x}$ sum to $\mathbf{0}^T$. But if $\mathbf{x} \leq \gamma n$, the fact that $U(S)$ is non-empty implies that there is at least one column such that exactly one of these rows is non-zero in that column, which implies that the sum of all these rows cannot be $\mathbf{0}^T$. So $\|\mathbf{x}\| > \gamma n$, and the distance of the code is more than γn. □

The following is implied immediately by Lemmas 2.7 and 2.8 and says that with high probability the output of SMSamp defines a code with high distance.

Lemma 2.9 (Sampling codes)**.** *For any constant $c > 0$, set $n = k^c$, and $d = \log^2(k)$. For a matrix $\mathbf{H}$, let $\delta(\mathbf{H})$ denote the minimum distance of the code whose parity check matrix is $\mathbf{H}$. Then, there is a negligible function ν such that for any k that is a power of two,*

$$\Pr_{\mathbf{H}\leftarrow\mathsf{SMSamp}(n,k,d)}\left[\delta(\mathbf{H}) \geq \frac{k}{\log^3(k)}\right] \geq 1 - \nu(k)$$

Recall that a *δ-wise independent* distribution over n bits is a distribution whose marginal distribution on any set of δ bits is the uniform distribution.

Lemma 2.10 (Distance and Independence)**.** *Let* $\mathbf{H}$ *(of dimension* $n \times k$*) be the parity check matrix of an* $[n, (n-k)]_2$ *linear code of minimum distance more than* δ*. Then, the distribution of* $\mathbf{Hx}$ *is* δ*-wise independent when* $\mathbf{x}$ *is chosen uniformly at random from* $\{0,1\}^k$*.*

Proof. The distance of the $[n, (n-k)]_2$ code being more than δ implies that there is no non-zero vector $\mathbf{y}$ in $\{0,1\}^n$ such that $\mathbf{y}^T\mathbf{H} = \mathbf{0}^T$ and $\|\mathbf{y}\| \leq \delta$. In particular, this implies that any set of δ rows of $\mathbf{H}$ are linearly independent. Hence, the restriction of $h(\mathbf{x}) = \mathbf{Hx}$ to any set of δ bits is a full rank linear transformation, and if $\mathbf{x}$ is distributed uniformly at random, then so are these bits. This implies that the distribution of $\mathbf{Hx}$ is δ-wise independent. □

The following is immediately implied by Lemmas 2.9, 2.10 and Theorem 2.2. It effectively says that AC^0 circuits cannot distinguish between $(\mathbf{A}, \mathbf{As})$ and $(\mathbf{A}, \mathbf{r})$ when $\mathbf{A}$ is sampled using SRSamp and $\mathbf{s}$ and $\mathbf{r}$ are chosen uniformly at random.

Lemma 2.11. *For any polynomial* n*, there is a negligible function* ν *such that for any Boolean family* $\mathcal{G} = \{g_\lambda\} \in \mathsf{AC}^0$*, and for any* k *that is a power of 2, when* $\mathbf{A} \leftarrow \mathsf{SMSamp}(n(k), k, \log^2(k))$*,* $\mathbf{s} \leftarrow \{0,1\}^k$ *and* $\mathbf{r} \leftarrow \{0,1\}^{n(k)}$*,*

$$|\Pr[g_\lambda(\mathbf{A}, \mathbf{As}) = 1] - \Pr[g_\lambda(\mathbf{A}, \mathbf{r}) = 1]| \leq \nu(\lambda)$$

2.4 Adversaries

Definition 2.2 (Function Family)**.** A *function family* is a family of (possibly randomized) functions $\mathcal{F} = \{f_\lambda\}_{\lambda \in \mathbb{N}}$, where for each λ, f_λ has domain D^f_λ and co-domain R^f_λ.

In most of our considerations, D^f_λ and R^f_λ will be $\{0,1\}^{d^f_\lambda}$ and $\{0,1\}^{r^f_\lambda}$ for some sequences $\{d^f_\lambda\}_{\lambda \in \mathbb{N}}$ and $\{r^f_\lambda\}_{\lambda \in \mathbb{N}}$. Wherever function families are seen to act as adversaries to cryptographic objects, we shall use the terms *adversary* and *function family* interchangeably.

The following are some examples of natural classes of function families. These are in the vein of classes like FP and FNP, which are defined starting from P and NP, respectively. For the sake of brevity, we refer to classes of function families as function classes. Also, we will abuse taxonomy and use the same name for a class of languages and its corresponding class of functions (we would, for instance, simply say P instead of FP for the class of deterministic polynomial-time computable functions).

Definition 2.3 (AC^0)**.** The class of (non-uniform) AC^0 function families is the set of all function families $\mathcal{F} = \{f_\lambda\}$ for which there is a polynomial p and constant d such that for each λ, f_λ can be computed by a (randomized) circuit of size $p(\lambda)$, depth d and unbounded fan-in using AND, OR and NOT gates.

Definition 2.4 ($\mathsf{AC}^0[2]$)**.** The class of (non-uniform) $\mathsf{AC}^0[2]$ function families is the set of all function families $\mathcal{F} = \{f_\lambda\}$ for which there is a polynomial p and constant d such that for each λ, f_λ can be computed by a (randomized) circuit of size $p(\lambda)$, depth d and unbounded fan-in using AND, OR, NOT and PARITY gates.

Definition 2.5 (NC^1)**.** The class of (non-uniform) NC^1 function families is the set of all function families $\mathcal{F} = \{f_\lambda\}$ for which there is a polynomial p and constant c such that for each λ, f_λ can be computed by a (randomized) circuit of size $p(\lambda)$, depth $c\log(\lambda)$ and fan-in 2 using AND, OR and NOT gates

Definition 2.6 ($\oplus$L/poly)**.** $\oplus$L/poly is the set of all Boolean function families $\mathcal{F} = \{f_\lambda\}$ for which there is a constant c such that for each λ, there is a Non-Deterministic Turing Machine M_λ such that for each input x of length λ, $M_\lambda(x)$ uses at most $c\log(\lambda)$ space, and $f_\lambda(x)$ is equal to the PARITY of the number of accepting paths of $M_\lambda(x)$.

Definition 2.7 (BPP)**.** The class of BPP function families is the set of all function families $\mathcal{F} = \{f_\lambda\}$ for which there is a randomized polynomial-time algorithm $B_\mathcal{F}$ such that for each λ and x, $f_\lambda(x) \equiv B_\mathcal{F}(1^\lambda, x)$.

2.5 Primitives Against Bounded Adversaries

In this section, we generalize the standard definitions of several standard cryptographic primitives to talk about security against different classes of adversaries. In the following definitions, $\mathcal{C}_1$ and $\mathcal{C}_2$ are two function classes, and $l, s : \mathbb{N} \to \mathbb{N}$ are some functions.

Implicit (and hence left unmentioned) in each definition are the following conditions:

- *Computability*, which says that the function families that are part of the primitive are in the class $\mathcal{C}_1$. Additional restrictions are specified when they apply.
- *Non-triviality*, which says that the security condition in each definition is not vacuously satisfied – that there is at least one function family in $\mathcal{C}_2$ whose input space corresponds to the output space of the appropriate function family in the primitive.

Definition 2.8 (One-Way Function)**.** Let $\mathcal{F} = \left\{ f_\lambda : \{0,1\}^\lambda \to \{0,1\}^{l(\lambda)} \right\}$ be a function family. $\mathcal{F}$ is a $\mathcal{C}_1$*-One-Way Function (OWF) against* $\mathcal{C}_2$ if:

- **Computability:** For each λ, f_λ is deterministic.
- **One-wayness:** For any $\mathcal{G} = \left\{ g_\lambda : \{0,1\}^{l(\lambda)} \to \{0,1\}^\lambda \right\} \in \mathcal{C}_2$, there is a negligible function ν such that for any $\lambda \in \mathbb{N}$:

$$\Pr_{x \leftarrow U_\lambda} \left[f_\lambda(g_\lambda(y)) = y \mid y \leftarrow f_\lambda(x) \right] \leq \nu(\lambda)$$

For a function class $\mathcal{C}$, we sometimes refer to a $\mathcal{C}$-OWF or an OWF against $\mathcal{C}$. In both these cases, both $\mathcal{C}_1$ and $\mathcal{C}_2$ from the above definition are to be taken to be $\mathcal{C}$. To be clear, this implies that there is a family $\mathcal{F} \in \mathcal{C}$ that realizes the primitive and is secure against all $\mathcal{G} \in \mathcal{C}$. We shall adopt this abbreviation also for other primitives defined in the above manner.

Definition 2.9 (Pseudo-Random Generator)**.** Let $\mathcal{F} = \left\{ f_\lambda : \{0,1\}^\lambda \to \{0,1\}^{l(\lambda)} \right\}$ be a function family. $\mathcal{F}$ is a $\mathcal{C}_1$*-Pseudo-Random Generator (PRG) against* $\mathcal{C}_2$ if:

- **Computability:** For each λ, f_λ is deterministic.
- **Expansion:** $l(\lambda) > \lambda$ for all λ. $a(\lambda) = (l(\lambda) - \lambda)$ is called the additive stretch of the PRG, and $m(\lambda) = \frac{l(\lambda)}{\lambda}$ is called its multiplicative stretch.
- **Pseudo-randomness:** For any $\mathcal{G} = \left\{ g_\lambda : \{0,1\}^{l(\lambda)} \to \{0,1\} \right\} \in \mathcal{C}_2$, there is a negligible function ν such that for any $\lambda \in \mathbb{N}$:

$$\left| \Pr_{x \leftarrow U_\lambda} \left[g_\lambda(f_\lambda(x)) = 1 \right] - \Pr_{y \leftarrow U_{l(\lambda)}} \left[g_\lambda(y) = 1 \right] \right| \leq \nu(\lambda)$$

Definition 2.10 (Collision Resistant Hashing)**.** Let $\mathcal{K}eyGen = \{\mathsf{KeyGen}_\lambda : \varnothing \to K_\lambda\}$ and $\mathcal{E}val = \left\{\mathsf{Eval}_\lambda : K_\lambda \times \{0,1\}^\lambda \to \{0,1\}^{l(\lambda)}\right\}$ be function families. For a function $s : \mathbb{N} \to \mathbb{N}$, $(\mathcal{K}eyGen, \mathcal{E}val)$ is a $\mathcal{C}_1$-*Collision Resistant Hash Family (CRHF) against* $\mathcal{C}_2$ *with compression* s if:

- **Computability:** For each λ, Eval_λ is deterministic.
- **Compression:** For all large enough λ, $l(\lambda) \leq \frac{\lambda}{s(\lambda)} < \lambda$.
- **Collision Resistance:** For any $\mathcal{G} = \left\{g_\lambda : K_\lambda \to \{0,1\}^\lambda \times \{0,1\}^\lambda\right\} \in \mathcal{C}_2$, there is a negligible function ν such that for any $\lambda \in \mathbb{N}$:

$$\Pr_{k \leftarrow \mathsf{KeyGen}_\lambda}\left[\mathsf{Eval}_\lambda(k,x) = \mathsf{Eval}_\lambda(k,y) \mid (x,y) \leftarrow g_\lambda(k)\right] \leq \nu(\lambda)$$

Definition 2.11 (Weak Pseudo-Random Functions)**.** Let $\mathcal{K}eyGen = \{\mathsf{KeyGen}_\lambda : \varnothing \to K_\lambda\}$ and $\mathcal{E}val = \left\{\mathsf{Eval}_\lambda : K_\lambda \times \{0,1\}^\lambda \to \{0,1\}^{l(\lambda)}\right\}$ be function families. $(\mathcal{K}eyGen, \mathcal{E}val)$ is a *Weak* $\mathcal{C}_1$-*Pseudo-Random Function Family (Weak PRF) against* $\mathcal{C}_2$ if:

- **Computability:** For each λ, Eval_λ is deterministic.
- **Pseudo-randomness:** Let $F_{l,\lambda}$ be the set of all functions from $\{0,1\}^\lambda$ to $\{0,1\}^{l(\lambda)}$. For any function $n : \mathbb{N} \to \mathbb{N}$ and any $\mathcal{G} = \left\{g_\lambda : \left(\{0,1\}^\lambda \times \{0,1\}^{l(\lambda)}\right)^{n(\lambda)} \to \{0,1\}\right\} \in \mathcal{C}_2$, there is a negligible function ν such that for any $\lambda \in \mathbb{N}$:

$$\left| \Pr_{\substack{k \leftarrow \mathsf{KeyGen}_\lambda \\ x_1,\ldots,x_{n(\lambda)} \leftarrow U_\lambda}}\left[g_\lambda\left(\{(x_i, \mathsf{Eval}_\lambda(k,x_i))\}\right) = 1\right] - \Pr_{\substack{f \leftarrow F_{l,\lambda} \\ x_1,\ldots,x_{n(\lambda)} \leftarrow U_\lambda}}\left[g_\lambda\left(\{(x_i, f(x_i))\}\right) = 1\right] \right| \leq \nu(\lambda)$$

The adjective *Weak* in the above definition is present because the adversary gets evaluations of the PRF or a random function on randomly chosen input values. In the standard definition of a PRF, the adversary is allowed to choose inputs at which to receive functions evaluations. While this is a natural definition for polynomial-time adversaries, it is unclear how to extend this definition to other classes of adversaries, especially to classes like AC^0.

Definition 2.12 (Symmetric Key Encryption)**.** Consider a triple of function families $\mathcal{K}eyGen = \{\mathsf{KeyGen}_\lambda : \varnothing \to K_\lambda\}$, $\mathcal{E}nc = \{\mathsf{Enc}_\lambda : K_\lambda \times \{0,1\} \to C_\lambda\}$, and $\mathcal{D}ec = \{\mathsf{Dec}_\lambda : K_\lambda \times C_\lambda \to \{0,1\}\}$. $(\mathcal{K}eyGen, \mathcal{E}nc, \mathcal{D}ec)$ is a $\mathcal{C}_1$-*Symmetric Key Encryption Scheme against* $\mathcal{C}_2$ if:

- **Correctness:** There is a negligible function ν such that for any $\lambda \in \mathbb{N}$ and any $b \in \{0,1\}$:

$$\Pr\left[\mathsf{Dec}_\lambda(k,c) = b \;\middle|\; \begin{array}{c} k \leftarrow \mathsf{KeyGen}_\lambda \\ c \leftarrow \mathsf{Enc}_\lambda(k,b) \end{array}\right] \geq 1 - \nu(\lambda)$$

- **Semantic Security:** For any polynomials $n_0, n_1 : \mathbb{N} \to \mathbb{N}$, and any family $\mathcal{G} = \left\{ g_\lambda : C_\lambda^{n_0(\lambda)+n_1(\lambda)+1} \to \{0,1\} \right\}$ $\mathcal{C}_2$, there is a negligible function ν' such that for any $\lambda \in \mathbb{N}$:

$$\Pr\left[g_\lambda\left(\{c_i^0\}, \{c_i^1\}, c\right) = b \;\middle|\; \begin{array}{c} k \leftarrow \mathsf{KeyGen}_\lambda, b \leftarrow U_1 \\ c_1^0, \ldots, c_{n_0(\lambda)}^0 \leftarrow \mathsf{Enc}_\lambda(k, 0) \\ c_1^1, \ldots, c_{n_1(\lambda)}^1 \leftarrow \mathsf{Enc}_\lambda(k, 1) \\ c \leftarrow \mathsf{Enc}_\lambda(k, b) \end{array} \right] \leq \frac{1}{2} + \nu'(\lambda)$$

Definition 2.13 (Public Key Encryption)**.** Let $\mathcal{K}eyGen = \{\mathsf{KeyGen}_\lambda : \varnothing \to SK_\lambda \times PK_\lambda\}$, $\mathcal{E}nc = \{\mathsf{Enc}_\lambda : PK_\lambda \times \{0,1\} \to C_\lambda\}$, and $\mathcal{D}ec = \{\mathsf{Dec}_\lambda : SK_\lambda \times C_\lambda \to \{0,1\}\}$ be function families. $(\mathcal{K}eyGen, \mathcal{E}nc, \mathcal{D}ec)$ is a $\mathcal{C}_1$-*Public Key Encryption scheme against* $\mathcal{C}_2$ if:

- **Correctness:** There is a negligible function ν such that for any $\lambda \in \mathbb{N}$ and any $b \in \{0,1\}$:

$$\Pr\left[\mathsf{Dec}_\lambda(\mathsf{sk}, c) = b \;\middle|\; \begin{array}{c} (\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}_\lambda \\ c \leftarrow \mathsf{Enc}_\lambda(\mathsf{pk}, b) \end{array} \right] \geq 1 - \nu(\lambda)$$

- **Semantic Security:** For any $\mathcal{G} = \{g_\lambda : PK_\lambda \times C_\lambda \to \{0,1\}\} \in \mathcal{C}_2$, there is a negligible function ν' such that for any $\lambda \in \mathbb{N}$:

$$\Pr\left[g_\lambda(\mathsf{pk}, c) = b \;\middle|\; \begin{array}{c} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}_\lambda, b \leftarrow U_1 \\ c \leftarrow \mathsf{Enc}_\lambda(\mathsf{pk}, b) \end{array} \right] \leq \frac{1}{2} + \nu'(\lambda)$$

2.6 Randomized Encodings

The notion of randomized encodings of functions was introduced by Ishai and Kushilevitz [IK00] in the context of secure multi-party computation. Roughly, a randomized encoding of a deterministic function f is another deterministic function $\widehat{f}$ that is easier to compute by some measure, and is such that for any input x, the distribution of $\widehat{f}(x, r)$ (when r is chosen uniformly at random) reveals the value of $f(x)$ and nothing more. This reduces the computation of $f(x)$ to determining some property of the distribution of $\widehat{f}(x, r)$. Hence, randomized encodings offer a flavor of worst-to-average case reduction — from computing $f(x)$ from x to that of computing $f(x)$ from random samples of $\widehat{f}(x, r)$.

We work with the following definition of *Perfect Randomized Encodings* from [App14]. We note that constructions of such encodings for $\oplus\mathsf{L}/\mathsf{poly}$ which are computable in NC^0 were presented in [IK00].

Definition 2.14 (Perfect Randomized Encodings)**.** Consider a deterministic function $f : \{0,1\}^n \to \{0,1\}^t$. We say that the deterministic function $\widehat{f} : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^s$ is a *Perfect Randomized Encoding (PRE)* of f if the following conditions are satisfied.

- **Input independence:** For every $x, x' \in \{0,1\}^n$ such that $f(x) = f(x')$, the random variables $\widehat{f}(x, U_m)$ and $\widehat{f}(x', U_m)$ are identically distributed.
- **Output disjointness:** For every $x, x' \in \{0,1\}^n$ such that $f(x) \neq f(x')$, $\mathrm{Supp}(\widehat{f}(x, U_m)) \cap \mathrm{Supp}(\widehat{f}(x', U_m)) = \phi$.
- **Uniformity:** For every x, $\widehat{f}(x, U_m)$ is uniform on its support.
- **Balance:** For every $x, x' \in \{0,1\}^n$, $\left|\mathrm{Supp}(\widehat{f}(x, U_m))\right| = \left|\mathrm{Supp}(\widehat{f}(x', U_m))\right|$

- **Stretch preservation:** $s - (n + m) = t - n$

Additionally, the PRE is said to be *surjective* if it also has the following property.

- **Surjectivity:** For every $y \in \{0,1\}^s$, there exist x and r such that $\widehat{f}(x,r) = y$.

We naturally extend the definition of PREs to function families – a family $\widehat{\mathcal{F}} = \left\{\widehat{f_\lambda}\right\}$ is a PRE of another family $\mathcal{F} = \{f_\lambda\}$ if for all large enough λ, $\widehat{f_\lambda}$ is a PRE of f_λ. Note that this notion only makes sense for deterministic functions, and the functions and families we assume or claim to have PREs are to be taken to be deterministic.

3 OWFs from worst-case assumptions

In this section and in Section 4, we describe some constructions of cryptographic primitives against bounded adversaries starting from worst-case hardness assumptions. The existence of Perfect Randomized Encodings (PREs) can be leveraged to construct one-way functions and pseudo-random generators against bounded adversaries starting from a function that is hard in the worst-case for these adversaries. We describe this construction below.

Remark 3.1 (Infinitely often primitives)**.** *For a class $\mathcal{C}$, the statement $\mathcal{F} = \{f_\lambda\} \notin \mathcal{C}$ implies that for any family $\mathcal{G} = \{g_\lambda\}$ in $\mathcal{C}$, there are an infinite number of values of λ such that $f_\lambda \not\equiv g_\lambda$. Using such a worst case assumption, we only know how to obtain primitives whose security holds for an infinite number of values of λ, as opposed to holding for all large enough λ. Such primitives are called* infinitely-often*, and all primitives constructed in this section and Section 4 are infinitely-often primitives.*

On the other hand, if we assume that for every $\mathcal{G} \in \mathcal{C}$, there exists λ_0 such that for all $\lambda > \lambda_0$, $f_\lambda \not\equiv g_\lambda$ we can achieve the regular stronger notion of security (that holds for all large enough security parameters) in each case by the same techniques.

Theorem 3.1 (OWFs, PRGs from PREs)**.** *Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be two function classes satisfying the following conditions:*

1. *Any function family in $\mathcal{C}_2$ has a surjective PRE computable in $\mathcal{C}_1$.*
2. *$\mathcal{C}_2 \not\subseteq \mathcal{C}_1$.*
3. *$\mathcal{C}_1$ is closed under a constant number of compositions.*
4. *$\mathcal{C}_1$ is non-uniform or randomized.*
5. *$\mathcal{C}_1$ can compute arbitrary thresholds.*

Then:

1. *There is a $\mathcal{C}_1$-OWF against $\mathcal{C}_1$.*
2. *There is a $\mathcal{C}_1$-PRG against $\mathcal{C}_1$ with non-zero additive stretch.*

Theorem 3.1 in effect shows that the existence of a language with PREs outside $\mathcal{C}_1$ implies the existence of one way functions and pseudorandom generators computable in $\mathcal{C}_1$ secure against $\mathcal{C}_1$. Instances of classes that satisfy its hypothesis (apart from $\mathcal{C}_2 \not\subseteq \mathcal{C}_1$) include NC^1 and BPP. Note that this theorem does not provide constructions against AC^0 because AC^0 cannot compute arbitrary thresholds.

Proof. Let $\mathcal{F} = \left\{ f_\lambda : \{0,1\}^{n(\lambda)} \to \{0,1\} \right\}$ be a function family in $\mathcal{C}_2$ that is not in $\mathcal{C}_1$, and let $\widehat{\mathcal{F}} = \left\{ \widehat{f}_\lambda : \{0,1\}^{n(\lambda)} \times \{0,1\}^{m(\lambda)} \to \{0,1\}^{s(\lambda)} \right\}$ be its PRE that is in $\mathcal{C}_1$. We define the family $\mathcal{G} = \left\{ g_\lambda : \{0,1\}^{m(\lambda)} \to \{0,1\}^{s(\lambda)} \right\}$ as:

$$g_\lambda(x) = \widehat{f}_\lambda(0^{n(\lambda)}, x)$$

We claim that $\mathcal{G}$ is both a $\mathcal{C}_1$-OWF and a $\mathcal{C}_1$-PRG against $\mathcal{C}_1$ with non-zero additive stretch. In both cases, computability and non-triviality are easily seen to be satisfied. The non-zero additive stretch follows from the stretch-preserving property of $\widehat{f}_\lambda$, which guarantees that $(s(\lambda) - m(\lambda)) = 1$.

We now show the pseudorandomness of $\mathcal{G}$ against adversaries in $\mathcal{C}_1$. It is easily shown by standard arguments that this implies that $\mathcal{G}$ is also one-way against adversaries in $\mathcal{C}_1$.

Suppose there is a family $\mathcal{A} = \left\{ a_\lambda : \{0,1\}^{s(\lambda)} \to \{0,1\} \right\}$ in $\mathcal{C}_1$ such that a_λ distinguishes between the output of g_λ and the uniform distribution with non-negligible advantage. We show how to use $\mathcal{A}$ to show that $\mathcal{F} \in \mathcal{C}_1$, which is a contradiction.

The advantage a_λ has in distinguishing between the output of g_λ and the uniform distribution is given by:

$$\left| \Pr_{x \leftarrow U_\lambda} [a_\lambda(g_\lambda(x)) = 1] - \Pr_{y \leftarrow U_{s(\lambda)}} [a_\lambda(y) = 1] \right| = \left| \Pr_{x \leftarrow U_\lambda} \left[a_\lambda(\widehat{f}_\lambda(0^{n(\lambda)}, x)) = 1 \right] - \Pr_{y \leftarrow U_{s(\lambda)}} [a_\lambda(y) = 1] \right|$$

which is assumed to be non-negligible. Due to the surjectivity of $\widehat{f}_\lambda$, the uniform distribution over $\{0,1\}^{s(\lambda)}$ is the same as the equal convex combination of the distributions of $\widehat{f}_\lambda(0^{n(\lambda)}, r)$ and $\widehat{f}_\lambda(z_1, r)$ for any z_1 such that $f_\lambda(z_1) = 1$. So we can rewrite the above advantage as:

$$\begin{aligned} &\left| \Pr_{x \leftarrow U_\lambda} \left[a_\lambda(\widehat{f}_\lambda(0^{n(\lambda)}, x)) = 1 \right] - \left(\frac{1}{2} \Pr_{x \leftarrow U_\lambda} \left[a_\lambda(\widehat{f}_\lambda(0^{n(\lambda)}, x)) = 1 \right] + \frac{1}{2} \Pr_{x \leftarrow U_\lambda} \left[a_\lambda(\widehat{f}_\lambda(z_1, x)) = 1 \right] \right) \right| \\ &\qquad = \frac{1}{2} \left| \Pr_{x \leftarrow U_\lambda} \left[a_\lambda(\widehat{f}_\lambda(0^{n(\lambda)}, x)) = 1 \right] - \Pr_{x \leftarrow U_\lambda} \left[a_\lambda(\widehat{f}_\lambda(z_1, x)) = 1 \right] \right| \end{aligned}$$

which is non-negligible. Let $p = \Pr_{x \leftarrow U_\lambda} \left[a_\lambda(\widehat{f}_\lambda(0^{n(\lambda)}, x)) = 1 \right]$ and $q = \Pr_{x \leftarrow U_\lambda} \left[a_\lambda(\widehat{f}_\lambda(z_1, x)) = 1 \right]$.

To decide $f_\lambda(z) = f_\lambda(0^{n(\lambda)})$ given z, by the input independence property of $\widehat{f}_\lambda$, it is sufficient to determine whether $\Pr_{x \leftarrow U_\lambda} \left[a_\lambda(\widehat{f}_\lambda(z, x)) = 1 \right]$ is less than $\left(\frac{p+q}{2}\right)$. This may be done by taking several samples from $a_\lambda(\widehat{f}_\lambda(z, x))$ and using the threshold function to check whether more than a $\left(\frac{p+q}{2}\right)$ fraction of these are 1. The fact that p and q are non-negligibly separated implies that some $poly(\lambda)$ samples should suffice to be able to do this with exponentially small failure probability.

By the hypothesis, the function family that performs all these operations is in $\mathcal{C}_1$, and the non-uniformity of $\mathcal{C}_1$ implies that $f_\lambda(0^{n(\lambda)})$ can be used as non-uniform advice to actually decide f_λ, and as noted in Section 1, the randomness involved above can be traded for non-uniformity. This implies that $\mathcal{F}$ is in $\mathcal{C}_1$, which is a contradiction. This proves the pseudo-randomness of $\mathcal{G}$ for adversaries in $\mathcal{C}_1$ (though only in the weak sense mentioned in Remark 3.1).

□

4 PKE against NC^1 from worst-case assumptions

In Theorem 3.1 we saw that we can construct one way functions and PRGs with a small stretch generically from Perfect Randomized Encodings (PREs) starting from worst-case hardness assumptions. We do not know how to construct Public Key Encryption (PKE) in a similar black-box fashion. In this section, we use certain algebraic properties of a specific construction of PREs for functions in $\oplus\mathsf{L}/\mathsf{poly}$ due to Ishai-Kushilevitz [IK00] to construct Public Key Encryption and Collision Resistant Hash Functions against NC^1 that are computable in $\mathsf{AC}^0[2]$ under the assumption that $\oplus\mathsf{L}/\mathsf{poly} \not\subseteq \mathsf{NC}^1$. We state the necessary implications of their work here. We start by describing sampling procedures for some relevant distributions in Construction 4.1.

Construction 4.1 Sampling distributions from [IK00]

Let $\mathbf{M}_0^n$ and $\mathbf{M}_1^n$ be the following $n \times n$ matrices:

$$\mathbf{M}_0 = \begin{pmatrix} 0 & & \cdots & 0 & 0 \\ 1 & 0 & & & 0 \\ 0 & 1 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & 0 & \\ 0 & \cdots & 0 & 1 & 0 \end{pmatrix}, \mathbf{M}_1 = \begin{pmatrix} 0 & & \cdots & 0 & 1 \\ 1 & 0 & & & 0 \\ 0 & 1 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & 0 & \\ 0 & \cdots & 0 & 1 & 0 \end{pmatrix}$$

$\mathsf{LSamp}(n)$:

1. Output an $n \times n$ upper triangular matrix where all entries in the diagonal are 1 and all other entries in the upper triangular part are chosen at random.

$\mathsf{RSamp}(n)$:

1. Sample at random $\mathbf{r} \leftarrow \{0,1\}^{n-1}$.
2. Output the following $n \times n$ matrix:

$$\left(\begin{array}{cccc|c} 1 & 0 & \cdots & 0 & \\ 0 & 1 & \ddots & \vdots & r \\ \vdots & \ddots & \ddots & 0 & \\ 0 & \cdots & 0 & 1 & \\ 0 & \cdots & 0 & 0 & 1 \end{array}\right)$$

In the randomized encodings of [IK00], the output of the encoding of a function f on input x is a matrix $\mathbf{M}$ sampled identically to $\mathbf{R}_1\mathbf{M}_0^\lambda\mathbf{R}_2$ when $f(x) = 0$ and identically to $\mathbf{R}_1\mathbf{M}_1^\lambda\mathbf{R}_2$ when $f(x) = 1$, where $\mathbf{R}_1 \leftarrow \mathsf{LSamp}(\lambda)$ and $\mathbf{R}_2 \leftarrow \mathsf{RSamp}(\lambda)$. Notice that $\mathbf{R}_1\mathbf{M}_1^\lambda\mathbf{R}_2$ is full rank, while $\mathbf{R}_1\mathbf{M}_0^\lambda\mathbf{R}_2$ has rank $(\lambda-1)$. The public key in our encryption scheme is a sample $\mathbf{M}$ from $\mathbf{R}_1\mathbf{M}_0^\lambda\mathbf{R}_2$, and the secret key is a vector $\mathbf{k}$ in the kernel of $\mathbf{M}$. An encryption of 0 is a random vector in the row-span of $\mathbf{M}$ (whose inner product with $\mathbf{k}$ is hence 0), and an encryption of 1 is a random vector that is not in the row-span of $\mathbf{M}$ (whose inner product with $\mathbf{k}$ is non-zero). Decryption is simply inner product with $\mathbf{k}$. (This is very similar to the cryptosystem in [ABW10] albeit without the noise that is added there.)

Security follows from the fact that under our hardness assumption $\mathbf{M}$ is indistinguishable from $\mathbf{R}_1\mathbf{M}_1^\lambda\mathbf{R}_2$ (see Theorem 4.2), which has an empty kernel, and so when used as the public key results in identical distributions of encryptions of 0 and 1.

Construction 4.2 Public Key Encryption

Let λ be the security parameter. Let $\mathbf{M}_0^\lambda$ be the $\lambda \times \lambda$ matrix described in Construction 4.1. Define the families $\mathcal{K}eyGen = \{\mathsf{KeyGen}_\lambda\}$, $\mathcal{E}nc = \{\mathsf{Enc}_\lambda\}$, and $\mathcal{D}ec = \{\mathsf{Dec}_\lambda\}$ as follows.

KeyGen_λ:

1. Sample $\mathbf{R}_1 \leftarrow \mathsf{LSamp}(\lambda)$ and $\mathbf{R}_2 \leftarrow \mathsf{RSamp}(\lambda)$.
2. Let $\mathbf{k} = (\mathbf{r}\ 1)^T$ be the last column of $\mathbf{R}_2$.
3. Compute $\mathbf{M} = \mathbf{R}_1\mathbf{M}_0^\lambda\mathbf{R}_2$.
4. Output $(\mathsf{pk} = \mathbf{M}, \mathsf{sk} = \mathbf{k})$.

$\mathsf{Enc}_\lambda(\mathsf{pk} = \mathbf{M}, b)$:

1. Sample $\mathbf{r} \in \{0,1\}^\lambda$.
2. Let $\mathbf{t}^T = (0\ \ldots\ 0\ 1)$, of length λ.
3. Output $\mathbf{c}^T = \mathbf{r}^T\mathbf{M} + b\mathbf{t}^T$.

$\mathsf{Dec}_\lambda(\mathsf{sk} = \mathbf{k}, \mathbf{c})$:

1. Output $\langle \mathbf{c}, \mathbf{k} \rangle$.

In randomized encodings, encoding is efficient while decoding is not. But notice that this is not an issue in our case, as our scheme never tries to decode any encoding - we rely on the correctness of the randomized encoding only for its implication of Theorem 4.2.

Theorem 4.1 (Public Key Encryption Against NC^1)**.** *Assume* $\oplus\mathsf{L/poly} \not\subseteq \mathsf{NC}^1$*. Then, the tuple of families* $(\mathcal{K}eyGen, \mathcal{E}nc, \mathcal{D}ec)$ *defined in Construction 4.2 is an* $\mathsf{AC}^0[2]$*-Public Key Encryption Scheme against* NC^1*.*

Before beginning with the proof, we describe some properties of the construction. We first begin with two sampling procedures that correspond to sampling from $\widehat{f}(x, \cdot)$ when $f(x) = 0$ or $f(x) = 1$ as described earlier. We describe these again in Construction 4.3.

Construction 4.3 Sampling procedures

$\mathsf{ZeroSamp}(n)$: $\widehat{f}(x, r)$ where $f(x) = 0$

1. Sample $\mathbf{R}_1 \leftarrow \mathsf{LSamp}(n)$ and $\mathbf{R}_2 \leftarrow \mathsf{RSamp}(n)$.
2. Output $\mathbf{R}_1\mathbf{M}_0\mathbf{R}_2$.

$\mathsf{OneSamp}(n)$: $\widehat{f}(x, r)$ where $f(x) = 1$

1. Sample $\mathbf{R}_1 \leftarrow \mathsf{LSamp}(n)$ and $\mathbf{R}_2 \leftarrow \mathsf{RSamp}(n)$.
2. Output $\mathbf{R}_1\mathbf{M}_1\mathbf{R}_2$.

Theorem 4.2 ([IK00, AIK04]). *For any boolean function family $\mathcal{F} = \{f_\lambda\}$ in $\oplus\mathsf{L}/\mathsf{poly}$, there is a polynomial n such that for any λ, f_λ has a PRE $\widehat{f_\lambda}$ such that the distribution of $\widehat{f_\lambda}(x)$ is identical to $\mathsf{ZeroSamp}(n(\lambda))$ when $f_\lambda(x) = 0$ and is identical to $\mathsf{OneSamp}(n(\lambda))$ when $f_\lambda(x) = 1$.*

This implies that if some function in $\oplus\mathsf{L}/\mathsf{poly}$ is hard to compute on the worst-case then it is hard to distinguish between samples from ZeroSamp and OneSamp. In particular, the following lemma follows immediately from the observation that ZeroSamp and OneSamp can be computed in NC^1.

Lemma 4.3. *If $\oplus\mathsf{L}/\mathsf{poly} \not\subseteq \mathsf{NC}^1$, then there is a polynomial n and a negligible function ν such that for any family $\mathcal{F} = \{f_\lambda\}$ in NC^1, for an infinite number of values of λ,*

$$\left| \Pr_{\mathbf{M} \leftarrow \mathsf{ZeroSamp}(n(\lambda))} [f_\lambda(\mathbf{M}) = 1] - \Pr_{\mathbf{M} \leftarrow \mathsf{OneSamp}(n(\lambda))} [f_\lambda(\mathbf{M}) = 1] \right| \leq \nu(\lambda)$$

Now we are in a position to use the indistinguishability result in Lemma 4.3 to prove Theorem 4.1.

Proof of Theorem 4.1. To prove the theorem, we need to show that the functions in the construction are computable in $\mathsf{AC}^0[2]$ and that they are secure against NC^1 adversaries. It is straightforward to see that $\mathcal{K}eyGen$, $\mathcal{E}nc$, and $\mathcal{D}ec$ are in $\mathsf{AC}^0[2]$, as multiplication of any constant number of matrices can be done in constant depth with PARITY gates, and LSamp and RSamp simply involve sampling random bits. Non-triviality is also easily seen to be satisfied.

Let t be as described in Construction 4.2. For any $(\mathbf{M}, \mathbf{k}) \leftarrow \mathsf{KeyGen}_\lambda$, note that $\mathbf{M} = \mathbf{R}_1 \mathbf{M}_0^\lambda \mathbf{R}_2$, and $\mathbf{k} = (r\ 1)^T$ is the last column of R_2. It can be verified easily that $\mathbf{Mk} = \mathbf{R}_1(\mathbf{M}_0^\lambda \mathbf{R}_2 s) = \mathbf{0}$, and that $\langle \mathbf{t}, \mathbf{k} \rangle = 1$. So for any b, $\mathsf{Dec}_\lambda(\mathbf{k}, \mathsf{Enc}_\lambda(\mathbf{M}, b)) = \langle \mathbf{M}^T\mathbf{r} + b\mathbf{t}, \mathbf{k} \rangle = \mathbf{r}^T\mathbf{Mk} + b\langle \mathbf{t}, \mathbf{k} \rangle = b$. This proves correctness.

To prove semantic security, we need to show that the distributions of $(\mathsf{pk}, \mathsf{Enc}_\lambda(\mathsf{pk}, 0))$ and $(\mathsf{pk}, \mathsf{Enc}(\mathsf{pk}, 1))$ are indistinguishable to adversaries in NC^1. Note that by the action of KeyGen_λ and Enc_λ,

$$(\mathsf{pk}, \mathsf{Enc}_\lambda(\mathsf{pk}, 0)) = (\mathbf{M}, \mathbf{r}^T\mathbf{M} \mid \mathbf{M} \leftarrow \mathsf{ZeroSamp}(\lambda), \mathbf{r} \leftarrow \{0,1\}^\lambda)$$

For any adversary in NC^1, we know by Lemma 4.3 that there are an infinite number of values of λ for which:

$$(\mathbf{M}, \mathbf{r}^T\mathbf{M} \mid \mathbf{M} \leftarrow \mathsf{ZeroSamp}(\lambda), \mathbf{r}) \approx (\mathbf{M}, \mathbf{r}^T\mathbf{M} \mid \mathbf{M} \leftarrow \mathsf{OneSamp}(\lambda), \mathbf{r})$$

But the output of OneSamp is always full rank. Hence the distribution of $\mathbf{r}^T\mathbf{M}$ is uniform over $\{0,1\}^\lambda$. Then,

$$(\mathbf{M}, \mathbf{r}^T\mathbf{M} \mid \mathbf{M} \leftarrow \mathsf{OneSamp}(\lambda), \mathbf{r}) = (\mathbf{M}, \mathbf{r}^T\mathbf{M} + \mathbf{t}^T \mid \mathbf{M} \leftarrow \mathsf{OneSamp}(\lambda), \mathbf{r})$$

For the same adversary and the same infinite set of values of λ as before,

$$(\mathbf{M}, \mathbf{r}^T\mathbf{M} + \mathbf{t}^T \mid \mathbf{M} \leftarrow \mathsf{OneSamp}(\lambda), \mathbf{r}) \approx (\mathbf{M}, \mathbf{r}^T\mathbf{M} + \mathbf{t}^T \mid \mathbf{M} \leftarrow \mathsf{ZeroSamp}(\lambda), \mathbf{r})$$

which is the distribution of $(\mathsf{pk}, \mathsf{Enc}_\lambda(\mathsf{pk}, 1))$. This proves the indistinguishability necessary for semantic security. □

Remark 4.1. *The computation of the PRE from [IK00] can be moved to* NC^0 *by techniques noted in [IK00] itself. Using similar techniques with Construction 4.2 gives us a Public Key Encryption scheme with encryption in* NC^0 *and decryption and key generation in* $\mathsf{AC}^0[2]$*. The impossibility of decryption in* NC^0*, as noted in [AIK04], continues to hold in our setting.*

Remark 4.2. *(This was pointed out to us by Abhishek Jain.) The above PKE scheme has what are called, in the terminology of [PVW08], "message-lossy" public keys – in this case, this is simply* $\mathbf{M}$ *when sampled from* $\mathsf{OneSamp}$*, as in the proof above. Such schemes may be used, again by results from [PVW08], to construct protocols for Oblivious Transfer where the honest parties are computable in* NC^1 *and which are secure against semi-honest* NC^1 *adversaries under the same assumptions (that* $\oplus\mathsf{L}/\mathsf{poly} \not\subseteq \mathsf{NC}^1$*).*

4.1 Collision Resistant Hashing

Note that again, due to the linearity of decryption, Construction 4.2 is additively homomorphic – if c_1 and c_2 are valid encryptions of m_1 and m_2, $(c_1 \oplus c_2)$ is a valid encryption of $(m_1 \oplus m_2)$. Furthermore, the size of ciphertexts does not increase when this operation is performed. Given these properties, we can use the generic transformation from additively homomorphic encryption to collision resistance due to [IKO05], along with the observation that all operations involved in the transformation can still be performed in $\mathsf{AC}^0[2]$, to get the following.

Theorem 4.4. *Assume* $\oplus\mathsf{L}/\mathsf{poly} \not\subseteq \mathsf{NC}^1$*. Then, for any constant* $c < 1$ *and function* s *such that* $s(n) = O(n^c)$*, there exists an* $\mathsf{AC}^0[2]$*-CRHF against* NC^1 *with compression* s*.*

5 Cryptography Without Assumptions

In this section, we present some constructions of primitives unconditionally secure against AC^0 adversaries that are computable in AC^0. This is almost the largest complexity class (after AC^0 with MOD gates) for which we can hope to get such unconditional results owing to a lack of better lower bounds. In this section, we present constructions of PRGs with arbitrary polynomial stretch, Weak PRFs, Symmetric Key Encryption, and Collision Resistant Hash Functions. We end with a candidate for Public Key Encryption against AC^0 that we are unable to prove secure, but also do not have an attack against.

5.1 High-Stretch Pseudo-Random Generators

We present here a construction of Pseudo-Random Generators against AC^0 with arbitrary polynomial stretch that can be computed in AC^0. In fact, the same techniques can be used to obtain constant stretch generators computable in NC^0— see Remark 5.1 for details..

The key idea behind the construction is the following: [Bra10] implies that for any constant ϵ, an n^ϵ-wise independent distribution will fool AC^0 circuits of arbitrary constant depth. So, being able to sample such distributions in AC^0 suffices to construct good PRGs. As shown in Section 2.3, if $\mathbf{H}$ is the parity-check matrix of a code with large distance d, then the distribution $\mathbf{Hx}$ is d-wise independent for $\mathbf{x}$ being a uniformly random vector (by Lemma 2.10). Further, as was also shown in Section 2.3, even for rather large d there are such matrices $\mathbf{H}$ that are sparse, allowing us to compute the product $\mathbf{Hx}$ in AC^0.

Construction 5.1 AC^0-PRG against AC^0

For any polynomial l, we define the family $\mathcal{F}^l = \left\{ f^l_\lambda : \{0,1\}^\lambda \to \{0,1\}^{l(\lambda)} \right\}$ as follows.
Lemma 2.9 implies for large λ, there is an $[l(\lambda), (l(\lambda)-\lambda)]_2$ linear code with minimum distance at least $\frac{\lambda}{\log^3(\lambda)}$ whose parity check matrix has $\log^2(\lambda)$ non-zero entries in each row. Denote this parity check matrix by $\mathbf{H}_{l,\lambda}$. The dimensions of $\mathbf{H}_{l,\lambda}$ are $l(\lambda) \times \lambda$.

$$f^l_\lambda(\mathbf{x}) = \mathbf{H}_{l,\lambda}\mathbf{x}$$

Theorem 5.1 (PRGs against AC^0)**.** *For any polynomial l, the family $\mathcal{F}^l$ from Construction 5.1 is an AC^0-PRG with multiplicative stretch $\left(\frac{l(\lambda)}{\lambda}\right)$.*

Proof. For any l, the most that needs to be done to compute $f^l_\lambda(x)$ is computing the product $\mathbf{H}_{l,\lambda}\mathbf{x}$. We know that each row of $\mathbf{H}_{l,\lambda}$ contains at most $\log^2(\lambda)$ non-zero entries. Hence, by Lemma 2.5, $\mathcal{F}^l$ is in AC^0. The multiplicative stretch being $\left(\frac{l(\lambda)}{\lambda}\right)$ is also easily verified.

For pseudo-randomness, we observe that the product $\mathbf{H}_{l,\lambda}\mathbf{x}$ is $\Omega\left(\frac{\lambda}{\log^3(\lambda)}\right)$-wise independent, by Lemma 2.10. And hence, Theorem 2.2 implies that this distribution is pseudo-random to adversaries in AC^0. □

Remark 5.1. *For any constant c, NC^0-PRGs against AC^0 that provide a multiplicative stretch of k^{c-1} can be obtained by noting that if we strengthen the hypothesis of Lemma 2.7 by setting $d = 8c$, $\alpha = 6c$, and $\gamma = \frac{\omega(\mathsf{polylog}(k))}{n}$, the lemma still holds, except that the negligible function is now replaced with an inverse polynomial function, which is still smaller than 1 for large enough k. As all we need for Construction 5.1 is that a matrix of the necessary form exists, this suffices to construct a PRG that is computable in NC^0 using the same techniques.*

5.2 Weak Pseudo-Random Functions

In this section, we describe our construction of Weak Pseudo-Random Functions against AC^0 computable in AC^0 (Construction 5.2). Roughly, we know that for a random sparse matrix $\mathbf{H}$, $(\mathbf{H}, \mathbf{Hk})$ is indistinguishable from $(\mathbf{H}, \mathbf{r})$ where $\mathbf{r}$ and $\mathbf{k}$ are chosen uniformly at random. We choose the key of the PRF to be a random vector $\mathbf{k}$. On an input $\mathbf{x}$, the strategy is to use the input $\mathbf{x}$ to generate a sparse vector $\mathbf{y}$ and then take the inner product $\langle \mathbf{y}, \mathbf{k} \rangle$.

Theorem 5.2 (PRFs against AC^0)**.** *The pair of families $(\mathcal{K}eyGen, \mathcal{E}val)$ defined in Construction 5.2 is a Weak AC^0-PRF against AC^0.*

The intuitive reason one would think Construction 5.2 might be pseudo-random is that a collection of random function values from a randomly sampled key seems to contain the same information as $(\mathbf{H}, \mathbf{Hk})$ where $\mathbf{k}$ is sampled uniformly at random and $\mathbf{H}$ is sampled using SMSamp: a matrix with sparse rows. We know from Lemma 2.9 that except with negligible probability, $\mathbf{H}$ is going to be the parity check matrix of a code with large distance, and when it is, the arguments from Section 5.1 show that $(\mathbf{H}, \mathbf{Hk})$ is indistinguishable from $(\mathbf{H}, \mathbf{r})$, where $\mathbf{r}$ is sampled uniformly at random.

Construction 5.2 AC^0-PRF against AC^0

Let $\mathcal{I}^t = \left\{ip^t_\lambda\right\}$ be the inner product family with threshold promise t described in Lemma 2.5. Define families $\mathcal{K}eyGen = \{\mathsf{KeyGen}_\lambda\}$ and $\mathcal{E}val = \{\mathsf{Eval}_\lambda\}$ as follows.

KeyGen_λ:

1. Output a random vector $\mathbf{k} \leftarrow \{0,1\}^{\lfloor\lambda\rfloor_2}$.

$\mathsf{Eval}_\lambda(\mathbf{k}, \mathbf{r})$:

1. Compute $\mathbf{y} \leftarrow \mathsf{SRSamp}(\lfloor\lambda\rfloor_2, \log^2(\lfloor\lambda\rfloor_2), \mathbf{r})$.
2. Output $ip^{\log^2(\lambda)}_{\lfloor\lambda\rfloor_2}(\mathbf{k}, \mathbf{y})$.

The only fact that prevents this from functioning as a proof is that what the adversary gets is not $(\mathbf{y}, \langle\mathbf{y}, \mathbf{k}\rangle)$ where $\mathbf{y}$ is an output of SRSamp, but rather $(\mathbf{r}, \langle\mathbf{y}, \mathbf{k}\rangle)$, where $\mathbf{r}$ is randomness that when used in SRSamp gives $\mathbf{y}$. One way to show that this is still pseudo-random is to reduce the case where the input is $(\mathbf{y}, \langle\mathbf{y}, \mathbf{x}\rangle)$ to the case where the input is $(\mathbf{r}, \langle\mathbf{y}, \mathbf{x}\rangle)$ using an AC^0-reduction. To do this, one would need an AC^0 circuit that would, given $\mathbf{y}$, sample from a distribution close to the uniform distribution over $\mathbf{r}$'s that cause SRSamp to output $\mathbf{y}$ when used as randomness. We implement this proof strategy below.

To prove Theorem 5.2, we first show that there exists an AC^0 circuit that would, given $\mathbf{y}$, sample from a distribution close to the uniform distribution over $\mathbf{r}$'s that cause SRSamp to output $\mathbf{y}$ when used as randomness.

Lemma 5.3 (Inverting SRSamp)**.** *For any constant c, there exists another constant c' and a polynomial s such that for any k that is a power of 2 and $d = \Theta(\log^c(k))$, there is a (randomized) circuit $C^{inv}_{k,d}$ of size at most $s(k)$ and depth c', and a negligible function ν such that for any $\mathbf{y} \in \{0,1\}^k$ that has exactly d non-zero entries,*

$$\Delta(C^{inv}_{k,d}(y), U_{R_y}) \leq \nu(k)$$

where $R_y = \{\mathbf{r} \mid \mathsf{SRSamp}(k, d, \mathbf{r}) = \mathbf{y}\}$.

Proof. For any k that is a power of 2 and any d, given input $\mathbf{y}$ of length k and with exactly d non-zero entries, consider the following inverting procedure:

1. Let $z = (z_1, \ldots, z_d)$, where the z_j's are the indices (written as strings in $\{0,1\}^{\log(k)}$) of non-zero entries in y.
2. Permute the elements of z at random. Let z' be the result of this operation.
3. Generate d sets $\{v_i = (v_{i1}, \ldots, v_{id})\}_{i\in[d]}$, where each $v_{ij} \in \{0,1\}^{\log(k)}$. If there is no $i \in [d]$ such that there are no collisions among the elements of v_i, output $v = (v_1, \ldots, v_d)$.
4. Otherwise, replace the first $\mathbf{r}_i$ that has no collisions with z' and output as the inverse $v = (v_1, \ldots, v_{i-1}, z', v_{i+1}, \ldots, v_d)$.

We first describe why this samples from a favourable distribution and then how to sample from a distribution negligibly close to this in AC^0. If none of the $\mathbf{r}_i$'s are free of collisions, then the

output of this procedure when used as randomness will actually not cause SRSamp to output y. But the probability that this happens is at most $\left(\frac{d^2}{k}\right)^d$ (see proof of Lemma 2.6).

Conditioned on the above not happening, we claim that the distribution of outputs is uniform over R_y. It is clear by the definition of SRSamp that for any v' that is output by this procedure, $v \in R_y$, and also that any $v \in R_y$ is output by this procedure with non-zero probability.

Consider any $v' = (v'_1, \ldots, v'_d) \in R_y$. Let z be as described in the above procedure. The fact that v' is in R_y implies that there is some $i \in [d]$ such that v'_i is some permutation of z and for all $j < i$, $\mathbf{r}'_j$ contains a collision. The probability that v' is output by the above procedure is the probability that all of the following three events happen:

1. $v'_1, \ldots, v'_{i-1}, v'_{i+1}, \ldots, v'_d$ are sampled in step 3 of the procedure when a random v is sampled.
2. v_i is sampled to be free of collisions.
3. z' is equal to v'_i.

Note that all three of these events are independent, and their probabilities do not depend on the value of i or that of any of the v'_js. Hence, conditioned on outputting a $v' \in R_y$, the above procedure outputs a uniformly random element from R_y, and so the distance of its output distribution from the uniform distribution over R_y is at most the probability that it fails, which is $\left(\frac{d^2}{k}\right)^d$, which is negligible when $d = \Theta(\log^c(k))$ for some constant c.

Now we explain why each of the steps above can be performed by a constant depth circuit in the case where $d = O(\log^c(k))$ for some c, with a negligible probability of failure. Recall that the input is a string y of length k that has exactly d non-zero entries.

1. To compute z, let $Ham_{j-1,l-1}$ be the constant depth circuit that computes that takes inputs of length $(l-1)$ and checks whether the Hamming weight of its input is $(j-1)$. By Theorem 2.3, such a circuit exists for $j \leq d = O(\log^c(k))$. Note that out of all $l \in [d]$, exactly one of $(y_l \wedge Ham_{j-1,l-1}(y_1 \ldots y_{l-1}))$ is true. So z_j can be computed as follows:

$$z_j = \bigvee_{l \in [d]} [l \wedge (y_l \wedge Ham_{j-1,l-1}(y_1 \ldots y_{l-1}))]$$

   where by $(l \wedge \phi)$, we mean the $\log(k)$-bit string whose ith bit is equal to the ith bit of l if ϕ is true, and is 0 otherwise.

2. While it is not clear how to sample uniformly from the set of all permutations of a given tuple of elements in constant depth, it turns out to be possible to sample from a distribution sufficiently close to this when there are only $O(\log^c(k))$ elements that are all distinct.

   - Choose d numbers $p_1, \ldots, p_d \in [k]$. The probability that two of them are equal is at most $\left(\frac{d^2}{k}\right)$.
   - Repeat this process at most d times, till a set of $p_1, \ldots, p_d$ are chosen without collisions. If no such set is found, output $z' = z$ itself.
   - Note that all this - checking whether there are any collisions and picking the first set without any - can be done in parallel in constant depth, and the probability that the above step fails is at most $\left(\frac{d^2}{k}\right)^d$.

- Compute the string $s \in \{0,1\}^k$ where $s_i = 1$ iff there is a j such that $p_j = i$. Also compute $o \in \{0,1\}^{k \log(k)}$ where $o_i = j$ if $p_j = i$, and $o_j = 0^{\log(k)}$ otherwise.
- Compute the permuted string z' as:

$$z'_j = \bigvee_{l \in [d]} [o_l \wedge (s_l \wedge Ham_{j-1,l-1}(s_1 \dots s_{l-1}))]$$

3. As noted in the point above, steps 3 and 4 of the procedure, which involve finding and replacing with z' a set that has no collisions, can also be done in constant depth.

The above randomized circuit computes the same distribution as the inversion procedure described above except with probability at most $\left(\frac{d^2}{k}\right)^d$. So the distance of the distribution produced from the uniform distribution over R_y is at most $O\left(\left(\frac{d^2}{k}\right)^d\right) = O\left(\left(\frac{\log^{2c}(k)}{k}\right)^{\log^c(k)}\right)$, which is negligible. □

Proof of Theorem 5.2. $\mathcal{K}eyGen$ and $\mathcal{E}val$ are both in AC^0 because KeyGen_λ simply outputs random strings, and Eval_λ first calls SRSamp, which can be done in constant depth and outputs a vector with at most $\log^2(\lambda)$ non-zero entries, and then computes inner product of this sparse vector with another vector using $ip^{\log^2(\lambda)}_{\lfloor\lambda\rfloor_2}$, which can again be done in constant depth as noted in Lemma 2.5. Non-triviality is also easily seen to be satisfied.

Consider any AC^0 family $\mathcal{G} = \left\{g_\lambda : \{0,1\}^{(\lambda+1)n(\lambda)} \to \{0,1\}\right\}$, where n is some polynomial. To simplify presentation, we prove pseudo-randomness when λ is a power of 2; the other case may be proven very similarly.

We show that any pair of consecutive distributions among the following are indistinguishable by AC^0 adversaries for large enough λ. Below, $k, \mathbf{x}_i \leftarrow \{0,1\}^\lambda$, $y_i \leftarrow SpR_{\lambda,\log^2(\lambda)}$, $w_i \leftarrow R_{y_i}$, $z_i \leftarrow C^{inv}_{\lambda,\log^2(\lambda)}(y_i)$, $\mathbf{r}_i \leftarrow \{0,1\}^{\lambda - \log^5(\lambda)}$, and $b_i \leftarrow \{0,1\}$.

D_1: $\{(\mathbf{x}_i, \mathsf{Eval}_\lambda(k, \mathbf{x}_i))\}_{i \in [n(\lambda)]}$

D_2: $\{(w_i || \mathbf{r}_i, \mathsf{Eval}_\lambda(k, w_i))\}_{i \in [n(\lambda)]}$

D_3: $\{(w_i || \mathbf{r}_i, \langle k, y_i \rangle)\}_{i \in [n(\lambda)]}$

D_4: $\{(z_i || \mathbf{r}_i, \langle k, y_i \rangle)\}_{i \in [n(\lambda)]}$

D_5: $\{(z_i || \mathbf{r}_i, b_i)\}_{i \in [n(\lambda)]}$

D_6: $\{(w_i || \mathbf{r}_i, b_i)\}_{i \in [n(\lambda)]}$

D_7: $\{(\mathbf{x}_i, b_i)\}_{i \in [n(\lambda)]}$

D_1 and D_2 are statistically close because w_i is uniformly distributed over strings that do not cause $\mathsf{SRSamp}(\lambda, \log^2(\lambda), w_i)$ to fail, which is shown to be a negligible fraction in Lemma 2.6. D_3 is simply a re-writing of D_2 because of how Eval_λ works.

D_3 and D_4 are statistically close by Lemma 5.3. D_4 and D_5 are indistinguishable by AC^0 adversaries because Lemma 2.11 says that $\{(y_i, \langle k, y_i \rangle)\}$ and $\{(y_i, b_i)\}$ are indistinguishable by AC^0,

and D_4 and D_5 can be sampled using samples from these two distributions, respectively, in AC^0, as shown in Lemma 5.3.

D_5 and D_6 are statistically close by Lemma 5.3. D_6 and D_7 are statistically close for the same reason as D_1 and D_2.

D_1 is the distribution of random evaluations of Construction 5.2, and D_7 is the distribution of random evaluations of a random function. So we have shown that g_λ cannot distinguish between these, which proves the pseudo-randomness of Construction 5.2 against AC^0 adversaries. □

Remark 5.2. *Note that this construction cannot be a strong PRF for any reasonable definition of that notion. If the adversary is able to select the inputs for function evaluations, it could easily distinguish a function from Construction 5.2 from a random function by choosing* $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$ *such that* $\mathsf{SRSamp}(\lambda, \log^2 \lambda, \mathbf{x}_1) + \mathsf{SRSamp}(\lambda, \log^2 \lambda, \mathbf{x}_2) = \mathsf{SRSamp}(\lambda, \log^2 \lambda, \mathbf{x}_3)$ *and then checking if* $f(\mathbf{x}_1) + f(\mathbf{x}_2) = f(\mathbf{x}_3)$.

Construction 5.2 of Weak PRFs achieves only quasi-polynomial security — that is, there is no guarantee that some AC^0 adversary may not have an inverse quasi-polynomial advantage in distinguishing between the PRF and a random function. Due to the seminal work of Linial-Mansour-Nisan [LMN93] and subsequent improvements in [Tal14], we know that this barrier is inherent and we cannot hope for exponential security — see Observation 5.4.

Observation 5.4. *For any set of Boolean functions, all of which are computable by circuits of size* m *and depth* d*, there is a circuit of size* $poly(m)$ *and depth* $O(d)$ *which can distinguish a random function from this set from a random function with an advantage of* $\frac{1}{m^{\Omega(\log^{d-1} m)}}$ *given only function evaluations on randomly chosen inputs.*

Proof. By [LMN93], any Boolean function f computed by a circuit of size m and depth d has over a constant fraction of its Fourier mass on coefficients of degree $O(\log^{d-1} m)$. So, there is a Fourier Coefficient of degree $\leq O(\log^{d-1} m)$ with over $\frac{1}{m^{\Omega(\log^{d-1} m)}}$ Fourier mass.

This gives us a simple AC^0 attack - try to guess this Fourier coefficient and estimate the correlation using two samples - $\mathbf{r}_1, \mathbf{r}_2 \leftarrow \{0,1\}^\lambda$.

$A((\mathbf{r}_1, f(\mathbf{r}_1)), (\mathbf{r}_2, f(\mathbf{r}_2))$:

1. Guess a Fourier coefficient s of degree $\leq O(\log^{d-1} m)$
2. If $(\chi_s(\mathbf{r}_1) \oplus f(\mathbf{r}_1)) = (\chi_s(\mathbf{r}_2) \oplus f(\mathbf{r}_2))$ Output 1 else 0.

For a random function, this adversary would output 1 exactly with probability 1/2. On the other hand, let $\widehat{f}(s) = \mathbf{E}[f(x) \oplus \chi_s(x)]$ be the Fourier coefficient. Then the probability of two samples being equal would be $(\frac{1-\widehat{f}(s)}{2})^2 + (\frac{1+\widehat{f}(s)}{2})^2 = \frac{1}{2} + \frac{\widehat{f}(s)^2}{2}$. So, in expectation over s, the distinguishing advantage of this adversary would be $\mathbf{E}_s\left[\widehat{f}(s)^2\right] = \Omega(\frac{1}{m^{\Omega(\log^{d-1} m)}})$. □

5.3 Symmetric Key Encryption

In this section, we present a Symmetric Key Encryption scheme against AC^0 computable in AC^0, which is also additively homomorphic – a property that shall be useful in constructing Collision Resistant Hash Functions later on.

In Section 5.2, we saw a construction of Weak PRFs. And Weak PRFs give us Symmetric Key Encryption generically (where $\mathsf{Enc}(\mathbf{k}, b) = (\mathbf{r}, \mathsf{PRF}(\mathbf{k}, \mathbf{r}) \oplus b)$). For the Weak PRF construction from

Section 5.2, this implied scheme also happens to be additively homomorphic. But it has the issue that the last bit of the ciphertext is an almost unbiased bit and hence it is not feasible to do more than $\mathsf{polylog}(\lambda)$ homomorphic evaluations on collections of ciphertexts in AC^0. So, we construct a different Symmetric Key Encryption scheme that does not suffer from this drawback and is still additively homomorphic. Then we will use this scheme to construct Collision Resistant Hash Functions. This scheme is described in Construction 5.3. In this scheme we choose the ciphertext by performing rejection sampling in parallel. For encrypting a bit b, we sample a ciphertext $\mathbf{c}$ such that $\mathbf{c}$ is sparse and $\langle \mathbf{c}, \mathbf{k} \rangle = b$. This ensures that the we have an additively homomorphic scheme where all the bits are sparse.

Construction 5.3 AC^0-Symmetric Key Encryption against AC^0

Let $\mathcal{I}^t = \left\{ip_\lambda^t\right\}$ be the inner product family with threshold promise t described in Lemma 2.5. Define families $\mathcal{K}eyGen = \{\mathsf{KeyGen}_\lambda\}$, $\mathcal{E}nc = \{\mathsf{Enc}_\lambda\}$, and $\mathcal{D}ec = \{\mathsf{Dec}_\lambda\}$ as below.

KeyGen_λ:

1. Output $\mathbf{k} \leftarrow \{0,1\}^{\lfloor \lambda \rfloor_2}$.

$\mathsf{Enc}_\lambda(\mathbf{k}, b)$:

1. For $i \in [\lambda]$, sample $\mathbf{c}_i \leftarrow \mathsf{SRSamp}(\lfloor \lambda \rfloor_2, \log^2(\lfloor \lambda \rfloor_2))$.
2. Choose the first i such that $\langle \mathbf{c}_i, \mathbf{k} \rangle = b$.
3. If such an i exists, output $\mathbf{c}_i$, else output $0^{\lfloor \lambda \rfloor_2}$.

$\mathsf{Dec}_\lambda(\mathbf{k}, \mathbf{c})$:

1. Output $ip_{\lfloor \lambda \rfloor_2}^{\log^2(\lambda)}(\mathbf{k}, \mathbf{c})$.

Theorem 5.5 (Symmetric Encryption Against AC^0)**.** *The tuple of families $(\mathcal{K}eyGen, \mathcal{E}nc, \mathcal{D}ec)$ defined in Construction 5.3 is an AC^0-Symmetric-Key Encryption Scheme against AC^0.*

The key idea behind the proof is showing that for most keys $\mathbf{k}$, the distribution of a uniformly random bit along with its encryption, that is,

$$D_1 = \{(b, \mathsf{Enc}_\lambda(\mathbf{k}, b)) \mid b \leftarrow \{0,1\}\}$$

is statistically close to the distribution of a random sparse vector along with its inner product with $\mathbf{k}$, that is,

$$D_2 = \left\{(\langle \mathbf{r}, \mathbf{k} \rangle, \mathbf{r}) \mid \mathbf{r} \leftarrow \mathsf{SRSamp}(\lambda, \log^2 \lambda)\right\}$$

The second distribution is similar to the one that came up in the security proof of the weak PRF construction earlier, where we effectively showed that we can replace $\langle \mathbf{r}, \mathbf{k} \rangle$ with an independent random bit without being caught by AC^0 adversaries. We now prove the statistical closeness and complete the above idea to a proof.

We begin by establishing some properties of the encryption scheme. This first lemma says that most keys generated by KeyGen_λ are balanced. It follows easily from the Chernoff bound when applied to the Hamming weight of a random string of length λ.

Lemma 5.6. *For the family $KeyGen = \mathsf{KeyGen}_\lambda$ defined in Construction 5.3, for any λ,*

$$\Pr_{\mathbf{k}\leftarrow\mathsf{KeyGen}_\lambda}\left[\|\mathbf{k}\| \in \left(\frac{\lambda}{2} - \sqrt{\lambda}\log^2\lambda, \frac{\lambda}{2} + \sqrt{\lambda}\log^2\lambda\right)\right] > 1 - \mathsf{negl}(\lambda)$$

The following lemma states that for a vector $\mathbf{k}$ that is almost balanced, its inner product with a random sparse vector is almost unbiased.

Lemma 5.7. *For $\mathbf{k}$ of length λ such that $\|\mathbf{k}\| \in \left(\frac{\lambda}{2} - \sqrt{\lambda}\log^2\lambda, \frac{\lambda}{2} + \sqrt{\lambda}\log^2\lambda\right)$,*

$$\left|\Pr_{\mathbf{r}\leftarrow\mathsf{SRSamp}(\lambda,\log^2(\lambda))}[\langle\mathbf{r},\mathbf{k}\rangle = 0] - \Pr_{\mathbf{r}\leftarrow\mathsf{SRSamp}(\lambda,\log^2(\lambda))}[\langle\mathbf{r},\mathbf{k}\rangle = 1]\right| < \mathsf{negl}(\lambda)$$

Proof. Let $n = \|\mathbf{k}\|$ and $m = \lambda - \|\mathbf{k}\|$. Below, $\mathbf{r}$ is sampled using $\mathsf{SRSamp}(\lambda, \log^2(\lambda))$.

In the inner product $\langle\mathbf{r},\mathbf{k}\rangle$, we start with an almost balanced $\mathbf{k}$, the randomness $\mathbf{r}$ is used to choose $\log^2\lambda$ coordinates of the key and then we xor these. We want to show that his output is almost unbiased.

The number of possibilities for $\langle\mathbf{r},\mathbf{k}\rangle = 0$ is $\sum_{i \text{ is even}}\binom{n}{i}\binom{m}{d-i}$. That is number of ones chosen is even. This gives us:

$$\Pr_{\mathbf{r}}[\langle\mathbf{r},\mathbf{k}\rangle = 0] - \Pr_{\mathbf{r}}[\langle\mathbf{r},\mathbf{k}\rangle = 1] = \frac{\sum_{i+j=d}(-1)^i\binom{n}{i}\binom{m}{j}}{\binom{m+n}{d}}$$

We want to show that this is negligible. The way we do this is by considering the generating function of the term and interpreting it differently. So, consider the generating function of $\sum_{i+j=d}(-1)^i\binom{n}{i}\binom{m}{j}$. It is $(1-x)^n(1+x)^m$ with the coefficient of x^d being the required value. Without loss of generality say $m \geq n$. We get an identity by rewriting $(1-x)^m(1+x)^n = (1-x^2)^n(1+x)^{m-n}$ and then looking at the coefficient of x^d. It is - $\sum_{i\leq d/2}(-1)^i\binom{n}{i}\binom{m-n}{d-2i}$. So we get the identity:

$$\sum_{i+j=d}(-1)^i\binom{n}{i}\binom{m}{j} = \sum_{i\leq d/2}(-1)^i\binom{n}{i}\binom{m-n}{d-2i}$$

We know that $\sum_{i\leq d/2}(-1)^i\binom{n}{i}\binom{m-n}{d-2i} < d\binom{n}{d/2}\binom{m-n}{d}$. We want to show that this is negligible compared to $\binom{m+n}{d}$.

$$\frac{d\binom{n}{d/2}\binom{m-n}{d}}{\binom{m+n}{d}} < \frac{d\left(\frac{\sqrt{n}e}{d}\right)^d\left(\frac{ne}{d/2}\right)^{d/2}}{\left(\frac{2n}{d}\right)^d} < \left(\frac{e^{3/2}}{d}\right)^d$$

which is negligible. □

The following lemma states that the rejection sampling performed in Enc_λ fails only with negligible probability for balanced keys. It follows from the statement of Lemma 5.7 that each $\langle \mathbf{r}_i, \mathbf{k} \rangle$ is an independent coin flip with negligible bias. Hence the probability that none of them would equal b is exponentially small.

Lemma 5.8. *For every $b \in \{0,1\}$ and any $\mathbf{k}$ of length λ such that $\|\mathbf{k}\| \in \left(\frac{\lambda}{2} - \sqrt{\lambda}\log^2\lambda, \frac{\lambda}{2} + \sqrt{\lambda}\log^2\lambda\right)$,*

$$\Pr_{\mathbf{r}_1,\ldots,\mathbf{r}_\lambda \leftarrow \mathsf{SRSamp}(\lambda, \log^2(\lambda))} [\forall i : \langle \mathbf{r}_i, \mathbf{k} \rangle \neq b] < \mathsf{negl}(\lambda)$$

The following lemma states that for keys that are almost balanced in Hamming weight, the distribution of random bits with their encryptions under a key is similar to the distribution of inner products of random sparse vectors with the key with the sparse vectors.

Lemma 5.9. *For any $\mathbf{k} \in \{0,1\}^\lambda$ such that $\|\mathbf{k}\| \in \left(\frac{\lambda}{2} - \sqrt{\lambda}\log^2\lambda, \frac{\lambda}{2} + \sqrt{\lambda}\log^2\lambda\right)$, define the following distributions:*

- $D_1 = \{(b, \mathsf{Enc}_\lambda(b, \mathbf{k})) \mid b \leftarrow \{0,1\}\}$
- $D_2 = \left\{(\langle \mathbf{r}, \mathbf{k} \rangle, \mathbf{r}) \mid \mathbf{r} \leftarrow \mathsf{SRSamp}(\lambda, \log^2\lambda)\right\}$

Then,

$$\Delta(D_1, D_2) < \mathsf{negl}(\lambda)$$

Proof. It follows from the definition of Enc_λ and Lemma 5.8 that the when conditioned on $\langle \mathbf{r}, \mathbf{k} \rangle = b$, the distributions of $\mathbf{r}$ and $\mathsf{Enc}_\lambda(\mathbf{k}, b)$ are negligibly close. According to Lemma 5.7, $\langle \mathbf{r}, \mathbf{k} \rangle$ is almost unbiased, and its distribution is negligibly close to that of b. These two facts together imply that D_1 and D_2 are negligibly close. □

The semantic security definition is presented as a game in Figure 1. Here we choose a non-adaptive notion of security because it is a bit-encryption scheme and for the given adversary model considered - AC^0 adversaries, adaptivity can only be very limited because it increases depth. We also define two other games as shown in the same figure. The advantage of the adversary in each game is: $\left|\Pr[b' = b] - \frac{1}{2}\right|$. The following claims state that the advantage of AC^0 adversaries in all these games are comparable, a fact that will be useful in proving Theorem 5.5.

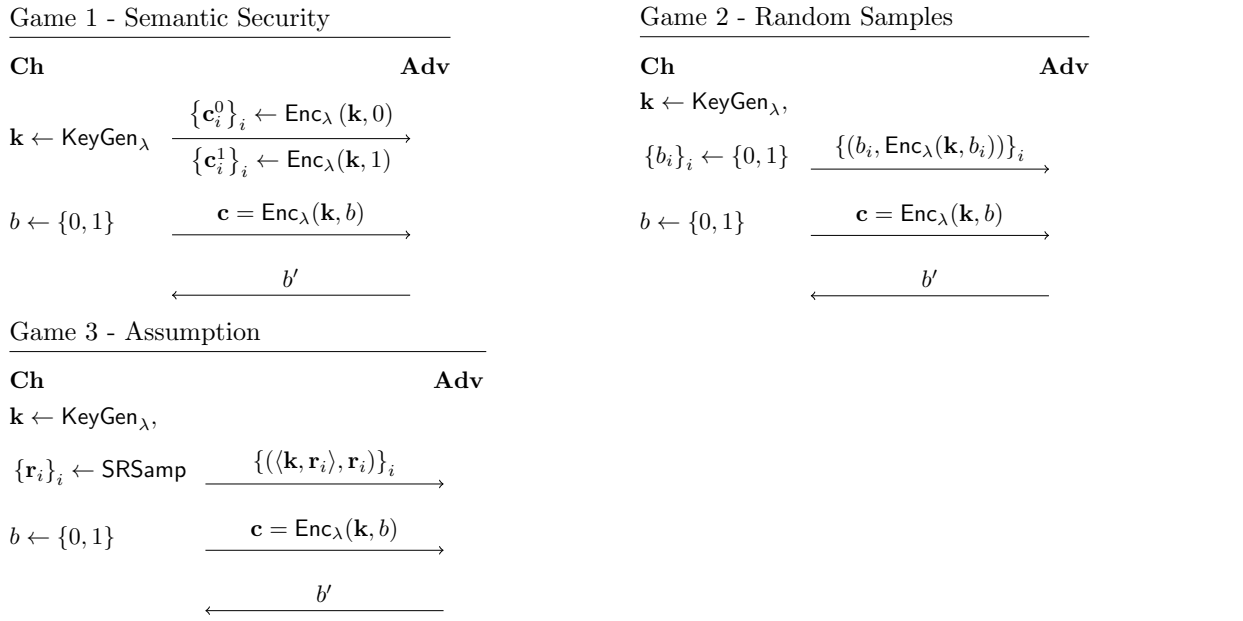

Figure 1: Security Games

Claim 5.1. *For any adversary $\mathcal{A} \in \mathsf{AC}^0$ with advantage $\epsilon(\lambda)$ in the Game 1, we can construct an adversary $\mathcal{B} \in \mathsf{AC}^0$ that has advantage $\left(\epsilon(\lambda) - 2^{-\lambda}(n_0(\lambda) + n_1(\lambda))\right)$ in Game 2.*

Proof. The adversary B takes $(n_1(\lambda) + n_2(\lambda))\lambda$ samples and then selects $n_0(\lambda)$ samples with $b = 0$ and $n_1(\lambda)$ samples with $b = 1$ and feeds them to A along with the challenge b. The way it does this is it for every sample required, it takes λ samples from the set of samples it has and selects the first one that has the required bit b encrypted. It can do this for all the required bits in parallel and constant depth. The only case when B diverges from A is when all of some set of λ samples are of $\overline{b}$. Happens with probability 2^λ. We take a union bound over the $(n_0(\lambda) + n_1(\lambda))$ samples. □

Claim 5.2. *Any adversary $\mathcal{A} \in \mathsf{AC}^0$ has comparable advantage in Game 2 and Game 3.*

Proof. The input distributions - $\{(b_i, \mathsf{Enc}_\lambda(\mathbf{k}, b_i))\}_i$ and $\{(\langle \mathbf{k}, \mathbf{r}_i \rangle, \mathbf{r}_i)\}_i$ have negligible statistical distance. Because from Lemma 5.9 we know that for balanced keys the distributions $\{(b_i, \mathsf{Enc}_\lambda(\mathbf{k}, b_i))\}$ and $\{(\langle \mathbf{k}, \mathbf{r}_i \rangle, \mathbf{r}_i)\}$ have negligible statistical distance. Hence the input distributions which are $m(\lambda)$ independent copies of the distributions also have negligible statistical distance. We also know from Lemma 5.6 that the key is balanced except with negligible probability. And hence any adversary cannot have a non-negligible difference in the advantage. □

Proof of Theorem 5.5. It is easy to see that $\mathcal{KeyGen} \in \mathsf{AC}^0$. Enc_λ can be computed in constant depth because SRSamp can, the inner product in step 2 is with an output of SRSamp, which is sparse, and the first i such that $\langle \mathbf{r}_i, \mathbf{k} \rangle = b$ can also be found in constant depth. Dec_λ can be computed in constant depth as stated in Lemma 2.5. So computability is satisfied. Non-triviality is also easily seen to be satisfied.

Correctness follows from Lemma 5.8, which implies that if the key generated by KeyGen_λ is balanced, the ciphertext is generated correctly by Enc_λ except with negligible probability. Lemma 5.6 says that the key is unbalanced only with negligible probability. Also, for any $\mathbf{k}$ and any

ciphertext $\mathbf{c}$ generated by Enc_λ, $\mathsf{Dec}_\lambda(\mathbf{k}, \mathbf{c})$ is actually equal to $\langle \mathbf{k}, \mathbf{c} \rangle$ because the are outputs of Enc_λ are always sparse. So except with negligible probability over the randomness in the generation of keys and encryption, decryption is correct.

Due to Claims 5.1 and 5.2, it is sufficient to prove that any AC^0 adversary has negligible advantage in Game 3 to prove semantic security. We prove this for the case where λ is a power of 2; the proof for the other case then follows immediately from the observation that $\lfloor \lambda \rfloor_2 = \Theta(\lambda)$.

Essentially in the form of samples, the adversary gets $(\mathbf{R}, \mathbf{Rk})$ where $\mathbf{R}$ is sampled using $\mathsf{SMSamp}(m(\lambda), \lambda, \log^2 \lambda)$ and $\mathbf{k} \leftarrow \{0,1\}^\lambda$. Let the challenge be $(\mathbf{r}^\star, b)$, where $b \leftarrow \{0,1\}$ and $\mathbf{r}^\star \leftarrow \mathsf{Enc}_\lambda(\mathbf{k}, b)$. Consider the matrix $\begin{pmatrix} \mathbf{R} & \mathbf{Rk} \\ \mathbf{r}^\star & b \end{pmatrix}$. This matrix is indistinguishable from $\begin{pmatrix} \mathbf{R} & \mathbf{r} \\ \mathbf{r}^\star & \mathbf{r}' \end{pmatrix}$ where $\mathbf{r} \leftarrow \{0,1\}^\lambda$, $\mathbf{r}' \leftarrow \{0,1\}$, and $\mathbf{r}^\star \leftarrow \mathsf{SRSamp}(\lambda, \log^2(\lambda))$ for AC^0 circuits from Lemmas 2.11 and 5.9. Any adversary that has non-negligible advantage in Game 3 can be used to distinguish these two distributions. Hence no adversary has non-negligible advantage in Game 3. □

We further claim that the adversary cannot distinguish between encryptions of two messages of its own choosing. We formalize this using the security game in Theorem 5.10.

Theorem 5.10. *No AC^0 adversary $(\mathcal{A}_1, \mathcal{A}_2)$ has non-negligible advantage against the Symmetric Key Encryption scheme from Construction 5.3 in the Multibit Semantic Security game given below.*

1. $\mathbf{k} \leftarrow \mathsf{KeyGen}_\lambda$
2. $(\mathbf{m}_0, \mathbf{m}_1, \mathsf{state}) \leftarrow \mathcal{A}_1$ *where* $|\mathbf{m}_0| = |\mathbf{m}_1| = \mathsf{poly}(\lambda)$.
3. $b' = \mathcal{A}_2(\mathsf{Enc}_\lambda(\mathbf{k}, \mathbf{m}_b), \mathsf{state})$ *where* $b \leftarrow \{0,1\}$.
4. *Adversary wins if* $b = b'$.

Proof. The way we prove this is by a sequence of hybrids. Consider $\mathbf{m}'_i$ to be the message obtained by concatenating the first i bits of $\mathbf{m}_0$ and the last $n - i$ bits of $\mathbf{m}_1$.

$$\mathbf{m}'_i(x) = \begin{cases} \mathbf{m}_0(x) & \text{if } x \leq i \\ \mathbf{m}_1(x) & \text{if } x > i \end{cases}$$

In hybrid i, the adversary has to distinguish the $\mathsf{Enc}(\mathbf{k}, \mathbf{m}'_i)$ from $\mathsf{Enc}(\mathbf{k}, \mathbf{m}'_{i+1})$. The advantage the adversary has in distinguishing between $\mathsf{Enc}(\mathbf{k}, \mathbf{m}_0)$ and $\mathsf{Enc}(\mathbf{k}, \mathbf{m}_1)$ is at most the sum of advantages in each of the hybrids. We show that the adversary has negligible advantage in each of the hybrids from the fact that the messages $\mathbf{m}'_i$ and $\mathbf{m}'_{i+1}$ differ only in one bit.

If there exists an AC^0 adversary A that has a non-negligible advantage in distinguishing between $\mathsf{Enc}(\mathbf{k}, \mathbf{m}'_i)$ and $\mathsf{Enc}(\mathbf{k}, \mathbf{m}'_{i+1})$ we use the adversary to construct an adversary B that has non-negligible advantage in the Single bit semantic security game. Let $\mathbf{c}$ be the challenge. We use the encryptions $\{\mathbf{c}_i^0\}$ of 0 and $\{\mathbf{c}_i^1\}$ of 1 to construct the input -

$$\mathbf{c}^\star(j) = \begin{cases} \mathbf{c} & \text{If } j = i+1 \\ \mathsf{Enc}(\mathbf{k}, \mathbf{m}_0(j)) & \text{If } j \leq i \\ \mathsf{Enc}(\mathbf{k}, \mathbf{m}_1(j)) & \text{If } j > i+1 \end{cases}$$

$\mathbf{c}^\star$ has the same distribution as $\{\mathbf{m}'_i, \mathbf{m}'_{i+1}\}$ and hence we can use this adversary to distinguish in the one-bit semantic security game.

Hence the advantage in each of the hybrids is negligible. □

5.4 Collision Resistant Hash Functions

To construct Collision Resistant Hash Functions (CRHFs), we use the additive homomorphism of the Symmetric Key Encryption scheme constructed in Section 5.3. Each function in the family of hash functions is given by a matrix whose columns are ciphertexts from the encryption scheme, and evaluation is done by treating the input as a column vector and computing its product with this matrix (effectively computing a linear combination of ciphertexts). To find collisions, the adversary needs to come up with a vector in the kernel of this matrix. We show that constant depth circuits of polynomial size cannot do this for most such matrices. This is because the all-zero vector is a valid encryption of 0 in our encryption scheme, and as this scheme is additively homomorphic, finding a subset of ciphertexts that sum to zero is roughly the same as finding a subset of the corresponding messages that sum to 0, and this is a violation of semantic security.

Construction 5.4 AC^0-CRHFs against AC^0

Let $\mathcal{I}^t = \left\{ip^t_\lambda\right\}$ be the inner product family with threshold promise t described in Lemma 2.5. Let $(\mathcal{K}eyGen^{Enc}, \mathcal{E}nc^{Enc})$ be the SKE scheme from Construction 5.3. Let $l(\lambda) = \left\lfloor \frac{\lambda}{s(\lambda)} \right\rfloor_2$.

For any $s : \mathbb{N} \to \mathbb{N}$ such that $s(\lambda) = O(\log^c(\lambda))$ for some constant c, we define the families $\mathcal{K}eyGen^s = \{\mathsf{KeyGen}^s_\lambda\}$ and $\mathcal{E}val^s = \{\mathsf{Eval}^s_\lambda\}$ as follows.

$\mathsf{KeyGen}^s_\lambda$:

1. Sample $\mathbf{k} \leftarrow \mathsf{KeyGen}^{Enc}_{l(\lambda)}$, and $b_1, \ldots, b_\lambda \leftarrow \{0,1\}$.
2. Output $\mathbf{M} = (\mathbf{m}_1, \ldots, \mathbf{m}_\lambda)$, where $\mathbf{m}_i \leftarrow \mathsf{Enc}^{Enc}_{l(\lambda)}(\mathbf{k}, b_i)$.

$\mathsf{Eval}^s_\lambda(\mathbf{M}, \mathbf{x})$:

1. Note that $\mathbf{M} = (\mathbf{m}_1, \ldots, \mathbf{m}_\lambda)$, where each $\mathbf{m}_i$ is of length $l(\lambda)$.
2. For $j \in [l(\lambda)]$, let $r_j = (\mathbf{m}_{1j}, \ldots, \mathbf{m}_{\lambda j})$ (the jth bit of each $\mathbf{m}_i$).
3. Output $(h_1, \ldots, h_{l(\lambda)})$, where $h_j = ip^{4s(\lambda)\log^2(\lambda)}_\lambda(r_j, \mathbf{x})$.

Theorem 5.11 (CRHFs Against AC^0)**.** *For any polylogarithmic function s, the pair of families $(\mathcal{K}eyGen^s, \mathcal{E}val^s)$, from Construction 5.4 is an AC^0-CRHF with compression s.*

Proof. Throughout this proof, it will be useful to think of the hash function index $\mathbf{M}$ as a matrix whose columns are the $\mathbf{m}_i$'s, and whose rows are the r_j's. What $\mathsf{Eval}^s_\lambda(\mathbf{M}, \mathbf{x})$ effectively does now is compute the product $\mathbf{Mx}$. As in the construction, let $l(\lambda) = \left\lfloor \frac{\lambda}{s(\lambda)} \right\rfloor_2$.

We first observe that we can actually compute both the function families $\mathcal{K}eyGen^s$ and $\mathcal{E}val^s$ in AC^0. This is easy to observe for $\mathsf{KeyGen}^s_\lambda$ because of Construction 5.3 being in AC^0, and for Eval^s_λ because of Lemma 2.5. Non-triviality and compression are easily seen to be satisfied.

Observe about that since we choose $b_1, \ldots, b_\lambda$ and $\mathbf{k}$ at random, the distribution of the matrices $\mathbf{M}$ is negligibly close to that of the transpose of matrices sampled from $SpM_{\lambda, l(\lambda), \log^2(l(\lambda))}$, by Lemmas 5.6, 5.9, and 2.6. Now we use a Chernoff bound to see that every row will also have less than $\left(2\lambda \frac{\log^2(l(\lambda))}{l(\lambda)}\right) \leq 4s(\lambda)\log^2(\lambda)$ Hamming weight with all but negligible probability. So, except with negligible probability, $\mathsf{Eval}^s_\lambda(\mathbf{M}, \mathbf{x}) = \mathbf{Mx}$.

Hence, to prove security, it is sufficient to show that for an adversary in AC^0, for most such $\mathbf{M}$, it is hard to find an $\mathbf{x} \in \{0,1\}^\lambda$, $\mathbf{x} \neq 0$ such that $\mathbf{Mx} = 0$, except with negligible probability. This is because our hash function is linear with high probability and when it is, finding a collision is the same as finding a non-zero pre-image for $0^{l(\lambda)}$.

If possible, let $\mathcal{A}$ be an adversary given $\mathbf{M}$ of dimension $l(\lambda) \times \lambda$ whose columns are sampled from $\mathsf{KeyGen}^s_\lambda$, can actually find a vector in the kernel. We will use this adversary to break the semantic security of Construction 5.3. Say there is a polynomial p such that

$$\Pr_{\mathbf{M} \leftarrow \mathsf{KeyGen}^s_\lambda} [\mathbf{M} \cdot \mathcal{A}(\mathbf{M}) = 0] \geq \frac{1}{p(\lambda)}$$

We know from Theorem 5.10 that the symmetric encryption scheme is semantically secure even for chosen multi-bit messages.

From the given adversary $\mathcal{A}$ that breaks the collision resistant hash function, we construct an adversary B that breaks the multibit semantic security. B works as follows:

1. It chooses $\mathbf{m}_0 \leftarrow \{0,1\}^\lambda$ and $\mathbf{m}_1 \leftarrow \mathsf{SRSamp}(\lambda, 1)$ and sends them to the challenger
2. Upon receiving ciphertext $\mathbf{M}$ from the challenger, it computes $\mathbf{v} = \mathcal{A}(\mathbf{M})$.
3. If $\mathbf{Mv} = 0$ and $\langle \mathbf{v}, \mathbf{m}_1 \rangle \neq 0$, it sets $b' = 0$.
4. Else, it sets b' at random.

We need to show that this adversary breaks multibit semantic security with a polynomial advantage and that it can be computed in AC^0. That it can be computed in AC^0 is easy to see because SRSamp and $\mathcal{A}$ can, and the inner product in step 3 is with $\mathbf{m}_1$, which has at most one non-zero entry.

Note that by the correctness of Construction 5.3, the probability that $\mathsf{Enc}^{Enc}_{l(\lambda)}$ produces a ciphertext that does not decrypt correctly is negligible. This means that except with negligible probability, if $\mathbf{c}$ is an encryption of $\mathbf{m}$ under key $\mathbf{k}$, then $\langle \mathbf{c}, \mathbf{k} \rangle = \mathbf{m}$; in this case the construction is additively homomorphic - if $\mathbf{c}_1$ and $\mathbf{c}_2$ are encryptions of $\mathbf{m}_1$ and $\mathbf{m}_2$ under the same key, then $(\mathbf{c}_1 \oplus \mathbf{c}_2)$ is an encryption of $(\mathbf{m}_1 \oplus \mathbf{m}_2)$. Then the following arguments follow except with negligible probability.

If $b = 1$ ($\mathbf{m}_1$ was encrypted), then $\mathbf{Mv} = 0$ implies that $\langle \mathbf{m}_1, \mathbf{v} \rangle = 0$ and hence B will always output a random bit b' and hence gains or loses no advantage.

On the other hand, if $b = 0$ ($\mathbf{m}_0$ was encrypted), then the distribution of $\mathbf{M}$ is the same as that generated by $\mathsf{KeyGen}^s_\lambda$. In this case, the adversary $\mathcal{A}$ will generate a vector $\mathbf{v} \neq 0$ in the kernel of $\mathbf{M}$ with probability at least $\frac{1}{p(\lambda)}$. Conditioned on this happening, if it turns out that $\langle \mathbf{m}_1, \mathbf{v} \rangle \neq 0$, then B will guess b correctly as 0. If this does not happen, then again B guess randomly and loses nothing. So we would be done if we can show that in this case the inner product $\langle \mathbf{m}_1, \mathbf{v} \rangle$ is non-zero with some inverse polynomial probability.

When $b = 0$, $\mathbf{m}_1$ - the sparse vector - is sampled independently from $\mathbf{M}$ and hence from $\mathbf{v}$. As observed earlier, by Lemmas 5.6 and 5.9, the distribution of $\mathbf{M}^T$ is negligibly close to that of $\mathsf{SMSamp}(\lambda, l(\lambda), \log^2(l(\lambda)))$. So Lemma 2.9 implies that except with negligible probability, the code whose parity check matrix is $\mathbf{M}^T$ has distance at least $\frac{\lambda}{\log^3 \lambda}$. In this case, $\|\mathbf{v}\|_0 \geq \frac{\lambda}{\log^3 \lambda}$.

Since the Hamming weight of $\mathbf{m}_1$ is 1, the probability of the $\langle \mathbf{m}_1, \mathbf{v} \rangle$ being non-zero is simply $\frac{\|\mathbf{v}\|_0}{\lambda} \geq \frac{1}{\log^3 \lambda}$. So, B achieves non-negligible advantage (almost $\frac{1}{p(\lambda) \log^3(\lambda)}$) in the multibit semantic security game, which contradicts the semantic security of Construction 5.3 that was established in

Theorem 5.10, which is a contradiction. This completes the argument, demonstrating the collision resistance of Construction 5.4. □

5.5 Candidate Public Key Encryption Scheme

In Lemma 2.11 we showed that the distribution $(\mathbf{A}, \mathbf{Ak})$ where $\mathbf{A}$ was sampled as a sparse matrix and $\mathbf{k}$ was a random vector is indistinguishable from $(\mathbf{A}, \mathbf{r})$ where $\mathbf{r}$ is also a random vector, for a wide range of parameters. We need at least one of the two $\mathbf{A}$ or $\mathbf{k}$ to be sparse to enable the computation of $\mathbf{Ak}$ in AC^0. If we make the analogous indistinguishability assumption with the key being sparse – that is, that $(\mathbf{A}, \mathbf{Ak})$ is indistinguishable from $(\mathbf{A}, \mathbf{r})$ where $\mathbf{A} \leftarrow \{0,1\}^{\lambda \times \lambda}$, $\mathbf{k} \leftarrow \mathsf{SRSamp}(\lambda, \log^2 \lambda)$ and $\mathbf{r} \leftarrow \{0,1\}^{\lambda}$ – this allows us to construct a Public Key Encryption scheme against AC^0 computable in AC^0.

This is presented in Construction 5.5, and is easily seen to be secure under Assumption 5.12. This candidate is very similar to the LPN based cryptosystem due to Alekhnovich [Ale03]. Note that while the correctness of decryption in Construction 5.5 is not very good, this may be easily amplified by repetition without losing security, as the error is one-sided.

Assumption 5.12. *Distributions $D_1 = (\mathbf{A}, \mathbf{Ak})$ where $\mathbf{A} \leftarrow \{0,1\}^{\lambda \times \lambda}$, $\mathbf{k} \leftarrow \mathsf{SRSamp}(\lambda, \log^2 \lambda)$ and $D_2 = (\mathbf{A}, \mathbf{r})$ where $\mathbf{r} \leftarrow \{0,1\}^{\lambda}$ are indistinguishable by AC^0 adversaries with non-negligible advantage.*

Construction 5.5 Public key encryption

Let $\mathcal{I}^t = \{ip^t_\lambda\}$ be the inner product family with threshold promise t described in Lemma 2.5. Define families $\mathcal{K}eyGen = \{\mathsf{KeyGen}_\lambda\}$, $\mathcal{E}nc = \{\mathsf{Enc}_\lambda\}$, and $\mathcal{D}ec = \{\mathsf{Dec}_\lambda\}$ as below.

KeyGen_λ:

1. Sample $\mathbf{A} \leftarrow \{0,1\}^{\lambda \times \lambda - 1}$, $\mathbf{k} \leftarrow \mathsf{SRSamp}(\lambda - 1, \log^2 \lambda)$
2. Output $(\mathsf{pk}, \mathsf{sk}) = ((\mathbf{A}, \mathbf{Ak}), \mathbf{k} \circ 1)$.

$\mathsf{Enc}_\lambda(\mathsf{pk}, b)$:

1. If $b = 0$, sample $\mathbf{t} \leftarrow \mathsf{SRSamp}(\lambda, \log^2 \lambda)$ and output $\mathbf{t}^T \mathsf{pk}$
2. Else if $b = 1$, output $\mathbf{t} \leftarrow \{0,1\}^{\lambda}$

$\mathsf{Dec}_\lambda(\mathsf{sk}, \mathbf{c})$:

1. Output $ip^{\log^2(\lambda)}_{\lfloor \lambda \rfloor_2}(\mathsf{sk}, \mathbf{c})$.

The most commonly used proof technique in this paper – showing k-wise independence for a large k – cannot be used to prove the security of this scheme because due to the sparsity of the key, the distribution $(\mathbf{A}, \mathbf{Ak})$ is not k-wise independent.

Conclusions and Open Questions. We construct various cryptographic primitives secure against parallel-time-bounded adversaries. Our constructions against AC^0 are unconditional whereas our constructions against NC^1 require the assumption that $\mathsf{NC}^1 \neq \oplus\mathsf{L/poly}$. Our constructions make use of circuit lower bounds [Bra10] and non-black-box use of randomized encodings for logspace classes [IK00].

There are several open questions that arise out of this work. Perhaps the most immediate are:

1. Unconditional lower-bounds are known for slightly larger classes like $\mathsf{AC}^0[p]$ when p is a prime power. Can we get cryptographic primitives from those lower-bounds?

2. Construct a public key encryption scheme secure against AC^0, either by proving the security of our candidate proposal (see Section 5.5) or by completely different means.

   Natural ways of doing this lead us to a fascinating question about the complexity of AC^0 circuits. Braverman [Bra10] shows that *any* n^ϵ-wise independent distribution fools all AC^0 circuits. Our candidate encryption, however, produces ciphertexts that come from a $\log^c(n)$-wise distribution for some constant c. This raises the following question: *Can we show* some *fixed* $\mathsf{polylog}$*-wise independent distribution (that is* not n^ϵ*-wise independent) that fools* AC^0 *circuits of arbitrary depth?* (This question came up during discussions with Li-Yang Tan.)

3. We relied on the assumption that $\oplus\mathsf{L}/\mathsf{poly} \not\subset \mathsf{NC}^1$ to construct primitives secure against NC^1. It would be desirable to relax the assumption to $\mathsf{P} \not\subset \mathsf{NC}^1$.

A related extension of Merkle's work is to construct a public-key encryption scheme resistant against $O(n^c)$ time adversaries (for some $c > 2$) under worst-case hardness assumptions.

References

[AB84] Miklós Ajtai and Michael Ben-Or. A theorem on probabilistic constant depth computations. In *Proceedings of the 16th Annual ACM Symposium on Theory of Computing, April 30 - May 2, 1984, Washington, DC, USA*, pages 471–474, 1984.

[ABW10] Benny Applebaum, Boaz Barak, and Avi Wigderson. Public-key cryptography from different assumptions. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 171–180. ACM, 2010.

[AGGM06] Adi Akavia, Oded Goldreich, Shafi Goldwasser, and Dana Moshkovitz. On basing one-way functions on np-hardness. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing, Seattle, WA, USA, May 21-23, 2006*, pages 701–710, 2006.

[AGHP93] Noga Alon, Oded Goldreich, Johan Håstad, and René Peralta. Addendum to "simple construction of almost k-wise independent random variables". *Random Struct. Algorithms*, 4(1):119–120, 1993.

[AIK04] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography in nco. In *FOCS 2004: 45th Annual IEEE Symposium on Foundations of Computer Science: proceedings: 17-19 October, 2004, Rome, Italy*, page 166. IEEE Computer Society Press, 2004.

[Ajt83] M. Ajtai. 11-formulae on finite structures. *Annals of Pure and Applied Logic*, 24(1):1 – 48, 1983.

[Ale03] Michael Alekhnovich. More on average case vs approximation complexity. In *Foundations of Computer Science, 2003. Proceedings. 44th Annual IEEE Symposium on*, pages 298–307. IEEE, 2003.

[App14] Benny Applebaum. Cryptography in nc 0. In *Cryptography in Constant Parallel Time*, pages 33–78. Springer, 2014.

[AR99] Yonatan Aumann and Michael O Rabin. Information theoretically secure communication in the limited storage space model. In *Advances in CryptologyCRYPTO99*, pages 65–79. Springer, 1999.

[AR15] Benny Applebaum and Pavel Raykov. On the relationship between statistical zero-knowledge and statistical randomized encodings. *Electronic Colloquium on Computational Complexity (ECCC)*, 22:186, 2015.

[AW85] Miklós Ajtai and Avi Wigderson. Deterministic simulation of probabilistic constant depth circuits (preliminary version). In *26th Annual Symposium on Foundations of Computer Science, Portland, Oregon, USA, 21-23 October 1985*, pages 11–19, 1985.

[Bar86] David A. Mix Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in nc^1. In *Proceedings of the 18th Annual ACM Symposium on Theory of Computing, May 28-30, 1986, Berkeley, California, USA*, pages 1–5, 1986.

[BB15] Andrej Bogdanov and Christina Brzuska. On basing size-verifiable one-way functions on np-hardness. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *Theory of Cryptography - 12th Theory of Cryptography Conference, TCC 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part I*, volume 9014 of *Lecture Notes in Computer Science*, pages 1–6. Springer, 2015.

[BGI+01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In *Advances in Cryptology - CRYPTO 2001*, pages 1–18, 2001.

[BGI08] Eli Biham, Yaron J. Goren, and Yuval Ishai. Basing weak public-key cryptography on strong one-way functions. In Ran Canetti, editor, *Theory of Cryptography, Fifth Theory of Cryptography Conference, TCC 2008, New York, USA, March 19-21, 2008.*, volume 4948 of *Lecture Notes in Computer Science*, pages 55–72. Springer, 2008.

[BM84] Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM J. Comput.*, 13(4):850–864, 1984.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *CCS '93, Proceedings of the 1st ACM Conference on Computer and Communications Security, Fairfax, Virginia, USA, November 3-5, 1993.*, pages 62–73. ACM, 1993.

[Bra10] Mark Braverman. Polylogarithmic independence fools AC^0 circuits. *J. ACM*, 57(5), 2010.

[BT03] Andrej Bogdanov and Luca Trevisan. On worst-case to average-case reductions for NP problems. In *44th Symposium on Foundations of Computer Science (FOCS 2003), 11-14 October 2003, Cambridge, MA, USA, Proceedings*, pages 308–317. IEEE Computer Society, 2003.

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In Rafail Ostrovsky, editor, *FOCS*, pages 97–106. IEEE, 2011. Invited to SIAM Journal on Computing.

[CM97] Christian Cachin and Ueli Maurer. Unconditional security against memory-bounded adversaries. In *Advances in CryptologyCRYPTO'97*, pages 292–306. Springer, 1997.

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[DM04] Stefan Dziembowski and Ueli Maurer. On generating the initial key in the bounded-storage model. In *Advances in Cryptology-EUROCRYPT 2004*, pages 126–137. Springer, 2004.

[FSS84] Merrick L. Furst, James B. Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. *Mathematical Systems Theory*, 17(1):13–27, 1984.

[Gal62] Robert G. Gallager. Low-density parity-check codes. *IRE Trans. Information Theory*, 8(1):21–28, 1962.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, pages 169–178, 2009.

[GGH+13] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *FOCS 2013*, pages 40–49, 2013.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *Journal of the ACM (JACM)*, 33(4):792–807, 1986.

[GM82] Shafi Goldwasser and Silvio Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In *STOC 1982*, pages 365–377, 1982.

[GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In *Proceedings of the 17th Annual ACM Symposium on Theory of Computing, May 6-8, 1985, Providence, Rhode Island, USA*, pages 291–304, 1985.

[GR12] Shafi Goldwasser and Guy N. Rothblum. How to compute in the presence of leakage. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012*, pages 31–40, 2012.

[Hås86] Johan Håstad. Almost optimal lower bounds for small depth circuits. In *Proceedings of the 18th Annual ACM Symposium on Theory of Computing, May 28-30, 1986, Berkeley, California, USA*, pages 6–20, 1986.

[Has87] Johan Hastad. One-way permutations in nc 0. *Information Processing Letters*, 26(3):153–155, 1987.

[Hås14] Johan Håstad. On the correlation of parity and small-depth circuits. *SIAM J. Comput.*, 43(5):1699–1708, 2014.

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.

[IK00] Yuval Ishai and Eyal Kushilevitz. Randomizing polynomials: A new representation with applications to round-efficient secure computation. In *Foundations of Computer Science, 2000. Proceedings. 41st Annual Symposium on*, pages 294–304. IEEE, 2000.

[IKO05] Yuval Ishai, Eyal Kushilevitz, and Rafail Ostrovsky. Sufficient conditions for collision-resistant hashing. In *Theory of Cryptography, Second Theory of Cryptography Conference, TCC 2005, Cambridge, MA, USA, February 10-12, 2005, Proceedings*, pages 445–456, 2005.

[IL89] Russell Impagliazzo and Michael Luby. One-way functions are essential for complexity based cryptography (extended abstract). In *30th Annual Symposium on Foundations of Computer Science, Research Triangle Park, North Carolina, USA, 30 October - 1 November 1989*, pages 230–235. IEEE Computer Society, 1989.

[IR88] Russell Impagliazzo and Steven Rudich. Limits on the provable consequences of one-way permutations. In *Advances in Cryptology - CRYPTO '88, 8th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1988, Proceedings*, pages 8–26, 1988.

[LMN93] Nathan Linial, Yishay Mansour, and Noam Nisan. Constant depth circuits, fourier transform, and learnability. *Journal of the ACM (JACM)*, 40(3):607–620, 1993.

[Mau92] Ueli M Maurer. Conditionally-perfect secrecy and a provably-secure randomized cipher. *Journal of Cryptology*, 5(1):53–66, 1992.

[Mer78] Ralph C. Merkle. Secure communications over insecure channels. *Commun. ACM*, 21(4):294–299, 1978.

[MP06] Silvio Micali and Rafael Pass. Local zero knowledge. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing, Seattle, WA, USA, May 21-23, 2006*, pages 306–315, 2006.

[MST06] Elchanan Mossel, Amir Shpilka, and Luca Trevisan. On epsilon-biased generators in nc^0. *Random Struct. Algorithms*, 29(1):56–81, 2006.

[NW94] Noam Nisan and Avi Wigderson. Hardness vs randomness. *J. Comput. Syst. Sci.*, 49(2):149–167, 1994.

[PVW08] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In *Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings*, pages 554–571, 2008.

[RAD78] R. Rivest, L. Adleman, and M. Dertouzos. On data banks and privacy homomorphisms. In *Foundations of Secure Computation*, pages 169–177. Academic Press, 1978.

[Raz87] A. A. Razborov. Lower bounds on the size of bounded depth circuits over a complete basis with logical addition. *Mathematical notes of the Academy of Sciences of the USSR*, 41(4):333–338, 1987.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.

[RST15] Benjamin Rossman, Rocco A. Servedio, and Li-Yang Tan. An average-case depth hierarchy theorem for boolean circuits. *Electronic Colloquium on Computational Complexity (ECCC)*, 22:65, 2015.

[RW91] Prabhakar Ragde and Avi Wigderson. Linear-size constant-depth polylog-treshold circuits. *Inf. Process. Lett.*, 39(3):143–146, 1991.

[Smo87] Roman Smolensky. Algebraic methods in the theory of lower bounds for boolean circuit complexity. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, pages 77–82, 1987.

[SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In David B. Shmoys, editor, *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 475–484. ACM, 2014.

[Tal14] Avishay Tal. Tight bounds on the fourier spectrum of ac^0. *Electronic Colloquium on Computational Complexity (ECCC)*, 21:174, 2014.

[TX13] Luca Trevisan and Tongke Xue. A derandomized switching lemma and an improved derandomization of AC0. In *Proceedings of the 28th Conference on Computational Complexity, CCC 2013, K.lo Alto, California, USA, 5-7 June, 2013*, pages 242–247, 2013.

[Vad04] Salil P Vadhan. Constructing locally computable extractors and cryptosystems in the bounded-storage model. *Journal of Cryptology*, 17(1):43–77, 2004.

[Vio12] Emanuele Viola. The complexity of distributions. *SIAM Journal on Computing*, 41(1):191–218, 2012.

[Yao82] Andrew Chi-Chih Yao. Theory and applications of trapdoor functions (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3-5 November 1982*, pages 80–91. IEEE Computer Society, 1982.

A Håstad's OWF Construction

Håstad([Has87]) constructed One-Way Functions (actually One Way Permutations) against AC^0 based on the average-case hardness of the PARITY function against low-depth circuits. This is presented in Construction A.1.

Theorem A.1 (OWFs against AC^0)**.** *The family $\mathcal{F}$ from Construction A.1 is an NC^0-One-Way Function against AC^0.*

Construction A.1 NC^0-One-Way Function against AC^0

$\mathcal{F} = \{f_\lambda\}$ is given by:

$$f_\lambda(x_1, \ldots, x_\lambda) = (x_1 \oplus x_2, x_2 \oplus x_3, \ldots, x_{\lambda-1} \oplus x_\lambda, x_\lambda)$$

The intuition behind the proof is that given the output $(x_1 \oplus x_2, x_2 \oplus x_3, \ldots x_{\lambda-1} \oplus x_\lambda, x_\lambda)$, the first bit of the inverse is the parity of the output. Since the output is uniform distribution for the input being uniform, the average case hardness of parity implies that AC^0 adversaries cannot invert this function.

Proof of Theorem A.1. Computability and non-triviality are easily seen to be satisfied - each output bit of f_λ for any λ depends on at most two input bits, f_λ is deterministic.

Note that f_λ is bijective. Any $y = (y_1, \ldots, y_\lambda)$ has a unique inverse under f_λ, which is $(\oplus_{i=1}^n y_i, \oplus_{i=2}^n y_i, \ldots, y_{n-1} \oplus y_n, y_n)$. In particular, the first bit of the inverse is $\mathsf{PARITY}(y)$.

Consider any AC^0 function family $\mathcal{G} = \{g_\lambda\}$. Define another function family $\mathcal{H} = \{h_\lambda\}$, where h_λ does the follwing on input y:

1. Compute $z \leftarrow g_\lambda(y)$.
2. Check whether $f_\lambda(z) = y$.
3. If so, output the first bit of z.
4. If not, output a random bit.

$\mathcal{H}$ is also an AC^0 function family because f_λ and g_λ can be computed in constant depth, and so can checking equality. This means that there is a polynomial s and constant d such that g_λ is computed by a circuit of size at most $s(\lambda)$ and depth at most d.

By the above observation about the first bit of f_λ, we have for any λ:

$$\begin{aligned}\Pr_{y \leftarrow \{0,1\}^\lambda}[h_\lambda(y) = \mathsf{PARITY}(y)] &= \Pr_{y \leftarrow \{0,1\}^\lambda}\left[g_\lambda(y) = f_\lambda^{-1}(y)\right] \\ &\quad + \frac{1}{2}\Pr_{y \leftarrow \{0,1\}^\lambda}\left[g_\lambda(y) \neq f_\lambda^{-1}(y)\right] \\ &= \frac{1}{2} + \frac{1}{2}\Pr_{y \leftarrow \{0,1\}^\lambda}\left[g_\lambda(y) = f_\lambda^{-1}(y)\right]\end{aligned}$$

By the injectivity of f_λ, we have:

$$\Pr_{x \leftarrow \{0,1\}^\lambda}[f_\lambda(g_\lambda(y)) = y \mid y = f_\lambda(x)] = \Pr_{y \leftarrow \{0,1\}^\lambda}\left[g_\lambda(y) = f_\lambda^{-1}(y)\right]$$

Hence, to prove one-wayness, we only need to show that h_λ cannot compute PARITY correctly with probability noticeably greater than $\frac{1}{2}$. This follows directly from Theorem 2.1 as:

$$\Pr_{y \leftarrow \{0,1\}^\lambda}[h_\lambda(y) = \mathsf{PARITY}(y)] \leq \frac{1}{2} + 2^{-\Omega(\lambda/(\log s(\lambda))^{d-1})}$$

Hence, there cannot be an AC^0 family of functions $\mathcal{G}$ that has non-negligible advantage in inverting $\mathcal{F}$, and this proves the theorem. □

Remark A.1. *Construction A.1 is, in fact, a permutation.*

Remark A.2. *It is not difficult to generalise Construction A.1 by noting that the only properties of f_λ used in the proof are that:*

1. *$f_\lambda(x)$ is uniform over $\{0,1\}^{\lambda'}$ when x is uniform over $\{0,1\}^\lambda$.*
2. *$\mathsf{PARITY}(f_\lambda(x))$ is computable from x in AC^0.*

One example of a function with these properties is $f(\mathbf{x}) = \mathbf{A}\mathbf{x}$, where $\mathbf{A}$ is a full rank $\lambda' \times \lambda$ for some $\lambda' < \lambda$, and the sum of all the rows of $\mathbf{A}$ is a vector of low (polylog in λ) Hamming weight.