Fine-Grained Cryptography

by

Prashant Nalini Vasudevan

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2018

© Massachusetts Institute of Technology 2018. All rights reserved.

Author .....
Department of Electrical Engineering and Computer Science
August 15, 2018

Certified by .....
Vinod Vaikuntanathan
Associate Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by .....
Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee on Graduate Students

Fine-Grained Cryptography

by

Prashant Nalini Vasudevan

Submitted to the Department of Electrical Engineering and Computer Science
on August 15, 2018, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

Fine-grained cryptography is the study of cryptographic objects that are required to be secure only against adversaries that are moderately more powerful than the honest parties. This weakening in security requirements opens up possibilities for meaningful cryptographic constructions in various settings using hardness assumptions that are considerably weaker than those used in standard cryptography. In this thesis, we study these possibilities in two different settings.

First, we present functions that are hard to compute on average for algorithms running in some fixed polynomial time, assuming widely-conjectured worst-case hardness of certain problems from the study of fine-grained complexity. We also construct a proof-of-work protocol based on this hardness and certain structural properties of our functions.

Second, we construct several unconditionally secure cryptographic primitives that are computable by and secure against constant-depth circuits. Under a reasonable complexity-theoretic assumption, we do the same for log-depth circuits.

Thesis Supervisor: Vinod Vaikuntanathan
Title: Associate Professor of Electrical Engineering and Computer Science

Fine-Grained Cryptography

by

Prashant Nalini Vasudevan

Submitted to the Department of Electrical Engineering and Computer Science
on August 15, 2018, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

Fine-grained cryptography is the study of cryptographic objects that are required to be secure only against adversaries that are moderately more powerful than the honest parties. This weakening in security requirements opens up possibilities for meaningful cryptographic constructions in various settings using hardness assumptions that are considerably weaker than those used in standard cryptography. In this thesis, we study these possibilities in two different settings.

First, we present functions that are hard to compute on average for algorithms running in some fixed polynomial time, assuming widely-conjectured worst-case hardness of certain problems from the study of fine-grained complexity. We also construct a proof-of-work protocol based on this hardness and certain structural properties of our functions.

Second, we construct several unconditionally secure cryptographic primitives that are computable by and secure against constant-depth circuits. Under a reasonable complexity-theoretic assumption, we do the same for log-depth circuits.

Thesis Supervisor: Vinod Vaikuntanathan
Title: Associate Professor of Electrical Engineering and Computer Science

Acknowledgments

I should first thank my advisor, Vinod Vaikuntanathan, for supporting me for the past five years and teaching me much in spite of myself. Vinod has been a constant source of interesting problems (though I did not work on as many as I should have), and his infectious energy and enthusiasm have often served as great inspiration. I have learnt much from him, and I am sure I still have much to learn.

John Lennon said he might have been born in Liverpool, but he grew up in Hamburg. I am graduating from MIT, but I grew up in Tel Aviv. Much of the work in this thesis was done two years ago there with Marshall Ball, Alon Rosen, and Manuel Sabin, while I was visiting Alon at IDC Herzliya. Alon believed in me in times when even I did not believe in myself, and this has made all the difference.

I am grateful to Benny Applebaum, for hosting me several times in Israel and spending days and weeks in the trenches with me, working on problems with little more than hope, persistence, and linear algebra. It has been a great experience working with him, as it has with Andrej Bogdanov. It has been over a year since I spent a summer with Andrej in Hong Kong, and I still cannot think of concentration bounds without being reminded of him just a little.

I was very lucky to find others – Itay Berman, Akshay Degwekar, and Ron Rothblum – who shared my interests in understanding a number of aspects of the foundations of cryptography. Together, we have dug around the foothills and come up with numerous interesting connections and implications and continue to discover promising veins to mine. Ron has also been of great help in other respects, always available with advice, whether on research or otherwise.

I also thank all my other collaborators, especially Adam Bouland and Dhiraj Holden, with whom I have laid several sieges against looming fortifications, and while many of these fortresses have held up to our batteries, each of these attempts have left me with finer weapons for future assaults. As Alon once told me, paraphrasing Friedrich Nietzsche, “In the mountains of truth, we never walk in vain”.

No winter can be that cold, and no night too dark, with good company and

nothing much to do. Over the past five years I have spent several hours and days in conversations with many I am fortunate to call friends. Prabhanjan Ananth, Marshall Ball, Danny Blumberg, Michael Coulombe, Apoorvaa Deshpande, Sergey Gorbunov, Robin Hui, Pritish Kamath, Ranjit Kumaresan, Tianren Liu, Madalina Persu, Srinivasan Raghuraman, Manuel Sabin, Adam Sealfon, Rajan Udwani, the list goes on and there are many whom I am surely missing out who have softened the relentless ticking of time as it trundles along oblivious to our fears and failures. And there are my friends from my undergraduate times, whom I can always rely on when the path seems full of potholes and loose gravel.

I am also grateful to From Software for making one of the greatest computer games to exist, and to everyone who organised and participated in years of Theory Jam.

And most of all, I could not have made it anywhere without my family. My parents have encouraged and supported me selflessly for the past twenty six years, and of everything I have done, I owe them some part. My sister remains a persistent source of glee and mockery, and I am incredibly grateful to my grandmothers and Perippa and Perimma and everyone else in my family who have cared for me and remain pillars of support.

Dedicated to the couches on the sixth floor, the playground outside my window, and the movie theatre on Brattle Street.

Contents

1 Introduction **11**
- 1.1 Fine-Grained Cryptography . . . 12
- 1.2 Cryptography Against Bounded Running Time . . . 20
  - 1.2.1 Average-Case Hardness . . . 21
  - 1.2.2 Proofs of Work . . . 21
- 1.3 Cryptography Against Bounded Circuit Depth . . . 23
  - 1.3.1 Cryptography against AC^0 . . . 23
  - 1.3.2 Cryptography against NC^1 . . . 24

2 Average-Case Fine-Grained Hardness **27**
- 2.1 Worst-Case Conjectures . . . 33
- 2.2 Average-Case Fine-Grained Hardness . . . 37
  - 2.2.1 Orthogonal Vectors . . . 39
  - 2.2.2 3SUM and All-Pairs Shortest Path . . . 41
  - 2.2.3 SETH, 3SUM, and All-Pairs Shortest Path . . . 42
  - 2.2.4 CONVOLUTION-3SUM . . . 44
- 2.3 Evaluating Low Degree Polynomials . . . 46

3 Proofs of Work **49**
- 3.1 Definitions . . . 53
  - 3.1.1 Proofs of Work . . . 53
  - 3.1.2 Orthogonal Vectors . . . 55
- 3.2 Verifying $\mathcal{F}\mathsf{OV}^k$ . . . 57
- 3.3 The PoW Protocol . . . 63
- 3.4 A Direct Sum Theorem for $\mathcal{F}\mathsf{OV}$ . . . 66
- 3.5 Removing Interaction . . . 72
- 3.6 Zero-Knowledge Proofs of Work . . . 76

4 Cryptography Against Bounded Depth **83**
- 4.1 Definitions and Preliminaries . . . 93
  - 4.1.1 Bounded Adversaries . . . 94
  - 4.1.2 Constant-Depth Circuits . . . 98
  - 4.1.3 Graphs and Linear Codes . . . 100
  - 4.1.4 Randomized Encodings . . . 106
- 4.2 Cryptography Against AC^0 . . . 107

4.2.1 High-Stretch Pseudo-Random Generators . . . 108
4.2.2 Weak Pseudo-Random Functions . . . 109
4.2.3 Symmetric Key Encryption . . . 116
4.2.4 Collision Resistant Hash Functions . . . 123
4.2.5 Candidate Public Key Encryption Scheme . . . 126
4.3 Cryptography Against NC^1 . . . 127
4.3.1 OWFs from worst-case assumptions . . . 128
4.3.2 PKE and CRHF against NC^1 . . . 130

5 Conclusion and Future Directions **137**
5.1 Subsequent Work . . . 137
5.2 Towards Fine-Grained One-Way Functions . . . 138
5.2.1 Barriers and NSETH . . . 140
5.2.2 A Way Around . . . 140
5.3 Other Open Problems . . . 142

A Appendices to Chapter 2 **145**
A.1 An Average-Case Time Hierarchy . . . 145
A.2 On the Heuristic Falsifiability of Conjectures . . . 147
A.3 A Tighter Reduction for $\mathcal{F}\mathsf{OV}$ . . . 149
A.4 Polynomials Computing Sums . . . 154
A.5 Isolating Orthogonal Vectors . . . 155

B Appendices to Chapter 3 **161**
B.1 A Stronger Direct Sum Theorem for $\mathcal{F}\mathsf{OV}$ . . . 161

Chapter 1

Introduction

Cryptography is today a central part of commerce and communication around the world; from encrypting credit card numbers to authenticating web servers, cryptographic operations have become a part of everyday life for large sections of the population. The last four decades of research in the theory of cryptography has produced a host of fantastic notions that have enabled these numerous applications, such as digital signatures [DH76, RSA78, GMR88] and public-key encryption [DH76, RSA78, GM82], and continues to produce more that stand to shape the digital world in the years to come, such as zero-knowledge proofs [GMR85], fully homomorphic encryption [RAD78, Gen09, BV11], and program obfuscation [BGI+01, GGH+13, SW14]. 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 $\mathsf{P} \neq \mathsf{NP}$ [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 [Rab79, RSA78], discrete logarithms [DH76], quadratic residuosity [GM82], the approximate shortest vector and other problems for lattices and codes [Mce78, Ajt96], and, more recently, even more exotic assumptions [BFKL93, ABW10, MI88, Ale03]. 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 the relationships between most others. In particular, it is a wide open question whether $\mathsf{P} \neq \mathsf{NP}$ suffices to construct even the most basic cryptographic objects such as one-way functions, and 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]).

1.1 Fine-Grained Cryptography

In this thesis, we explore one possible approach towards weakening the assumptions necessary to do useful cryptography – that of weakening the requirements we place on the cryptographic objects we wish to construct.

Traditionally, in theoretical computer science, efficiency is associated with (arbitrarily large) polynomial time. Consequently, in cryptography, security is typically desired against all adversaries that run in polynomial time, even those that are vastly more powerful than the honest parties involved. We propose to instead study what we call *fine-grained* cryptography, where we care only about adversaries that are moderately more powerful than the honest parties.

For illustration, let us consider the case of encryption schemes. In the design of an encryption scheme, it is typically desired that the encryption and decryption procedures are efficient – that is, that they run in polynomial time – and that the ciphertexts are secure against any efficient adversary – that is, any adversary that runs in polynomial time. It could be that encryption and decryption run in $O(n)$ time (where n is a security parameter), and such a scheme would have to be secure against adversaries that run in $O(n^2)$, $O(n^{10})$, and even $O(n^{100})$ time.

In a fine-grained encryption scheme, the security and efficiency requirements become more fine-grained. For instance if, in such a scheme, encryption and decryption can be done in $O(n)$ time, we would ask for security only against adversaries that run in, say, $o(n^3)$ time. It may even be the case that an adversary running in $\Theta(n^3)$

time can already break the encryption, and this would be permissible. By setting the parameter n appropriately, it may be ensured that encryption and decryption are eminently feasible, while the scheme cannot be broken by all the computational power available in the world in a reasonable amount of time.

More generally, fine-grained cryptography is the study of cryptographic primitives that are:

1. Secure against adversaries with bounded resources (e.g. $o(n^3)$ time, log-space)
2. Computable with fewer resources than these adversaries

Cryptography even if $\mathsf{P} = \mathsf{NP}$. A significant benefit of considering such fine-grained cryptographic primitives is that this limited security, while still useful in practical settings, is a much weaker requirement than security against adversaries that run in arbitrary polynomial time. This opens up the possibility of constructing such primitives based on hardness assumptions that are much weaker than those used presently.

In particular, recall that, as mentioned earlier, the security of most standard cryptographic primitives already implies that $\mathsf{P} \neq \mathsf{NP}$. Fine-grained primitives, such as an encryption scheme that is secure against $o(n^3)$ adversaries, on the other hand, do not have this implication. Thus, we might very well be able to construct such objects unconditionally much earlier than we are able to resolve P vs. NP. In different terms, even in a world where $\mathsf{P} = \mathsf{NP}$ and standard polynomial-time cryptography is not possible, such fine-grained cryptographic objects could still exist.

Security based on more reliable assumptions. As mentioned earlier, by setting the parameters right, several fine-grained cryptographic primitives could be used in the place of standard ones. These parameter settings would perhaps require more computation effort of the honest parties, but they stand to offer better confidence in the security of the resulting primitives, if we are indeed able to realise our goal of building such primitives from weaker assumptions.

A major challenge to the hardness of any computational problem is the presence

of some "structure" or symmetry that can be exploited to design algorithms for it. At the same time, in the design of more advanced cryptographic primitives like public-key encryption with standard security guarantees, it seems, from our experience so far, that hard problems with a certain amount of structure (like DDH or LWE) are necessary. As a result, the hardness of problems that public-key cryptography is based on is much more suspect than those that the simpler private-key primitives are based on, at least where security against all polynomial-time adversaries is desired (see [Bar17] for a more detailed discussion of this subject).

On the other hand, Merkle [Mer78], in the 70's, already showed how an object as structureless as a random oracle could be used to construct a public-key primitive like key agreement where the honest parties run in $O(n)$ time and security holds against $o(n^2)$-time adversaries. This suggests that perhaps fine-grained cryptographic primitives could be based on problems that are qualitatively different, lacking much exploitable structure, which would let us rest more comfortably on beliefs that these problems are hard.

Ephemeral security. Primitives with limited security guarantees are most appropriate in applications where more security is not deemed necessary – where security that is *ephemeral*, lasting only for a brief period of time, is sufficient. For example, consider the stock market and, in particular, the setting of high-frequency trading. In this regime, there is large constant stream of information that is maintained and communicated by various parties regarding which products to trade, etc., all of which is relevant and valuable for little longer than mere seconds. In such a setting, fine-grained encryption schemes would have all the other benefits mentioned here without any concerns regarding loss of security against stronger adversaries, as any adversary that spends too much time breaking these primitives would only end up with information that has already lost its value.

Another example of an application that only requires ephemeral security is interactive authentication [BR93]. Authentication protocols are run by communicating parties at the very beginning of a session of communication in order to authenticate

themselves to the each other. Any adversary that seeks to intrude into their communication by defeating the authentication protocol has to be able to break the security of the protocol in the small amount of time that the other parties involved are willing to wait before they decide to terminate the protocol. Thus, security is not needed against adversaries that run for longer than the pre-determined time-to-live in the protocol description.

Towards an economic model for security. Fine-grained primitives could also be used to make the cost of breaking security just high enough to render it unprofitable. If the cost, to any adversary that may be presumed to be rational, of the resources (computational power or even energy) required to decrypt a ciphertext without the key is greater than value it places on whatever was encrypted, then no such adversary will even attempt to break this security, even if it would be able to. Fine-grained control over the security of the primitives used would allow the honest parties to tune the parameters so that the work required on their part is minimised while the cost of an attack is still high enough to effect the above situation.

Win-win. While this is certainly not true in all cases, a large number of cryptographic constructions are built on the hardness of problems that are interesting for little reason other than our belief in their hardness itself. With such constructions, while the security of the resulting primitive is valuable, if it turns out that they are insecure, there are no interesting implications. For instance, if SHA-256 is secure then it has a number of applications, if it is not, then SHA-256 is broken. This is a Win-Lose situation – either we have a secure hash function, or a specific instantiation of a heuristic cryptographic hash function is broken and no new knowledge is gained.

On the other hand, many of the problems whose worst-case hardness we make use of in Chapter 2, such as Orthogonal Vectors or 3SUM for instance, are of independent interest and have several connections with other interesting problems [Wil15]. Building cryptographic primitives on the hardness of such problems would set up a Win-Win situation where, if these problems are hard then we have cryptography, and

if not, we end up with better algorithms for several useful problems or at least with better knowledge of their complexities.

Hybrid constructions. Another interesting possibility with fine-grained primitives (that was suggested to us independently by Ron Rothblum and Yuval Ishai) is using them in conjunction with other primitives that are secure against polynomial-time adversaries under stronger assumptions; this would result in hybrids that are secure against polynomial-time adversaries under these stronger assumptions while also being secure against bounded adversaries under weaker assumptions (or even unconditionally).

For instance, consider an encryption scheme where the message is first encrypted using the symmetric-key encryption scheme from Section 4.2.3 (that is unconditionally secure against AC^0 adversaries), and the resultant ciphertext is then encrypted using an encryption scheme that works in AC^0 and is secure against polynomial-time adversaries under some standard assumptions (see [AIK04] for such schemes). This hybrid scheme is now secure against polynomial-time adversaries under these standard assumptions while being unconditionally secure against AC^0 adversaries, and encryption and decryption can still be done in AC^0.

Examples of prior work. Of course, we are not the first to investigate the possibility of cryptography with security against a bounded class of adversaries; this question has been asked by several researchers ever since the big-bang of cryptography [Mer78], and some examples of such research are the following.

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 constant fan-in and NOT gates), but inverting which is hard for any AC^0 circuit (also constant-depth, but unbounded fan-in). 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].

4. Certain primitives, like proof of work and proof of space protocols and time-lock puzzles, are inherently fine-grained. There is a significant body of work studying these owing to their various applications. [DN92, JJ99, RSW00, MMV11, DFKP15, BGJ+16, ...]

Note that all these works share two common features. First, security is achieved against a class of adversaries with bounded resources (time, space and circuit-depth or parallel-time, respectively, in the first three works above). Secondly, the honest

algorithms use less resources than the class of adversaries they are trying to fool. These are the two essential properties of fine-grained cryptography as we study it. Our contributions in this thesis are to employ recent advances in the study of the fine-grained hardness of problems within P to construct some simple cryptographic objects against adversaries that run in some fixed polynomial time, and a number of constructions (some unconditional, some based on reasonable worst-case complexity-theoretic assumptions) against adversaries that are computable by low-depth circuits. These are described briefly in the following sections, and in detail in the following chapters.

Remark 1 (on the relative power of honest parties). 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. This is the setting that is most commonly studied in cryptography, though there are a number of exceptions [DPW09, GR15, ...]; and outside cryptography (in complexity theory, for instance), there are many cases where the "adversary" has less power than the "honest" algorithms. 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 former regime.

The Landscape of Hardness in Cryptography

Before describing our results, in order to place them in sufficient context, we provide a brief and informal overview of the different kinds of computational hardness and

how they relate to some basic cryptographic primitives. We refer the reader to any standard textbook on cryptography [Gol01, KL14, ...] for a more detailed discussion of the known relationships between these and also for references for other statements made below.

The simplest notion of hardness, and the one that is most prevalent in the study of complexity, algorithms, etc., is that of worst-case hardness. A problem is worst-case hard if every efficient algorithm fails to solve it correctly on some input of size n for all large enough n.

The notion of hardness that is of greater interest in the study of cryptography, however, is that of average-case hardness. A problem is average-case hard if, for all large enough n, there is a distribution over instances of size n such that no efficient algorithm solves the problem correctly with high success probability over random instances drawn according to this distribution.

The cleanest notion of hardness that is actually useful in a cryptographic sense is that of the one-way function, one of the simplest cryptographic primitives. A one-way function is a function that can be computed efficiently, but no efficient algorithm can, for large enough input lengths, invert a uniformly random output of the function. It is commonly known that if a problem is average-case hard and it is possible to efficiently sample an input from the hard distribution along with the corresponding solution, then this implies a one-way function.

The most sophisticated cryptographic primitive we discuss here is public-key encryption. In a loose sense, public-key encryption is known to follow from families of one-way functions that are permutations and have "trapdoors" – some additional information that makes the function efficiently invertible if known.

Each notion described above is stronger than the one before it, as represented below. No generic transformations are known from any of these notions to a stronger one, and most cryptographic constructions can be thought of as performing such a transformation in a special case. In our constructions, we start from worst-case hardness (either assumed or known) and obtain various primitives in various settings as described in the next few sections.

$$\text{Worst-Case Hardness} \lessdot \text{Average-Case Hardness} \lessdot \text{One-Way Functions} \lessdot \text{Public-Key Encryption} \lessdot \cdots$$

This picture is for illustrative purposes and is by no means complete, excluding as it does even common primitives like collision-resistant hash functions and several advanced primitives that are stronger than public-key encryption. Also, in the above description, we have left unmentioned the fact that generally, at least in complexity theory, when one says that a problem is hard, one means it is hard infinitely often – that any efficient algorithm does not solve it on instances of size n for some infinite sequence of n's; whereas, in cryptography, security is generally desired for all large enough n. In all our constructions described below (except the unconditional ones), the kind of security obtained (infinitely often or for all large n) corresponds directly to the kind of computational hardness assumption one starts with.

1.2 Cryptography Against Bounded Running Time

Perhaps the most natural computational resource one could consider is running-time, and indeed, much of complexity theory concerns itself with classifying problems according to how much time a Turing Machine takes to solve them or how large a Boolean circuit for them is. Typically, efficiency is associated with algorithms running in polynomial time (or circuits of polynomial size), and this has also been the view in cryptography, where the adversaries we seek to be secure against are allowed to run in arbitrary polynomial time.

Over the last couple of decades, however, there has been significant progress in the study of the fine-grained complexity of problems that are known to have polynomial-time algorithms (see [Wil15] for relevant references). Similar to polynomial-time reductions in the study of NP-completeness, several efficient reductions between problems of interest have been discovered that relate their complexities in a fine-grained manner. As the dust settles, a handful of these problems – such as Orthogonal Vectors (OV, a version of Hopcroft's problem), 3SUM, and All Pairs Shortest Paths (APSP) (all defined in Chapter 2) – have risen in prominence, both because of their importance

by themselves, and owing to their tight connections with the fine-grained complexities of other important problems. These problems have been extensively studied, and there are fine-grained conjectures regarding their complexities. We asked whether these conjectures could be used to do fine-grained cryptography, and have been able to make the following initial progress towards an answer.

1.2.1 Average-Case Hardness

The first step to constructing cryptographic objects is identifying problems that are hard in the average-case (rather than just in the worst-case). All previous study of fine-grained complexity, however, has been in the worst-case setting. Towards addressing this, we show several fine-grained worst-case to average-case reductions that, along with the conjectured worst-case hardness of the problems mentioned above, give us problems that are as hard to solve on random inputs. The following is an example of such a statement that we prove.

Informal Theorem 1. *There is a function $f\mathsf{OV}$ such that, if OV requires $n^{2-o(1)}$ time to decide in the worst-case, then $f\mathsf{OV}$ requires $n^{2-o(1)}$ time to compute on uniformly random inputs. Further, $f\mathsf{OV}$ can be computed on any input in $\widetilde{O}(n^2)$ time.*

Note that $n^{2-o(1)}$ is the conjectured complexity of the OV problem; this hardness of OV follows from the Strong Exponential Time Hypothesis (SETH, which states, very roughly, that k-SAT cannot be solved in the worst-case in time $2^{(1-\delta)n}$ for any constant $\delta > 0$) due to a reduction by Williams [Wil05].

We also construct such functions that are similarly average-case hard based on the conjectured worst-case hardness of $\mathsf{3SUM}$ and APSP. These average-case hard problems, in addition, happen to possess a number of desirable properties, which we take advantage of to construct Proof of Work protocols, as described next.

1.2.2 Proofs of Work

Proofs of Work (PoWs), introduced by Dwork and Naor [DN92], are protocols that allow a prover to prove to a verifier that it has performed a certain amount of com-

putational work. Proofs of Work have shown themselves to be an invaluable cryptographic primitive. Originally introduced to combat Denial of Service attacks and email spam, they now serve as the heart of a number of modern cryptocurrencies. By quickly generating easily verifiable challenges that require a certain amount of work to solve, PoWs ensure that adversaries attempting to swarm a system must have a large amount of computational power to do so.

One thing to note about a PoW is that it is inherently a fine-grained cryptographic primitive – we typically want the honest prover to run in a certain polynomial amount of time, while disabling any prover strategy that tries to run in less. Using the above fine-grained average-case reductions, along with a theorem of Williams [Wil16] and a direct sum theorem described shortly, we construct PoWs based on the conjectured worst-case hardness of OV, $\mathsf{3SUM}$, or APSP. The following is an example of a theorem we show, this one based on the hardness of OV.

Informal Theorem 2. *Suppose* OV *takes* $n^{2-o(1)}$ *time to decide. A challenge* $\boldsymbol{c}$ *can be generated in* $\widetilde{O}(n)$ *time such that:*

- *A valid solution* $\boldsymbol{\pi}$ *to* $\boldsymbol{c}$ *can be computed in* $\widetilde{O}(n^2)$ *time.*
- *The validity of a candidate solution to* $\boldsymbol{c}$ *can be verified in* $\widetilde{O}(n)$ *time.*
- *Any valid solution to* $\boldsymbol{c}$ *requires* $n^{2-o(1)}$ *time to compute.*

The hardness above can be scaled to $n^{k-o(1)}$ for any $k \in \mathbb{N}$ by employing a natural generalization of the OV problem, called the k-OV problem, whose hardness is also supported by SETH. As the verification in this case can still be done in $\widetilde{O}(n)$ time, this allows us to *tune* the hardness of the PoW.

An important property of PoWs is non-batchability; for several applications, it is desired that a prover not be able to participate successfully in some ℓ instances of a PoW protocol in less than ℓ times the time it would take to participate in one instance. We show that our PoW satisfies this property using the following lemma, which may be of independent interest, and is a direct sum theorem for computing the function $f\mathsf{OV}$ mentioned earlier.

Informal Theorem 3. *Suppose* OV *takes* $n^{2-o(1)}$ *time to decide. Then, for any polynomial* ℓ*, any algorithm that computes* $f\mathsf{OV}(\boldsymbol{x}_i)$*'s correctly on* ℓ *uniformaly random* x_i*'s with probability* $1/n^{O(1)}$ *takes time* $\ell \cdot n^{2-o(1)}$*.*

Further, our PoWs allow the prover to prove that it has done work with zero knowledge; that is, such that the proofs can be simulated in very low complexity – in time comparable to the verification time. While previously known PoWs could also be proved in zero knowledge as they essentially involve proving NP statements, the exact polynomial time complexities matter in this regime. We are able to use the algebraic structure of our problem to attain a notion of zero knowledge that makes sense in the fine-grained world.

An important open question following our work in this setting is the possibility of constructing other fine-grained cryptographic objects – say fine-grained one-way functions, to start with – based on the hardness of such well-studied problems from the area of fine-grained complexity. We discuss the possibilities and barriers in this regard in Section 5.2.

1.3 Cryptography Against Bounded Circuit Depth

Another measure of computational complexity we consider in this thesis is circuit-depth or parallel-time. That is, we consider adversaries (and honest parties) that are computable by Boolean circuits of some bounded depth.

We study two classes of low-depth circuits. 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*.

1.3.1 Cryptography against AC^0

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]. Other very recent progress on circuit lower bounds also show the hardness of computing explicit functions with a lot of structure [GK15, FGHK15]. 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 pseudo-randomness 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*? We show positive answers to some of these questions, and construct the following primitives that are computable in and (unconditionally) secure against AC^0:

- Weak Pseudo-Random Functions
- Symmetric Key Encryption
- Collision Resistant Hash Functions

A conspicuous question left open by our work here is the construction of Public-Key Encryption for AC^0. We have a candidate scheme that we are unable to prove secure, but also unable to break. Its security is connected to an interesting and simple open problem, which is stated in Section 4.2.5.

1.3.2 Cryptography against NC^1

Our second contribution in this direction 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 $\mathsf{ACC}^0[2]$ (constant-depth circuits with $\mathsf{MOD2}$ gates; note that $\mathsf{ACC}^0[2] \subsetneq \mathsf{NC}^1$ [Raz87, Smo87]), and the scheme is semantically secure against NC^1 adversaries. (For simplicity, $\oplus\mathsf{L}/\mathsf{poly}$ 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.)

Organization

The layout of the rest of the thesis is as follows:

1. Chapter 2 describes our fine-grained worst-case to average-case reductions from OV, $\mathsf{3SUM}$, and APSP.

2. Chapter 3 covers our construction of Proofs of Work using the above reductions and the relevant worst-case hardness conjectures.

3. Chapter 4 describes our constructions of fine-grained cryptographic primitives against low-depth circuits.

4. Chapter 5 concludes the thesis with a summary of subsequent related work by others, an outline of an approach for some future work, and a list of open problems.

The appendices contain some extensions and strengthenings of some theorems and lemmas from the thesis.

The content of this thesis is drawn from the following papers:

1. Marshall Ball, Alon Rosen, Manuel Sabin, and Prashant Nalini Vasudevan. Average-case fine-grained hardness. In Hamed Hatami, Pierre McKenzie, and

Valerie King, editors, *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 483–496. ACM, 2017

2. Marshall Ball, Alon Rosen, Manuel Sabin, and Prashant Nalini Vasudevan. Proofs of work from worst-case assumptions. *IACR Cryptology ePrint Archive*, 2018:559, 2018. To appear in CRYPTO 2018

3. Akshay Degwekar, Vinod Vaikuntanathan, and Prashant Nalini Vasudevan. Fine-grained cryptography. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part III*, volume 9816 of *Lecture Notes in Computer Science*, pages 533–562. Springer, 2016

Chapter 2

Average-Case Fine-Grained Hardness

In this chapter, we present our fine-grained worst-case to average-case reductions starting from a number of well-studied problems from the area of fine-grained complexity. Apart from our cryptographic motivations as outlined in Chapter 1, there are other reasons for interest in such reductions, as we discuss now.

Since the 1970s we have had a notion of what we consider "easy" and what we consider "hard." Polynomial-time computable has long been synonymous to efficient and easy, while showing a problem NP-complete was to condemn it as intractable. In our recent history, however, this categorization has been called into question: SAT instances, the flagship of NP-complete problems, are solved on the daily [BHvM09], while algorithms that run in as little as quadratic time may be prohibitively expensive for some practical problems such as DNA sequencing, due to large input sizes.

Thus, in the "real world," our notions of easy and hard may not always align with our classical views. The main problem here is our choice of analysis. For SAT, we classify it as "hard" when it often may be more appropriately classified as "easy" because complexity theory typically employs *worst-case* analysis. That is, we may be adhering to an overly-pessimistc metric, when, in practice, the SAT instances we come across may be much more benign. In part to combat this sort of problem, average-case complexity was introduced in [Lev86]. By considering distributions over problem instances, we can at least hope to argue about the performance of heuristic algorithms in practice.

Similarly, the *practical hardness* of a problem with quadratic time complexity is invisible to our typical "coarse-grained" analysis that only distinguishes between polynomial and not polynomial. Within the past decade, the field of fine-grained complexity has quickly developed [Wil15], mapping out (conditional) hardness of natural problems *within* P. By introducing fine-grained reductions, a picture is emerging of a few main islands amongst the web of reductions, giving us an increasingly clearer classification of the relative hardness of fine-grained problems. Through such reductions, the more exact practical hardness of problems, such as DNA sequencing's quadratic time barrier [BI14], has been given evidence for.

However, while average-case analysis and fine-grained analysis independently address issues in classical complexity theory, average-case analysis is still coarse-grained and fine-grained analysis is still worst-case. A more complete theory attempting to capture the notion of "complexity" in our world should begin by marrying average-case and fine-grained analysis.

In this chapter we do so by providing average-case fine-grained hardness conjectures and show them to follow from widely conjectured worst-case assumptions on well-studied fine-grained problems. Alternatively viewed, we give new routes for the falsifications of these worst-case conjectures.

We present worst-case–to–average-case fine-grained reductions from the three main islands of fine-grained complexity theory. We recall these three problems here to frame our work, and their relevance is discussed in Section 2.1.

- *Orthogonal Vectors:* The OV problem on vectors of dimension d (denoted OV_d) is to determine, given two sets U, V of n vectors from $\{0,1\}^{d(n)}$ each, whether there exist $u \in U$ and $v \in V$ such that $\langle u, v\rangle = 0$ (over $\mathbb{Z}$). (If left unspecified, d is to be taken to be $\lceil \log^2 n \rceil$.)

- *3SUM:* The $\mathsf{3SUM}$ problem is to determine whether a given set $S \subset \{-n^3, \ldots, n^3\}$ of size n contains three distinct elements a, b, c such that $a + b = c$.

- *All-Pairs Shortest Path*: Given an edge-weighted (undirected or directed) graph on n vertices, the APSP problem is to find the distances between every pair of

vertices, where edge weights are in $\{1, \ldots, n^c\}$ for some sufficiently large c.

We give a family of polynomials over finite fields corresponding to each of these, called $\mathcal{F}$OV, $\mathcal{F}$3SUM, and $\mathcal{F}$ZWT respectively, and conjecture these polynomials to be hard to evaluate on uniformly chosen inputs. To support these conjectures we prove worst-case–to–average-case fine-grained reductions from OV, 3SUM, and APSP to their respective families of polynomials (where 3SUM reduces also to $\mathcal{F}$ZWT). Specifically, we show:

- If OV requires $n^{2-o(1)}$ time to decide in the worst-case, then $\mathcal{F}$OV requires $n^{2-o(1)}$ time to evaluate with probability $3/4$ on uniformly chosen inputs.
- If 3SUM requires $n^{2-o(1)}$ time to decide in the worst-case, then $\mathcal{F}$3SUM requires $n^{2-o(1)}$ time to evaluate with probability $3/4$ on uniformly chosen inputs.
- If APSP requires $n^{3-o(1)}$ time *or* 3SUM requires $n^{2-o(1)}$ time to decide in the worst-case, then $\mathcal{F}$ZWT requires $n^{3-o(1)}$ time to evaluate with probability $3/4$ on uniformly chosen inputs.

Further, we conjecture a fourth family of polynomials, $\mathcal{F}$TC, to also be average-case hard to evaluate and support this with fine-grained reductions from 3SUM, and APSP, and k-SAT. The reduction from k-SAT makes $\mathcal{F}$TC hard under the *Strong Exponential Time Hypothesis* (SETH), which states that there is no $\epsilon > 0$ such that k-SAT can be solved in time $\widetilde{O}(2^{n(1-\epsilon)})$ for all values of k.

- If *either* APSP requires $n^{3-o(1)}$ time, 3SUM requires $n^{2-o(1)}$ time, *or* SETH holds, then $\mathcal{F}$TC requires $n^{3-o(1)}$ time to evaluate with probability $3/4$ on uniformly chosen inputs.

We note that SETH implies that OV requires $n^{2-o(1)}$ time to decide in the worst-case and so $\mathcal{F}$OV is also hard on average under the stronger assumption of SETH. Thus, $\mathcal{F}$OV and $\mathcal{F}$TC are our mostly strongly supported average-case hardness results. $\mathcal{F}$TC only can become easy if SETH breaks and both 3SUM and APSP become easy,

while $\mathcal{F}\mathsf{OV}$, even with a broken SETH, remains hard unless *all* first-order graph problems become easy since [GI16] shows that all such problems reduce to (moderate-dimension) OV.

Our results crucially rely on the fact that the polynomials in $\mathcal{F}\mathsf{OV}$, $\mathcal{F}\mathsf{3SUM}$, $\mathcal{F}\mathsf{ZWT}$, and $\mathcal{F}\mathsf{TC}$ have degree polylog(n), which is very low. This extremely low degree enables us to invoke *in a fine-grained way* the classic random self-reducibility of evaluating low-degree polynomials, first used to show the average-case hardness of computing the Permanent when assuming its worst-case hardness [Lip91, FF90], or more generally to show local correctability of Reed-Muller codes [GS92] (see also the work of Blum, Luby and Rubinfeld [BLR93] for related examples of self-reductions).

Beyond low degree, our polynomials are efficient to evaluate in time that *tightly matches* their conjectured average-case hardness. These two properties are what distinguishes our polynomials from naïvely representing a decision problem with a multilinear extension, which has "low" degree n and, having exponentially many terms, can take exponential time to compute, both of which are too large for our fine-grained analysis. A key technique we use to bypass the naïve construction, then, is to look at the structure of specific problems from fine-grained complexity to tailor *very* low-degree efficiently-computable polynomials to them. The matching upper and lower bounds are precisely what allow us to capture the complexity of our problems in the fine-grained setting and open the door for applications.

Extensions. We extend our results to a generalization of OV, called k-OV, whose hardness can also be based on the SETH. To this end, we define a corresponding polynomial $\mathcal{F}\mathsf{OV}^k$ which is computable in $\widetilde{O}(n^k)$ time in the worst-case. Using the same ideas as in the above worst-case to average-case reductions we show:

- If k-OV requires $n^{k-o(1)}$ time to decide in the worst-case, then $\mathcal{F}\mathsf{OV}^k$ requires $n^{k-o(1)}$ time to evaluate with probability $3/4$ on uniformly chosen inputs.

We note that this yields a tight average-case time hierarchy: an average-case problem computable in time n^k but not much faster for every integer k (these can be

extended to rational numbers through standard padding techniques). Unconditional average-case time hierarchies are known – e.g. [GGH94] – but these are based on canonical functions that have not found further use than as a proof of this concept, while our functions are closely related to well-studied problems and have considerable algebraic structure. We discuss this in more detail in Appendix A.1.

Building on [CPS99], we use local list decoding to show that our families of polynomials remain hard even when asking for their successful evaluation on just a $1/\text{polylog}(n)$ fraction of inputs. We extend this to attain a smooth trade-off between the running time and the upper bound on the probability of success of any average-case solver. Details may be found in Appendix A.3.

We additionally show, in Appendix A.5, that $\mathcal{F}\mathsf{OV}$ remains hard to evaluate even over very small field sizes by applying an isolation lemma to OV, which may be of independent interest by itself.

Applications. Leveraging the structure of our problems, and using ideas from [Wil16], in Chapter 3 we construct a *Proof of Work* scheme based on their average-case hardness. We pose the application of creating fine-grained cryptography as an open problem, and in Chapter 5 outline a structural barrier to achieving fine-grained one-way functions from our results.

Finally, we note that these reductions set up a win-win scenario: either the worst-case conjectures are true and we have average-case fine-grained hard problems *or* we can show them easy and thus break our worst-case conjectures, allowing a breakthrough in fine-grained complexity theory. We explore this notion further by considering the ideas introduced in [Nao03] of falsifiable assumptions to show that our results make the OV, $\mathsf{3SUM}$, and APSP conjectures falsifiable in a practical sense. Specifically, in Appendix A.2 we discuss how *empirically* evaluating our polynomial for OV faster than we conjecture possible would give strong heuristic evidence that SAT has faster worst-case algorithms – i.e. that the SETH is false. In this sense, we discuss how our results allow for the *heuristic falsifiability* of conjectures.

Related Work

Recently, and independently of our work, a sequence of papers [BK16b, GR17, Wil16] has also observed that a number of problems from the fine-grained world can be captured by the evaluation of efficiently computable low-degree polynomials.The focus of these papers has been on using the algebraic structure and low-degree of the polynomials to *delegate computation* of fine-grained problems in quickly verifiable ways. The papers were not, however, concerned with the hardness aspects of complexity and made no average-case claims or guarantees.

A key discovery, then, achieved here and independently in [BK16b, GR17, Wil16], is the utility of looking at the structure of specific computational problems to tailor *very* low-degree polynomials that are efficiently computable to them. From the algorithmic perspective, [BK16b, GR17, Wil16] find utility for delegation of computation, while, from the hardness perspective, we connect it to average-case complexity and applications thereof.

This gives a very rich framework that our results are applicable to, as our worst-case to average-case reductions can be adapted in a straightforward way to work for any problem appropriately expressible as a low-degree polynomial. Many other low-degree polynomials for interesting practical problems are found in [BK16b, GR17, Wil16], with [Wil16] independently discovering our $\mathcal{F}\mathsf{OV}$ polynomial and [BK16b] independently finding a polynomial similar to our $\mathcal{F}\mathsf{3SUM}$. The work of [GR17], in fact, identifies a natural class of "locally characterizable sets" that contains problems admitting low-degree polynomials akin to the ones considered here. Many of the above polynomials can fit into the framework of applications of average-case fine-grained hardness, such as the Proofs of Work we introduce.

The notion of precise cryptography introduced by Micali and Pass [MP06] studies reductions between cryptographic primitives that (among other things) 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. While they are also interested in tight

reductions (similar to our quasi-linear time average-case reductions), their notions are stronger as they refer to the complexity of reductions and simulations on a per-instance basis. Similar to our case, their considerations also remain relevant even if $\mathsf{P} \neq \mathsf{NP}$.

Lastly, [GH16] recently shows that fine-grained problems related to DNA sequencing are actually *easy* when given a batch of correlated instances. While these correlated instances are not typically what we consider in average-case complexity, they are distributional notions of inputs on fine-grained problems and so seems to be the closest existing work to average-case fine-grained complexity. The techniques used, however, are very different and focused on attaining *easiness* for specific problems with respect to specific distributions, whereas we focus on attaining hardness and applications of hardness and do so within the emerging low-degree polynomial framework. However, [GH16] can also be used to make claims similar to our notion of *heuristic falsifiability*, suggesting that whenever the intersection of average-case complexity and fine-grained complexity is considered it may immediately bear interesting fruit for this notion.

2.1 Worst-Case Conjectures

We present here a few well-studied problems in fine-grained complexity and conjectures about their worst-case hardness that we use to support our average-case hardness conjectures. For a more comprehensive survey of fine-grained complexity, connections between problems, and formal definitions of concepts like fine-grained reductions, see [Wil15].

(All our discussion will be in the Word RAM model of computation with $O(\log(n))$-bit words. When we speak of randomized algorithms in a worst-case setting, we mean algorithms that, for every input, output the correct answer with probability at least 2/3. And unless specified otherwise, all algorithms and conjectures about algorithms are randomized throughout the paper.)

First we recall the problems of OV, $\mathsf{3SUM}$, and APSP defined earlier. These

problems currently remain the three key problems of fine-grained complexity; there are no known reductions between them, but they reduce to many other problems and, thus, give us the basis for what we generally call hardness within P [Wil15]. This foundation is more formally given, after extensive attempts to find improved algorithms for them, through the following popular hardness conjectures (see [Wil15] for relevant discussion and references):

- OV Conjecture: For any $d = \omega(\log n)$, any algorithm for OV_d requires $n^{2-o(1)}$ time.
- 3SUM Conjecture: Any algorithm for 3SUM requires $n^{2-o(1)}$ time.
- APSP Conjecture: Any algorithm for APSP requires $n^{3-o(1)}$ time.

These conjectures are not only important because they help stratify P, but the truth or falsity of each of them has many ramifications to practical problems.

It has been shown that if the the OV conjecture is true, then many string processing problems, hugely relevant to DNA sequencing and data comparison, also have hardness bounds (typically sub-quadratic) [AWW14, ABW15b, BI14, BK15]. On the other hand, if a sub-quadratic algorithm for OV is found, Williams [Wil05] gave a reduction to show we would achieve improved algorithms for SAT; more specifically, the well-known Strong Exponential Time Hypothesis (SETH) would break. Thus, as stated in earlier in this chapter, SETH implies the OV conjecture.

(We note that the results in this chapter still go through under a slightly weaker variant of the OV conjecture: for all $\varepsilon > 0$, there is no $O(n^{2-\varepsilon}\mathsf{poly}(d))$ algorithm for OV_d. This problem, "Moderate Dimension" OV, where d could be as large as a small polynomial in n, was shown to be hard for the class of all first-order graph problems in [GI16].)

Similarly, if the 3SUM conjecture is true, problems in computational geometry [GO95] and exact weighted subgraph problems [AL13] are moderately hard. Further, [AL13] shows that if the 3SUM conjecture is false, we get improved algorithms for many of the same graph problems.

Finally, the APSP conjecture's truth would give lower bounds for many problems in dense graphs [WW10] and for dynamic problems [RZ04]. [WW10] also shows that the conjecture being false gives better algorithms for the dense graph problems.

We note that while it is common to make these conjectures only for deterministic algorithms, we see these as structural beliefs about the problems – that brute force is essentially necessary as there is no structure to algorithmically exploit – and so there is no reason to believe that allowing randomness will allow a significant speed-up. Indeed, these conjecture are often made against randomized expected running time machines as in [Wil15] and randomized one-sided error versions of SETH have been made in [DHW10] and [CIKP03]. Further, [CFK+15] conjectures and argues for a SETH under randomized two-sided error machines (which are the machines we assume and state our conjectures for).

Besides these three main islands of fine-grained complexity theory, a fourth seems to be emerging based on the k-CLIQUE problem. With the current best algorithm solving the problem in $n^{\omega k/3}$ time [NP85], where ω is the matrix multiplication constant, there has been recent work showing that conjecturing this to be optimal leads to interesting hardness results for other important problems such as parsing languages and RNA folding [ABW15a, BGL16, BDT16, BT16]. To explore delegating computation, [BK16b, GR17, Wil16] all introduce different families of polynomials to express the k-CLIQUE problem, yet none yield analysis akin to those above. The polynomials either yield average-case hardness via our techniques but cannot be computed efficiently enough to give matching upper bounds [GR17, Wil16], or they have too large of a degree for our worst-case to average-case reductions [BK16b]. We leave the open problem of finding a family of polynomials to represent k-CLIQUE that both are computable in time $n^{\omega k/3}$ and have degree $n^{o(1)}$.

For these practical connections, any reduction to or from the main island problems has interesting consequences (see [Wil15] for a more comprehensive treatment). In our results we achieve reductions *from* these problems and, to help facilitate that, we recall two more problems.

- *Zero-Weight Triangle:* Given an edge-weighted graph on n vertices, the ZWT

problem is to decide whether there exists a triangle with edge weights w_1, w_2, w_3 such that $w_1 + w_2 = -w_3$, where edge weights are in $\{-n^c, \ldots, n^c\}$ for some sufficiently large c.

- *Triangle-Collection:* Given an graph on n vertices and a partition C of the vertices into colors, the Triangle-Collection problem is to decide whether for each triple of three colors $a, b, c \in C$, there exists vertices x, y, z in the graph that form a triangle and $x \in a$, $y \in b$, and $z \in c$. That is, each triplet of colors is 'collected' by some triangle.

We will follow the approach in [CGI+16] of, at times, using ZWT as a proxy for both 3SUM and APSP. That is, both reduce in a fine-grained way to ZWT: APSP reduces to (Negative Weight Triangle [WW10] and then to) ZWT [VW09], and 3SUM has a randomized (which suits our purposes) reduction to ZWT in [VW09, PĬ0]. Thus reducing *from* ZWT reduces from both 3SUM and APSP simultaneously. It then follows that if *either* the 3SUM or APSP conjecture are true, then ZWT requires $n^{3-o(1)}$ time.

Similarly, the Triangle-Collection problem is introduced in [AWY15] as a way to base hardness on the believable conjecture that *at least one of* the SETH, 3SUM, or APSP conjectures are true. To do this, they give fine-grained reductions from *all three* of k-SAT, 3SUM, and APSP so that if any of their conjectures are true, then Triangle-Collection requires $n^{3-o(1)}$ time.

In general, it is better to reduce from problems furthest down a chain of reductions, as assuming those problems to be hard will then be the weakest assumption required - e.g. assuming Triangle-Collection requires $n^{3-o(1)}$ time is a *weaker* assumption than assuming that at least one of k-SAT, 3SUM, or APSP are hard. It is an interesting direction to base average-case fine-grained hardness on increasingly weaker assumptions.

For this reason, it would be desirable to reduce from some very practical DNA sequencing problems (e.g. EDIT-DISTANCE and LCS) that are reduced to *from* OV (and thus k-SAT). Further, there is mounting evidence that, regardless of the status

of k-SAT's complexity, these DNA sequencing problems are in fact *very* likely to be hard [GI16, AHWW15]. We remark, however, that there is a barrier to representing these problems with low-degree polynomials [Abb17]. Namely, representing them with low-degree polynomials would allow for small speedups – i.e. by using the polynomial method [CW16] – but such speedups (of just shaving some logarithmic factors off of the runtime) have been show to imply new breakthroughs in circuit lower bounds [AHWW15].

2.2 Average-Case Fine-Grained Hardness

We now define the notion of average-case complexity that we shall use and describe the technique we use for our worst-case to average-case reductions. Then, we describe the problems we conjecture to be hard on average and show reductions from the worst-case problems described in Section 2.1 in support of these conjectures.

Definition 1. A family of functions $\mathcal{F} = \{f_n\}$ is *computable in time t on average* if there is an algorithm that runs in $t(n)$ time on the domain of f_n and, for all large enough n, computes f_n correctly with probability at least $3/4$ over the uniform distribution of inputs in its domain.

For broader definitions that are more useful when one is concerned with whole classes of problems rather than a handful of specific ones, and for extensive discussions of the merits of the same, we refer the reader to Bogdanov and Trevisan's survey [BT06].

To achieve average-case hardness for our fine-grained problems, our main technique will be to "express" these problems as low-degree polynomials and then use the random self-reducibility of evaluating these polynomials to attain average-case hard problems.

We now recall the classic random self-reducibility of evaluating low-degree polynomials, first used to show the average-case hardness of computing the Permanent when assuming its worst-case hardness [Lip91, FF90]. We can more generally view

this as the local correctability of Reed-Muller codes first shown by Gemmell and Sudan [GS92] and get better error rates using techniques from this perspective. We repeat the proof in Section 2.3 in order to accurately assess the running time of the algorithm involved.

Lemma 1. *Consider positive integers N, D, and p, and an $\varepsilon \in (0, 1/3)$ such that $D > 9$, p is prime and $p > 12D$. Suppose that for some polynomial $f : \mathbb{F}_p^N \to \mathbb{F}_p$ of degree D, there is an algorithm A running in time t such that A computes f correctly on average. That is,*

$$\Pr_{\boldsymbol{x} \leftarrow \mathbb{F}_p^N} [A(\boldsymbol{x}) = f(\boldsymbol{x})] \geq 1 - \varepsilon$$

Then there is a randomized algorithm B that runs in time $O(ND^2 \log^2 p + D^3 + tD)$ such that B computes f in the worst-case. That is, for any $x \in \mathbb{F}_p^N$:

$$\Pr [B(\boldsymbol{x}) = f(\boldsymbol{x})] \geq \frac{2}{3}$$

Remark 2. The range of ε being $(0, 1/3)$ is arbitrary to some extent. It could be any constant smaller than $1/2$ at the cost of p having to be slightly larger.

Remark 3. An important thing to note here is how B's runtime depends on f's degree D. Assuming tD is the high-order term in the runtime, B runs in time $O(tD)$. So if we want our reductions to have low overhead, we will need D to be rather small. For our fine-grained purposes, we need to be careful in what we consider "low" and we will see that we always have D polylogarithmic in N.

We now introduce three families of polynomials that we conjecture average-case hard to evaluate and then give evidence for this by reducing to them from the worst-case problems OV, ZWT, and Triangle-Collection, respectively, and then applying the random self-reducibility of low-degree polynomials as just described. Another family of polynomials arising from 3SUM is presented in Section 2.2.4. The landscape of these reductions is seen in Figure 2-1.

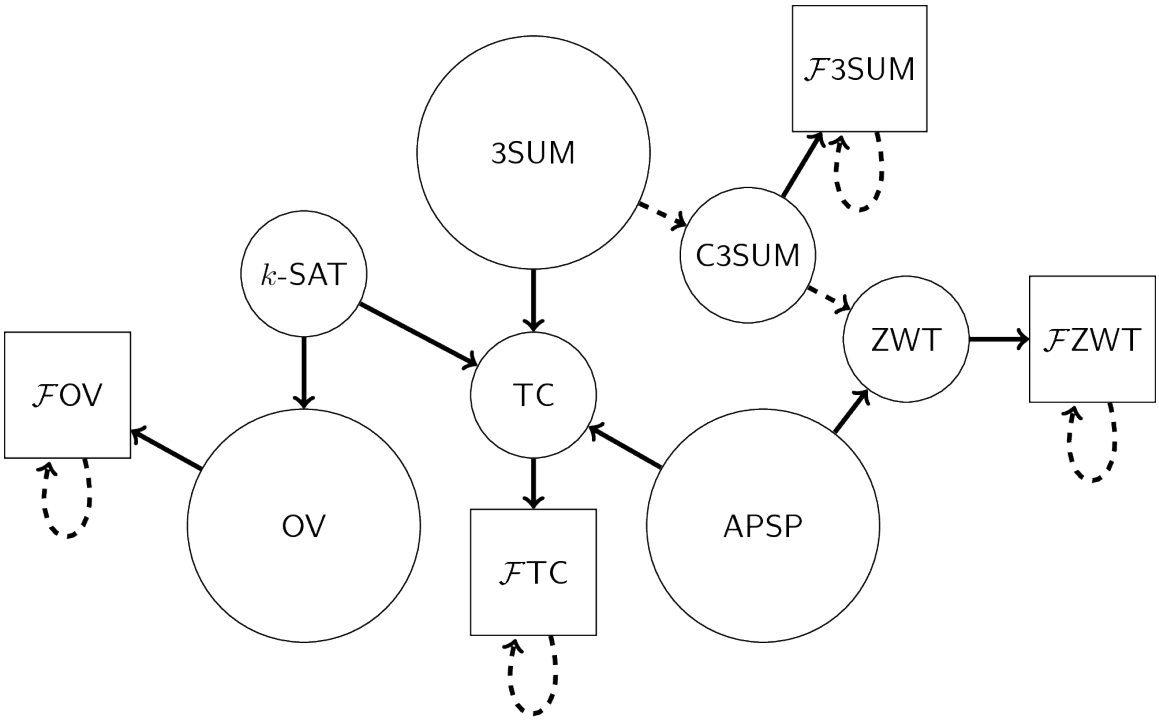

Figure 2-1: Arrows represent (fine-grained) reductions and dashed means they're randomized. A dashed self-loop is a worst-case to average-case self-reduction. Our work introduces $\mathcal{F}\mathsf{OV}$, $\mathcal{F}\mathsf{3SUM}$, $\mathcal{F}\mathsf{ZWT}$, and $\mathcal{F}\mathsf{TC}$ and the reductions involving them.

2.2.1 Orthogonal Vectors

For any n, let $p(n)$ be the smallest prime number larger than n^2, and $d(n) = \lceil \log^2 n \rceil$ (for brevity, we shall write just p and d). We define polynomials $f\mathsf{OV}_n : \mathbb{F}_p^{2nd} \to \mathbb{F}_p$ over $2nd$ variables. We view these variables as representing the input to OV – we separate the variables into two matrices $U, V \in \mathbb{F}_p^{n \times d}$. The polynomial $f\mathsf{OV}_n$ is then defined as follows:

$$f\mathsf{OV}_n(U,V) = \sum_{i,j \in [n]} \prod_{\ell \in [d]} (1 - u_{i\ell} v_{j\ell})$$

A similar polynomial was used independently by Williams [Wil16] to construct coMA proof systems for OV with efficient verifiers. Given an OV instance $(U,V) \in \{0,1\}^{2nd}$, $f\mathsf{OV}_n(U,V)$ counts the number of pairs of orthogonal vectors in it – for

each pair $i, j \in [n]$, the corresponding summand is 1 if $\langle u_i, v_j \rangle = 0$, and 0 otherwise (there is no modular wrap-around of the sum as $p > n^2$). Also, $f\mathsf{OV}_n$ has degree at most $2d$, which is rather low.

Define the family of polynomials $\mathcal{F}\mathsf{OV} = \{f\mathsf{OV}_n\}$. We show a worst-case to average-case reduction from OV to $\mathcal{F}\mathsf{OV}$ that, given an algorithm that computes $f\mathsf{OV}_n$ well on average, decides OV on instances of length n without much overhead. This is stated as the following theorem.

Theorem 1. *If $\mathcal{F}\mathsf{OV}$ can be computed in $O(n^{1+\alpha})$ time on average for some $\alpha > 0$, then OV can be decided in $\widetilde{O}(n^{1+\alpha})$ time in the worst case.*

Proof. Suppose there were an algorithm A that ran in $O(n^{1+\alpha})$ time and computed $f\mathsf{OV}_n$ correctly on more than a 3/4 fraction of inputs for all large enough n.

In order to be able to use such an average-case algorithm, however, one has to be able to write down inputs to run it on. These inputs to $f\mathsf{OV}_n$ are in $\mathbb{F}_p^{2nd}$, and so to work with them it is necessary to know $p = p(n)$, the smallest prime number larger than n^2. Further, p would have to be computable from n rather efficiently for a reduction that uses A to be efficient. As the following lemma states, this turns out to be possible to do.

Lemma 2 (Implied by [LO87])**.** *The smallest prime number greater than m can be computed deterministically in $\widetilde{O}(m^{1/2+\alpha})$ time for any $\alpha > 0$.*

We will then use A and this lemma to decide OV as follows. Given an input $(U, V) \in \{0,1\}^{2nd}$, first compute $p = p(n)$ – this can be done in $\widetilde{O}(n^{1+\alpha})$ time by Lemma 2. Once p is known, A can be used along with Lemma 1 to compute $f\mathsf{OV}_n(U, V)$ in $O(n(2d)^2 \log^2 p + (2d)^3 + 2dn^{1+\alpha}) = \widetilde{O}(n^{1+\alpha})$ time, and this immediately indicates membership in OV as observed above. □

Corollary 1. *If OV requires $n^{2-o(1)}$ time to decide, $\mathcal{F}\mathsf{OV}$ requires $n^{2-o(1)}$ time to compute on average.*

Note that our result is then *tight* under the OV conjecture in the sense that our polynomial is computable in $\widetilde{O}(n^2)$ time, but in no less (even on average) assuming sub-quadratic hardness of OV. That is, we demonstrate a problem that is

quadratic-computable but sub-quadratic-hard on average. It should also be noted that our results can adapted the Moderate Dimension OV problem (as mentioned in Section 2.1) and thus an appropriately parametrized variant of $\mathcal{F}\mathsf{OV}$ is average-case hard for the class of all first-order graph problems as defined in [GI16].

2.2.2 3SUM and All-Pairs Shortest Path

Recall from Section 2.1 that both $\mathsf{3SUM}$ and APSP have fine-grained reductions to ZWT, and so we restrict our attention to ZWT. We now show a family of polynomials that can count Zero Weight Triangles.

For any n, let $p(n)$ denote the smallest prime number larger than n^3 and let $d = \lceil \log(2(2n^c+1)) \rceil + 3$ (c being the constant from the definition of ZWT). We define the polynomial $f\mathsf{ZWT}_n : \mathbb{F}_p^{n^2 d} \to \mathbb{F}_p$ as taking in a set E of $n^2 d$ variables where we split them into n^2 sets, w_{ij}, of d variables each for all $i, j \in [n]$:

$$f\mathsf{ZWT}_n(E) = \sum_{i,j,k \in [n]} \prod_{\ell \in [d]} \left(1 - \left(s_\ell\left(w_{ij}, w_{jk}\right) - s_\ell\left(\overline{w}_{ik}, 0 \ldots 01\right)\right)^2\right)$$

where $s_\ell : \mathbb{F}_p^{2d} \to \mathbb{F}_p$ is the polynomial such that if $x, y \in \{0,1\}^d$, then $s_\ell(x,y)$ equals the ℓ^{th} bit of $(x+y)$ as long as x and y represent numbers in $[-n^c, n^c]$. Such polynomials exist, have degree at most $2d$, and are computable in $O(d \log^2 p)$ time – see Appendix A.4. Further, $\overline{w}_{ik}$ represents the set of linear polynomials that toggle all the bits in a boolean valued w_{ij}; so $s_\ell\left(\overline{w}_{ik}, 0 \ldots 01\right)$ effectively takes the one's complement of w_{ij} and then adds 1, which is exactly the two's complement of w_{ij}.

Now, considering a graph on n vertices with edges weighted from $[-n^c, \ldots, n^c]$. We use this polynomial to count zero weight triangles in it: For an edge-weight between nodes i and j we decompose the value to its bit representation in two's complement notation and now have d boolean inputs for w_{ij}. If an edge does not exist between an i and j, we similarly put the bit decomposition of the value $2n^c + 1$ into w_{ij} (note that $i = j$ is possible and we consider there to *not* be an edge for this). Conceptually,

we now have weights w_{ij} corresponding to a complete graph on n vertices with the the non-edges added at weight $2n^c + 1$. Note that each triangle in it is zero weight if and only if it was a zero weight triangle in the original graph. Thus, collecting these all together we have boolean input $E \in \{0,1\}^{n^2 d}$. This reduction certainly takes sub-cubic time.

Then, given the binary representation of a ZWT instance, the ℓ^{th} term in the product above checks whether the ℓ^{th} bit of the sum of w_{ij} and w_{jk} equals that of the negation of w_{ik}. If all d bits are equal, then, and only then, the summand is 1, otherwise it is 0. So the sum counts the number of triples of distinct (i,j), (j,k), and (i,k) such that $w_{ij} + w_{jk} = -w_{ik}$. Also, the degree of $f\mathsf{ZWT}_n$ is at most $4d^3 = O(\log^3 n)$.

Define the family of polynomials $\mathcal{F}\mathsf{ZWT} = \{f\mathsf{ZWT}_n\}$. The following theorem can be proved identically to Theorem 1.

Theorem 2. *If $\mathcal{F}$ZWT can be computed in $O(n^{1.5+\alpha})$ time on average for some $\alpha > 0$, then ZWT can be decided in $\widetilde{O}(n^{1.5+\alpha})$ time in the worst case.*

Corollary 2. *If ZWT requires $n^{3-o(1)}$ time to decide, $\mathcal{F}$ZWT requires $n^{3-o(1)}$ time to compute on average.*

Thus, assuming the ZWT conjecture, using the fact that $f\mathsf{ZWT}_n$ has n^3 terms and each s_ℓ is computable in $O(d)$ time, we again achieve tightness where $\mathcal{F}$ZWT is cubic-computable but sub-cubic-hard. It is also worth noting that the following corollary frames our result in the more familiar problems of 3SUM and APSP.

Corollary 3. *If* either 3SUM *requires $n^{2-o(1)}$ time or* APSP *requires $n^{3-o(1)}$ time, then $\mathcal{F}$ZWT takes $n^{3-o(1)}$ time to compute on average.*

2.2.3 SETH, 3SUM, and All-Pairs Shortest Path

We now give our most encompassing worst-case–to–average-case result. Recall from Section 2.1 that if *any* of k-SAT, 3SUM, or APSP are hard then the Triangle-Collection problem is also hard [AWY15], thus so would be any polynomial based on it. We can

hence focus our attention on Triangle-Collection. More specifically, we will look at a restricted version of the problem called Triangle-Collection* shown to be equivalent to Triangle-Collection in [AWY15, Abb17], whose extra structure we will use to construct low-degree polynomials.

- *Triangle-Collection*:* Given an undirected tripartite node-colored graph G with n colors and $m = n\log^2 n + 2n\log^4 n$ nodes and with partitions A, B, C of the form:

  - A contains $n\log^2 n$ nodes $a_{\ell,i}$ where $i \in [n], \ell \in [\log^2 n]$ and $a_{\ell,i}$ is colored with color i.
  - B (respectively C) contains $n\log^4 n$ nodes $b_{\ell,i,x}$ (respectively $c_{\ell,i,x}$) where $i \in [n], \ell \in [\log^2 n], x \in [\log^2 n]$ and $b_{\ell,i,x}$ (respectively $c_{\ell,i,x}$) is colored with color i.
  - For each node $a_{\ell,i}$ and colors $j, k \in [n]$, there is exactly one edge from A to B of the form $(a_{\ell,i}, b_{\ell,j,x})$ and exactly one edge from A to C of the form $(a_{\ell,i}, c_{\ell,k,y})$, for some $x, y \in [\log^2 n]$.
  - A node $b_{\ell,j,x}$ can only be connected to nodes of the form $c_{\ell,k,y}$ in C. (There no edges across disparate ℓ's.)

  For all triples of distinct colors i, j, k, is there a triangle (u, v, w) in G where u has color i, v has color j, and w has color k?

We now give a polynomial whose evaluation would allow us to decide Triangle-Collection*. For any n, let $p(n)$ denote the smallest prime number larger than n^3. We define the polynomial $f\mathsf{TC}_n : \mathbb{F}_p^m \to \mathbb{F}_p$ as taking in a set E of $m = (n\log^2 n + 2n\log^4 n)^2$ variables (corresponding to entries in the adjacency matrix of an input graph to the above problem):

$$f\mathsf{TC}_n(E) = \sum_{1\le i<j<k\le n} \prod_{\substack{\ell,x,y\in[\log^2 n]\\ \pi\in S_{\{i,j,k\}}}} \left(1 - e_{a_{\ell,\pi(i)},b_{\ell,\pi(j),x}} e_{a_{\ell,\pi(i)},c_{\ell,\pi(k),y}} e_{b_{\ell,\pi(j),x},c_{\ell,\pi(k),y}}\right)$$

(Note that for a set X, S_X denotes the set of permutations on X.)

Consider a tripartite graph as defined above with adjacency matrix E. For each triple of colors, $(i, j, k) \in [n]^3$, if there is a corresponding triangle in the graph then it zeroes out that particular term, otherwise it will evaluate to one. Thus, $f\mathsf{TC}_n$ counts the number of colors not collected by a triangle – i.e. the number of violations to being a YES instance – and so, for boolean E, $f\mathsf{TC}_n(E) = 0$ if and only if E corresponds to a YES instance of Triangle-Collection*. Moreover, the degree of $f\mathsf{TC}_n$ is at most $18 \log^6 n$.

Define the family of polynomials $\mathcal{F}\mathsf{TC} = \{f\mathsf{TC}_n\}$. The following theorem can be proved identically to Theorem 1.

Theorem 3. *If $\mathcal{F}\mathsf{TC}$ can be computed in $O(n^{1.5+\alpha})$ time on average for some $\alpha > 0$, then TC can be decided in $\widetilde{O}(n^{1.5+\alpha})$ time in the worst case.*

Corollary 4. *If TC requires $n^{3-o(1)}$ time to decide, $\mathcal{F}\mathsf{TC}$ requires $n^{3-o(1)}$ time to compute on average.*

Thus, $f\mathsf{TC}_n$ only having n^3 many summands with each being computable in polylog(n) time, it is easily seen that we again achieve tightness where $\mathcal{F}\mathsf{TC}$ is cubic-computable but sub-cubic-hard. More recognizably we attain the following.

Corollary 5. *If* either SETH *holds,* $\mathsf{3SUM}$ *takes $n^{2-o(1)}$ time,* or APSP *takes $n^{3-o(1)}$ time, then $\mathcal{F}\mathsf{TC}$ takes $n^{3-o(1)}$ time to compute on average.*

Note that this does not subsume the hardness of $\mathcal{F}\mathsf{ZWT}$ as, even if SETH fails and $\mathsf{3SUM}$ and APSP become easy, the ZWT problem may still be hard and yield hardness for $\mathcal{F}\mathsf{ZWT}$.

2.2.4 CONVOLUTION-3SUM

Here we give another average-case fine-grained hard problem based on the hardness of $\mathsf{3SUM}$. While Section 2.2.2 already has a polynomial whose average-case hardness is based on the $\mathsf{3SUM}$ conjecture, we include this one for completeness as it is possible that either is independently hard even if the other is shown to be easy. We first

recall the CONVOLUTION-3SUM (C3SUM) problem introduced by [PĬ0] where it was shown that 3SUM has a (randomized) fine-grained reduction to C3SUM, thus allowing us to restrict our attention to it.

- C3SUM*:* Determine whether, when given three n-element arrays, A, B, and C, with entries in $\{-n^3, \ldots, n^3\}$, there exist $i, j \in [n]$ such that $A[i] + B[j] = C[i + j]$.

We now define a family of polynomials that can count solutions to a C3SUM instance (when given in binary) and refer the reader to Section 2.2.2 for all notation and discussion due to its similarity to $\mathcal{F}$ZWT.

For any n, let $p(n)$ denote the smallest prime number larger than n^2 and let $d = \lceil 3 \log n \rceil + 3$. We define the polynomial $f\mathsf{3SUM}_n : \mathbb{F}_p^{3nd} \to \mathbb{F}_p$ as taking in sets A, B, and C of nd variables each and where we split each set into n groups of d variables – e.g. $A[i]$ is the i^{th} group of d variables of the nd variables in A.

$$f\mathsf{3SUM}_n(A, B, C) = \sum_{i,j \in [n]} \prod_{\ell \in [d]} \left(1 - \left(s_\ell\left(A[i], B[j]\right) - C[i + j]_\ell\right)^2\right)$$

From the same arguments in Section 2.2.2, we get the following theorem for $\mathcal{F}\mathsf{3SUM} = \{f\mathsf{3SUM}_n\}$.

Theorem 4. *If* $\mathcal{F}$3SUM *can be computed in* $O(n^{1+\alpha})$ *time on average for some* $\alpha > 0$*, then* 3SUM *can be decided in* $\widetilde{O}(n^{1+\alpha})$ *time in the worst case.*

Corollary 6. *If* 3SUM *requires* $n^{2-o(1)}$ *time to decide,* $\mathcal{F}$3SUM *requires* $n^{2-o(1)}$ *time to compute on average.*

Note that if we did not use C3SUM, we would have had n^3 terms from a more naïve construction from 3SUM and thus a gap between $\mathcal{F}$3SUM's computability and its hardness. But, with our current construction having only n^2 terms, we achieve tightness where $\mathcal{F}$3SUM is quadratic-computable but sub-quadratic-hard.

2.3 Evaluating Low Degree Polynomials

Here we recall and prove Lemma 1.

Lemma 1. *Consider positive integers N, D, and p, and an $\varepsilon \in (0, 1/3)$ such that $D > 9$, p is prime and $p > 12D$. Suppose that for some polynomial $f : \mathbb{F}_p^N \to \mathbb{F}_p$ of degree D, there is an algorithm A running in time t such that A computes f correctly on average. That is,*

$$\Pr_{\boldsymbol{x} \leftarrow \mathbb{F}_p^N} [A(\boldsymbol{x}) = f(\boldsymbol{x})] \geq 1 - \varepsilon$$

Then there is a randomized algorithm B that runs in time $O(ND^2 \log^2 p + D^3 + tD)$ such that B computes f in the worst-case. That is, for any $x \in \mathbb{F}_p^N$:

$$\Pr [B(\boldsymbol{x}) = f(\boldsymbol{x})] \geq \frac{2}{3}$$

Proof. The algorithm B works as follows on input $\boldsymbol{x}$:

1. Draw two random points $\boldsymbol{y}_1, \boldsymbol{y}_2 \in \mathbb{F}_p^N$ and define the curve $\boldsymbol{c}(w) = \boldsymbol{x} + w\boldsymbol{y}_1 + w^2\boldsymbol{y}_2$ for $w \in \mathbb{F}_p$.

2. For a value of m ($< p$) to be determined later, compute $(A(\boldsymbol{c}(1)), \ldots, A(\boldsymbol{c}(m)))$ to get $(z_1, \ldots, z_m) \in \mathbb{F}_p$.

3. Run Berlekamp-Welch on $(z_1, \ldots, z_m)$. If it succeeds and outputs a polynomial $\hat{g}$, output $\hat{g}(0)$. Otherwise output 0.

To see why the above algorithm works, define the polynomial $g(w) = f(\boldsymbol{c}(w))$. g is a polynomial of degree $2D$ over the single variable w, with the property that $g(0) = f(\boldsymbol{x})$. So if we had $(2D+1)$ evaluations of g at different points in $\mathbb{F}_p$, we could retrieve g and compute $f(\boldsymbol{x})$. While we may not be able to obtain these evaluations directly (since they involve computing f), we do have access to A, which promises to be correct about the value of f on a random point with probability $2/3$.

So we replace the values $g(w) = f(\boldsymbol{c}(w))$ with $A(\boldsymbol{c}(w))$, which will hopefully be correct at several points. Now our problem is to retrieve g given a set of its alleged evaluations at m points, some of which may be wrong.

We do this by interpreting $(g(1), g(2), \ldots, g(m))$ as a Reed-Solomon encoding of g and running its decoding algorithm on $(A(\boldsymbol{c}(1)), \ldots, A(\boldsymbol{c}(m)))$, which now corresponds to a corrupt codeword. The Berlekamp-Welch algorithm can do this as long as less than $(m - 2D)/2$ of the m values of $A(\boldsymbol{c}(w))$ are wrong. We now bound the probability of too many of these values being wrong.

Let Q_w be an indicator variable such that $Q_w = 1$ if and only if $A(\boldsymbol{c}(w)) \neq f(\boldsymbol{c}(w))$. Let $Q = \sum_{w=1}^{m} Q_w$. We note at this point the fact that over the randomness of $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$, the distributions of $\boldsymbol{c}(w)$ and $\boldsymbol{c}(w')$ for any two distinct $w, w' \in \mathbb{F}_p$ are uniform and independent. This gives us the following statistics:

$$\begin{aligned} \mathrm{E}[Q] &= \varepsilon m \\ \mathrm{Var}[Q] &= m\varepsilon(1-\varepsilon) \end{aligned}$$

Thus, by the Chebyshev inequality, we have that the probability that more than a δ $(> \varepsilon)$ fraction of $A(\boldsymbol{c}(w))$'s disagree with $f(\boldsymbol{c}(w))$ is:

$$\begin{aligned} \Pr\left[Q > \delta m\right] &\leq \Pr\left[|Q - \varepsilon m| > (\delta - \varepsilon)m\right] \\ &\leq \frac{\varepsilon(1-\varepsilon)}{(\delta-\varepsilon)^2 m} \leq \frac{1}{4(\delta-\varepsilon)^2 m} \end{aligned}$$

We are interested in $\delta = \frac{m-2D}{2m} = \frac{1}{2} - \frac{D}{m}$. So if we set, for instance, $m = 12D$, the bound on the probability above is at most $1/3$ if $D > 9$ and $\varepsilon < 1/3$.

So except with this small probability, the decoding algorithm correctly recovers g as $\hat{g}$, and consequently B computes $f(\boldsymbol{x})$ correctly.

Generating each $\boldsymbol{c}(w)$ and running A on it takes $O(ND \log^2 p + t)$ time. The Berlekamp-Welch algorithm then takes $O(m^3)$ time, and the final evaluation of $\widehat{g}$ takes $O(D \log^2 p)$. Hence, given A with the above properties, the running time of B

is:

$$O(m(ND\log^2 p + t) + m^3 + D\log^2 p) = O(ND^2\log^2 p + D^3 + tD)$$

□

Chapter 3

Proofs of Work

Proofs of Work (PoWs), introduced in [DN92], have shown themselves to be an invaluable cryptographic primitive. Originally introduced to combat Denial of Service attacks and email spam, their key notion now serves as the heart of most modern cryptocurrencies (when combined with additional desired properties for this application).

By quickly generating easily verifiable challenges that require some quantifiable amount of work, PoWs ensure that adversaries attempting to swarm a system must have a large amount of computational power to do so. Practical uses aside, PoWs at their core ask a foundational question of the nature of hardness: Can you prove that a certain amount of work t was completed? In the context of complexity theory for this theoretical question, it suffices to obtain a computational problem whose (moderately) hard instances are easy to sample such that solutions are quickly verifiable.

Unfortunately, implementations of PoWs in practice stray from this theoretical question and, as a consequence, have two main drawbacks. First, they are often based on *heuristic* assumptions that have no quantifiable guarantees. One commonly used PoW is the problem of simply finding a value s so that hashing it together with the given challenge (e.g. with SHA-256) maps to anything with a certain amount of leading 0's. This is based on the *heuristic* belief that SHA-256 seems to behave unpredictably with no provable guarantees.

Secondly, since these PoWs are not provably secure, their heuristic sense of security

stems from, say, SHA-256 not having much discernible *structure* to exploit. This lack of structure, while hopefully giving the PoW its heuristic security, limits the ability to use the PoW in richer ways. That is, heuristic PoWs do not seem to come with a structure to support any useful properties beyond the basic definition of PoWs.

In this chapter, building on the techniques results in Chapter 2, we address both of these problems by constructing PoWs that are based on *worst-case complexity theoretic assumptions* in a provable way while also having considerable *algebraic structure*. This simultaneously moves PoWs in the direction of modern cryptography by basing our primitives on well-studied worst-case problems and expands the usability of PoWs by exploiting our algebraic structure to create, for example, PoWs that can be proved in Zero Knowledge or that can be distributed across many workers in a way that is robust to Byzantine failures. Our biggest use of our problems' structure is in proving a direct sum theorem to show that our proofs are non-amortizable across many challenges.

While all of our results and techniques will be analogous for 3SUM and APSP, we will use OV as our running example for our proofs and results statements. Recall that OV (defined again in Section 3.1.2) is a well-studied problem that is conjectured to require $n^{2-o(1)}$ time in the *worst-case* [Wil15]. Roughly, we show the following.

Informal Theorem 4. *Suppose* OV *takes* $n^{2-o(1)}$ *time to decide for sufficiently large* n*. A challenge* $\boldsymbol{c}$ *can be generated in* $\widetilde{O}(n)$ *time such that:*

- *A valid proof* $\boldsymbol{\pi}$ *to* $\boldsymbol{c}$ *can be computed in* $\widetilde{O}(n^2)$ *time.*
- *The validity of a candidate proof to* $\boldsymbol{c}$ *can be verified in* $\widetilde{O}(n)$ *time.*
- *Any valid proof to* $\boldsymbol{c}$ *requires* $n^{2-o(1)}$ *time to compute.*

This can be scaled to $n^{k-o(1)}$ hardness for all $k \in \mathbb{N}$ by a natural generalization of the OV problem to the k-OV problem, whose hardness is also supported by SETH. Thus fine-grained complexity theory props up PoWs of any complexity that is desired.

Further, we show that the verification can still be done in $\widetilde{O}(n)$ time for all of our $n^{k-o(1)}$ hard PoWs, allowing us to *tune* hardness. The corresponding PoW for this is

interactive but we show how to remove this interaction in the Random Oracle model in Section 3.5.

We also note that a straightforward application of [BK16b] allows our PoWs to be distributed amongst many workers in a way that is robust to byzantine failure or errors and can detect malicious party members. Namely, that a challenge can be broken up amongst a group of provers so that partial work can be error-corrected into a full proof.

Further, our PoWs admit zero knowledge proofs such that the proofs can be simulated in *very low* complexity – i.e. in time comparable to the verification time. While heuristic PoWs can be proved in zero knowledge as they are NP statements, the exact polynomial time complexities matter in this regime. We are able to use the algebraic structure of our problem to attain a notion of zero knowledge that makes sense in the fine-grained world.

A main lemma which may be of independent interest is a direct sum theorem on evaluating a specific low-degree polynomial $f\mathsf{OV}^k$.

Informal Theorem 5. *Suppose k-OV takes $n^{k-o(1)}$ time to decide. Then, for any polynomial ℓ, any algorithm that computes $f\mathsf{OV}^k(\boldsymbol{x}_i)$'s correctly on ℓ uniformaly random x_i's with probability $1/n^{O(1)}$ takes time $\ell(n) \cdot n^{k-o(1)}$.*

On Security From Worst-Case Assumptions. We make a point here that if SHA-256 is secure then it can be made into the aforementioned PoW whereas, if it is not, then SHA-256 is broken. While tautological, we point out that this is a Win-Lose situation. That is, either we have a PoW, or a specific instantiation of a heuristic cryptographic hash function is broken and no new knowledge is gained.

This is in contrast to our provably secure PoWs, in which we either have a PoW, or we have a breakthrough in complexity theory. For example, if we base a PoW on the Orthogonal Vectors problem, then either we have a PoW or the Orthogonal Vectors problem can be solved in sub-quadratic time which has been shown [Wil05] to be sufficient to break the Strong Exponential Time Hypothesis (SETH), giving a faster-than-brute-force algorithm for k-SAT formulas and thus a major insight to the

P vs NP problem.

By basing our PoWs on well-studied complexity theoretic problems, we position our conditional results to be in the desirable position for cryptography and complexity theory: a Win-Win. Orthogonal Vectors, 3SUM, and All-Pairs Shortest Path are the central problems of fine-grained complexity theory precisely because of their many quantitative connections to many other computational problems and so breaking any of their associated conjectures would give considerable insight into computation. Breaking a heuristic PoW like SHA-256, however, would simply say that that specific design for that specific input size happened to not be as secure as we thought.

Related Work

As mentioned earlier, PoWs were introduced by Dwork and Naor [DN92]. Definitions similar to ours were studied by Jakobsson and Juels [JJ99], Bitansky et al [BGJ+16], and (under the name Strong Client Puzzles) Stebila et al [SKR+11] (also see the last paper for some candidate constructions and further references).

We note that, while PoWs are often used in cryptocurrencies, the literature studying them in that context have more properties than the standard notion of a PoW (e.g. [BK16a]) that are desirable for their specific use within cryptocurrency and blockchain frameworks. We do not consider these and instead focus on the foundational cryptographic primitive that is a PoW.

Provably secure PoWs have been considered before in [BGJ+16] where PoWs are achieved from cryptographic assumptions. Namely, they show that if there is a worst-case hard problem that is non-amortizable *and* succinct randomized encodings exist, then PoWs are achievable. In contrast, our PoWs are based on solely on *worst-case* assumptions on well-studied problems from fine-grained complexity theory.

Subsequent to our work, Goldreich and Rothblum [GR18] have constructed (implicitly) a PoW protocol based on the worst-case hardness of the problem of counting t-cliques in a graph (for some constant t); they show a worst-case to average-case reduction for this problem, a doubly efficient interactive proof, and that the average-case problem is somewhat non-amortizable, which are the properties needed to go

from worst-case hardness to PoWs.

3.1 Definitions

3.1.1 Proofs of Work

Syntactically, a Proof of Work scheme involves three algorithms:

- $\mathsf{Gen}(1^n)$ produces a *challenge* $\boldsymbol{c}$.
- $\mathsf{Solve}(\boldsymbol{c})$ solves the challenge $\boldsymbol{c}$, producing a *proof* $\boldsymbol{\pi}$.
- $\mathsf{Verify}(\boldsymbol{c}, \boldsymbol{\pi})$ verifies the proof $\boldsymbol{\pi}$ to the challenge $\boldsymbol{c}$.

Taken together, these algorithms should result in an efficient proof system whose proofs are hard to find. This is formalized as follows.

Definition 2 (Proof of Work)**.** A $(t(n), \delta(n))$-*Proof of Work (PoW)* consists of three algorithms $(\mathsf{Gen}, \mathsf{Solve}, \mathsf{Verify})$. These algorithms must satisfy the following properties for large enough n:

- **Efficiency:**
  - $\mathsf{Gen}(1^n)$ runs in time $\widetilde{O}(n)$.
  - For any $\boldsymbol{c} \leftarrow \mathsf{Gen}(1^n)$, $\mathsf{Solve}(\boldsymbol{c})$ runs in time $\widetilde{O}(t(n))$.
  - For any $\boldsymbol{c} \leftarrow \mathsf{Gen}(1^n)$ and any $\boldsymbol{\pi}$, $\mathsf{Verify}(\boldsymbol{c}, \boldsymbol{\pi})$ runs in time $\widetilde{O}(n)$.
- **Completeness:** For any $\boldsymbol{c} \leftarrow \mathsf{Gen}(1^n)$ and any $\boldsymbol{\pi} \leftarrow \mathsf{Solve}(\boldsymbol{c})$,

$$\Pr\left[\mathsf{Verify}(\boldsymbol{c}, \boldsymbol{\pi}) = \mathit{accept}\right] = 1$$

  where the probability is taken over Verify's randomness.
- **Hardness:** For any polynomial ℓ, any constant $\epsilon > 0$, and any algorithm Solve^*_ℓ

that runs in time $\ell(n) \cdot t(n)^{1-\epsilon}$ when given $\ell(n)$ challenges of size n as input,

$$\Pr\left[\forall i : \mathsf{Verify}(\boldsymbol{c}_i, \boldsymbol{\pi}_i) = \text{acc} \;\middle|\; \begin{array}{l} (\boldsymbol{c}_i \leftarrow \mathsf{Gen}(1^n))_{i\in[\ell(n)]} \\ \boldsymbol{\pi} \leftarrow \mathsf{Solve}^*_\ell(\boldsymbol{c}_1, \ldots, \boldsymbol{c}_{\ell(n)}) : \\ \boldsymbol{\pi} = (\boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_{\ell(n)}) \end{array}\right] < \delta(n)$$

where the probability is taken over Gen and Verify's randomness.

The efficiency requirement above guarantees that the verifier in the Proof of Work scheme runs in nearly linear time. Together with the completeness requirement, it also ensures that a prover who actually spends roughly $t(n)$ time can convince the verifier that it has done so. The hardness requirement says that any attempt to convince the verifier without actually spending the prescribed amount of work has only a small probability of succeeding, and that this remains true even when amortized over several instances. That is, even a prover who gets to see several independent challenges and respond to them together will be unable to reuse any work across the challenges, and is effectively forced to spend the sum of the prescribed amount of work on all of them.

In some of the PoWs we construct, Solve and Verify are not algorithms, but are instead parties in an interactive protocol. The requirements of such interactive PoWs are the natural generalizations of those in the definition above, with Verify deciding whether to accept after interacting with Solve. And the hardness requirement applies to the numerous interactive protocols being run in any form of composition – serial, parallel, or otherwise. We will, however, show how to remove interaction in Section 3.5.

Heuristic constructions of PoWs, such as those based on SHA-256, easily satisfy efficiency and completeness (although not formally, given their lack of asymptotics), yet their hardness guarantees are based on nothing but the heuristic assumption that the PoW itself is a valid PoW.

3.1.2 Orthogonal Vectors

We now formally define the k-Orthogonal Vectors (k-OV) problem (which was mentioned briefly in Chapter 2), whose hardness we use to construct our PoW scheme. As the name suggests, k-OV is a generalisation of the Orthogonal Vectors (OV) problem defined and used in Chapter 2.

The properties possessed by OV (and k-OV) that enable our construction are also shared by other well-studied problems mentioned earlier, including 3SUM and APSP as noted in [BRSV17], and an array of other problems [BK16b, GR17, Wil16]. Consequently, while we focus on OV, PoWs based on the hardness of these other problems can be constructed along the lines of the one here. Further, the security of these constructions would also follow from the hardness of other problems that reduce to OV, 3SUM, etc. in a fine-grained manner with little, if any, degradation of security. Of particular interest, deciding graph properties that are statable in first-order logic all reduce to (moderate-dimensional) OV [GI16], and so we can obtain PoWs if *any* problem statable as a first-order graph property is hard.

All the algorithms we consider henceforth – reductions, adversaries, etc. – are *non-uniform Word-RAM algorithms* (with words of size $O(\log n)$ where n will be clear from context) unless stated otherwise, both in our hardness assumptions and our constructions. Security against such adversaries is necessary for PoWs to remain hard in the presence of pre-processing, which is typical in the case of cyrptocurrencies, for instance, where specialized hardware is often used. In the case of reductions, this non-uniformity is solely used to ensure that specific parameters determined completely by instance size (such as the prime $p(n)$ in Definition 5) are known to the reductions.

Remark 4. All of our reductions, algorithms, and assumptions can easily be made uniform by having an extra Setup procedure that is allowed to run in $t(n)^{1-\epsilon}$ for some $\epsilon > 0$ for a $(t(n), \delta(n))$-PoW. In our setting, this will just be used to find a prime on which to base a field extension for the rest of the PoW to satisfy the rest of its conditions. This makes sense for a PoW scheme to do and, for all the problems we consider, this can be done be done so that all the conjectures can be made uniformly.

We leave everything non-uniform, however, for the sake of exposition.

For convenience, we restate the definition of the OV problem below.

Definition 3 (Orthogonal Vectors)**.** The OV problem on vectors of dimension d (denoted OV_d) is to determine, given two sets U, V of n vectors from $\{0,1\}^{d(n)}$ each, whether there exist $u \in U$ and $v \in V$ such that $\langle u, v\rangle = 0$ (over $\mathbb{Z}$). If left unspecified, d is to be taken to be $\lceil \log^2 n \rceil$.

OV is commonly conjectured to require $n^{2-o(1)}$ time to decide, which many conditional fine-grained hardness results are based on [Wil15], and which has been shown to be true if the Strong Exponential Time Hypothesis (SETH) holds [Wil05]. This hardness and the hardness of its generalization to k-OV of requiring $n^{k-o(1)}$ time (which also holds under SETH) are what we base the hardness of our PoWs on. We now define k-OV.

Definition 4 (k-Orthogonal Vectors)**.** For an integer $k \geq 2$, the k-OV problem on vectors of dimension d is to determine, given k sets $(U_1, \ldots, U_k)$ of n vectors from $\{0,1\}^{d(n)}$ each, whether there exist $u^s \in U_s$ for each $s \in [k]$ such that over $\mathbb{Z}$,

$$\sum_{\ell \in [d(n)]} u^1_\ell \cdots u^k_\ell = 0$$

We say that such a set of vectors is *k-orthogonal*. If left unspecified, d is to be taken to be $\lceil \log^2 n \rceil$.

While these problems are conjectured worst-case hard, there are currently no widely-held beliefs for distributions that it may be average-case hard over. In Chapter 2, however, we defined a related problem that we showed to be average-case hard when assuming the worst-case hardness of k-OV. This problem is that of evaluating the following polynomial, which we write here with more parameters than earlier.

For any prime number p, we define the polynomial $f\mathsf{OV}^k_{n,d,p} : \mathbb{F}_p^{knd} \to \mathbb{F}_p$ as follows. Its inputs are parsed in the manner that those of k-OV are: below, for any $s \in [k]$ and $i \in [n]$, u^s_i represents the i^{th} vector in U_s, and for $\ell \in [d]$, $u^s_{i\ell}$ represents its ℓ^{th} coordinate.

$$f\mathsf{OV}^k_{n,d,p}(U_1,\ldots,U_k) = \sum_{i_1,\ldots,i_k\in[n]} \prod_{\ell\in[d]} \left(1 - u^1_{i_1\ell}\cdots u^k_{i_k\ell}\right)$$

When given an instance of k-OV (from $\{0,1\}^{knd}$) as input, $f\mathsf{OV}^k_{n,d,p}$ counts the number of tuples of k-orthogonal vectors (modulo p). Note that the degree of this polynomial is kd; for small d (e.g. $d = \lceil \log^2 n \rceil$), this is a fairly low-degree polynomial. The following definition gives the family of such polynomials parameterized by input size.

Definition 5 ($\mathcal{F}\mathsf{OV}^k$)**.** Consider an integer $k \geq 2$. Let $p(n)$ be the smallest prime number larger than $n^{\log n}$, and $d(n) = \lceil \log^2 n \rceil$. $\mathcal{F}\mathsf{OV}^k$ is the family of functions $\left\{f\mathsf{OV}^k_{n,d(n),p(n)}\right\}$.

Remark 5. We note that most of our results would hold for a much smaller choice of $p(n)$ above – anything larger than n^k would do. The reason we choose p to be this large is to achieve negligible soundness error in interactive protocols we shall be designing for this family of functions (see Protocol 3.1). Another way to achieve this is to use large enough extension fields of $\mathbb{F}_p$ for smaller p's; this is actually preferable, as the value of $p(n)$ as defined now is much harder to compute for uniform algorithms.

3.2 Verifying $\mathcal{F}\mathsf{OV}^k$

Our final protocol and its security consists, essentially, of two components – the hardness of evaluating $f\mathsf{OV}^k$ on random inputs, and the the ability to certify the correct evaluation of $f\mathsf{OV}^k$ in an efficiently verifiable manner. We explain the former in the next section; here, we describe the protocol for the latter (Protocol 3.1), which we will use as a sub-routine in our final PoW protocol. This protocol is a $(k-1)$-round interactive proof that, given $U_1,\ldots,U_k \in \mathbb{F}_p^{nd}$ and $y \in \mathbb{F}_p$, proves that $f\mathsf{OV}^k_{n,d,p}(U_1,\ldots,U_k) = y$.

In the special case of $k = 2$, a non-interactive (MA) protocol for OV was shown in [Wil16] and this MA protocol was used to construct a PoW scheme based on

OV, 3SUM, and APSP in [BRSV17], albeit one that only satisfies a weaker hardness requirement (i.e. non-batchability was not considered or proved). We introduce interaction to greatly improve the verifier's efficiency and show how interaction can be removed in Section 3.5. The following interactive proof is essentially the sum-check protocol, but in our case we need to pay close attention to the complexity of the prover and the verifier and so use ideas from [Wil16].

We will set up the following definitions before describing the protocol. For each $s \in [k]$, consider the univariate polynomials $\phi_1^s, \ldots, \phi_d^s : \mathbb{F}_p \to \mathbb{F}_p$, where ϕ_ℓ^s represents the ℓ^{th} column of U_s – that is, for $i \in [n]$, $\phi_\ell^s(i) = u_{i\ell}^s$. Each ϕ_ℓ^s has degree at most $(n-1)$. $f\mathsf{OV}_{n,d,p}^k$ can now be written as:

$$\begin{aligned} f\mathsf{OV}_{n,d,p}^k(U_1, \ldots, U_k) &= \sum_{i_1,\ldots,i_k \in [n]} \prod_{\ell \in [d]} \left(1 - u_{i_1\ell}^1 \cdots u_{i_k\ell}^k\right) \\ &= \sum_{i_1,\ldots,i_k \in [n]} \prod_{\ell \in [d]} \left(1 - \phi_\ell^1(i_1) \cdots \phi_\ell^k(i_k)\right) \\ &= \sum_{i_1,\ldots,i_k \in [n]} q(i_1, \ldots, i_k) \end{aligned}$$

where q is defined for convenience as:

$$q(i_1, \ldots, i_k) = \prod_{\ell \in [d]} \left(1 - \phi_\ell^1(i_1) \cdots \phi_\ell^k(i_k)\right)$$

The degree of q is at most $D = k(n-1)d$. Note that q can be evaluated at any point in $\mathbb{F}_p^k$ in time $\widetilde{O}(knd \log p)$, by evaluating all the $\phi_\ell^s(i_s)$'s (these polynomials can be found using fast interpolation techniques for univariate polynomials [Hor72]), computing each term in the above product and then multiplying them.

For any $s \in [k]$ and $\alpha_1, \ldots, \alpha_{s-1} \in \mathbb{F}_p$, define the following univariate polynomial:

$$q_{s,\alpha_1,\ldots,\alpha_{s-1}}(x) = \sum_{i_{s+1},\ldots,i_k \in [n]} q(\alpha_1, \ldots, \alpha_{s-1}, x, i_{s+1}, \ldots, i_k)$$

Every such q_s has degree at most $(n-1)d$ – this can be seen by inspecting the definition of q. With these definitions, the interactive proof is described as Protocol 3.1

below. The completeness and soundness of this interactive proof is then asserted by Theorem 5, which is proven in Section 3.2.

Interactive Proof for $\mathcal{F}\mathsf{OV}^k$:
The inputs to the protocol are $(U_1, \ldots, U_k) \in \mathbb{F}_p^{knd}$ (a valid input to $f\mathsf{OV}^k_{n,d,p}$), and a field element $y \in \mathbb{F}_p$. The polynomials q are defined as in the text.

- The prover sends the coefficients of a univariate polynomial q_1^* of degree at most $(n-1)d$.
- The verifier checks that $\sum_{i_1 \in [n]} q_1^*(i_1) = y$. If not, it rejects.
- For s from 1 up to $k-2$:
  - The verifier sends a random $\alpha_s \leftarrow \mathbb{F}_p$.
  - The prover sends the coefficients of a polynomial $q^*_{s+1,\alpha_1,\ldots,\alpha_s}$ of degree at most $(n-1)d$.
  - The verifier checks that $\sum_{i_{s+1} \in [n]} q^*_{s+1,\alpha_1,\ldots,\alpha_s}(i_{s+1}) = q^*_{s,\alpha_1,\ldots,\alpha_{s-1}}(\alpha_s)$. If not, it rejects.
- The verifier picks $\alpha_{k-1} \leftarrow \mathbb{F}_p$ and checks that $q^*_{k-1,\alpha_1,\ldots,\alpha_{k-2}}(\alpha_{k-1}) = q_{k-1,\alpha_1,\ldots,\alpha_{k-2}}(\alpha_{k-1})$, computed using the fact that $q_{k-1,\alpha_1,\ldots,\alpha_{k-2}}(\alpha_{k-1}) = \sum_{i_k \in [n]} q_{k,\alpha_1,\ldots,\alpha_{k-1}}(i_k)$. If not, it rejects.
- If the verifier hasn't rejected yet, it accepts.

Protocol 3.1: Interactive Proof for $\mathcal{F}\mathsf{OV}^k$.

Theorem 5. *For any $k \geq 2$, let d and p be as in Definition 5. Protocol 3.1 is a $(k-1)$-round interactive proof for proving that $y = \mathcal{F}\mathsf{OV}^k(x)$. This protocol has perfect completeness and soundness error at most $\left(\frac{knd}{p}\right)$. The prover runs in time $\widetilde{O}(n^k d \log p)$, and the verifier in time $\widetilde{O}(knd^2 \log p)$.*

As observed earlier, Protocol 3.1 is non-interactive when $k = 2$. We then get the following corollary for $\mathcal{F}\mathsf{OV}$.

Corollary 7. *For $k = 2$, let d and p be as in Definition 5. Protocol 3.1 is an MA proof for proving that $y = \mathcal{F}\mathsf{OV}(x)$. This protocol has perfect completeness and soundness error at most $\left(\frac{2nd}{p}\right)$. The prover runs in time $\widetilde{O}(n^2)$, and the verifier in time $\widetilde{O}(n)$.*

Proof of Theorem 5. We show each of the required properties in turn.

Completeness. If indeed $y = f\mathsf{OV}^k_{n,d,p}(U_1, \ldots, U_k)$, the prover can make the verifier in the protocol accept by using the polynomials $(q_1, q_{2,\alpha_1}, \ldots, q_{k,\alpha_1,\ldots,\alpha_k})$ in place of $(q^*_1, q^*_{2,\alpha_1}, \ldots, q^*_{k,\alpha_1,\ldots,\alpha_k})$. Perfect completeness is then seen to follow from the definitions of these polynomials and their relation to q and hence $f\mathsf{OV}^k_{n,d,p}$.

Soundness. Suppose $y \neq f\mathsf{OV}^k_{n,d,p}(U_1, \ldots, U_k)$. We now analyze the probability with which a cheating prover could make the verifier accept.

To start with, note that the prover's q^*_1 has to be different from q_1, as otherwise the check in the second step would fail. Further, as the degree of these polynomials is less than nd, the probability that the verifier will then choose an α_1 such that $q^*_1(\alpha_1) = q_1(\alpha_1)$ is less than $\frac{nd}{p}$.

If this event does not happen, then the prover has to again send a q^*_{2,α_1} that is different from q_{2,α_1}, which again agree on α_2 with probability less than $\frac{nd}{p}$. This goes on for $(k-1)$ rounds, at the end of which the verifier checks whether $q^*_{k-1}(\alpha_{k-1})$ is equal to $q_{k-1}(\alpha_{k-1})$, which it computes by itself. If at least one of these accidental equalities at a random point has not occurred throughout the protocol, the verifier will reject. The probability that no violations occur over the $(k-1)$ rounds is, by the union bound, less than $\frac{knd}{p}$.

Efficiency. Next we discuss details of how the honest prover and the verifier are implemented, and analyze their complexities. To this end, we will need the following algorithmic results about computations involving *univariate* polynomials over finite fields.

Lemma 3 (Fast Multi-point Evaluation [Fid72])**.** *Given the coefficients of a univariate polynomial $q : \mathbb{F}_p \to \mathbb{F}_p$ of degree at most N, and N points $x_1, \ldots, x_N \in \mathbb{F}_p$, the set of evaluations $(q(x_1), \ldots, q(x_N))$ can be computed in time $O(N \log^3 N \log p)$.*

Lemma 4 (Fast Interpolation [Hor72])**.** *Given $N+1$ evaluations of a univariate polynomial $q : \mathbb{F}_p \to \mathbb{F}_p$ of degree at most N, the coefficients of q can be computed in time $O(N \log^3 N \log p)$.*

To start with, both the prover and verifier compute the coefficients of all the ϕ_ℓ^s's. Note that, by definition, they know the evaluation of each ϕ_ℓ^s on n points, given by $\{(i, u_{i\ell}^s)\}_{i\in[n]}$. This can be used to compute the coefficients of each ϕ_ℓ^s in time $\widetilde{O}(n \log p)$ by Lemma 4. The total time taken is hence $\widetilde{O}(knd \log p)$.

The proof of the following proposition specifies further details of the prover's workings.

Proposition 1. *The coefficients of the polynomial $q_{s,\alpha_1,\ldots,\alpha_{s-1}}$ can be computed in time $\widetilde{O}((n^{k-s+1}d + nd^2) \log p)$ given the above preprocessing.*

Proof. The procedure to do the above is as follows:

1. Fix some value of $s, \alpha_1, \ldots, \alpha_{s-1}$.
2. For each $\ell \in [d]$, compute the evaluation of ϕ_ℓ^s on nd points, say $\{1, \ldots, nd\}$.
   - Since its coefficients are known, the evaluations of each ϕ_ℓ^s on these nd points can be computed in time $\widetilde{O}(nd \log p)$ by Lemma 3, for a total of $\widetilde{O}(nd^2 \log p)$ for all the ϕ_ℓ^s's.
3. For each setting of $i_{s+1}, \ldots, i_k$, compute the evaluations of the polynomial $\rho_{i_{s+1},\ldots,i_k}(x) = q(\alpha_1, \ldots, \alpha_{s-1}, x, i_{s+1}, \ldots, i_k)$, on the points $\{1, \ldots, nd\}$.
   - First substitute the constants $\alpha_1, \ldots, \alpha_{s-1}, i_{s+1}, \ldots, i_k$ into the definition of q.
   - This requires computing, for each $\ell \in [d]$ and $s' \in [k]\setminus\{s\}$, either $\phi_\ell^{s'}(\alpha_s)$ or $\phi_\ell^{s'}(i_s)$. All of this can be done in time $\widetilde{O}(knd \log p)$ by direct polynomial evaluations since the coefficients of the $\phi_\ell^{s'}$'s are known.
   - This reduces q to a product of d univariate polynomials of degree less than n, whose evaluations on the nd points can now be computed in time $\widetilde{O}(knd \log p)$ by multiplying the constants computed in the above step with the evaluations of $\phi_\ell^{s'}$ on these points, and subtracting from 1.
   - The product of the evaluations can now be computed in time $\widetilde{O}(nd^2 \log p)$ to get what we need.

4. Add up the evaluations of $\rho_{i_{s+1},\ldots,i_k}$ pointwise over all settings of $(i_{s+1},\ldots,i_k)$.

   - There are n^{k-s} possible settings of $(i_{s+1},\ldots,i_k)$, and for each of these we have nd evaluations. All the additions hence take $\widetilde{O}(n^{k-s+1}d\log p)$ time.

5. This gives us nd evaluations of $q_{s,\alpha_1,\ldots,\alpha_{s-1}}$, which is a univariate polynomial of degree at most $(n-1)d$. So its coefficients can be computed in time $\widetilde{O}(nd\log p)$ by Lemma 4.

It can be verified from the intermediate complexity computations above that all these operations together take $\widetilde{O}((n^{k-s+1}d + nd^2)\log p)$ time. This proves the proposition.

□

Recall that what the honest prover has to do is compute $q_1, q_{2,\alpha_1},\ldots,q_{k,\alpha_1,\ldots,\alpha_{k-1}}$ for the α_s's specified by the verifier. By the above proposition, along with the preprocessing, the total time the prover takes is:

$$\widetilde{O}(knd\log p + (n^k d + nd^2)\log p) = \widetilde{O}(n^k d\log p)$$

The verifier's checks in steps (2) and (3) can each be done in $\widetilde{O}(n\log p)$ time using Lemma 3. Step (4), finally, can be done by using the above proposition with $s = k$ in time $\widetilde{O}(nd^2\log p)$. Even along with the preprocessing, this leads to a total time of $\widetilde{O}(knd^2\log p)$.

□

Remark 6. Note the Prover's work of finding coefficients of polynomials is mainly done by evaluating the polynomial on many points and interpolating. Similarly to [BK16b], this opens the door to distributing the Prover's work. Namely, the individual evaluations can be split amongst a group of workers which can then be recombined to find the final coefficients. Further, since the evaluations of a polynomial is a Reed-Solomon code, this allows for error correction in the case that the group of provers make errors or have some malicious members. Thus, the Prover's work can

be distributed in a way that is robust to Byzantine errors and can identify misbehaving members.

3.3 The PoW Protocol

We now present Protocol 3.2, which we show to be a Proof of Work scheme assuming the hardness of k-OV.

Proof of Work based on hardness of k-OV:

- $\mathsf{Gen}(1^n)$:
  - Output a random $\boldsymbol{c} \in \mathbb{F}_p^{knd}$.
- $(\mathsf{Solve}, \mathsf{Verify})$ work as follows given $\boldsymbol{c}$:
  - Solve computes $z = f\mathsf{OV}^k_{n,d,p}(\boldsymbol{c})$ and outputs it.
  - Solve and Verify run Protocol 3.1 with input $(\boldsymbol{c}, z)$, Solve as prover, and Verify as verifier.
  - Verify accepts iff the verifier in the above instance of Protocol 3.1 accepts.

Protocol 3.2: Proof of Work based on the hardness of k-OV.

Theorem 6. *For some $k \geq 2$, suppose k-OV takes $n^{k-o(1)}$ time to decide for all but finitely many input lengths for any $d = \omega(\log n)$. Then, Protocol 3.2 is an (n^k, δ)-Proof of Work scheme for any function $\delta(n) > 1/n^{o(1)}$.*

Remark 7. As is, this will be an interactive Proof of Work protocol. In the special case of $k = 2$, Corollary 7 gives us a non-interactive PoW. If we want to remove interaction for general k-OV, however, we could use the MA proof in [Wil16] at the cost of verification taking time $\widetilde{O}(n^{k/2})$ as was done in [BRSV17]. To keep verification time at $\widetilde{O}(n)$, we instead show how to remove interaction in the Random Oracle model in Section 3.5. This will allow us to *tune* the gap between the parties – we can choose k and thus the amount of work, $n^{k-o(1)}$, that must be done by the prover while always only needing $\widetilde{O}(n)$ time for verification.

Remark 8. We can also exploit this PoW's algebraic structure on the Prover's side. Using techniques from [BK16b], the Prover's work can be distributed amongst a group of provers. While, cumulatively, they must complete the work required of the PoW, they can each only do a portion of it. Further, this can be done in a way robust to Byzantine errors amongst the group. See Remark 6 for further details.

We will use Theorem 5 to argue for the completeness and soundness of Protocol 3.2. In order to prove the hardness, we will need lower bounds on how well the problem that Solve is required to solve can be batched. We first define what it means for a function to be non-batchable in the average-case in a manner compatible with the hardness requirement. Note that this requirement is stronger than being non-batchable in the worst-case.

Definition 6. Consider a function family $\mathcal{F} = \{f_n : \mathcal{X}_n \to \mathcal{Y}_n\}$, and a family of distributions $\mathcal{D} = \{D_n\}$, where D_n is over $\mathcal{X}_n$. $\mathcal{F}$ is *not* (ℓ, t, δ)*-batchable on average over* $\mathcal{D}$ if, for any algorithm Batch that runs in time $\ell(n)t(n)$ when run on $\ell(n)$ inputs from $\mathcal{X}_n$, when it is given as input $\ell(n)$ independent samples from D_n, the following is true for all large enough n:

$$\Pr_{x_i \leftarrow D_n}\left[\mathsf{Batch}(x_1, \ldots, x_{\ell(n)}) = (f_n(x_1), \ldots, f_n(x_{\ell(n)}))\right] < \delta(n)$$

We will be concerned with the case where the batched time $t(n)$ is less than the time it takes to compute f_n on a single instance. This sort of statement is what a direct sum theorem for $\mathcal{F}$'s hardness would guarantee. Theorem 7, then, claims that we achieve this non-batchability for $\mathcal{F}\mathsf{OV}^k$ and, as $\mathcal{F}\mathsf{OV}^k$ is one of the things that Solve is required to evaluate, we will be able to show the desired hardness of Protocol 3.2. We prove Theorem 7 via a direct sum theorem in Appendix B.1, and prove a weaker version for illustrative purposes in Section 3.4.

Theorem 7. *For some* $k \geq 2$*, suppose* k-OV *takes* $n^{k-o(1)}$ *time to decide for all but finitely many input lengths for any* $d = \omega(\log n)$*. Then, for any constants* $c, \epsilon > 0$ *and* $\delta < \epsilon/2$*,* $\mathcal{F}\mathsf{OV}^k$ *is* not $(n^c, n^{k-\epsilon}, 1/n^\delta)$*-batchable on average over the uniform distribution over its inputs.*

We now put all the above together to prove Theorem 6 as follows.

Proof of Theorem 6. We prove that Protocol 3.2 satisfies the various requirements demanded of a Proof of Work scheme assuming the hardness of k-OV.

Efficiency:

- $\mathsf{Gen}(1^n)$ simply samples knd uniformly random elements of $\mathbb{F}_p$. As $d = \log^2 n$ and $p \leq 2n^{\log n}$ (by Bertrand-Chebyshev's Theorem), this takes $\widetilde{O}(n)$ time.
- Solve computes $f\mathsf{OV}^k_{n,d,p}(\boldsymbol{c})$, which can be done in $\widetilde{O}(n^k)$ time. It then runs the prover in an instance of Protocol 3.1, which can be done in $\widetilde{O}(n^k)$ time by Theorem 5. So in all it takes takes $\widetilde{O}(n^k)$ time.
- Verify runs the verifier in an instance of Protocol 3.1, taking $\widetilde{O}(n)$ time, again by Theorem 5.

Completeness: This follows immediately from the completeness of Protocol 3.1 as an interactive proof for $\mathcal{F}\mathsf{OV}^k$, as stated in Theorem 5, as this is the protocol that Solve and Verify engage in.

Hardness: We proceed by contradiction. Suppose there is a polynomial ℓ, an (interactive) algorithm Solve^*, and a constant $\epsilon > 0$ such that Solve^* runs in time $\ell(n)n^{k-\epsilon}$ and makes Verify accept on $\ell(n)$ independent challenges generated by $\mathsf{Gen}(1^n)$ with probability at least $\delta(n) > 1/n^{o(1)}$ for infinitely many input lengths n.

For each of these input lengths, let the set of challenges (which are $f\mathsf{OV}$ inputs) produced by $\mathsf{Gen}(1^n)$ be $\left\{\boldsymbol{c}_1, \ldots, \boldsymbol{c}_{\ell(n)}\right\}$, and the corresponding set of solutions output by Solve^* be $\left\{z_1, \ldots, z_{\ell(n)}\right\}$. So Solve^* succeeds as a prover in Protocol 3.1 for *all* the instances $\{(\boldsymbol{c}_i, z_i)\}$ with probability at least $\delta(n)$.

By the negligible soundness error of Protocol 3.1 guaranteed by Theorem 5, in order to do this, Solve^* has to use the correct values $f\mathsf{OV}^k_{n,d,p}(\boldsymbol{c}_i)$ for all the z_i's with probability negligibly close to $\delta(n)$ and definitely more than, say, $\delta(n)/2$. In particular, with this probability, it has to explicitly compute $f\mathsf{OV}^k_{n,d,p}$ at $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_{\ell(n)}$, all of which are independent uniform points in $\mathbb{F}_p^{knd}$ for all of these infinitely many input

lengths n. But this is exactly what Theorem 7 says is impossible under our assumptions. So such a Solve^* cannot exist, and this proves the hardness of Protocol 3.2.

We have thus proven all the properties necessary and hence Protocol 3.2 is indeed an (n^k, δ)-Proof of Work under the hypothesised hardness of k-OV for any $\delta(n) > 1/n^{o(1)}$. □

3.4 A Direct Sum Theorem for $\mathcal{F}\mathsf{OV}$

A direct sum theorem for a problem roughly states that solving m independent instances of a problem takes m times as long as a single instance. The converse of this is attaining a non-trivial speed-up when given a *batch* of instances. In this section we prove a direct sum theorem for the problem of evaluating $\mathcal{F}\mathsf{OV}$ and thus its non-batchability.

Direct sum theorems are typically elusive in complexity theory and so our results, which we prove for generic problems with a certain set of properties, may be of independent interest to the study of hardness amplification. That our results show that batch-evaluating our multivariate low-degree polynomials is *hard* may be particularly surprising since batch-evaluation for *univariate* low-degree polynomials is known to be *easy* [Fid72, Hor72] and, further, [BK16b, GR17, Wil16] show that batch-evaluating multivariate low-degree polynomials (including our own) is *easy to delegate*. For more rigorous definitions of direct sum and direct product theorems, see [She12].

We now prove the following weaker version of Theorem 7 on $\mathcal{F}\mathsf{OV}$'s non-batchability (Theorem 7 is proven in Appendix B.1 using an extension of the techniques employed here). The notion of non-batchability used below is defined in Definition 6 in Section 3.3.

Theorem 8. *For some $k \geq 2$, suppose k-OV takes $n^{k-o(1)}$ time to decide for all but finitely many input lengths for any $d = \omega(\log n)$. Then, for any constants $c, \epsilon > 0$, $\mathcal{F}\mathsf{OV}^k$ is* not *$(n^c, n^{k-\epsilon}, 7/8)$-batchable on average over the uniform distribution over its inputs.*

Throughout this section, $\mathcal{F}$, $\mathcal{F}'$ and $\mathcal{G}$ are families of functions $\{f_n : \mathcal{X}_n \to \mathcal{Y}_n\}$, $\{f'_n : \mathcal{X}'_n \to \mathcal{Y}'_n\}$ and $\left\{g_n : \hat{\mathcal{X}}_n \to \hat{\mathcal{Y}}_n\right\}$, and $\mathcal{D} = \{D_n\}$ is a family of distributions where D_n is over $\hat{\mathcal{X}}_n$.

Theorem 8 is the result of two properties possessed by $\mathcal{F}\mathsf{OV}^k$. We define these properties below, prove a more general lemma about functions that have these properties, and use it to prove this theorem.

Definition 7. $\mathcal{F}$ is said to be (s, ℓ)-*downward reducible* to $\mathcal{F}'$ in time t if there is a pair of algorithms $(\mathsf{Split}, \mathsf{Merge})$ satisfying:

- For all large enough n, $s(n) < n$.
- Split on input an $x \in \mathcal{X}_n$ outputs $\ell(n)$ instances from $\mathcal{X}'_{s(n)}$.

$$\mathsf{Split}(x) = (x_1, \ldots, x_{\ell(n)})$$

- Given the value of $\mathcal{F}'$ at these $\ell(n)$ instances, Merge can reconstruct the value of $\mathcal{F}$ at x.

$$\mathsf{Merge}(x, f'_{s(n)}(x_1), \ldots, f'_{s(n)}(x_{\ell(n)})) = f_n(x)$$

- Split and Merge together run in time at most $t(n)$.

If $\mathcal{F}'$ is the same as $\mathcal{F}$, then $\mathcal{F}$ is said to be *downward self-reducible*.

Definition 8. $\mathcal{F}$ is said to be ℓ-*robustly reducible* to $\mathcal{G}$ in time t if there is a pair of algorithms $(\mathsf{Split}, \mathsf{Merge})$ satisfying:

- Split on input an $x \in \mathcal{X}_n$ (and randomness r) outputs $\ell(n)$ instances from $\hat{\mathcal{X}}_n$.

$$\mathsf{Split}(x; r) = (x_1, \ldots, x_{\ell(n)})$$

- For such a tuple $(x_i)_{i \in [\ell(n)]}$ and any function g^* such that $g^*(x_i) = g_n(x_i)$ for at

least 2/3 of the x_i's, Merge can reconstruct the function value at x as:

$$\mathsf{Merge}(x, r, g^*(x_1), \ldots, g^*(x_{\ell(n)})) = f_n(x)$$

- Split and Merge together run in time at most $t(n)$.
- Each x_i is distributed according to D_n, and the x_i's are pairwise independent.

The above is a more stringent notion than the related non-adaptive random self-reducibility as defined in [FF93]. We remark that to prove what we need, it can be shown that it would have been sufficient if the reconstruction above had only worked for *most* r's.

Lemma 5. *Suppose $\mathcal{F}$, $\mathcal{F}'$ and $\mathcal{G}$ have the following properties:*

- *$\mathcal{F}$ is (s_d, ℓ_d)-downward reducible to $\mathcal{F}'$ in time t_d.*
- *$\mathcal{F}'$ is ℓ_r-robustly reducible to $\mathcal{G}$ over $\mathcal{D}$ in time t_r.*
- *$\mathcal{G}$ is $(\ell_a, t_a, 7/8)$-batchable on average over $\mathcal{D}$, and $\ell_a(s_d(n)) = \ell_d(n)$.*

Then $\mathcal{F}$ can be computed in the worst-case in time:

$$t_d(n) + \ell_d(n)t_r(s_d(n)) + \ell_r(s_d(n))\ell_d(n)t_a(s_d(n))$$

We note, that the condition $\ell_a(s_d(n)) = \ell_d(n)$ above can be relaxed to $\ell_a(s_d(n)) \leq \ell_d(n)$ at the expense of a factor of 2 in the worst-case running time obtained for $\mathcal{F}$. We now show how to prove Theorem 8 using Lemma 5, and then prove the lemma itself.

Proof of Theorem 8. Fix any $k \geq 2$. Suppose, towards a contradiction, that for some $c, \epsilon > 0$, $\mathcal{F}\mathsf{OV}^k$ is $(n^c, n^{k-\epsilon}, 7/8)$-batchable on average over the uniform distribution. In our arguments we will refer to the following function families:

- $\mathcal{F}$ is k-OV with vectors of dimension $d = \left(\frac{k}{k+c}\right)^2 \log^2 n$.
- $\mathcal{F}'$ is k-OV with vectors of dimension $\log^2 n$.

- $\mathcal{G}$ is $\mathcal{F}\mathsf{OV}^k$ (over $\mathbb{F}_p^{knd}$ for some p that definitely satisfies $p > n$).

Let $m = n^{k/(k+c)}$. Note the following two properties :

- $\frac{n}{m^{c/k}} = m$
- $d = \left(\frac{k}{k+c}\right)^2 \log^2 n = \log^2 m$

We now establish the following relationships among the above function families.

Proposition 2. *$\mathcal{F}$ is (m, m^c)-downward reducible to $\mathcal{F}'$ in time $\widetilde{O}(m^{c+1})$.*

Split_d, when given an instance $(U_1, \ldots, U_k) \in \{0,1\}^{k(n\times d)}$, first divides each U_i into $m^{c/k}$ partitions $U_{i1}, \ldots, U_{im^{c/k}} \in \{0,1\}^{m\times d}$. It then outputs the set of tuples $\left\{(U_{1j_1}, \ldots, U_{kj_k}) \mid j_i \in [m^{c/k}]\right\}$. Each U_{ij} is in $\{0,1\}^{m\times d}$ and, as noted earlier, $d = \log^2 m$. So each tuple in the set is indeed an instance of $\mathcal{F}'$ of size m. Further, there are $(m^{c/k})^c = m^c$ of these.

Note that the original instance has a set of k-orthogonal vectors if and only if at least one of the m^c smaller instances produced does. So Merge_d simply computes the disjunction of the $\mathcal{F}'$ outputs to these instances.

Both of these can be done in time $O(m^c \cdot k \cdot md + m^c) = \widetilde{O}(m^{c+1})$.

Proposition 3. *$\mathcal{F}'$ is $12kd$-robustly reducible to $\mathcal{G}$ over the uniform distribution in time $\widetilde{O}(m)$.*

Notice that for any $U_1, \ldots, U_k \in \{0,1\}^{m\times d}$, we have that $k\text{-}\mathsf{OV}(U_1, \ldots, U_k) = f\mathsf{OV}_m^k(U_1, \ldots, U_k)$. So it is sufficient to show such a robust reduction from $\mathcal{G}$ to itself. We do this now.

Given input $\boldsymbol{x} \in \mathbb{F}_p^{knd}$, Split_r picks two uniformly random $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{F}_p^{knd}$ and outputs the set of vectors $\{\boldsymbol{x} + t\boldsymbol{x}_1 + t^2\boldsymbol{x}_2 \mid t \in \{1, \ldots, 12kd\}\}$. Recall that our choice of p is much larger than $12kd$ and hence this is possible. The distribution of each of these vectors is uniform over $\mathbb{F}_p^{knd}$, and they are also pairwise independent as they are points on a random quadratic curve through $\boldsymbol{x}$.

Define the univariate polynomial $g_{\boldsymbol{x},\boldsymbol{x}_1,\boldsymbol{x}_2}(t) = f\mathsf{OV}_m^k(\boldsymbol{x} + t\boldsymbol{x}_1 + t^2\boldsymbol{x}_2)$. Note that its degree is at most $2kd$. When Merge_r is given $(y_1, \ldots, y_{12kd})$ that are purported to be the evaluations of $f\mathsf{OV}_m^k$ on the points produced by Split, these can be seen

as purported evaluations of $g_{\boldsymbol{x},\boldsymbol{x}_1,\boldsymbol{x}_2}$ on $\{1,\ldots,12kd\}$. This can, in turn, be treated as a corrupt codeword of a Reed-Solomon code, which under these parameters has distance $10kd$.

The Berlekamp-Welch algorithm can be used to decode any codeword that has at most $5kd$ corruptions, and if at least $2/3$ of the evaluations are correct, then at most $4kd$ evaluations are wrong. Hence Merge_r uses the Berlekamp-Welch algorithm to recover $g_{\boldsymbol{x},\boldsymbol{x}_1,\boldsymbol{x}_2}$, which can be evaluated at 0 to obtain $f\mathsf{OV}^k_n(\boldsymbol{x})$.

Thus, Split_r takes $\widetilde{O}(12kd \cdot kmd) = \widetilde{O}(m)$ time to compute all the vectors it outputs. Merge_r takes $\widetilde{O}((12kd)^3)$ time to run Berlekamp-Welch, and $\widetilde{O}(12kd)$ time to evaluate the resulting polynomial at 0. So in all both algorithms take $\widetilde{O}(m)$ time.

By our assumption at the beginning, $\mathcal{G}$ is $(n^c, n^{k-\epsilon}, 7/8)$-batchable on average over the uniform distribution. Together with the above propositions, this satisfies all the requirements in the hypothesis of Lemma 5, which now tells us that $\mathcal{F}$ can be computed in the worst-case in time:

$$\begin{aligned}\widetilde{O}(m^{c+1} + m^c \cdot m + 12kd \cdot m^c \cdot m^{k-\epsilon}) &= \widetilde{O}(m^{c+1} + m^{c+k-\epsilon}) \\ &= \widetilde{O}(n^{k(c+1)/(k+c)} + n^{k(k+c-\epsilon)/(k+c)}) \\ &= \widetilde{O}(n^{k-\epsilon'})\end{aligned}$$

for some $\epsilon' > 0$. But this is what the hypothesis of the theorem says is not possible. So $\mathcal{F}\mathsf{OV}^k$ cannot be $(n^c, n^{k-\epsilon}, 7/8)$-batchable on average, and this argument applies for any $c, \epsilon > 0$. □

Proof of Lemma 5. Given the hypothesised downward reduction (Split_d, Merge_d), robust reduction ($\mathsf{Split}_r, \mathsf{Merge}_r$) and batch-evaluation algorithm Batch for $\mathcal{F}$, f_n can be computed as follows (for large enough n) on an input $x \in \mathcal{X}_n$:

- Run $\mathsf{Split}_d(x)$ to get $x_1, \ldots, x_{\ell_d(n)} \in \mathcal{X}'_{s_d(n)}$.
- For each $i \in [\ell_d(n)]$, run $\mathsf{Split}_r(x_i; r_i)$ to get $x_{i1}, \ldots, x_{i\ell_r(s_d(n))} \in \hat{\mathcal{X}}_{s_d(n)}$.

- For each $j \in [\ell_r(s_d(n))]$, run $\mathsf{Batch}(x_{1j}, \ldots, x_{\ell_d(n)j})$ to get the outputs $y_{1j}, \ldots, y_{\ell_d(n)j} \in \hat{\mathcal{Y}}_{s_d(n)}$.
- For each $i \in [\ell_d(n)]$, run $\mathsf{Merge}_r(x_i, r_i, y_{i1}, \ldots, y_{i\ell_r(s_d(n))})$ to get $y_i \in \mathcal{Y}'_{s_d(n)}$.
- Run $\mathsf{Merge}_d(x, y_1, \ldots, y_{\ell_d(n)})$ to get $y \in \mathcal{Y}_n$, and output y as the alleged $f_n(x)$.

We will prove that with high probability, after the calls to Batch, enough of the y_{ij}'s produced will be equal to the respective $g_{s_d(n)}(x_{ij})$'s to be able to correctly recover all the $f'_{s_d(n)}(x_i)$'s and hence $f_n(x)$.

For each $j \in [\ell_r(s_d(n))]$, define I_j to be the indicator variable that is 1 if $\mathsf{Batch}(x_{1j}, \ldots, x_{\ell_d(n)j})$ is correct and 0 otherwise. Note that by the properties of the robust reduction of $\mathcal{F}'$ to $\mathcal{G}$, for a fixed j each of the x_{ij}'s is independently distributed according to $D_{s_d(n)}$ and further, for any two distinct j, j', the tuples (x_{ij}) and $(x_{ij'})$ are independent.

Let $I = \sum_j I_j$ and $m = \ell_r(s_d(n))$. By the aforementioned properties and the correctness of Batch, we have the following:

$$\begin{aligned} \mathrm{E}[I] &\geq \frac{7}{8}m \\ \mathrm{Var}[I] &\leq \frac{7}{64}m \end{aligned}$$

Note that as long as Batch is correct on more than a 2/3 fraction of the j's, Merge_r will get all of the y_i's correct, and hence Merge_d will correctly compute $f_n(x)$. The probability that this does not happen is bounded using Chebyshev's inequality as:

$$\begin{aligned} \Pr\left[I \leq \frac{2}{3}m\right] &\leq \Pr\left[|I - \mathrm{E}[I]| \geq \left(\frac{7}{8} - \frac{2}{3}\right)m\right] \\ &\leq \frac{\mathrm{Var}[I]}{(5m/24)^2} \\ &\leq \frac{63}{25 \cdot m} < \frac{3}{m} \end{aligned}$$

As long as $m > 9$, this probability of failure is less than 1/3, and hence $f_n(x)$ is computed correctly in the worst-case with probability at least 2/3. If it is the case

that $\ell_r(s_d(n)) = m$ happens to be less than 9, then instead of using Merge_r directly in the above algorithm, we would use Merge'_r that runs Merge_r several times so as to get more than 9 samples in total and takes the majority answer from all these runs.

The time taken is $t_d(n)$ for the downward reduction, $t_r(s_d(n))$ for each of the $\ell_d(n)$ robust reductions on instances of size $s_d(n)$, and $\ell_d(n)t_a(s_d(n))$ for each of the $\ell_r(s_d(n))$ calls to Batch on sets of $\ell_d(n) = \ell_a(s_d(n))$ instances, summing up to the total time stated in the lemma. □

3.5 Removing Interaction

In this section we show how to remove the interaction in Protocol 3.2 via the Fiat-Shamir heuristic [FS86] and thus prove security of our non-interactive PoW in the Random Oracle model.

Remark 9. Recent papers have constructed hash functions for which provably allow the Fiat-Shamir heuristic to go through [KRR17, CCRR18]. Both of these constructions require a variety of somewhat non-standard sub-exponential security assumptions: [KRR17] uses sub-exponentially secure indistinguishability obfuscation, sub-exponentially secure input-hiding point function obfuscation, and sub-exponentially secure one-way functions; while [CCRR18] needs symmetric encryption schemes with strong guarantees against key recovery attacks (they specifically propose two instantiating assumptions that are variants on the discrete-log assumption and the learning with errors assumption). While for simplicity we present our work in the context of the random oracle model, [KRR17, CCRR18] give evidence that our scheme can be made non-interactive in the plain model.

We also note that our use of a Random Oracle here is quite different from its possible direct use in a Proof of Work similar to those currently used, for instance, in the cryptocurrency blockchains. There, the task is to find a pre-image to H such that its image starts (or ends) with at least a certain number of 0's. In order to make this only moderately hard for PoWs, the security parameter of the chosen instantiation of

the Random Oracle (which is typically a hash function like SHA-256) is necessarily not too high. In our case, however, there is no such need for such a task to be feasible, and this security parameter can be set very high, so as to be secure even against attacks that could break the above kind of PoW.

It is worth noting that because of this use of the RO and the soundness properties of the interactive protocol, the resulting proof of work is effectively unique in the sense that it is computationally infeasible to find two accepting proofs. This is markedly different from proof of work described above, where random guessing for the same amount of time is likely to yield an alternate proof.

In what follows, we take H to be a random oracle that outputs an element of $\mathbb{F}_p$, where p is as in Definition 5 and n will be clear from context. Informally, as per the Fiat-Shamir heuristic, we will replace all of the verifier's random challenges in the interactive proof (Protocol 3.1) with values output by H so that secure challenges can be gotten without interaction. Using the definitions of the polynomials $q(i_1, \ldots, i_k)$ and $q_{s,\alpha_1,\ldots,\alpha_{s-1}}(x)$ from Section 3.3, the non-interactive proof scheme for $\mathcal{F}\mathsf{OV}^k$ is described as Protocol 3.3.

Overloading the definition, we now consider Protocol 3.2 as our PoW as before except that we now use the *non-interactive* Protocol 3.3 as the the basis of our Solve and Verify algorithms. The following theorem states that this substitution gives us a *non-interactive* PoW in the Random Oracle model.

Theorem 9. *For some $k \geq 2$, suppose k-OV takes $n^{k-o(1)}$ time to decide for all but finitely many input lengths for any $d = \omega(\log n)$. Then, Protocol 3.2, when using Protocol 3.3 in place of Protocol 3.1, is a* non-interactive *(n^k, δ)-Proof of Work for k-OV in the Random Oracle model for any function $\delta(n) > 1/n^{o(1)}$.*

Efficiency and completeness of our now non-interactive Protocol 3.2 are easily seen to follow identically as in the proof of Theorem 6 in Section 3.3. Hardness also follow identically to the proof of Theorem 6's hardness except that the proof there required the *soundness* of Protocol 3.1, the interactive proof of $\mathcal{F}\mathsf{OV}^k$ that was previously used to implement Solve and Verify. To complete the proof of Theorem 9, then, we prove

Non-Interactive Proof for $\mathcal{F}\mathsf{OV}^k$:
The inputs to the protocol are $\boldsymbol{x} = (U_1, \ldots, U_k) \in \mathbb{F}_p^{knd}$ (a valid input to $f\mathsf{OV}^k_{n,d,p}$), and a field element $y \in \mathbb{F}_p$. The polynomials q are defined as in the text.

Prover$(\boldsymbol{x}, y)$**:**

- Compute coefficients of q_1. Let $\tau_1 = (q_1)$.
- For s from 1 to $k-2$:
  - Compute $\alpha_s = H(\boldsymbol{x}, y, \tau_s)$.
  - Compute coefficients of $q_{s+1} = q_{s+1,\alpha_1,\ldots,\alpha_s}$, with respect to $\boldsymbol{x}$.
  - Set $\tau_{s+1} = (\tau_s, \alpha_s, q_{s+1})$.
- Output τ_{k-1}

Verifier$(\boldsymbol{x}, y, \tau^*)$:

Given $\tau^* = (q_1, \alpha_1, q_2, \ldots, \alpha_{k-2}, q_{k-1})$, do the following:

- Check $\sum_{i_1 \in [n]} q_1(i_1) = y$. If check fails, reject.
- For s from 1 up to $k-2$:
  - Check that $\alpha_s = H(\boldsymbol{x}, y, q_1, \alpha_1, \ldots, \alpha_{s-1}, q_s)$.
  - Check that $\sum_{i_{s+1} \in [n]} q_{s+1}(i_{s+1}) = q_s(\alpha_s)$. If check fails, reject.
- Pick $\alpha_{k-1} \leftarrow \mathbb{F}_p$.
- Check that $q_{k-1}(\alpha_{k-1}) = \sum_{i_k \in [n]} q_{k,\alpha_1,\ldots,\alpha_{k-1}}(i_k)$. If check fails, reject.

If verifier has yet to reject, accept.

Protocol 3.3: A Non-Interactive Proof for $\mathcal{F}\mathsf{OV}^k$

the following lemma that Protocol 3.3 is also sound.

Lemma 6. *For any $k \geq 2$, if Protocol 3.1 is sound as an interactive proof, then Protocol 3.3 is sound as a non-interactive proof system in the Random Oracle model.*

Proof. Let P be a cheating prover for the non-interactive proof (Protocol 3.3) that breaks soundness with non-negligible probability $\varepsilon(n)$. We will construct a prover, P', that then also breaks soundness in the *interactive* proof (Protocol 3.1) with non-negligible probability.

Suppose P makes at most $m = \mathsf{poly}(n)$ queries to the random oracle, H; call them $\rho_1, \ldots, \rho_m$, and call the respective oracle answers $\beta_1, \ldots, \beta_m$.

For each $s \in [k-2]$, in order for the check on α_s to pass with non-negligible probability, the prover P must have queried the point $(\boldsymbol{x}, y, q_1, \alpha_1, \ldots, q_s)$. Hence, when P is able to make the verifier accept, except with negligible probability, there are $j_1, \ldots, j_{k-2} \in [m]$ such that the query ρ_{j_s} is actually $(\boldsymbol{x}, y, q_1, \alpha_1, \ldots, q_s)$, and β_{j_s} is α_s.

Further, for any $s < s'$, note that α_s is part of the query whose answer is $\alpha_{s'}$. So again, when P is able to make the verifier accept, except with negligible probability, $j_1 < j_2 < \cdots < j_{k-2}$. The interactive prover P' now works as follows:

- Select $(k-1)$ of the m query indices, and guess these to be the values of $j_1 < \cdots < j_{k-1}$.
- Run P until it makes the j_1^{th} query. To all other queries, respond uniformly at random as an actual random oracle would.
- If ρ_{j_1} is not of the form $(\boldsymbol{x}, y, q_1)$, abort. Else, sent q_1 to the verifier.
- Set the response to this query β_{j_1} to be the message α_1 sent by the verifier.
- Resume execution of P until it makes the j_2^{th} query from which q_2 can be obtained, and so on, proceeding in the above manner for each of the $(k-1)$ rounds of the interactive proof.

As the verifier's messages $\alpha_1, \ldots, \alpha_{k-2}$ are chosen completely at random, the oracle that P' is simulating for P is identical to the actual random oracle. So P would still be producing accepting proofs with probability $\varepsilon(n)$. By the earlier arguments, with probability nearly $\varepsilon(n)$, there are $(k-1)$ oracle queries of P that contain all the q_s's that make up the proof that it eventually produces. Whenever this is the case, if P' guesses the positions of these oracle queries correctly, the transcript of the interactive proof that it produces is the same as the proof produced by P, and is hence an accepting transcript.

Hence, when all of the above events happen, P succeeds in fooling the verifier. The probability of this happening is $\Omega(\varepsilon(n)/m^{k-1})$, which is still non-negligible as k

is a constant. This contradicts the soundness of the interactive proof, proving our lemma.

□

3.6 Zero-Knowledge Proofs of Work

In this section we show that the algebraic structure of our protocols can easily be exploited with mainstream cryptographic techniques to yield new protocols with desirable properties. In particular, we show that our Proof of Work scheme can be combined with ElGamal encryption and a zero-knowledge proof of discrete logarithm equality to get an non-repudiatable, non-transferable Proof of Work from the Decisional Diffie-Hellman assumption on Schnorr groups.

It should be noted that while general transformations are known for zero-knowledge protocols, many such transformations involve generic reductions with (relatively) high overhead. In the Proof of Work regime, we are chiefly concerned with the *exact* complexity of the prover and verifier. Even efficient transformations that go through circuit satisfiability must be adapted to this setting where no efficient deterministic verification circuit is known. That all said, the chief aim of this section is to exhibit the ease with which known cryptographic techniques can be used in conjunction the algebraic structure of the aforementioned protocols.

For simplicity of presentation, we demonstrate a protocol for $\mathcal{F}\mathsf{OV}^2$, however the techniques can easily be adapted to the protocol for general $\mathcal{F}\mathsf{OV}^k$.

Preliminaries. We begin by introducing a notion of honest verifier zero-knowledge scaled down to our setting. As the protocols under consideration have polynomial time provers, they are, in the traditional sense, trivially zero-knowledge. However, this is not a meaningful notion of zero-knowledge in this setting, because we are concerned with the exact complexity of the verifier. In order to achieve a meaningful notion of zero-knowledge, we must restrict ourselves to considering simulators of comparable complexity to the verifier (in this case, running in quasi-linear time). Similar notions

are found in [Pas03, BDSKM17], and [MP06] studies an interesting strenghtening of this notion..

Definition 9. An interactive protocol, $\Pi = \langle P, V \rangle$, for a function family, $\mathcal{F} = \{f_n\}$, is *T(n)-simulatable*, if for any $f_n \in \mathcal{F}$ there exists a simulator, $\mathcal{S}$, such that any x in the domain of f_n the following distributions are computationally indistinguishable,

$$\text{View}_{P,V}(x) \qquad \mathcal{S}(x),$$

where $\text{View}_{P,V}(x)$ denotes the distribution interactions between (honest) P and V on input x and $\mathcal{S}$ is randomized algorithm running in time $O(T(n))$.

Given the exposition above it would be meaningful to consider such a definition where we instead simply require the distributions to be indistinguishable with respect to distinguishers running in time $O(T(n))$. However, given that our protocol satisfies the stronger, standard notion of computational indistinguishability, we will stick with that.

Recall that *El Gamal encryption* consists of the following three algorithms for a group G of order p_λ with generator g.

$\mathsf{Gen}(\lambda; y) = (\mathsf{sk} = y, \mathsf{pk} = (g, g^y))$.

$\mathsf{Enc}(m, (a, b); r) = (a^r, mb^r)$.

$\mathsf{Dec}((c, d), y) = dc^{-y}$

El Gamal is a semantically secure cryptosystem (encryptions of different messages are computationally indistinguishable) if the *Decisional Diffie-Hellman assumption (DHH)* holds for the group G. Recall that DDH on G with generator g states that the following two distributions are compuationally indistinguishable:

- (g^a, g^b, g^{ab}) where a, b are chosen uniformly,
- (g^a, g^b, g^c) where a, b, c are chosen uniformly.

Protocol. Let $\mathbb{Z}_p$ be a Schnorr group such of size $p = qm+1$ such that $p = \Theta(n^{\log n})$ and $q > n^2$ are prime, and DDH holds with generator g. Let (E, D) denote an ElGamal encryption system on G. In what follows, we will take $R_{U,V}$ (or R^* for the honest prover) to be q (or q_1) as defined in Section 3.2. The protocol is described as Protocol 3.4.

Zero-Knowledge Proof of Work protocol:

- Challenge is issued as before: $(U, V) \leftarrow \mathbb{Z}_q^{2nd}$.
- Prover generates a secret key $x \leftarrow \mathbb{Z}_{p-1}$, and sends encryptions of the coefficients of the challenge response over the subgroup of size q to Verifier with the public key $(g, h = g^x)$ (and also sends the public key):

$$\begin{aligned}\mathsf{E}(R^*(\cdot); S(\cdot)) &= \mathsf{E}(mr_0^*; s_0), \ldots, \mathsf{E}(mr_{nd-1}^*; s_{nd-1}) \\ &= (g^{s_0}, g^{mr_0^*} h^{xs_0}), \ldots, (g^{s_{nd-1}}, g^{mr_{nd-1}^*} h^{xs_{nd-1}}).\end{aligned}$$

  Prover additionally draws $t \leftarrow \mathbb{Z}_{p-1}$ and sends $a_1 = g^t, a_2 = h^t$.
- Verifier draws random $z \leftarrow \mathbb{Z}_q$ and challenge $c \leftarrow \mathbb{Z}_p^*$ and sends to Prover.
- Prover sends $w = t + cS(z)$ to verifier.
- Verifier evaluates $y = f\mathsf{OV}_V(\phi_1(z), \ldots, \phi_d(z))$ to get g^{my}. Then, homomorphically evaluates $\mathsf{E}(R^*; S)$ on z so that $\mathsf{E}(R^*(z); S(z))$ equals

$$\begin{aligned}&\left((g^{s_0})(g^{s_1})^z \cdots (g^{s_{nd-1}})^{z^{nd-1}}, (g^{mr_0^*} h^{s_0})(g^{mr_1^*} h^{s_1})^z \cdots (g^{mr_{nd-1}^*} h^{s_{nd-1}})^{z^{nd-1}}\right) \\ &= (u_1, u_2)\end{aligned}$$

  Then, Verifier accepts if and only if

$$g^w = a_1 (u_1)^c \quad \& \quad h^w = a_2 (u_2 / g^{my})^c.$$

Protocol 3.4: Zero-Knowledge Proof of Work protocol based on the hardness of OV and the DDH assumption in Schnorr groups.

Remark 10. Note that the above protocol is public coin. Therefore, we can apply the Fiat-Shamir heuristic, and use a random oracle on partial transcripts to make the protocol non-interactive.

More explicitly, let H be a random oracle. Then:

- Prover computes

$$\begin{aligned}
&(g, h),\\
&\mathsf{E}(R^*; S),\\
&a_1 = g^t, a_2 = h^t,\\
&z = H(U, V, g, h, \mathsf{E}(R^*; S), a_1, a_2),\\
&c = H(U, V, g, h, \mathsf{E}(R^*; S), a_1, a_2, z),\\
&w = t + cS(z)
\end{aligned}$$

  and sends $(g, h, \mathsf{E}(R^*; S), a_1, a_2, w)$.

- Verifier calls random oracle twice to get

$$z = H(U, V, g, h, \mathsf{E}(R^*; S), a_1, a_2), c = H(U, V, g, h, \mathsf{E}(R^*; S), a_1, a_2, z).$$

  Then, the verifier homomorphically evaluates $\mathsf{E}(R^*; S)(z) = (u_1, u_2)$, it then computes the value $y = f\mathsf{OV}_V(\phi_1(z), \ldots, \phi_d(z))$. Finally, accepts if and only if

$$g^w = a_1(u_1)^c \quad \& \quad h^w = a_2(u_2/g^{my})^c.$$

Theorem 10. *Suppose* OV *takes* n^2 *time to decide for all but finitely many input lengths for any* $d = \omega(\log n)$ *and the DDH assumption holds in Schnorr groups, then Protocol 3.4 is a* $\tilde{O}(n)$*-simulatable* (n^2, δ)*-interactive Proof of Work scheme for any function* $\delta(n) > 1/n^{o(1)}$.

Proof. We show that Protocol 3.4 has all the necessary properties in turn.

Completeness: From before, if $R^* \equiv R_{U,V}$ as is the case for an honest prover, then for any $z \in \mathbb{Z}_q$ we have $R^*(z) = R_{U,V}(z) = f\mathsf{OV}_V(\phi_1(z), \ldots, \phi_d(z))$. Moreover

$$g^w = g^{t+cS(z)} = g^t(g^{S(z)})^c = a_1\left((g^{s_0})(g^{s_1})^z \cdots (g^{s_{nd-1}})^{z^{nd-1}}\right)^c,$$

and

$$\begin{aligned} h^w &= h^{t+cS(z)} \\ &= h^t(g^0 h^{S(z)})^c \\ &= a_2 \left((g^{mr_0^*} h^{s_0})(g^{mr_1^*} h^{s_1})^z \cdots (g^{mr_{nd-1}^*} h^{s_{nd-1}})^{z^{nd-1}} g^{-mf\mathsf{OV}_V(\phi_1(z),\ldots,\phi_d(z))} \right)^c. \end{aligned}$$

Hardness: Suppose a cheating prover runs in subquadratic time and succeeds in convincing the verifier. Then, by the hardness of Protocol 3.2, with probability at least $(1 - \delta(n))$ for some δ that is $1/n^{o(1)}$, we have that $R^* \not\equiv R_{U,V}$, in which case for random z, $R^*(z) \neq f\mathsf{OV}_V(\phi_1(z), \ldots, \phi_d(z))$ with overwhelming probability. Suppose this is the case in what follows, namely: $R^*(z) = y^* \neq y = f\mathsf{OV}_V(\phi_1(z), \ldots, \phi_d(z))$. In particular,

$$\log_g u_1 \neq \log_h (u_2 / g^{mf\mathsf{OV}_V(\phi_1(z),\ldots,\phi_d(z))}).$$

Note that $u_1, u_2/g^{mf\mathsf{OV}_V(\phi_1(z),\ldots,\phi_d(z))}$ can be calculated from the Prover's first message.

As is standard, we will fix the prover's first message and (assuming $y \neq y^*$) rewind any two accepting transcripts with distinct challenges to show that $\log_g u_1 = \log_h u_2/g^{my}$. Fix a_1, a_2 as above and let $(c, w), (c', w')$ be the two transcripts. Recall that if a transcript is accepted, $g^w = a_1 u_1^c$ and $h^w = a_2(u_2/g^{my})^c$. Then,

$$g^{w-w'} = u_1^{c-c'} \Rightarrow \log_g u_1 = \frac{w - w'}{c - c'} = \log_h u_2/g^{my} \Leftarrow h^{w-w'} = (u_2/g^{my})^{c-c'}.$$

Therefore, for any z such that $u_1 \neq u_2/g^{my}$, there can be at most one c for which a prover can convince the verifier. Such a c is chosen with negligible probability.

Thus, the cheating prover, who only succeeds when $R^* \not\equiv R_{U,V}$ or when a z is chosen such that $u_1 = u_2/g^{my}$ or when the above bad c is chosen otherwise, only does so with probability negligibly more than $\delta(n)$.

$\tilde{O}(nd)$-simulation: Given the verifier's challenge z, c, (which can simply be sampled uniformly, as above) we can efficiently simulate the transcript with respect to an honest prover as follows:

- Draw public key (g, h).
- Compute the ElGamal Encryption $\mathsf{E}_{g,h}(R'; S)$ where R' is the polynomial with constant term $f\mathsf{OV}_V(\phi_1(z), \ldots, \phi_d(z))$ and zeros elsewhere.
- Draw random w.
- Compute $a_1 = \frac{g^w}{g^{cS(z)}}$ and $a_2 = \frac{h^w}{h^{cS(z)}}$.
- Output $((g, h), \mathsf{E}_{g,h}(R'; S), a_1, a_2, z, c, w)$.

Notice that due to the semantic security of ElGamal, the transcript output is computationally indistinguishable from that of an actual transcript with the honest prover. Moreover, the simulator runs in $\tilde{O}(nd)$ time – this is the time taken to compute R', encrypt, evaluate S and exponentiate. Thus, the protocol is $\tilde{O}(nd)$-simulatable.

Efficiency: The honest prover runs in time $\tilde{O}(n^2)$, because the nd encryptions can be performed in time polylog(n) each. The verifier takes $\tilde{O}(nd)$ time as well. Note that the homomorphic evaluation requires $2nd$ exponentiations by $z, \ldots, z^{nd-1}$, each of which takes $O(\log(p)) = \text{polylog}(n)$ multiplications, followed by $d = \text{polylog}(n)$ multiplications.

□

Chapter 4

Cryptography Against Bounded Depth

In this chapter, we present constructions of a number of cryptographic primitives that are secure against low-depth circuits (and are also computable by such circuits). Our results are grouped into two classes. 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*. Our constructions for the former are unconditionally secure and build upon known lower bounds for this AC^0, while those for the latter rely on some minimal worst-case assumptions. Alternatively, these classes can be thought of as functions computable by machines that have polynomially many processors in parallel, where the resource we care about is the number of sequential levels of computation that is performed.

We note that, 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.

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 (weak) 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.

Our constructions against AC^0 adversaries are primarily based on the theorem of Braverman [Bra10] (and its recent sharpening by Tal [Tal14]) regarding the pseudorandomness of polylog-wise independent distributions against constant depth circuits. We use this to show that AC^0 circuits cannot distinguish between the distribution $(\boldsymbol{A}, \boldsymbol{A}\mathbf{k})$, where $\boldsymbol{A}$ is a random "sparse" matrix of dimension $\mathsf{poly}(n) \times n$ and $\mathbf{k}$ is a uniformly random secret vector, from the distribution $(\boldsymbol{A}, \boldsymbol{r})$, where $\boldsymbol{r}$ is a uniformly random vector. Sparse here means that each row of $\boldsymbol{A}$ 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 $\mathrm{polylog}(n)$ rows is linearly independent (Lemma 12). Note that when a set of rows of $\boldsymbol{A}$ is linearly independent, the corresponding set of bits in $\boldsymbol{A}\mathbf{k}$ is uniformly distributed. This implies that if all $\mathrm{polylog}(n)$-sized sets

of rows of $\boldsymbol{A}$ are linearly independent, then $\boldsymbol{Ak}$ is 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 $\boldsymbol{A}$ with the linear independence properties mentioned above, and the PRG output on seed $\mathbf{k}$ is $\boldsymbol{Ak}$. Pseudo-randomness follows from the indistinguishability arguments above. This is stated in the following informal restatement of Theorem 21 (along with Remark 11). We need to show that there exist such matrices $\boldsymbol{A}$ in which any polylog-sized set of rows are linearly independent, and yet are sparse. As we show in Section 4.1.3, there are bipartite expander graphs whose adjacency matrices have these properties.

Theorem 11 (Informal)**.** *For any constant c, there exists a family of Boolean 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, and 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 $\boldsymbol{x}$ to a sparse vector $\boldsymbol{a}$ and computing the inner product $\langle \boldsymbol{a}, \mathbf{k} \rangle$ over $\mathbb{F}_2$. Given polynomially many samples of the form $(\boldsymbol{a}, \langle \boldsymbol{a}, \mathbf{k} \rangle)$, one can write this as $(\boldsymbol{A}, \boldsymbol{A}\mathbf{k})$, where $\boldsymbol{A}$ is a matrix with random sparse rows. Our mapping of $\boldsymbol{x}$'s to $\boldsymbol{a}$'s is such that this is indistinguishable from $(\boldsymbol{A}, \boldsymbol{r})$ where $\boldsymbol{r}$ is uniformly random, via the arguments mentioned earlier in this section. The following is an informal restatement of Theorem 22.

Theorem 12 (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 polylog(n). So there is at least one coefficient in the case of such a function that is at least $\Omega\left(\frac{1}{2^{\text{polylog}(n)}}\right)$ 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 polylog(n) (which can be done in AC^0), one can distinguish functions computed in AC^0 from a random function with some $\Omega\left(\frac{1}{2^{\text{polylog}(n)}}\right)$ advantage. See Observation 1 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 sparse vector $\boldsymbol{c}$ such that $\langle \boldsymbol{c}, \mathbf{k} \rangle = b$ 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 $\boldsymbol{c}$, and decryption is performed by computing $\langle \boldsymbol{c}, \mathbf{k} \rangle$, which can be done in constant depth owing to the sparsity of $\boldsymbol{c}$. We reduce the semantic security of this construction to the indistinguishability of the distributions $(\boldsymbol{A}, \boldsymbol{A}\mathbf{k})$ and $(\boldsymbol{A}, \boldsymbol{r})$ mentioned earlier. Note that this scheme is additively homomorphic, a property that will be useful later. The following is an informal restatement of Theorem 23.

Theorem 13 (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 $\boldsymbol{M}$ that consists of encryptions of the bits of a string m, the evaluation of the function on input $\boldsymbol{x}$ is $\boldsymbol{M}\boldsymbol{x}$. 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 $\boldsymbol{M}$ is equivalent to finding a vector $\boldsymbol{u}$ such that $\boldsymbol{M}\boldsymbol{u} = 0$. By the additive homomorphism of the encryption scheme, and the fact that $\vec{0}$ is a valid encryption of 0, this implies that $\langle m, \boldsymbol{u} \rangle = 0$. 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 $\boldsymbol{m}_0$ and $\boldsymbol{m}_1$ at random and sends them to the challenger who responds with $\boldsymbol{M} = \mathsf{Enc}(\boldsymbol{m}_b)$. B then forwards this to A who would then return, with non-negligible probability, a vector $\boldsymbol{u}$ such that $\langle \boldsymbol{u}, \boldsymbol{m}_b \rangle = 0$. If B could compute $\langle \boldsymbol{u}, \boldsymbol{m}_0 \rangle$ and $\langle \boldsymbol{u}, \boldsymbol{m}_1 \rangle$, B would then be able to guess b correctly with non-negligible advantage. The problem with this approach is that $\boldsymbol{u}$, $\boldsymbol{m}_0$ and $\boldsymbol{m}_1$ 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 $\boldsymbol{m}_0$ to be a sparse vector and $\boldsymbol{m}_1$ to be a random vector and repeat the same procedure. This way, B can compute $\langle \boldsymbol{u}, \boldsymbol{m}_0 \rangle$, and while it still cannot check whether $\langle \boldsymbol{u}, \boldsymbol{m}_1 \rangle = 0$, it can instead check whether $\boldsymbol{M}\boldsymbol{u} = \vec{0}$ and use this information. If it turns out that $\boldsymbol{M}\boldsymbol{u} = \vec{0}$ and $\langle \boldsymbol{u}, \boldsymbol{m}_0 \rangle \neq 0$, then B knows that $b = 1$, due to the additive homomorphism of the encryption scheme. Also, when $b = 1$, since $\boldsymbol{m}_0$ is independent of $\boldsymbol{m}_1$, the probability that A outputs $\boldsymbol{u}$ such that $\langle \boldsymbol{u}, \boldsymbol{m}_0 \rangle \neq 0$ 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 25.

Theorem 14 (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 $\boldsymbol{M} = (\boldsymbol{A}, \boldsymbol{A}\mathbf{k})$ where $\boldsymbol{A}$ 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 $\boldsymbol{r}$ and output $\boldsymbol{c}^T = \boldsymbol{r}^T\boldsymbol{M}$, and to encrypt 1 we just output a random vector $\boldsymbol{c}^T$ of length $(n+1)$. Decryption is simply the inner product of $\boldsymbol{c}$ and the vector $(\mathbf{k}\ 1)^T$,

and can be done in AC^0 because $\mathbf{k}$ is sparse.

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) 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 15 (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 15 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 4.3.1 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 $\boldsymbol{M}$ 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 $\mathsf{ACC}^0[2]$.

The public key in our scheme is such a matrix $\boldsymbol{M}$, and the secret key is the corresponding $\mathbf{k}$. Encryption of a bit b is a vector $\boldsymbol{r}^T\boldsymbol{M} + b\boldsymbol{t}^T$, where $\boldsymbol{r}$ is a random vector[1] and $\boldsymbol{t}$ is a vector such that $\langle \boldsymbol{t}, \mathbf{k} \rangle = 1$. In effect, the encryption of 0 is a random vector in the row-span of $\boldsymbol{M}$ while the encryption of 1 is a random vector outside. Decryption of a ciphertext $\vec{c}$ is simply the inner product $\langle \vec{c}, \mathbf{k} \rangle$. 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

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

a random full rank matrix $\boldsymbol{M}'$ chosen from D_n^1; and (2) with $\boldsymbol{M}'$ as the public key, encryptions of 0 are identically distributed to the encryptions of 1. The following is an informal restatement of Theorem 27.

Theorem 16 (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)* $\mathsf{ACC}^0[2]$*.*

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 17 (Informal)**.** *If* $\oplus\mathsf{L}/\mathsf{poly} \neq \mathsf{NC}^1$*, then 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)* $\mathsf{ACC}^0[2]$*.*

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.

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 ACC^0[2].

Related Work

Arguably, 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.

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].

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].

4.1 Definitions and Preliminaries

In this section we establish notation that shall be used throughout the rest of this chapter 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 4.2. In Section 4.1.1, we present formal definitions of a complexity theoretic 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).

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 chapter will be

over $\mathbb{F}_2$, unless specified otherwise.

4.1.1 Bounded Adversaries

Definition 10 (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 11 (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 12 ($\mathsf{AC}^0[2]$)**.** 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 13 (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 14 ($\oplus$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 15 (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, $f_\lambda(x) \equiv B_\mathcal{F}(1^\lambda, x)$.

Next, 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 16 (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 negl such that for any $\lambda \in \mathbb{N}$:

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

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 17 (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 negl such that for any $\lambda \in \mathbb{N}$:

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

Definition 18 (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 negl such that for any $\lambda \in \mathbb{N}$:

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

Definition 19 (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 $\mathcal{G} = \left\{ g_\lambda : \left(\{0,1\}^\lambda \times \{0,1\}^{l(\lambda)}\right)^{n(\lambda)} \to \{0,1\} \right\} \in \mathcal{C}_2$ and any function $n : \mathbb{N} \to \mathbb{N}$, there is a negligible function negl such that for any $\lambda \in \mathbb{N}$:

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

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

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

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

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

for any $\lambda \in \mathbb{N}$:

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

Definition 21 (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 negl such that for any $\lambda \in \mathbb{N}$ and any $b \in \{0,1\}$:

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

- **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 negl$'$ such that for any $\lambda \in \mathbb{N}$:

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

4.1.2 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 18 (Hardness of Parity, [Hås14])**.** *For any circuit C with n inputs, size s and depth d,*

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

Theorem 19 (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 20 (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 $\mathcal{H}^t = \{h^t_\lambda\}$ 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 7 (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 $\mathcal{G}^t = \{g^t_\lambda\}$ 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 20 for function t' by $\mathcal{H}^{t'} = \left\{h^{t'}_\lambda\right\}$. Then, for any t satisfying the hypothesis for Lemma 7, a family $\mathcal{G}^t = \{g^t_\lambda\}$ 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 8 (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 $\mathcal{I}^t = \{ip^t_\lambda\}$ 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 7 and the fact that $\langle x, y \rangle = \mathsf{PARITY}(x_1 \wedge y_1, \ldots, x_\lambda \wedge y_\lambda)$. □

4.1.3 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 22 (Bipartite Expander)**.** An $(n, k, d, \gamma, \alpha)$-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.

Construction 4.1.1 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 $u^l_j = r_{((l-1)d+j-1)\lceil \log(k) \rceil + 1} \ldots r_{((l-1)d+j)\lceil \log(k) \rceil}$.
3. If there is no l such that for all distinct $j_1, j_2 \in [d]$, $u^l_{j_1} \neq u^l_{j_2}$, 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 $u^{l_0}_j = i$ (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.

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 9 (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. $\Delta(\mathsf{SRSamp}(k,d,r), SpR_{k,d}) \leq \nu(k)$
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 10 (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 9, 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:

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

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

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

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.

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

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:

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

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

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

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

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

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 11. *Let G be an $(n, k, d, \gamma, \alpha)$ 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 $(n, k, d, \gamma, \alpha)$-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 $\boldsymbol{x}$ in the $[n, (n-k)]_2$ code that has A_G as its parity check matrix. The fact that $\boldsymbol{x}$ belongs to the code implies that the rows indexed by non-zero positions in $\boldsymbol{x}$ sum to $\vec{0}^T$. But if $\boldsymbol{x} \leq \gamma n$, 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 $\vec{0}^T$. So $\|\boldsymbol{x}\| > \gamma n$, and the distance of the code is more than γn. □

The following is implied immediately by Lemmas 10 and 11 and says that with high probability the output of SMSamp defines a code with high distance.

Lemma 12 (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 one whose marginal distribution on any set of δ bits is the uniform distribution.

Lemma 13 (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* $\mathbf{H}\boldsymbol{x}$ *is* δ*-wise independent when* $\boldsymbol{x}$ *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 $\boldsymbol{y}$ in $\{0,1\}^n$ such that $\boldsymbol{y}^T\mathbf{H} = \vec{0}^T$ and $\|\vec{y}\| \leq \delta$. In particular, this implies that any set of δ rows of $\mathbf{H}$ are linearly independent. Hence, the restriction of $h(\boldsymbol{x}) = \mathbf{H}\boldsymbol{x}$ to any set of δ bits is a full rank linear transformation, and if $\boldsymbol{x}$ is distributed uniformly at random, then so are these bits. This implies that the distribution of $\mathbf{H}\boldsymbol{x}$ is δ-wise independent. □

The following is immediately implied by Lemmas 12, 13 and Theorem 19. It effectively says that AC^0 circuits cannot distinguish between $(\boldsymbol{A}, \boldsymbol{A}\boldsymbol{s})$ and $(\boldsymbol{A}, \boldsymbol{r})$ when $\boldsymbol{A}$ is sampled using SRSamp and $\boldsymbol{s}$ and $\boldsymbol{r}$ are chosen uniformly at random.

Lemma 14. *For any polynomial* n*, there is a negligible function* negl *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* $\boldsymbol{A} \leftarrow \mathsf{SMSamp}(n(k), k, \log^2(k))$*,* $\boldsymbol{s} \leftarrow \{0,1\}^k$ *and* $\boldsymbol{r} \leftarrow \{0,1\}^{n(k)}$*,*

$$|\Pr[g_\lambda(\boldsymbol{A}, \boldsymbol{A}\boldsymbol{s}) = 1] - \Pr[g_\lambda(\boldsymbol{A}, \boldsymbol{r}) = 1]| \leq \mathrm{negl}(\lambda)$$

4.1.4 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 23 (Perfect Randomized Encodings)**.** Let $f : \{0,1\}^n \to \{0,1\}^t$ be a deterministic function. 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.

4.2 Cryptography Against AC^0

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. The randomized encoding construction of [IK00] along with the lower bounds of [FSS84, Ajt83, Hås86] and

Theorem 26 implies the existence of One-Way Functions against AC^0. In this section, we present Pseudo-Random Generators with much better stretch than what is implied by Theorem 26, and constructions of 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.

4.2.1 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 11 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. Expander codes provide one such construction. If $\boldsymbol{H}$ is the parity-check matrix of a code with large distance d, then the distribution $\boldsymbol{H}\vec{x}$ is d-wise independent for $\vec{x}$ being a uniformly random vector (by Lemma 13). For expander codes, the $\boldsymbol{H}$ is sparse and hence we can compute the product $\boldsymbol{H}\vec{x}$.

Construction 4.2.1 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 12 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 $l(\lambda) \times \lambda$.

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

Theorem 21 (PRGs against AC^0)**.** *For any polynomial l, the family $\mathcal{F}^l$ from Construction 4.2.1 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 $\mathbf{H}_{l,\lambda}\vec{x}$. We know that each row of $\mathbf{H}_{l,\lambda}$ contains at most $\log^2(\lambda)$ non-zero entries. Hence, by Lemma 8, $\mathcal{F}^l$ 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 $\mathbf{H}_{l,\lambda}\vec{x}$ is $\Omega\left(\frac{\lambda}{\log^3(\lambda)}\right)$-wise independent, by Lemmas 13. And hence, Theorem 19 implies that this distribution is pseudo-random to adversaries in AC^0. □

Remark 11. 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 10 by setting $d = 8c$, $\alpha = 6c$, and $\gamma = \frac{\omega(\text{polylog}(k))}{n}$, 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 4.2.1 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.

4.2.2 Weak Pseudo-Random Functions

In this section, we describe our construction of Weak Pseudo-Random Functions against AC^0 computable in AC^0 (Construction 4.2.2). Roughly, we know that for a sparse matrix $\boldsymbol{H}$, $(\boldsymbol{H}, \boldsymbol{H}\mathbf{k})$ is indistinguishable from $(\boldsymbol{H}, \boldsymbol{r})$ where $\boldsymbol{r}$ 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 $\vec{x}$, the strategy is to use the input $\vec{x}$ to generate a sparse vector $\boldsymbol{y}$ and then take the inner product $\langle \boldsymbol{y}, \mathbf{k} \rangle$. The security of this construction would stem from the fact that after getting a lot of these random input-output pairs we can construct a sparse matrix $\boldsymbol{H}$ whose rows are the $\boldsymbol{y}$'s and then apply the above indistinguishability.

Theorem 22 (PRFs against AC^0)**.** *The pair of families* $(\mathcal{K}eygen, \mathcal{E}val)$ *defined in Construction 4.2.2 is a Weak* AC^0*-PRF against* AC^0*.*

The intuitive reason one would think Construction 4.2.2 might be pseudo-random is that a collection of random function values from a randomly sampled key seems to contain the same information as $(\boldsymbol{H}, \boldsymbol{H}\mathbf{k})$ where $\mathbf{k}$ is sampled uniformly at random

Construction 4.2.2 AC^0-PRF against AC^0

Let $\mathcal{I}^t = \{ip^t_\lambda\}$ be the inner product family with threshold promise t described in Lemma 8. Define families $\mathcal{K}eygen = \{\mathsf{KeyGen}_\lambda\}$ and $\mathcal{E}val = \{\mathsf{Eval}_\lambda\}$ as follows.
KeyGen_λ:

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

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

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

and $\boldsymbol{H}$ is sampled using SMSamp: a matrix with sparse rows. We know from Lemma 12 that except with negligible probability, H is going to be the parity check matrix of a code with large distance, and when it is, the arguments from Section 4.2.1 show that $(\boldsymbol{H}, \boldsymbol{H}\mathbf{k})$ is indistinguishable from $(\boldsymbol{H}, \boldsymbol{r})$, where $\boldsymbol{r}$ is sampled uniformly at random.

The only fact that prevents this from functioning as a proof is that what the adversary gets is not $(\boldsymbol{y}, \langle \boldsymbol{y}, \mathbf{k}\rangle)$ where $\boldsymbol{y}$ is an output of SRSamp, but rather $(\boldsymbol{r}, \langle \boldsymbol{y}, \mathbf{k}\rangle)$, where $\boldsymbol{r}$ is randomness that when used in SRSamp gives $\boldsymbol{y}$. One way to show that this is still pseudo-random is to reduce the case where the input is $(\boldsymbol{y}, \langle \boldsymbol{y}, \vec{x}\rangle)$ to the case where the input is $(\mathbf{r}, \langle \boldsymbol{y}, \vec{x}\rangle)$ using an AC^0-reduction. To do this, one would need an AC^0 circuit that would, given $\boldsymbol{y}$, sample from a distribution close to the uniform distribution over r's that cause SRSamp to output $\boldsymbol{y}$ when used as randomness. We implement this proof strategy below. To prove Theorem 22, we first show that there exists an AC^0 circuit that would, given $\boldsymbol{y}$, sample from a distribution close to the uniform distribution over $\mathbf{r}$'s that cause SRSamp to output $\boldsymbol{y}$ when used as randomness.

Lemma 15 (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 (randomised) circuit $C^{inv}_{k,d}$ of size at most $s(k)$ and depth c', and a negligible function ν such that for any $\boldsymbol{y} \in \{0,1\}^k$ that has exactly d non-zero entries,*

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

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

Proof. For any k that is a power of 2 and any d, given input $\boldsymbol{y}$ 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 $v = (v_1, \ldots, v_{i-1}, z', v_{i+1}, \ldots, v_d)$.

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 9).

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 20, such a circuit exists for $j \leq d = O(\log^c(k))$. Note that out of all $l \in [d]$, exactly one of $(y_l \wedge Ham_{j-1,l-1}(y_1 \dots y_{l-1}))$ is true. So z_j can be computed as follows:

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

   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 $p_1, \dots, p_d \in [k]$. 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:

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

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 randomised 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 22. *KeyGen* and *Eval* 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 8. 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, $k, \vec{x}_i \leftarrow \{0,1\}^\lambda$, $y_i \leftarrow SpR_{\lambda,\log^2(\lambda)}$, $w_i \leftarrow R_{y_i}$, $z_i \leftarrow C^{inv}_{\lambda,\log^2(\lambda)}(y_i)$, $\mathbf{r}_i \leftarrow \{0,1\}^{\lambda-\log^5(\lambda)}$, and $b_i \leftarrow \{0,1\}$.

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

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

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 9. 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 15. D_4 and D_5 are indistinguishable by AC^0 adversaries because Lemma 14 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 15.

D_5 and D_6 are statistically close by Lemma 15. 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 4.2.2, 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 4.2.2 against AC^0 adversaries. □

Remark 12. 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 4.2.2

from a random function by choosing $\vec{x}_1, \vec{x}_2, \vec{x}_3$ such that $\mathsf{SRSamp}(\lambda, \log^2 \lambda, \vec{x}_1) + \mathsf{SRSamp}(\lambda, \log^2 \lambda, \vec{x}_2) = \mathsf{SRSamp}(\lambda, \log^2 \lambda, \vec{x}_3)$ and then checking if $f(\vec{x}_1) + f(\vec{x}_2) = f(\vec{x}_3)$.

Construction 4.2.2 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, as detailed in Observation 1.

Observation 1. *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 $\mathbf{E}_s\left[\widehat{f}(s)^2\right] = \Omega(\frac{1}{m^{\Omega(\log^{d-1} m)}})$. □

4.2.3 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 4.2.2, 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 4.2.2, 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 polylog(λ) 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 4.2.3. 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 4.2.3 AC^0-Symmetric Key Encryption against AC^0

Let $\mathcal{I}^t = \{ip^t_\lambda\}$ be the inner product family with threshold promise t described in Lemma 8. 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 $ip^{\log^2(\lambda)}_{\lfloor \lambda \rfloor_2}(\mathbf{k}, \mathbf{c})$.

Theorem 23 (Symmetric Encryption Against AC^0)**.** *The tuple composed of the families* $(\mathcal{K}eygen, \mathcal{E}nc, \mathcal{D}ec)$ *defined in Construction 4.2.3 is an* AC^0*-Symmetric-Key Bit-Encryption Scheme against* AC^0*.*

The key idea behind the proof is showing that for most keys, encryption via rejection sampling for a randomly chosen bit b, that is,

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

is statistically close to the distribution where we pick the encryption randomness $\mathbf{r}$ first and then the bit is the inner product, 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 weak PRF distribution whose security we have already proved. We implement this strategy below.

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 16. *For the family* $\mathcal{K}eyGen = \mathsf{KeyGen}_\lambda$ *defined in Construction 4.2.3, 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 17. *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 17 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 18. *For every $b \in \{0,1\}$ and any vector $\mathbf{k}$ 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 19. *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 18 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 17, $\langle \mathbf{r}, \mathbf{k} \rangle$ 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 4-1. 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 23.

Game 1 - Semantic Security

Ch		**Adv**
$\mathbf{k} \leftarrow \mathsf{KeyGen}_\lambda$	$\xrightarrow{\{\mathbf{c}_i^0\}_i \leftarrow \mathsf{Enc}_\lambda(\mathbf{k}, 0)}$	
	$\xrightarrow{\{\mathbf{c}_i^1\}_i \leftarrow \mathsf{Enc}_\lambda(\mathbf{k}, 1)}$	
$b \leftarrow \{0,1\}$	$\xrightarrow{\mathbf{c} = \mathsf{Enc}_\lambda(\mathbf{k}, b)}$	
	$\xleftarrow{b'}$	

Game 2 - Random Samples

Ch		**Adv**
$\mathbf{k} \leftarrow \mathsf{KeyGen}_\lambda$,		
$\{b_i\}_i \leftarrow \{0,1\}$	$\xrightarrow{\{(b_i, \mathsf{Enc}_\lambda(\mathbf{k}, b_i))\}_i}$	
$b \leftarrow \{0,1\}$	$\xrightarrow{\mathbf{c} = \mathsf{Enc}_\lambda(\mathbf{k}, b)}$	
	$\xleftarrow{b'}$	

Game 3 - Assumption

Ch		**Adv**
$\mathbf{k} \leftarrow \mathsf{KeyGen}_\lambda$,		
$\{\mathbf{r}_i\}_i \leftarrow \mathsf{SRSamp}$	$\xrightarrow{\{(\langle \mathbf{k}, \mathbf{r}_i \rangle, \mathbf{r}_i)\}_i}$	
$b \leftarrow \{0,1\}$	$\xrightarrow{\mathbf{c} = \mathsf{Enc}_\lambda(\mathbf{k}, b)}$	
	$\xleftarrow{b'}$	

Figure 4-1: Security Games

Claim 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 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 19 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 16 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 23. It is easy to see that $\mathcal{K}eyGen \in \mathsf{AC}^0$. 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 8. So computability is satisfied. Non-triviality is also easily seen to be satisfied.

Correctness follows from Lemma 18, 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 16 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 1 and 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 $(\boldsymbol{R}, \boldsymbol{R}\mathbf{k})$ where $\boldsymbol{R} \leftarrow \mathsf{SMSamp}(m(\lambda), \lambda, \log^2 \lambda)$ 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 $\begin{pmatrix} \boldsymbol{R} & \boldsymbol{R}\mathbf{k} \\ \mathbf{r}^\star & b \end{pmatrix}$. This is indistinguishable from $\begin{pmatrix} \boldsymbol{R} & \mathbf{r} \\ \mathbf{r}^\star & \mathbf{r}' \end{pmatrix}$ 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 14 and 19. 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 24.

Theorem 24. *No* AC^0 *adversary has non-negligible advantage against the Symmetric Key Encryption scheme from Construction 4.2.3 in the Multibit Semantic Security game given below.*

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

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

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

In hybrid i, the adversary has to distinguish the $\mathsf{Enc}(\mathbf{k}, \boldsymbol{m}'_i)$ from $\mathsf{Enc}(\mathbf{k}, \boldsymbol{m}'_{i+1})$. The advantage the adversary has in distinguishing between $\mathsf{Enc}(\mathbf{k}, \boldsymbol{m}_0)$ and $\mathsf{Enc}(\mathbf{k}, \boldsymbol{m}_1)$ 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 $\boldsymbol{m}'_i$ and $\boldsymbol{m}'_{i+1}$ differ only in one bit.

If there exists an AC^0 adversary A that has a non-negligible advantage in distinguishing between $\mathsf{Enc}(\mathbf{k}, \boldsymbol{m}'_i)$ and $\mathsf{Enc}(\mathbf{k}, \boldsymbol{m}'_{i+1})$ 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 -

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

$\mathbf{c}^\star$ has the same distribution as $\left\{\boldsymbol{m}'_i, \boldsymbol{m}'_{i+1}\right\}$ 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. □

4.2.4 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 4.2.3. 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.

Theorem 25 (CRHFs Against AC^0)**.** *For any polylogarithmic function s, the pair of families $(\mathcal{KeyGen}^s, \mathcal{Eval}^s)$, from Construction 4.2.4 is an AC^0-CRHF with compression s.*

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

Construction 4.2.4 AC^0-CRHFs against AC^0

Let $\mathcal{I}^t = \{ip^t_\lambda\}$ be the inner product family with threshold promise t described in Lemma 8. Let $(\mathcal{K}eyGen^{Enc}, \mathcal{E}nc^{Enc})$ be the SKE scheme from Construction 4.2.3. 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 $\boldsymbol{M} = (\boldsymbol{m}_1, \ldots, \boldsymbol{m}_\lambda)$, where $\boldsymbol{m}_i \leftarrow \mathsf{Enc}^{Enc}_{l(\lambda)}(\mathbf{k}, b_i)$.

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

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

---

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 4.2.3 being in AC^0, and for Eval^s_λ because of Lemma 8. 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 $\boldsymbol{M}$ is negligibly close to that of the transpose of matrices sampled from $SpM_{\lambda, l(\lambda), \log^2(l(\lambda))}$, by Lemmas 16, 19, and 9. 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, $\mathsf{Eval}^s_\lambda(\boldsymbol{M}, \vec{x}) = \boldsymbol{M}\vec{x}$.

Hence, to prove security, it is sufficient to show that for an adversary in AC^0, for most such $\boldsymbol{M}$, it is hard to find an $\vec{x} \in \{0,1\}^\lambda$, $\vec{x} \neq 0$ such that $\boldsymbol{M}\vec{x} = 0$, 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 $\boldsymbol{M}$ 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 4.2.3. Say there is a polynomial p such that

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

We know from Theorem 24 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 $\boldsymbol{m}_0 \leftarrow \{0,1\}^\lambda$ and $\boldsymbol{m}_1 \leftarrow \mathsf{SRSamp}(\lambda, 1)$ and sends them to the challenger
2. Upon receiving ciphertext $\boldsymbol{M}$ from the challenger, it computes $\vec{v} = \mathcal{A}(\boldsymbol{M})$.
3. If $\boldsymbol{M}\vec{v} = 0$ and $\langle \vec{v}, \boldsymbol{m}_1 \rangle \neq 0$, 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 $\boldsymbol{m}_1$, which has at most one non-zero entry.

Note that by the correctness of Construction 4.2.3, the probability that $\mathsf{Enc}_{l(\lambda)}^{Enc}$ produces a ciphertext that does not decrypt correctly is negligible. This means that except with negligible probability, if $\mathbf{c}$ is an encryption of $\boldsymbol{m}$ under key $\mathbf{k}$, then $\langle \mathbf{c}, \mathbf{k} \rangle = \boldsymbol{m}$; in this case the construction is additively homomorphic - if $\mathbf{c}_1$ and $\mathbf{c}_2$ are encryptions of $\boldsymbol{m}_1$ and $\boldsymbol{m}_2$ under the same key, then $(\mathbf{c}_1 \oplus \mathbf{c}_2)$ is an encryption of $(\boldsymbol{m}_1 \oplus \boldsymbol{m}_2)$. Then the following arguments follow except with negligible probability.

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

On the other hand, if $b = 0$ ($\boldsymbol{m}_0$ was encrypted), then the distribution of $\boldsymbol{M}$ is the same as that generated by $\mathsf{KeyGen}_\lambda^s$. In this case, the adversary $\mathcal{A}$ will generate

a vector $\vec{v} \neq 0$ in the kernel of $\boldsymbol{M}$ with probability at least $\frac{1}{p(\lambda)}$. Conditioned on this happening, if it turns out that $\langle \boldsymbol{m}_1, \vec{v} \rangle \neq 0$, 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 $\langle \boldsymbol{m}_1, \vec{v} \rangle$ is non-zero with some inverse polynomial probability.

When $b = 0$, $\boldsymbol{m}_1$ - the sparse vector - is sampled independently from $\boldsymbol{M}$ and hence from $\vec{v}$. As observed earlier, by Lemmas 16 and 19, the distribution of $\boldsymbol{M}^T$ is negligibly close to that of $\mathsf{SMSamp}(\lambda, l(\lambda), \log^2(l(\lambda)))$. So Lemma 12 implies that except with negligible probability, the code whose parity check matrix is $\boldsymbol{M}^T$ has distance at least $\frac{\lambda}{\log^3 \lambda}$. In this case, $\|\vec{v}\|_0 \geq \frac{\lambda}{\log^3 \lambda}$.

Since the Hamming weight of $\boldsymbol{m}_1$ is 1, the probability of the $\langle \boldsymbol{m}_1, \vec{v} \rangle$ being non-zero is simply $\frac{\|\vec{v}\|_0}{\lambda} \geq \frac{1}{\log^3 \lambda}$. 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 4.2.3 that was established in Theorem 24, which is a contradiction. This completes the argument, demonstrating the collision resistance of Construction 4.2.4. □

4.2.5 Candidate Public Key Encryption Scheme

In Lemma 14 we showed that the distribution $(\boldsymbol{A}, \boldsymbol{A}\mathbf{k})$ where $\boldsymbol{A}$ was sampled as a sparse matrix and $\mathbf{k}$ was a random binary string is indistinguishable from $(\boldsymbol{A}, \mathbf{r})$ where $\mathbf{r}$ is also a random string, for a wide range of parameters. We need atleast one of the two $\boldsymbol{A}$ or $\mathbf{k}$ to be sparse to enable computation in AC^0. If we make the analogous assumption with the key being sparse, that is $(\boldsymbol{A}, \boldsymbol{A}\mathbf{k})$ is indistinguishable from $(\boldsymbol{A}, \mathbf{r})$ where $\boldsymbol{A} \leftarrow \{0,1\}^{\lambda \times \lambda}$, $\mathbf{k} \leftarrow \mathsf{SRSamp}(\lambda, \log^2 \lambda)$ 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 4.2.5, and is easily seen to be secure under Assumption 1. This candidate is very similar to the LPN based cryptosystem due to Alekhnovich [Ale03]. Note that while the correctness of decryption in Construction 4.2.5 is not very good, this may be easily amplified by repetition without losing

security, as the error is one-sided.

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

Construction 4.2.5 Public key encryption

Let $\mathcal{I}^t = \{ip^t_\lambda\}$ be the inner product family with threshold promise t described in Lemma 8. 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 $\boldsymbol{A} \leftarrow \{0,1\}^{\lambda \times \lambda - 1}$, $\mathbf{k} \leftarrow \mathsf{SRSamp}(\lambda - 1, \log^2 \lambda)$
2. Output $(\mathsf{pk}, \mathsf{sk}) = ((\boldsymbol{A}, \boldsymbol{A}\mathbf{k}), \mathbf{k} \circ 1)$.

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

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

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

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

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

4.3 Cryptography Against NC^1

In this section, we describe some constructions of cryptographic primitives against bounded adversaries starting from worst-case hardness assumptions. In Section 4.3.1, we describe a set of generic conditions involving Randomized Encodings that imply One-Way Functions and Pseudo-Random Generators against such adversaries; the intended application of this theorem is to construct such primitives against NC^1 under the assumption that this class does not contain the class $\oplus\mathsf{L}/\mathsf{poly}$. Under the same assumption, in Section 4.3.2, we construct Public-Key Encryption and Collision Resistant Hash Functions against NC^1.

4.3.1 OWFs from worst-case 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 13 (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 4.3 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 26 (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 26 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 $\oplus\mathsf{L}/\mathsf{poly}$ (following known constructions of Randomized Encodings for this class [IK00]). 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:

$$\left| \Pr_{x \leftarrow U_\lambda} \left[a_\lambda(\widehat{f_\lambda}(0^{n(\lambda)}, x)) = 1 \right] - \left(\frac{1}{2} \Pr_{x \leftarrow U_\lambda} \left[a_\lambda(\widehat{f_\lambda}(0^{n(\lambda)}, x)) = 1 \right] + \frac{1}{2} \Pr_{x \leftarrow U_\lambda} \left[a_\lambda(\widehat{f_\lambda}(z_1, x)) = 1 \right] \right) \right|$$
$$= \frac{1}{2} \left| \Pr_{x \leftarrow U_\lambda} \left[a_\lambda(\widehat{f_\lambda}(0^{n(\lambda)}, x)) = 1 \right] - \Pr_{x \leftarrow U_\lambda} \left[a_\lambda(\widehat{f_\lambda}(z_1, x)) = 1 \right] \right|$$

which is non-negligible. Denote the probability $\Pr_{x \leftarrow U_\lambda} \left[a_\lambda(\widehat{f_\lambda}(0^{n(\lambda)}, x)) = 1 \right]$ by p, and denote $\Pr_{x \leftarrow U_\lambda} \left[a_\lambda(\widehat{f_\lambda}(z_1, x)) = 1 \right]$ by q.

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 earlier in the chapter, 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 13).

□

4.3.2 PKE and CRHF against NC^1

In Theorem 26 we saw that we can construct one way functions and PRGs with a small stretch generically from Perfect Randomised 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 $\mathsf{ACC}^0[2]$ 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 4.3.1.

Construction 4.3.1 Sampling distributions from [IK00]

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

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

$\mathsf{LSamp}(n)$:

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 $\vec{r} \leftarrow \{0,1\}^{n-1}$.
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 $\boldsymbol{M}$ sampled identically to $\boldsymbol{R}_1\boldsymbol{M}_0^\lambda\boldsymbol{R}_2$ when $f(x) = 0$ and identically to $\boldsymbol{R}_1\boldsymbol{M}_1^\lambda\boldsymbol{R}_2$ when $f(x) = 1$, where $\boldsymbol{R}_1 \leftarrow \mathsf{LSamp}(\lambda)$ and $\boldsymbol{R}_2 \leftarrow \mathsf{RSamp}(\lambda)$. Notice that $\boldsymbol{R}_1\boldsymbol{M}_1^\lambda\boldsymbol{R}_2$ is full rank, while $\boldsymbol{R}_1\boldsymbol{M}_0^\lambda\boldsymbol{R}_2$ has rank $(\lambda - 1)$. The public key in our encryption scheme is a sample $\boldsymbol{M}$ from $\boldsymbol{R}_1\boldsymbol{M}_0^\lambda\boldsymbol{R}_2$, and the secret key is a vector $\mathbf{k}$ in the kernel of $\boldsymbol{M}$. An encryption of 0 is a random vector in the row-span of $\boldsymbol{M}$ (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 $\boldsymbol{M}$ (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 $\boldsymbol{M}$ is indistinguishable from $\boldsymbol{R}_1\boldsymbol{M}_1^\lambda\boldsymbol{R}_2$ (see Theorem 28), which has an empty kernel, and so when used as the public key results in identical distributions of encryptions of 0 and 1.

Construction 4.3.2 Public Key Encryption

Let λ be the security parameter. Let $\boldsymbol{M}_0^\lambda$ be the $\lambda \times \lambda$ matrix described in Construction 4.3.1. 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 $\boldsymbol{R}_1 \leftarrow \mathsf{LSamp}(\lambda)$ and $\boldsymbol{R}_2 \leftarrow \mathsf{RSamp}(\lambda)$.
2. Let $\mathbf{k} = (\boldsymbol{r}\ 1)^T$ be the last column of $\boldsymbol{R}_2$.
3. Compute $\boldsymbol{M} = \boldsymbol{R}_1\boldsymbol{M}_0^\lambda\boldsymbol{R}_2$.
4. Output $(\mathsf{pk} = \boldsymbol{M}, \mathsf{sk} = \mathbf{k})$.

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

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

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

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

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 28.

Theorem 27 (Public Key Encryption Against NC^1)**.** *Assume* $\oplus\mathsf{L}/\mathsf{poly} \not\subseteq \mathsf{NC}^1$*. Then, the tuple of families* $(\mathcal{K}eygen, \mathcal{E}nc, \mathcal{D}ec)$ *defined in Construction 4.3.2 is an* $\mathsf{ACC}^0[2]$*-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 4.3.3.

Construction 4.3.3 Sampling procedures

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

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

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

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

Theorem 28 ([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 the inability to compute some function in $\oplus\mathsf{L}/\mathsf{poly}$ tranlates to the inability 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 20. *If $\oplus\mathsf{L}/\mathsf{poly} \not\subseteq \mathsf{NC}^1$, then there is a polynomial n and a negligible function* negl *such that for any family $\mathcal{F}=\{f_\lambda\}$ in NC^1, for an infinite number of values of λ,*

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

Now we are in a position to use the indistinguishability result in Lemma 20 to prove Theorem 27.

Proof of Theorem 27. To prove the theorem, we need to show that the functions in the construction are computable in $\mathsf{ACC}^0[2]$ 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 $\mathsf{ACC}^0[2]$, 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 4.3.2. For any $(\boldsymbol{M}, \mathbf{k}) \leftarrow \mathsf{KeyGen}_\lambda$, note that $\boldsymbol{M} = \boldsymbol{R}_1 \boldsymbol{M}_0^\lambda \boldsymbol{R}_2$, and $\mathbf{k} = (r\ 1)^T$ is the last column of R_2. It can be verified easily that $\boldsymbol{M}\mathbf{k} = \boldsymbol{R}_1(\boldsymbol{M}_0^\lambda \boldsymbol{R}_2 s) = \vec{0}$, and that $\langle \vec{t}, \mathbf{k} \rangle = 1$. So for any b, $\mathsf{Dec}_\lambda(\mathbf{k}, \mathsf{Enc}_\lambda(\boldsymbol{M}, b)) = \langle \boldsymbol{M}^T \vec{r} + b\vec{t}, \mathbf{k} \rangle = \vec{r}^T \boldsymbol{M}\mathbf{k} + b\langle \vec{t}, \mathbf{k} \rangle = b$. This proves correctness.

To prove semantic security, we need to show that the distributions $(\mathsf{pk}, \mathsf{Enc}_\lambda(\mathsf{pk}, 0))$ 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,

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

For any adversary in NC^1, we know by Lemma 20 that there are an infinite number of values of λ for which:

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

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

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

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

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

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

Remark 14. The computation of the PRE from [IK00] can be moved to NC^0 by techniques noted in [IK00] itself. Using similar techniques with Construction 4.3.2 gives us a Public Key Encryption scheme with encryption in NC^0 and decryption and key generation in ACC^0[2]. The impossibility of decryption in NC^0, as noted in [AIK04], continues to hold in our setting.

Remark 15. (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 $\oplus\mathsf{L/poly} \not\subseteq \mathsf{NC}^1$).

Note that again, due to the linearity of decryption, Construction 4.3.2 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 $(m_1 \oplus m_2)$. Further, the size of ciphertexts does not increase when this operation is performed. Now 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 $\mathsf{ACC}^0[2]$, to get the following.

Theorem 29. *Assume* $\oplus\mathsf{L/poly} \not\subseteq \mathsf{NC}^1$. *Then, for any constant* $c < 1$ *and function* s *such that* $s(n) = O(n^c)$, *there exists an* $\mathsf{ACC}^0[2]$*-CRHF against* NC^1 *with compression* s.

Chapter 5

Conclusion and Future Directions

So far we have described our attempts to construct fine-grained cryptographic objects in a number of different settings, starting from various worst-case hardness assumptions (an in some cases no assumptions). We believe this line of research could be the way forward towards weakening, and eventually removing, the computational assumptions necessary for doing cryptography. There are many interesting questions to be asked here, the answers to which stand to several yield useful insights and implications, whichever way they may go.

In this section, we briefly review some of the work that has been done in this area since the work in this thesis, present some directions onward, and pose a number of questions that we believe are significant.

5.1 Subsequent Work

Subsequently and in parallel to our work, a number of others have explored questions that are directly or indirectly related to fine-grained cryptography. Here we briefly describe a few of them.

Goldreich and Rothblum, in a series of recent papers, broadly study fine-grained average-case reductions for problems within P and interactive proofs for some of these problems, both of which are relevant to our study. In [GR17], building on ideas from Williams [Wil16], they construct doubly-efficient interactive proofs for problems

that are "locally-characterizable". Roughly, a language is locally-characterizable if membership in it can be determined by a conjunction of a polynomial number of predicates that each take logarithmically many bits, and are applied to different predetermined parts of the input. Essentially, these are problems that can be reduced to the evaluation of a low-degree polynomial in the bits of the input, and this includes the problems that we use in our constructions such as OV, 3SUM, etc. (and their complements). A doubly-efficient interactive proof is one in which the prover runs in polynomial time, and the verifier in near-linear time. Their constructions enable the use of other plausibly moderately hard problems that are locally-characterizable to be used to construct Proofs of Work along the lines of our work.

In [GR18], they generalize our average-case reductions, define a hierarchy of classes of counting problems that also have the flavour of the locally characterizable sets, and show that in each of these classes, solving any problem in the worst-case can be reduced in near-linear time to solving some other problem in the class on average. Finally, in [GR18], they partially answer Question 5, showing a fine-grained reduction from the problem of counting the number of t-cliques in a graph to itself on average.

Using our average-case reduction and the structure of the problems that we reduce to, Carmosino et al [CIS18] show uniform derandomisation results for BPP based on SETH and other conjectures from fine-grained complexity.

Campanelli and Gennaro [CG18] study fine-grained secure computation against low-depth circuits. They present constructions of fully homomorphic encryption and verifiable computation computable in and secure against NC^1 starting from the worst-case assumption we start with in this setting – that NC^1 is not contained in $\oplus\mathsf{L}/\mathsf{poly}$.

5.2 Towards Fine-Grained One-Way Functions

In this section, to illustrate the kinds of objects, approaches, and difficulties fine-grained cryptography might involve, we define a notion of fine-grained one-way function (OWF), discuss a natural approach to realizing it, show a structural barrier to basing this approach on certain problems, and outline as an open problem a way to

circumventing this barrier. A fine-grained OWF would capture the same concept as a standard OWF – easy to compute yet hard to invert – but with a more fine-grained interpretation of "easy" and "hard".

Definition 24. We say a function $f : \{0,1\}^* \to \{0,1\}^*$ is (t, ϵ)-*one-way* if it can be computed in $O(t(n)^{1-\delta})$ time for some $\delta > 0$, but for any $\delta' > 0$, any $O(t(n)^{1-\delta'})$-time algorithm A, and all sufficiently large n,

$$\Pr_{x \leftarrow \{0,1\}^n} \left[A(f(x)) \in f^{-1}(f(x)) \right] \leq \epsilon(n, \delta')$$

The primary question we ask and discuss in this section is the following. We consider this to be the most significant open problem at present in the area of fine-grained cryptography. Though one-way functions by themselves are not immediately applicable in practice, they are among the simplest cryptographic primitives, and trying to construct them seems like the way ahead towards understanding what we have to work with and constructing more sophisticated primitives.

Question 1. *Can we construct fine-grained one-way functions based on the worst-case hardness of well-studied problems from fine-grained complexity?*

We discuss possibilities for the above question starting from the OV problem, as it leads to some other questions that are interesting in their own right. One approach to constructing such OWFs (that has perhaps been part of folklore for decades) is as follows. Take one of our hard-on-average to evaluate polynomials – e.g. $f\mathsf{OV}_n$ – and suppose there was an input-output sampling algorithm $S \equiv (S_1, S_2)$ that runs in sub-quadratic time in n such that, on uniform input r, $S_1(r)$ is distributed uniformly over the appropriate domain, and $S_2(r) = f\mathsf{OV}_n(S_1(r))$. By our results it can be seen that S_1 is $(n^2, 3/4)$-one-way if we assume the OV conjecture (strengthened to assume hardness for all sufficiently large input sizes).

5.2.1 Barriers and NSETH

However, as stated, there turn out to be certain barriers to this approach. For instance, if $S(r) = (x, y)$, then r would be a certificate that $f\mathsf{OV}_n(x) = y$ that can be verified in sub-quadratic time. In particular, if $x \in \{0,1\}^{2nd}$ is a NO instance of OV, then r is a certificate for this that is verifiable in deterministic sub-quadratic time – i.e. that $f\mathsf{OV}_n(x) = 0$.

Further, tracing this back through the reduction of k-SAT to OV (see [Wil15]), this gives us a certificate for NO instances of k-SAT (for any k) that are verifiable in $O(2^{n(1-\epsilon)})$ time for some $\epsilon > 0$ – i.e. short and quickly verifiable certificates for $\mathsf{CNF\text{-}UNSAT}$ instances. Interestingly, the impossibility of this was recently conjectured and formalized as NSETH (the Non-deterministic Strong Exponential Time Hypothesis) in [CGI+16], where it was noted that its falsification would yield breakthroughs in both circuit complexity and proof complexity.

An alternative view of matters would be that this presents another approach to breaking NSETH. In fact, something weaker would suffice for this purpose – one only needs a sampler that runs in sub-quadratic time and samples $(x, f\mathsf{OV}_n^k(x))$ for some k such that the distribution of x has $\{0,1\}^{knd}$ in its support.

Note that while a sampler based on OV would, because of its relation to k-SAT, break NSETH, a sampler based on ZWT would only yield quick non-deterministic certification for APSP instances (the reduction to ZWT from $\mathsf{3SUM}$ is randomized). As APSP (and also $\mathsf{3SUM}$) is known to have small co-nondeterministic complexity [CGI+16], the above barrier does not show up here. Still, $\mathcal{F}\mathsf{OV}$ is much simpler algebraically and so seems to hold more hope for constructing a fine-grained OWF in this manner. We now discuss ways in which we may still salvage a sampler for $\mathcal{F}\mathsf{OV}$.

5.2.2 A Way Around

There are visible ways to skirt this NSETH barrier: Suppose that the sampler S was not perfect – that with some small probability over r it outputs (x, y) such that $f\mathsf{OV}_n(x) \neq y$. Immediately, NSETH no longer applies, as now an r such that

$S(r) = (x, y)$ may no longer be a sound certificate that $f\mathsf{OV}_n(x) = y$. And, as explained below, such a sampler still gives us a fine-grained *distributional* OWF based on the hardness of OV as long as the probability that it is wrong is small enough.

Informally, a distributional OWF is a function f such that $f(x)$ is easy to compute, but for most y's it is hard to sample a (close to) uniformly random x such that $f(x) = y$. A distributional OWF might not be a OWF itself, but it is known how to derive a OWF from any distributional OWF [IL89]. And further, this transformation works with quasi-linear overhead using constructions of hash functions from [IKOS08].

We claim that S_1 is now a fine-grained distributional OWF. Intuitively, if it were not, then we would, for many x's, be able to sub-quadratically sample r almost uniformly from those obeying $S_1(r) = x$. But, since the probability over r is low that S errs, the r we sampled is likely a non-erring one. That is, it is likely that we would have sub-quadratically obtained an r that gives us $S_2(r) = f\mathsf{OV}_n(x)$ and thus we likely can sub-quadratically compute $f\mathsf{OV}_n(x)$. So if $f\mathsf{OV}_n$ is actually hard to compute on average, then such a distributional inverter cannot exist.

While, as mentioned earlier, such an erring sampler would no longer give efficiently verifiable certificates for NO instances of OV, it turns out that it would still yield a coAM protocol for OV with sub-quadratic verification.

First we note that we get an AM protocol with a sub-quadratic time verifier that takes input (x, y) and proves that $f\mathsf{OV}_n(x) = y$, and is complete and sound *for most values* of x. Since the sampler is wrong with only with a very small probability over r, most values of x's have the property that most values of r satisfying $S_1(r) = x$ also satisfy $S_2(r) = f\mathsf{OV}_n(x)$. In the protocol with input (x, y), the verifier simply asks the prover to prove that for most r's such that $S_1(r) = x$, it is also the case that $S_2(r) = y$. If S_1 is indeed distributed uniformly, then this comes down to proving a lower bound on the number of r's such that $S_1(r) = x$ and $S_2(r) = y$, and this can be done in AM using the protocol from [GS86], in a single round (verifier sends a message and prover responds) and with the verifier running in sub-quadratic time. With a little more analysis and appropriate setting of parameters, this protocol can be shown to work even if S_1 is only close to uniform.

Further, from this protocol one can get a protocol that works *for all values* of x by following the approach in the proof of Lemma 1 – given an input (x, y), the verifier runs the random self-reduction for x and, for each evaluation query to $f\mathsf{OV}_n$ that comes up in its course, it asks the prover for the answer along with an AM proof of its correctness. Thus, done all at once, this gives a single-round coAM protocol for OV with a sub-quadratic time verifer (this is when $y = 0$). This in turns leads to a single-round coAM protocol for k-SAT with a verifier that runs in $\widetilde{O}(2^{n(1-\epsilon)})$ time for some $\epsilon > 0$.

The impossibility of the existence of such a protocol could be conjectured as a sort of $\mathsf{AM}[2]\mathsf{SETH}$. Williams [Wil16] gives a non-interactive MA protocol that comes close to breaking this conjecture, but standard approaches for converting MA protocols to AM protocols [Bab85] seem to incur a prohibitively large overhead in our fine-grained setting.

Question 2. *Is it possible to construct a single-round* coAM *protocol for* OV *(or for* $\mathcal{F}\mathsf{OV}$*) with a sub-quadratic time verifier (or for* k-SAT *with a* $2^{(1-\delta)n}$*-time verifier for some constant* $\delta > 0$*)?*

5.3 Other Open Problems

In fine-grained cryptography for low-depth circuits, the major problem left conspicuously open by our work (despite considerable effort) is the following.

Question 3. *Construct a public-key encryption scheme for* AC^0*.*

In Section 4.2.5 we presented a candidate construction for the above that we have not been able to prove secure, but also not been able to break. It would be very interesting to see either of these done. Proving this construction secure necessarily involves answering the following question, which seems interesting in its own right, in the affirmative.

Question 4. *Show some fixed* polylog*-wise independent distribution that fools* AC^0 *circuits of arbitrary depth.*

In all of our fine-grained worst-case to average-case reductions, we start from well-studied problems of independent interest like OV and 3SUM, and reduce to problems tailored to allow our reductions. While such a reduction would be a sufficient starting point for cryptography, a reduction to the original problems themselves would be of greater interest to those interested in algorithms for these problems and their complexities. As noted earlier, such a reduction has been successfully carried out for the k-CLIQUE problem by Goldreich and Rothblum [GR18].

Question 5. *Show a fine-grained worst-case to average-case reduction from* OV *(or* 3SUM*,* APSP*, etc.) to itself.*

Standard OWFs are sufficient for secret-key cryptography as they are equivalent to Pseudo-Random Generators and Pseudo-Random Functions [HILL99, GGM86]. One can ask to what extent similar equivalences hold in the fine-grained world and what fine-grained cryptography can be accomplished from fine-grained OWFs. More generally, one can ask the following.

Question 6. *Which generic cryptographic results translate over to fine-grained cryptography and are there any that hold only in the fine-grained world?*

Finally, we ask the less theoretical question of whether the heuristic falsifiability of conjectures that follows from our average-case reductions can actually be utilized.

Question 7. *Can we heuristically falsify any of the big worst-case fine-grained conjectures about* k-SAT*,* APSP*,* 3SUM*, etc.? Are there other ways in which we can develop techniques for practice to influence theory and give concrete and parameterizable evidence for theoretical conjectures?*

Appendix A

Appendices to Chapter 2

A.1 An Average-Case Time Hierarchy

In this section, we present an infinite collection of generalizations of $\mathcal{F}\mathsf{OV}$ that we conjecture form an average-case time hierarchy. That is, for every rational number k the collection contains a function $\mathcal{F}\mathsf{OV}^k$ such that $\mathcal{F}\mathsf{OV}^k$ is computable in $\widetilde{O}(n^k)$ time, but (we conjecture) requires $n^{k-o(1)}$ time to compute even on average. This conjecture is supported by SETH. We describe these generalizations below and indicate how this follows from SETH for integer values of $k \geq 2$ and note that this can be extended to all rational numbers using standard padding techniques.

- *k-Orthogonal Vectors:* For an integer $k \geq 2$, the k-OV problem on vectors of dimension d is to determine, given k sets $(U_1, \ldots, U_k)$ of n vectors from $\{0,1\}^{d(n)}$ each, whether there exist $u_i \in U_i$ for each i such that over $\mathbb{Z}$,

$$\sum_{\ell \in [d(n)]} u_{1\ell} \cdots u_{k\ell} = 0$$

  (As with OV, if left unspecified, d is to be taken to be $\lceil \log^2 n \rceil$.)

Similar to how it implies the hardness of OV, (the randomized version of) SETH also implies that for any integer $k \geq 2$, any randomized algorithm for k-OV requires $n^{k-o(1)}$ time – the proof is a natural generalization of that for OV shown in [Wil15].

We next take the same approach we did for OV and define for any integer $k \geq 2$ a family of polynomials $\mathcal{F}\mathsf{OV}^k = \left\{f\mathsf{OV}_n^k\right\}$, where with p being the smallest prime number larger than n^k and $d = \left\lceil \log^2(n) \right\rceil$, $f\mathsf{OV}_n^k : \mathbb{F}_p^{knd} \to \mathbb{F}_p$ is defined as:

$$f\mathsf{OV}_n^k(U_1, \ldots, U_k) = \sum_{u_1 \in U_1, \ldots, u_k \in U_k} \prod_{\ell \in [d]} (1 - u_{1\ell} \cdots u_{k\ell})$$

Similarly to the case with $f\mathsf{OV}_n$, when the input to $f\mathsf{OV}_n^k$ is a k-OV instance from $\{0,1\}^{knd}$, $f\mathsf{OV}_n^k(U_1, \ldots, U_k)$ counts the number of sets of "orthogonal" vectors in it. Note that the degree of $f\mathsf{OV}_n^k$ is at most kd. And also that by simply evaluating each summand and adding them up, the polynomial can be evaluated in $\widetilde{O}(n^k)$ time. The following theorem can again be proven in a manner identical to Theorem 1.

Theorem 30. *For any integer $k \geq 2$, if $\mathcal{F}\mathsf{OV}^k$ can be computed in $O(n^{k/2+\alpha})$ time on average for some $\alpha > 0$, then k-OV can be decided in $\widetilde{O}(n^{k/2+\alpha})$ time in the worst case.*

Corollary 8. *Suppose for every $k \geq 2$, k-OV requires $n^{k-o(1)}$ time to decide. Then for every such k, $\mathcal{F}\mathsf{OV}^k$ requires $n^{k-o(1)}$ to compute on average but can be computed in the worst case in $\widetilde{O}(n^k)$ time.*

Thus, we can attain the hierarchy from an infinite number of conjectures, one for each k, but, as noted earlier, the entire hierarchy is also implied by the single assumption SETH. We note that for k-OV (and all other problems) we could naïvely express k-OV with a polynomial via a multilinear extension, yet this polynomial would have exponentially many terms and degree n. Already the degree is too high for our purposes but not by much: we may not be too disappointed with $n^{(k-1)-o(1)}$ average-case hardness that degree n would still afford us. The main problem then is that the naïve polynomial may take exponential time to compute and so the upper bound is *very* far from the lower bound. The tightness of our hierarchy is a key feature then in capturing the hardness of our problems as well as for use in applications such as in those in Chapter 3.

Remark 16. We can also attain two semi-tight hierarchies from generalizations of the 3SUM problem. That is, if we assume the k-SUM conjecture proposed in [AL13], we get hardness for two infinite hierarchies, with one based on generalizing $\mathcal{F}$ZWT and one from generalizing $\mathcal{F}$3SUM (the proper generalizations can be based on problems found in [AL13]). These hierarchies, however, are loose, in that the k-SUM conjecture gives us $\left(n^{\lceil k/2 \rceil - o(1)}\right)$ hardness at the k^{th} level but, to our knowledge, our generalized polynomials are only $\widetilde{O}(n^{k-1})$ computable (as they have n^{k-1} many terms).

A.2 On the Heuristic Falsifiability of Conjectures

The Proof of Work protocol we construct in Chapter 3 – we take the case of the $(n^2, 1/n^{o(1)})$-PoW from OV throughout this section for simplicity – yields a win-win in the domain of algorithms and complexity. Namely, either we have a PoW *or* we have the existence of a *provably* sub-quadratic prover that can convince the verifer with sufficient probability which will in turn yield breakthroughs: a randomized sub-quadratic time algorithm for OV and thus a randomized $2^{n(1-\epsilon)}$ time algorithm for k-SAT for some $\epsilon > 0$. In particular, because the hardness of the PoW is over random instances, even a prover that can be *empirically* demonstrated to be sub-quadratic in practice will give *heuristic* evidence that the conjectured hardness of OV or k-SAT is false. In other words, if the above PoW is empirically (after working out the constants hidden by Big-Oh notation) insecure, then (randomized) SETH is heuristically falsified.

This notion of heuristic falsifiability sits directly at the intersection of average-case and fine-grained complexity that we study: Without average-case results, a worst-case conjecture could not be broken without a full proof that an algorithm works on *all* inputs, and, without fine-grained results, a fine-grained conjecture on complexity like SETH could not be feasibly be broken. Thus, it is *precisely* average-case fine-grained hardness that allows us to discuss the heuristic falsifiability of conjectures. While theory and practice can often influence each other indirectly, this marks an interesting connection, akin to the hard-sciences, in which empirical evidence can give concrete

and parameterizable theoretical evidence.

Remark 17. We note that while some may try to claim that SETH is already being falsified in practice – e.g. that we might seem to run in $2^{\sqrt{n}}$ time on practical inputs – there are two main points in which this is different from our heuristic falsifiability. One point is that, from our worst-case to average-case reductions, our notion would be heuristically breaking the *worst-case* version of SETH and achieve a complexity theoretic claim, as opposed to heuristically breaking an *average-case* notion of SETH on "nature's distribution" of "practical" inputs, which would only be a heuristic claim on how we perform in practice. Secondly, claims of breaking even such an *average-case* notion of SETH in practice cannot be given too much confidence to since the input sizes must remain very small to be feasible and so not many "data points" can be gathered to see the true shape of the exponential curve. In contrast, our heuristic falsifiability reduces to a quadratic time problem, so that many more "data points" can be gathered for runtimes on much larger input sizes, giving us much more confidence as to if we heuristically break the OV conjecture and thus SETH.

To put this observation in more concrete terms, we consider the notion of falsifiable assumptions introduced by Naor [Nao03]: Informally, an assumption A is efficiently falsifiable if there is an efficiently samplable distribution of challenges and an efficient verifier of solutions to these challenges such that, if and only if A is false (we consider the "only if" case), there is an efficient algorithm which causes the verifier to accept with high probability over the challenge distribution.

In our case, any algorithm that solves $\mathcal{F}\mathsf{OV}$ in sub-quadratic time falsifies the conjectured hardness of $\mathcal{F}\mathsf{OV}$ and thus OV and thus SETH. The sampler simply uniformly draws a set of inputs for $f\mathsf{OV}_n$, and the verifier simply evaluates $f\mathsf{OV}_n$ on the instances and compares with the sub-quadratic prover's answer.

Further, the problem underlying the PoW (to output a polynomial for a given $f\mathsf{OV}_n$ instance) is falsifiable with the added property that both the sampler *and* the verifier run in sub-quadratic time. Thus to heuristically falsify SETH, a challenger may deploy PoW challenges out into the world and, if they're often prematurely solved, we gather empirical evidence against SETH. Note that this can similarly be

done for 3SUM and APSP as their polynomials can also be used for PoWs.

We note that interesting applications of heuristic falsifiability may be inherent to the intersection of average-case complexity and fine-grained complexity: One of the only other works we are aware of that considered average-case fine-grained analysis, [GH16], immediately yields results that notions of heuristic falsifiability can apply to. That is, [GH16] shows that fine-grained problems related to DNA sequencing are actually *easy* when given a batch of correlated instances; thus, this analysis is average-case over a specific distribution of *correlated instances* for a fine-grained problem. This distribution for which easiness is achieved, however, is motivated to be "realistic" by the correlation of the instances attempting to match current evolutionary theory and how mutations occur within a phylogenetic tree. Thus, if a distribution over correlated instances matches well a theory of evolution *yet* [GH16]'s algorithm consistently under-performs on real-life data, this may suggest our current theory of evolution is *wrong*. Again, we can (efficiently) test a hypothesis in a concretely parameterizable way and gather evidence against it.

We find it interesting that combining average-case and fine-grained complexity seems to almost immediately bear interesting fruit in the context of heuristic falsifiability. We pose it as an open question to find more ways in which the "scientific method" can be introduced into the highly theoretical field of complexity theory so that conjectures can *tested* to give concrete parameters for our confidences for them.

A.3 A Tighter Reduction for $\mathcal{F}$OV

We show that sub-quadratic algorithms cannot compute $f\mathsf{OV}_n$ on even a $1/\operatorname{polylog}(n)$-fraction of inputs, assuming OV is hard on the worst case. Moreover, the techniques yield a tradeoff between adversarial complexity and provable hardness: less time implies lower success probability. Similar results can be achieved for our other polynomials, but we do not present them here.

Recall that the worst-case to average-case reduction used in Section 2.2 (as Lemma 1) works roughly as follows for a function f. Given an input x, the reduction produces

a set of inputs $y_1, \ldots, y_m$ such that $(f(x), f(y_1), \ldots, f(y_m))$ is a Reed-Solomon codeword. Then we said that if an algorithm is correct on a $(1-\delta)$ fraction of inputs, then it is correct on close to a $(1-\delta)$ fraction of the y_i's, and so only about a δ fraction of this codeword is erroneous. As long as δ is somewhat smaller than $1/2$, we can correct these errors to recover the whole codeword and hence $f(x)$. But notice that if δ is more than $1/2$, then there is no hope of correcting the codeword, and the reduction will not work.

Because of this, the approach used in Section 2.2 is limited in that it cannot show, for instance, that sub-quadratic algorithms cannot compute $f\mathsf{OV}_n$ on more than a $1/3$ fraction of inputs. One thing that can be done even if more than half the codeword is corrupted, however, is list decoding. And the Reed-Solomon code turns out to have rather efficient list decoding algorithms [Sud97, GS99]. This fact was used by Cai, Pavan, and Sivakumar [CPS99] to show rather strong average-case hardness results for the Permanent using its downward self-reducibility properties.

We use their techniques to prove that sub-quadratic algorithms cannot compute $f\mathsf{OV}_n$ on even a $1/\operatorname{polylog}(n)$ fraction of inputs. The primary issue that one has to deal with when using list decoding instead of decoding is that it will yield many candidate polynomials. The insight of [CPS99], building on previous work regarding enumerative counting, is that downward self-reducibility can be used to isolate the true polynomial via recursion. And $f\mathsf{OV}_n$ turns out to have the properties necessary to do this. And, although we do not show it here, the same methodology works for $f\mathsf{OV}_n^k$ and $f\mathsf{ZWT}_n$.

Before we begin, we will present a few Lemmas from the literature to make certain bounds explicit.

First, we present an inclusion-exclusion bound from [CPS99] on the polynomials consistent with a fraction of m input-output pairs, $(x_1, y_1), \ldots, (x_m, y_m)$. We include a laconic proof here with the given notation for convenience.

Lemma 21 ([CPS99])**.** *For any polynomial q over $\mathbb{F}_p$, define* $\operatorname{Graph}(q) := \{(i, q(i)) \mid i \in [p]\}$. *Let $c > 2$, $\delta/2 \in (0,1)$, and $m \leq p$ such that $m > \frac{c^2(d-1)}{\delta^2(c-2)}$ for some d. Finally, let $I \subseteq [p]$ such that $|I| = m$. Then, for any set $S = \{(i, y_i) \mid i \in I\}$, there are less*

than $\lceil c/\delta \rceil$ *polynomials* q *of degree at most* d *that satisfy* $|\operatorname{Graph}(q) \cap S| \geq m\delta/2$.

Corollary 9. *Let* S *be as in Lemma 21 with* $I = \{m+1, \ldots, p\}$*, for any* $m < p$*. Then for* $m > 9d/\delta^2$*, there are at most* $3/\delta$ *polynomials,* q*, of degree at most* d *such that* $|\operatorname{Graph}(q) \cap S| \geq m\delta/2$.

Proof. Reproduced from [CPS99] for convenience; see original for exposition.

Suppose, for contradiction, that there exists at least $\lceil c/\delta \rceil$ such polynomials. Consider a subset of exactly $N = \lceil c/\delta \rceil$ such polynomials, $\mathcal{F}$. Define $\mathsf{S}_f := \{(i, f(i)) \in \operatorname{Graph}(f) \cap S\}$, for each $f \in \mathcal{F}$.

$$\begin{aligned}
m \geq \left| \bigcup_{f \in \mathcal{F}} S_f \right| &\geq \sum_{f \in \mathcal{F}} |S_f| - \sum_{f, f' \in \mathcal{F} : f \neq f'} |S_f \cap S_{f'}| \\
&\geq N \frac{m\delta}{2} - \frac{N(N-1)(d-1)}{2} \\
&> \frac{N}{2} \left(m\delta - \frac{c(d-1)}{\delta} \right) \\
&\geq \frac{c}{2\delta} \left(m\delta - \frac{c(d-1)}{\delta} \right) \\
&= \frac{cm}{2} - \frac{c^2(d-1)}{2\delta^2} \\
&= m + \frac{1}{2} \left((c-2)m - \frac{c^2(d-1)}{\delta^2} \right) > m.
\end{aligned}$$

□

Now, we give a theorem based on an efficient list-decoding algorithm, related to Sudan's, from Roth and Ruckenstein. [RR00]

Lemma 22 ([RR00])**.** *List decoding for* $[n, k]$*-Reed-Solomon (RS) codes over* $\mathbb{F}_p$ *given a code word with almost* $n - \sqrt{2kn}$ *errors (for* $k > 5$*), can be performed in*

$$O\left(n^{3/2} k^{-1/2} \log^2 n + (n-k)^2 \sqrt{n/k} + (\sqrt{nk} + \log q) n \log^2(n/k) \right)$$

operations over $\mathbb{F}_q$.

Plugging in specific parameters and using efficient list decoding, we get the following corollary which will be useful below.

Corollary 10. *For parameters $n \in \mathbb{N}$ and $\delta \in (0,1)$, list decoding for $[m,k]$-RS codes over $\mathbb{F}_p$ where $m = \Theta(d \log n/\delta^2)$, $k = \Theta(d)$, $p = O(n^2)$, and $d = \Omega(\log n)$ can be performed in time*

$$O\left(\frac{d^2 \log^{5/2} n \mathrm{Arith}(n)}{\delta^5}\right),$$

where $\mathrm{Arith}(n)$ *is a time bound on arithmetic operations over prime fields size $O(n)$.*

Finally, we present a more explicitly parametrized variant of $\mathcal{F}\mathsf{OV}$, $\mathcal{G}\mathsf{OV} = \{g\mathsf{OV}_{n,d,p}\}_{n,d,p \in \mathbb{Z}^3}$, where

$$g\mathsf{OV}_{n,d,p} : \mathbb{F}_p^{2nd} \to \mathbb{F}_p$$

such that

$$g\mathsf{OV}_{n,d,p}\left(\boldsymbol{U} = \begin{bmatrix} \boldsymbol{u}_1 \\ \vdots \\ \boldsymbol{u}_n \end{bmatrix}, \boldsymbol{V} = \begin{bmatrix} \boldsymbol{v}_1 \\ \vdots \\ \boldsymbol{v}_n \end{bmatrix}\right) := \sum_{(\boldsymbol{u}_i, \boldsymbol{v}_j) \in U \times V} \prod_{\ell \in [d]} (1 - \boldsymbol{u}_{i\ell} \boldsymbol{v}_{j\ell}).$$

Theorem 31. *If there is an algorithm that runs in time $t(n,d,p)$ for $g\mathsf{OV}_{n,d,p}$ with success probability δ on the uniform distribution, then there is an algorithm that runs in time*

$$t'(n,d,p) = O(n^{1+\gamma} + td \log^2 n/\delta^2 + d^2/\delta^5 \log^{7/2} n \mathrm{Arith}(n^2))$$

for $g\mathsf{OV}_d$ with failure probility at most $\varepsilon < 4\delta \log n/d$ for any input. $\mathrm{Arith}(n)$ *is defined to be time bound on arithmetic operations over prime fields of size $O(n)$.*

Before jumping into the proof, we observe the following corollary that essentially provides a tradeoff between runtime and hardness. Moreover, it gives a tighter hardness result on algorithms allowed to run in slightly quadratic time.

Corollary 11. *Assume $t = \Omega(d/\delta^3 \log^3 n)$. If OV takes time $\Omega(n^{2-o(1)})$ time to decide, then any algorithm for $\mathcal{G}\mathsf{OV}$ that runs in time t with success probility δ on the uniform distribution must obey*

$$t/\delta^2 = \Omega(n^{2-o(1)}).$$

In particular, assuming OV *takes time* $\Omega(n^{2-o(1)})$*, any algorithm for* $\mathcal{G}\mathsf{OV}$ *running in time* $t = O(n^{2-\varepsilon})$*, cannot succeed on a* $1/n^{\gamma}$ *fraction of the instances for any* γ *such that* $0 < \gamma < \varepsilon/2$*.*

Proof. Let $(U, V) \in \{0,1\}^{2n \times d}$ be an instance of boolean-valued orthogonal vectors. Now, consider splitting these lists in half, $U = (U_0, U_1)$ and $V = (V_0, V_1)$, such that $(U_a, V_b) \in \{0,1\}^{n \times d}$ for $a, b \in \{0,1\}$. Then, define the following four sub-problems:

$$A^1 = (U_0, V_0), \qquad A^2 = (U_0, V_1), \qquad A^3 = (U_1, V_0), \qquad A^4 = (U_1, V_1).$$

Notice that given solutions to $\mathcal{G}\mathsf{OV}_d$ on A^1, A^2, A^3, A^4 we can trivially construct a solution to OV_d on (U, V).

Now, draw random $B, C \in \mathbb{F}_p^{n \times d}$ and consider the following degree 4 polynomial in x:

$$D(x) = \sum_{i=1}^{4} \delta_i(x) A^i + (B + xC) \prod_{i=1}^{4} (x - i),$$

where δ_i is the unique degree 3 polynomial over $\mathbb{F}_p$ that takes value 1 at $i \in \{1,2,3,4\}$ and 0 on all other values in $\{1,2,3,4\}$. Notice that $D(i) = A^i$ for $i = 1, 2, 3, 4$.

Let $m > 8d/\delta^2 \log n$. $D(5), D(6), \ldots, D(m+4)$. By the properties of $\mathcal{A}$ and because the $D(i)$'s are pairwise independent, $\mathcal{A}(D(i)) = g\mathsf{OV}(D(i))$ for $\delta m/2$ i's with probability $> 1 - \frac{4}{\delta m} = 1 - 1/\operatorname{polylog}(n)$, by a Chebyshev bound.

Now, because $\delta m/2 > \sqrt{16dm}$, we can run the list decoding algorithm of Roth and Ruckenstein, [RR00], to get a list of all polynomials with degree $\leq 8d$ that agree with at least $\delta m/2$ of the values. By Corollary 9, there are at most $L = 3/\delta$ such polynomials.

By a counting argument, there can be at most $4d\binom{L}{2} = O(dL^2)$ points in $\mathbb{F}_p$ on which any two of the L polynomials agree. Because $p > n^2 > 4d\binom{L}{2}$, we can find such a point, j, by brute-force in $O(L \cdot dL^2 \log^3(dL^2) \log p)$ time, via batch univariate evaluation [Fid72]. Now, to identify the correct polynomial $g\mathsf{OV}(D(\cdot))$, one only needs to determine the value $g\mathsf{OV}(D(j))$. To do so, we can recursively apply the above reduction to $D(j)$ until the number of vectors, n, is constant and $g\mathsf{OV}$ can be

evaluated in time $O(d \log p)$.

Because each recursive iteration cuts n in half, the depth of recursion is $\log(n)$. Additionally, because each iteration has error probability $< 4/(\delta m)$, taking a union bound over the $\log(n)$ recursive steps yields an error probability that is $\varepsilon < 4 \log n/(\delta m)$.

As noted above, we can find the prime p in time $O(n^{1+\gamma})$, for any constant $\gamma > 0$, by binary searching $\{n^2 + 1, \ldots, 2n^2\}$ with calls to [LO87]. Taking $m = 8d \log n/\delta^2$, Roth and Ruckenstein's algorithm takes time $O(d^2/\delta^5 \log^{5/2} n\text{Arith}(n^2))$, by Corollary 10, in each recursive call. The brute force procedure takes time $O(d/\delta^3 \log^3(d/\delta^2) \log n)$, which is dominated by list decoding time. Reconstruction takes time $O(\log n)$ in each round, and is also dominated.

$$t' = O(n^{1+\gamma} + td \log^2 n/\delta^2 + d^2/\delta^5 \log^{7/2} n\text{Arith}(n^2)),$$

with error probability $\varepsilon < 4 \log n\delta/d$.

□

A.4 Polynomials Computing Sums

In this section we write down polynomials that compute the bits of the sum of a pair of numbers that are given to them in bits. Without loss of generality, we will represent numbers in the two's complement form. In both cases where such representations are required (3SUM and ZWT), there are apriori bounds on the sizes of the numbers that come up – these are either numbers in the input or sums of pairs of these numbers. So we can assume that we always have enough bits to be able to represent these numbers. Under this assumption, addition in the two's complement representation is the same as adding unsigned numbers in the standard place-value representation (and ignoring the final carry). So we will present the polynomials $\{s_\ell\}_{\ell \in [d]}$ that correspond to unsigned addition (and are easier to describe) and these are the polynomials that will be used.

We label the bits of a d-bit number from 1 to d starting from the least significant

bit. We then translate the semantics of the ripple-carry adder into polynomials. The polynomial $s_1 : \mathbb{F}_p^{2d} \to \mathbb{F}_p$ corresponding to the first bit of the sum is:

$$s_1(x, y) = x_1(1 - y_1) + (1 - x_1)y_1$$

The carry from this operation is given by the following polynomial:

$$c_1(x, y) = x_1 y_1$$

For every other ℓ, this pair of polynomials can be computed from earlier polynomials and the inputs as follows (hiding the arguments x and y for convenience):

$$s_\ell = (1 - x_\ell)(1 - y_\ell)c_{\ell-1} + (1 - x_\ell)y_\ell(1 - c_{\ell-1}) + x_\ell(1 - y_\ell)(1 - c_{\ell-1}) + x_\ell y_\ell c_{\ell-1}$$
$$c_\ell = x_\ell y_\ell(1 - c_{\ell-1}) + x_\ell(1 - y_\ell)c_{\ell-1} + (1 - x_\ell)y_\ell c_{\ell-1} + x_\ell y_\ell c_{\ell-1}$$

It can now be seen that $deg(s_\ell) = deg(c_\ell) = deg(c_{\ell-1}) + 2$. Along with the fact that $deg(c_1) = 2$, this implies that $deg(s_\ell) = 2\ell \leq 2d$.

These polynomials can also be computed very easily by evaluating them in order. Given $c_{\ell-1}$, both s_ℓ and c_ℓ take only a constant number of operations to compute. Hence all the s_ℓ's can be computed is $O(d \log^2 p)$ time.

A.5 Isolating Orthogonal Vectors

In this section, we describe a randomized reduction from OV to uOV (*unique*-OV), which is the Orthogonal Vectors problem with the promise that there is at most one pair of orthogonal vectors in the given instance.

While interesting by itself, such a reduction is also relevant to the rest of our work for the following reason. Recall that the polynomial $f\mathsf{OV}_n$ is defined over the field $\mathbb{F}_p$ where $p > n^2$. The reason p had to be more than n^2 was so that $f\mathsf{OV}_n$ would count the number of orthogonal vectors when given a boolean input, and this number could be as large as n^2. If we wanted a polynomial that did this for uOV, this restriction

on the characteristic of the field wouldn't exist. p would have still to be $\Omega(d)$ for the random self-reduction to work, but this is much smaller than n^2 in our setting, and this could possibly allow applications of our results that would not be viable otherwise.

Recall that an important reason for believing that there is no sub-quadratic algorithm for OV is that such an algorithm would break SETH due to a fine-grained reduction from k-SAT [Wil05]. If all one wanted was a similar reason to believe that uOV was hard, one could attempt to reduce k-SAT to uOV. A natural approach to doing so would be to first reduce k-SAT to unique-k-SAT and then apply the reduction from [Wil05], as it translates the number of satisfying assignments to the number of orthogonal vectors.

However, the isolation lemma for k-SAT due to Valiant and Vazirani [VV85] turns out to not work for this purpose because it blows up the number of variables in the k-SAT instance it operates on, and the resulting reduction would not be fine-grained enough to provide the requisite lower bounds for uOV based on SETH. One way to circumvent this is that Calabro et al. [CIKP03] provide an alternative that does preserve the number of variables and shows that SETH implies an analogous conjecture for unique-k-SAT, and this can be used in the first step of the reduction so that the reduction chain would go from k-SAT to unique-k-SAT to uOV.

We instead start with OV itself and apply techniques from [VV85] directly to it, so a reduction chain of k-SAT to OV to uOV can be achieved. Throughout this section, we will use OV_d (uOV_d) to denote the OV (respectively uOV) problem over vectors of dimension d. We start by describing a search-to-decision reduction for OV/uOV.

Lemma 23. *If, for some $c, c' \geq 1$, there is an $(n^c d^{c'})$-time algorithm for OV_d, then there is an $O(n^c d^{c'})$-time algorithm that finds a pair of orthogonal vectors in any OV_d instance (if it exists) except with negligible probability. Further, the same is true for uOV_d.*

Proof. Let A be an algorithm for deciding OV_d that has negligible error and runs in time $n^c d^{c'}$. Given a YES instance (U, V) of OV_d, where U and V have n vectors each,

our search algorithm starts by dividing U and V into halves (U_0, U_1) and (V_0, V_1), where each half has roughly $\lfloor n/2 \rfloor$ vectors. Since there was a pair of orthogonal vectors in (U, V), at least one of (U_0, V_0), (U_0, V_1), (U_1, V_0), and (U_1, V_1) must contain a pair of orthogonal vectors. Run A on all four of these to identify one that does, and recurse on that one until the instance size reduces to a constant, at which point try all pairs of vectors and find an orthogonal pair. If at some point in this process A says that none of the four sub-instances contains a pair of orthogonal vectors, or if at the end there are no orthogonal pairs, give up and fail.

Since the size of the instances reduces by a constant factor each time, the number of calls made to A is $O(\log n)$. So since A makes mistakes with negligible probability, by union bound, the whole search algorithm fails only with a negligible probability. Copying over inputs to run A on takes only linear time in the input size. Accounting for this, the running time of the algorithm is:

$$\begin{aligned} T(n) &\leq 8\left(\frac{n}{2}\right)^c d^{c'} + 8\left(\frac{n}{4}\right)^c d^{c'} + \cdots + 8 \cdot O(d^{c'}) \\ &\leq 8n^c d^{c'} \left(\sum_{k=0}^{\infty} \frac{1}{2^{ck}}\right) = O(n^c d^{c'}) \end{aligned}$$

It can be seen that the same proof goes through for uOV_d as well. □

Theorem 32. *If, for some $c, c' \geq 1$, there is an $(n^c d^{c'})$-time algorithm for $\mathsf{uOV}_{d'}$, then there is an $\widetilde{O}(n^c d^{c'})$-time algorithm for OV_d, where $d' = d + 4\log n + 2$.*

The reduction is along the lines of that from SAT to unique-SAT from [VV85], and makes use of the following lemma, which is a special case of the one used there.

Lemma 24. *Let $S \subseteq \{0,1\}^d \times \{0,1\}^d$ be a set such that $2^{k-1} \leq |S| < 2^k$ for some k. With constant probability over randomly chosen $\boldsymbol{M}_0, \boldsymbol{M}_1 \in \{0,1\}^{(k+1)\times n}$ and $\boldsymbol{b} \in \{0,1\}^{(k+1)}$, there is a unique $(\boldsymbol{x}, \boldsymbol{y}) \in S$ such that $\boldsymbol{M}_0\boldsymbol{x} + \boldsymbol{b} = \boldsymbol{M}_1\boldsymbol{y}$ (over $\mathbb{F}_2$).*

The above lemma follows from the observation that over all $\boldsymbol{M}_0$, $\boldsymbol{M}_1$ and $\boldsymbol{b}$, the set

$$\{h_{\boldsymbol{M}_0, \boldsymbol{M}_1, \boldsymbol{b}}(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{M}_0\boldsymbol{x} + \boldsymbol{b} - \boldsymbol{M}_1\boldsymbol{y}\}$$

is a universal family of hash functions. We refer the reader to [VV85] for the proof.

Proof of Theorem 32. Let A be an $O(n^c d^{c'})$-time search algorithm for $\mathsf{uOV}_{d'}$ – such an algorithm exists by our hypothesis and Lemma 23. We would like to use it to decide an instance (U, V) of OV_d. What are the instances of $\mathsf{uOV}_{d'}$ that we could run A on to help us in our task?

Suppose we knew that in (U, V) there were exactly m pairs of orthogonal vectors. Let k be such that $2^{k-1} \leq m < 2^k$. Lemma 24 says that if we choose $\boldsymbol{M}_0, \boldsymbol{M}_1 \in \{0,1\}^{(k+1)\times d}$ and $\boldsymbol{b} \in \{0,1\}^{(k+1)}$ at random, then with some constant probability, there is exactly one pair of vectors $\boldsymbol{u} \in U$, $\boldsymbol{v} \in V$ such that $\langle \boldsymbol{u}, \boldsymbol{v} \rangle = 0$ and $\boldsymbol{M}_0\boldsymbol{u}+\boldsymbol{b} = \boldsymbol{M}_1\boldsymbol{v}$. If we could somehow encode the latter condition as part of the orthogonal vector problem, we could hope to get a uOV instance from (U, V).

Consider the encoding schemes E_0 and E_1 described next. For any vector $\boldsymbol{x}$, $E_0(\boldsymbol{x})$ is a vector twice as long as $\boldsymbol{x}$, where each 0 in $\boldsymbol{x}$ is replaced by "0 1" and each 1 is replaced by "1 0". $E_1(\boldsymbol{x})$ is similar, but here a 0 is replaced by "1 0" and a 1 is replaced by "0 1". The property of these encodings that make them useful for us is that $\langle E_0(\boldsymbol{x}), E_1(\boldsymbol{y}) \rangle = 0$ if and only if $\boldsymbol{x} = \boldsymbol{y}$.

Putting ideas from the above two paragraphs together, consider the process where we pick $\boldsymbol{M}_0, \boldsymbol{M}_1, \boldsymbol{b}$ at random, and to each $\boldsymbol{u} \in U$, we append the vector $E_0(\boldsymbol{M}_0\boldsymbol{u}+\boldsymbol{b})$, and to each $\boldsymbol{v} \in V$, we append $E_1(\boldsymbol{M}_1\boldsymbol{v})$. Let the entire resulting instance be (U', V').

For any $\boldsymbol{u}' \in U$ and $\boldsymbol{v}' \in V$, $\langle \boldsymbol{u}', \boldsymbol{v}' \rangle = \langle \boldsymbol{u}, \boldsymbol{v} \rangle + \langle E_0(\boldsymbol{M}_0\boldsymbol{u} + \boldsymbol{b}), E_1(\boldsymbol{M}_1\boldsymbol{v}) \rangle = 0$ if and only if $\langle \boldsymbol{u}, \boldsymbol{v} \rangle = 0$ and $\boldsymbol{M}_0\boldsymbol{u} + \boldsymbol{b} = \boldsymbol{M}_1\boldsymbol{v}$. So it follows that with some constant probability, (U', V') has a unique pair of orthogonal vectors.

Generalising slightly, if we knew that an instance (U, V) had either between 2^{k-1} and 2^k pairs of orthogonal vectors or none, then to decide which is the case, all we need to do is to do the above conversion to (U', V'), pad the vectors with 0's to get them to dimension d', and run A on it. If there were no orthogonal vectors, A can never return a valid answer, and in the other case, with a constant probability there will be a unique pair of orthogonal vectors that A will find. This can be repeated, say, $\log^2 n$ times to get a negligible probability of failure.

But we do not actually know much about the number of pairs of orthogonal vectors

in an instance that is given to us. This is easy to deal with, though – simply run the above algorithm for all possible values of k, from 0 to $2 \log n$. If there are indeed some pairs of orthogonal vectors, then one of these values of k was the right one to use and the corresponding iteration would give us a pair of orthogonal vectors, except with negligible probability. If there are no orthogonal vectors, then we will never find such a pair.

Each (U', V') takes at most $O(nd \log n)$ time to prepare, and an execution of A takes $O(n^c d^{c'})$ time. This is done $\log^2 n$ times for each value of k, which is from $[2 \log n]$. So the total time taken by the above algorithm is $O(\log^3 n(nd \log n + n^c d^{c'})) = \widetilde{O}(n^c d^{c'})$. □

Appendix B

Appendices to Chapter 3

B.1 A Stronger Direct Sum Theorem for $\mathcal{F}\mathsf{OV}$

In this section, we prove a stronger direct sum theorem (and, thus, non-batchable evaluation) for $\mathcal{F}\mathsf{OV}^k$. That is, we prove Theorem 7.

In particular, it is sufficient to define a notion of batchability for parametrized families of functions with a monotonicity constraint. In our case, monotonicity will essentially say "adding more vectors of the same dimension and field size does not make the problem easier." This is a natural property of most algorithms. Namely, it is the case if for any fixed d, p, $\mathcal{F}\mathsf{OV}^k_{n,d,p}$ is $(n, t, \delta) - batchable$.

Instead, we generalize batchability in a parametrized fashion for $\mathcal{F}\mathsf{OV}^k_{n,d,p}$.

Definition 25. A parametrized class, $\mathcal{F}_\rho$, is *not* (ℓ, t, δ)*-batchable on average over* $\mathcal{D}_\rho$*, a parametrized family of distributions* if, for any fixed parameter ρ and algorithm Batch_ρ that runs in time $\ell(\rho)t(\rho)$ when it is given as input $\ell(\rho)$ independent samples from D_ρ, the following is true for all large enough n:

$$\Pr_{x_i \leftarrow D_\rho}\left[\mathsf{Batch}(x_1, \ldots, x_{\ell(\rho)}) = (f_\rho(x_1), \ldots, f_\rho(x_{\ell(\rho)}))\right] < \delta(\rho).$$

Remark 18. We use a more generic parameterization of $\mathcal{F}_\rho$ by ρ rather than just n since we need the batch evaluation procedure to have the property that it should still run quickly as n shrinks, as we use downward self-reducibility of $\mathcal{F}\mathsf{OV}^k_{n,d,p}$, even when

p and d remain the same.

We now show how a generalization of the list decoding reduction of [BRSV17] yields strong batch evaluation bounds. Before we begin, we will present a few Lemmas from the literature to make certain bounds explicit.

First, we present an inclusion-exclusion bound from [CPS99] on the polynomials consistent with a fraction of m input-output pairs, $(x_1, y_1), \ldots, (x_m, y_m)$. We include a laconic proof here with the given notation for convenience.

Lemma 25 ([CPS99])**.** *Let q be a polynomial over $\mathbb{F}_p$, and define* $\mathrm{Graph}(q) := \{(i, q(i)) \mid i \in [p]\}$*. Let $c > 2$, $\delta/2 \in (0,1)$, and $m \leq p$ such that $m > \frac{c^2(d-1)}{\delta^2(c-2)}$ for some d. Finally, let $I \subseteq [p]$ such that $|I| = m$. Then, for any set $S = \{(i, y_i) \mid i \in I\}$, there are less than $\lceil c/\delta \rceil$ polynomials q of degree at most d that satisfy $|\mathrm{Graph}(q) \cap S| \geq m\delta/2$.*

Corollary 12. *Let S be as in Lemma 25 with $I = \{m+1, \ldots, p\}$, for any $m < p$. Then for $m > 9d/\delta^2$, there are at most $3/\delta$ polynomials, q, of degree at most d such that $|\mathrm{Graph}(q) \cap S| \geq m\delta/2$.*

Proof. Reproduced from [CPS99] for convenience; see original for exposition.

Suppose there exist at least $\lceil c/\delta \rceil$ such polynomials. Consider a subset of exactly $N = \lceil c/\delta \rceil$ such polynomials, $\mathcal{F}$. Define $\mathsf{S}_f := \{(i, f(i)) \in \mathrm{Graph}(f) \cap S\}$, for each $f \in \mathcal{F}$.

$$
\begin{aligned}
m \geq \left| \bigcup_{f \in \mathcal{F}} S_f \right| &\geq \sum_{f \in \mathcal{F}} |S_f| - \sum_{f, f' \in \mathcal{F} : f \neq f'} |S_f \cap S_{f'}| \\
&\geq N\frac{m\delta}{2} - \frac{N(N-1)(d-1)}{2} > \frac{N}{2}\left(m\delta - \frac{c(d-1)}{\delta}\right) \\
&\geq \frac{c}{2\delta}\left(m\delta - \frac{c(d-1)}{\delta}\right) = \frac{cm}{2} - \frac{c^2(d-1)}{2\delta^2} \\
&= m + \frac{1}{2}\left((c-2)m - \frac{c^2(d-1)}{\delta^2}\right) > m.
\end{aligned}
$$

□

Now, we give a theorem based on an efficient list-decoding algorithm, related to Sudan's, from Roth and Ruckenstein. [RR00]

Lemma 26 ([RR00])**.** *List decoding for $[n, k]$ Reed-Solomon (RS) codes over $\mathbb{F}_p$ given a code word with almost $n - \sqrt{2kn}$ errors (for $k > 5$), can be performed in*

$$O\left(n^{3/2}k^{-1/2}\log^2 n + (n-k)^2\sqrt{n/k} + (\sqrt{nk} + \log q)n\log^2(n/k)\right)$$

operations over $\mathbb{F}_q$.

Plugging in specific parameters and using efficient list decoding, we get the following corollary which will be useful below.

Corollary 13. *For parameters $n \in \mathbb{N}$ and $\delta \in (0, 1)$, list decoding for $[m, k]$ RS over $\mathbb{F}_p$ where $m = \Theta(d \log n/\delta^2)$, $k = \Theta(d)$, $p = O(n^2)$, and $d = \Omega(\log n)$ can be performed in time*

$$O\left(\frac{d^2 \log^{5/2} n \mathrm{Arith}(n)}{\delta^5}\right),$$

where $\mathrm{Arith}(n)$ is a time bound on arithmetic operations over prime fields size $O(n)$.

Theorem 33. *For some $k \geq 2$, suppose k-OV takes $n^{k-o(1)}$ time to decide for all but finitely many input lengths for any $d = \omega(\log n)$. Then, for any positive constants $c, \epsilon > 0$ and $0 < \delta < \varepsilon/2$, $\mathcal{F}\mathsf{OV}^k$ is* not

$$(n^c\mathsf{poly}(d, \log(p)), n^{k-\epsilon}\mathsf{poly}(d, \log(p)), n^{-\delta}\mathsf{poly}(d, \log(p)))$$

-batchable on average over the uniform distribution over its inputs.

Proof. Let $k = 2c' + c$ and $p > n^k$. Suppose for the sake of contradiction that $\mathcal{F}\mathsf{OV}_{n,d,p}$ is $(n^c\mathsf{poly}(d, \log(p)), n^{2c'+c-\epsilon}\mathsf{poly}(d, \log(p)), n^{-c'}\mathsf{poly}(d, \log(p)))$-batchable on average over the uniform distribution.

Let $m = n^{k/(k+c)}$, as before. By Proposition 2, k-OV with vectors of dimension $d = (\frac{k}{k+c}))^2 \log^2 n$ is (m, m^c)-downward reducible to k-OV with vectors of dimension $\log^2(n)$, in time $\tilde{O}(m^{c+1})$.

For each $j \in [m^c]$ $X_j = (U^{j1}, \ldots, U^{jk}) \in \{0,1\}^{kmd}$ is the instance of boolean-valued orthogonal vectors from the above reduction. Now, consider splitting these lists in half, $U^{ji} = (U_0^{ji}, U_1^{ji})$ $(i \in [k])$, such that $(U_{a_1}^{j1}, \ldots, U_{a_k}^{jk}) \in \{0,1\}^{kmd/2}$ for $\boldsymbol{a} \in \{0,1\}^k$. Interpret $\boldsymbol{a}$ as binary number in $\{0, \ldots, 2^k - 1\}$. Then, define the following 2^k sub-problems:

$$A^{\boldsymbol{a}} = ((U_{a_1}^{j1}, \ldots, U_{a_k}^{jk})), \forall \boldsymbol{a} \in \{0, \ldots, 2^k - 1\}$$

Notice that given solutions to $f\mathsf{OV}_d^k$ on $\{A^{\boldsymbol{a}}\}_{\boldsymbol{a} \in \{0,1\}^k}$ we can trivially construct a solution to OV_d^k on X_j.

Now, draw random $B_j, C_j \in \mathbb{F}_p^{kmd/2}$ and consider the following degree 2^k polynomial in x:

$$D_j(x) = \sum_{i=1}^{2^k} \delta_i(x) A^{i-1} + (B_j + xC_j) \prod_{i=1}^{2^k} (x - i),$$

where δ_i is the unique degree $2^k - 1$ polynomial over $\mathbb{F}_p$ that takes value 1 at $i \in [2^k]$ and 0 on all other values in $[2^k]$. Notice that $D_j(i) = A^{i-1}$ for $i \in [2^k]$.

Let $r > 2^{k+1} d/\delta^2 \log m$. $D_j(2^k + 1), D_j(6), \ldots, D_j(r + 2^k)$. By the properties of Batch and because the $D_j(\cdot)$'s are independent, $D_1(i), \ldots, D_{m^c}(i)$ are independent for any fixed i. Thus,

$$\mathsf{Batch}(D_1(i), \ldots, D_{m^c}(i)) = f\mathsf{OV}^k(D_1(i)), \ldots, f\mathsf{OV}^k(D_{m^c}(i))$$

for $\delta r/2$ i's with probability at least $1 - \frac{4}{\delta r} = 1 - 1/\operatorname{polylog}(m)$, by Chebyshev.

Now, because $\delta r/2 > \sqrt{16dr}$, we can run the list decoding algorithm of Roth and Ruckenstein, [RR00], to get a list of all polynomials with degree $\leq 2^{k+1} d$ that agree with at least $\delta r/2$ of the values. By Corollary 12, there are at most $L = 3/\delta$ such polynomials.

By a counting argument, there can be at most $2^k d\binom{L}{2} = O(dL^2)$ points in $\mathbb{F}_p$ on which any two of the L polynomials agree. Because $p > n^k > 2^k d\binom{L}{2}$, we can find such a point, ℓ, by brute-force in $O(L \cdot dL^2 \log^3(dL^2) \log p)$ time, via batch *univariate* evaluation [Fid72]. Now, to identify the correct polynomials $f\mathsf{OV}^k(D_j(\cdot))$, one only

needs to determine the value $f\mathsf{OV}^k(D_j(\ell))$. To do so, we can recursively apply the above reduction to all the $D_j(\ell)$s until the number of vectors, m, is constant and $f\mathsf{OV}^k$ can be evaluated in time $O(d \log p)$.

Because each recursive iteration cuts m in half, the depth of recursion is $\log(m)$. Additionally, because each iteration has error probability $< 4/(\delta r)$, taking a union bound over the $\log(m)$ recursive steps yields an error probability that is $\varepsilon < 4 \log m/(\delta r)$.

We can find the prime p via $O(\log m)$ random guesses in $\{m^k + 1, \ldots, 2m^k\}$ with overwhelming probability. By Corollary 13, taking $r = 8d \log m/\delta^2$, Roth and Ruckenstein's algorithm takes time $O(d^2/\delta^5 \log^{5/2} m\text{Arith}(m^k))$ in each recursive call. The brute force procedure takes time $O(d/\delta^3 \log^3(d/\delta^2) \log m)$, which is dominated by list decoding time. Reconstruction takes time $O(\log m)$ in each round, and is also dominated. Thus the total run time is

$$T = O(m^c(m^{k-\varepsilon} d \log^2 m/\delta^2 + d^2/\delta^5 \log^{7/2} m\text{Arith}(m^k))),$$

with error probability $\varepsilon < 4 \log m\delta/d$. □

Bibliography

[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.

[Abb17] Amir Abboud. Personal communication, 2017.

[ABW10] Benny Applebaum, Boaz Barak, and Avi Wigderson. Public-key cryptography from different assumptions. In Leonard J. Schulman, editor, *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 171–180. ACM, 2010.

[ABW15a] Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. If the current clique algorithms are optimal, so is valiant's parser. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pages 98–117. IEEE, 2015.

[ABW15b] Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. Quadratic-time hardness of LCS and other sequence similarity measures. *CoRR*, abs/1501.07053, 2015.

[AGGM06] Adi Akavia, Oded Goldreich, Shafi Goldwasser, and Dana Moshkovitz. On basing one-way functions on np-hardness. In Kleinberg [Kle06], pages 701–710.

[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.

[AHWW15] Amir Abboud, Thomas Dueholm Hansen, Virginia Vassilevska Williams, and Ryan Williams. Simulating branching programs with edit distance and friends or: A polylog shaved is a lower bound made. *arXiv preprint arXiv:1511.06022*, 2015.

[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. âĹŚ11-formulae on finite structures. *Annals of Pure and Applied Logic*, 24(1):1 – 48, 1983.

[Ajt96] Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In Gary L. Miller, editor, *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, pages 99–108. ACM, 1996.

[AL13] Amir Abboud and Kevin Lewi. Exact weight subgraphs and the k-sum conjecture. In *International Colloquium on Automata, Languages, and Programming*, pages 1–12. Springer Berlin Heidelberg, 2013.

[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 CryptologyâĂŤCRYPTOâĂŹ99*, 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.

[AWW14] Amir Abboud, Virginia Vassilevska Williams, and Oren Weimann. Consequences of faster alignment of sequences. In *International Colloquium on Automata, Languages, and Programming*, pages 39–51. Springer, 2014.

[AWY15] Amir Abboud, Virginia Vassilevska Williams, and Huacheng Yu. Matching triangles and basing hardness on an extremely popular conjecture. Manuscript: https://dl.dropboxusercontent.com/u/14999836/publications/MatchTria.pdf, 2015.

[Bab85] László Babai. Trading group theory for randomness. In Sedgewick [Sed85], pages 421–429.

[Bar86] David A. Mix Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in nc^1. In Hartmanis [Har86], pages 1–5.

[Bar17] Boaz Barak. The complexity of public-key cryptography. In Yehuda Lindell, editor, *Tutorials on the Foundations of Cryptography.*, pages 45–77. Springer International Publishing, 2017.

[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.

[BDSKM17] Marshall Ball, Dana Dachman-Soled, Mukul Kulkarni, and Tal Malkin. Non-malleable codes from average-case hardness: Ac0, decision trees, and streaming space-bounded tampering. Cryptology ePrint Archive, Report 2017/1061, 2017. https://eprint.iacr.org/2017/1061.

[BDT16] Arturs Backurs, Nishanth Dikkala, and Christos Tzamos. Tight hardness results for maximum weight rectangles. *arXiv preprint arXiv:1602.05837*, 2016.

[BFKL93] Avrim Blum, Merrick L. Furst, Michael J. Kearns, and Richard J. Lipton. Cryptographic primitives based on hard learning problems. In Douglas R. Stinson, editor, *Advances in Cryptology - CRYPTO '93, 13th Annual International Cryptology Conference, Santa Barbara, California, USA, August 22-26, 1993, Proceedings*, volume 773 of *Lecture Notes in Computer Science*, pages 278–291. Springer, 1993.

[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 Joe Kilian, editor, *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*, volume 2139 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 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.

[BGJ+16] Nir Bitansky, Shafi Goldwasser, Abhishek Jain, Omer Paneth, Vinod Vaikuntanathan, and Brent Waters. Time-lock puzzles from randomized encodings. In Sudan [Sud16], pages 345–356.

[BGL16] Karl Bringmann, Allan Grønlund, and Kasper Green Larsen. A dichotomy for regular expression membership testing. *arXiv preprint arXiv:1611.00918*, 2016.

[BHvM09] Armin Biere, Marijn Heule, and Hans van Maaren. *Handbook of satisfiability*, volume 185. ios press, 2009.

[BI14] Arturs Backurs and Piotr Indyk. Edit distance cannot be computed in strongly subquadratic time (unless SETH is false). *CoRR*, abs/1412.0348, 2014.

[BK15] Karl Bringmann and Marvin Künnemann. Quadratic conditional lower bounds for string problems and dynamic time warping. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pages 79–97. IEEE, 2015.

[BK16a] Alex Biryukov and Dmitry Khovratovich. Egalitarian computing. In Thorsten Holz and Stefan Savage, editors, *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016.*, pages 315–326. USENIX Association, 2016.

[BK16b] Andreas Björklund and Petteri Kaski. How proofs are prepared at camelot. In *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing*, pages 391–400. ACM, 2016.

[BLR93] Manuel Blum, Michael Luby, and Ronitt Rubinfeld. Self-testing/correcting with applications to numerical problems. *J. Comput. Syst. Sci.*, 47(3):549–595, 1993.

[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.

[BRSV17] Marshall Ball, Alon Rosen, Manuel Sabin, and Prashant Nalini Vasudevan. Average-case fine-grained hardness. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 483–496. ACM, 2017.

[BRSV18] Marshall Ball, Alon Rosen, Manuel Sabin, and Prashant Nalini Vasudevan. Proofs of work from worst-case assumptions. *IACR Cryptology ePrint Archive*, 2018:559, 2018. To appear in CRYPTO 2018.

[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.

[BT06] Andrej Bogdanov and Luca Trevisan. Average-case complexity. *Foundations and Trends in Theoretical Computer Science*, 2(1), 2006.

[BT16] Arturs Backurs and Christos Tzamos. Improving viterbi is hard: Better runtimes imply faster clique algorithms. *arXiv preprint arXiv:1607.04229*, 2016.

[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.

[CCRR18] Ran Canetti, Yilei Chen, Leonid Reyzin, and Ron D Rothblum. Fiat-shamir and correlation intractability from strong kdm-secure encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 91–122. Springer, 2018.

[CFK+15] M. Cygan, F.V. Fomin, Ł. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized Algorithms*. Springer International Publishing, 2015.

[CG18] Matteo Campanelli and Rosario Gennaro. Fine-grained secure computation. *IACR Cryptology ePrint Archive*, 2018:297, 2018.

[CGI+16] Marco L. Carmosino, Jiawei Gao, Russell Impagliazzo, Ivan Mihajlin, Ramamohan Paturi, and Stefan Schneider. Nondeterministic extensions of the strong exponential time hypothesis and consequences for non-reducibility. In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science, Cambridge, MA, USA, January 14-16, 2016*, pages 261–270, 2016.

[CIKP03] Chris Calabro, Russell Impagliazzo, Valentine Kabanets, and Ramamohan Paturi. The complexity of unique k-sat: An isolation lemma for k-cnfs. In *18th Annual IEEE Conference on Computational Complexity (Complexity 2003), 7-10 July 2003, Aarhus, Denmark*, page 135, 2003.

[CIS18] Marco L. Carmosino, Russell Impagliazzo, and Manuel Sabin. Fine-grained derandomization: From problem-centric to resource-centric complexity. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, volume 107 of *LIPIcs*, pages 27:1–27:16. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018.

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

[CPS99] Jin-yi Cai, Aduri Pavan, and D. Sivakumar. On the hardness of permanent. In Christoph Meinel and Sophie Tison, editors, *STACS 99, 16th Annual Symposium on Theoretical Aspects of Computer Science, Trier, Germany, March 4-6, 1999, Proceedings*, volume 1563 of *Lecture Notes in Computer Science*, pages 90–99. Springer, 1999.

[CW16] Timothy M Chan and Ryan Williams. Deterministic apsp, orthogonal vectors, and more: Quickly derandomizing razborov-smolensky. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1246–1255. Society for Industrial and Applied Mathematics, 2016.

[DFKP15] Stefan Dziembowski, Sebastian Faust, Vladimir Kolmogorov, and Krzysztof Pietrzak. Proofs of space. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II*, volume 9216 of *Lecture Notes in Computer Science*, pages 585–605. Springer, 2015.

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

[DHW10] Holger Dell, Thore Husfeldt, and Martin Wahlén. Exponential time complexity of the permanent and the tutte polynomial. In *International Colloquium on Automata, Languages, and Programming*, pages 426–437. Springer, 2010.

[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.

[DN92] Cynthia Dwork and Moni Naor. Pricing via processing or combatting junk mail. In Ernest F. Brickell, editor, *Advances in Cryptology - CRYPTO '92, 12th Annual International Cryptology Conference, Santa Barbara, California, USA, August 16-20, 1992, Proceedings*, volume 740 of *Lecture Notes in Computer Science*, pages 139–147. Springer, 1992.

[DPW09] Stefan Dziembowski, Krzysztof Pietrzak, and Daniel Wichs. Non-malleable codes. *IACR Cryptology ePrint Archive*, 2009:608, 2009.

[DVV16] Akshay Degwekar, Vinod Vaikuntanathan, and Prashant Nalini Vasudevan. Fine-grained cryptography. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August*

14-18, 2016, Proceedings, Part III, volume 9816 of *Lecture Notes in Computer Science*, pages 533–562. Springer, 2016.

[FF90] J Feigenbaum and L Fortnow. On the random-self-reducibility of complete sets. *University of Chicago Technical Report*, pages 90–22, 1990.

[FF93] Joan Feigenbaum and Lance Fortnow. Random-self-reducibility of complete sets. *SIAM J. Comput.*, 22(5):994–1005, 1993.

[FGHK15] Magnus Gausdal Find, Alexander Golovnev, Edward A. Hirsch, and Alexander S. Kulikov. A better-than-3n lower bound for the circuit complexity of an explicit function. *Electronic Colloquium on Computational Complexity (ECCC)*, 22:166, 2015.

[Fid72] Charles M. Fiduccia. Polynomial evaluation via the division algorithm: The fast fourier transform revisited. In Fischer et al. [FZUR72], pages 88–93.

[FS86] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, 1986.

[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.

[FZUR72] Patrick C. Fischer, H. Paul Zeiger, Jeffrey D. Ullman, and Arnold L. Rosenberg, editors. *Proceedings of the 4th Annual ACM Symposium on Theory of Computing, May 1-3, 1972, Denver, Colorado, USA*. ACM, 1972.

[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.

[GGH94] Mikael Goldmann, Per Grape, and Johan Håstad. On average time hierarchies. *Inf. Process. Lett.*, 49(1):15–20, 1994.

[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 *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 40–49. IEEE Computer Society, 2013.

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

[GH16] Shafi Goldwasser and Dhiraj Holden. On the fine grained complexity of polynomial time problems given correlated instances. In *Innovations in Theoretical Computer Science (ITCS)*, 2016.

[GI16] Jiawei Gao and Russell Impagliazzo. Orthogonal vectors is hard for first-order properties on sparse graphs. *Electronic Colloquium on Computational Complexity (ECCC)*, 23:53, 2016.

[GK15] Alexander Golovnev and Alexander S. Kulikov. Weighted gate elimination: Boolean dispersers for quadratic varieties imply improved circuit lower bounds. *Electronic Colloquium on Computational Complexity (ECCC)*, 22:170, 2015.

[GM82] Shafi Goldwasser and Silvio Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In Harry R. Lewis, Barbara B. Simons, Walter A. Burkhard, and Lawrence H. Landweber, editors, *Proceedings of the 14th Annual ACM Symposium on Theory of Computing, May 5-7, 1982, San Francisco, California, USA*, pages 365–377. ACM, 1982.

[GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In Sedgewick [Sed85], pages 291–304.

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17(2):281–308, 1988.

[GO95] Anka Gajentaan and Mark H Overmars. On a class of $O(n^2)$ problems in computational geometry. *Computational geometry*, 5(3):165–185, 1995.

[Gol01] Oded Goldreich. *The Foundations of Cryptography - Volume 1, Basic Techniques*. Cambridge University Press, 2001.

[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.

[GR15] Shafi Goldwasser and Guy N. Rothblum. How to compute in the presence of leakage. *SIAM J. Comput.*, 44(5):1480–1549, 2015.

[GR17] Oded Goldreich and Guy Rothblum. Simple doubly-efficient interactive proof systems for locally-characterizable sets. Electronic Colloquium on Computational Complexity Report TR17-018, February 2017.

[GR18] Oded Goldreich and Guy N. Rothblum. Counting t-cliques: Worst-case to average-case reductions and direct interactive proof systems. *Electronic Colloquium on Computational Complexity (ECCC)*, 25:46, 2018.

[GS86] Shafi Goldwasser and Michael Sipser. Private coins versus public coins in interactive proof systems. In *Proceedings of the eighteenth annual ACM symposium on Theory of computing*, pages 59–68. ACM, 1986.

[GS92] Peter Gemmell and Madhu Sudan. Highly resilient correctors for polynomials. *Information processing letters*, 43(4):169–174, 1992.

[GS99] Venkatesan Guruswami and Madhu Sudan. Improved decoding of reed-solomon and algebraic-geometry codes. *IEEE Trans. Information Theory*, 45(6):1757–1767, 1999.

[Har86] Juris Hartmanis, editor. *Proceedings of the 18th Annual ACM Symposium on Theory of Computing, May 28-30, 1986, Berkeley, California, USA*. ACM, 1986.

[Hås86] Johan Håstad. Almost optimal lower bounds for small depth circuits. In Hartmanis [Har86], pages 6–20.

[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 J. Comput.*, 28(4):1364–1396, 1999.

[Hor72] Ellis Horowitz. A fast method for interpolation using preconditioning. *Information Processing Letters*, 1(4):157–163, 1972.

[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.

[IKOS08] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Cryptography with constant computational overhead. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 433–442. ACM, 2008.

[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.

[JJ99] Markus Jakobsson and Ari Juels. Proofs of work and bread pudding protocols. In Bart Preneel, editor, *Secure Information Networks: Communications and Multimedia Security, IFIP TC6/TC11 Joint Working Conference on Communications and Multimedia Security (CMS '99), September 20-21, 1999, Leuven, Belgium*, volume 152 of *IFIP Conference Proceedings*, pages 258–272. Kluwer, 1999.

[KL14] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography, Second Edition*. CRC Press, 2014.

[Kle06] Jon M. Kleinberg, editor. *Proceedings of the 38th Annual ACM Symposium on Theory of Computing, Seattle, WA, USA, May 21-23, 2006*. ACM, 2006.

[KRR17] Yael Tauman Kalai, Guy N. Rothblum, and Ron D. Rothblum. From obfuscation to the security of fiat-shamir for proofs. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part II*, volume 10402 of *Lecture Notes in Computer Science*, pages 224–251. Springer, 2017.

[Lev86] Leonid A. Levin. Average case complete problems. *SIAM J. Comput.*, 15(1):285–286, 1986.

[Lip91] Richard Lipton. New directions in testing. *Distributed Computing and Cryptography*, 2:191–202, 1991.

[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.

[LO87] Jeffrey C Lagarias and Andrew M. Odlyzko. Computing π (x): An analytic method. *Journal of Algorithms*, 8(2):173–191, 1987.

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

[Mce78] Robert J Mceliece. A public-key cryptosystem based on algebraic. *Coding Thv*, 4244:114–116, 1978.

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

[MI88] Tsutomu Matsumoto and Hideki Imai. Public quadratic polynominal-tuples for efficient signature-verification and message-encryption. In Christoph G. Günther, editor, *Advances in Cryptology - EUROCRYPT '88, Workshop on the Theory and Application of of Cryptographic Techniques, Davos, Switzerland, May 25-27, 1988, Proceedings*, volume 330 of *Lecture Notes in Computer Science*, pages 419–453. Springer, 1988.

[MMV11] Mohammad Mahmoody, Tal Moran, and Salil P. Vadhan. Time-lock puzzles in the random oracle model. In Phillip Rogaway, editor, *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, volume 6841 of *Lecture Notes in Computer Science*, pages 39–50. Springer, 2011.

[MP06] Silvio Micali and Rafael Pass. Local zero knowledge. In Kleinberg [Kle06], pages 306–315.

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

[Nao03] Moni Naor. On cryptographic assumptions and challenges. In *Annual International Cryptology Conference*, pages 96–109. Springer, 2003.

[NP85] Jaroslav Nešetřil and Svatopluk Poljak. On the complexity of the subgraph problem. *Commentationes Mathematicae Universitatis Carolinae*, 26(2):415–419, 1985.

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

[Pas03] Rafael Pass. Simulation in quasi-polynomial time, and its application to protocol composition. In Eli Biham, editor, *Advances in Cryptology - EUROCRYPT 2003, International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4-8, 2003, Proceedings*, volume 2656 of *Lecture Notes in Computer Science*, pages 160–176. Springer, 2003.

[PĬ0] Mihai Pătraşcu. Towards polynomial lower bounds for dynamic problems. In *Proceedings of the Forty-second ACM Symposium on Theory of Computing*, STOC '10, pages 603–610, New York, NY, USA, 2010. ACM.

[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.

[Rab79] Michael O Rabin. Digitalized signatures and public-key functions as intractable as factorization. Technical report, MASSACHUSETTS INST OF TECH CAMBRIDGE LAB FOR COMPUTER SCIENCE, 1979.

[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.

[RR00] Ron M. Roth and Gitit Ruckenstein. Efficient decoding of reed-solomon codes beyond half the minimum distance. *IEEE Trans. Information Theory*, 46(1):246–257, 2000.

[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.

[RSW00] Ronald L Rivest, Adi Shamir, and David A Wagner. Time-lock puzzles and timed-release crypto. *Technical Report MIT/LCS/TR-684, MIT*, 2000.

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

[RZ04] Liam Roditty and Uri Zwick. On dynamic shortest paths problems. In *European Symposium on Algorithms*, pages 580–591. Springer, 2004.

[Sed85] Robert Sedgewick, editor. *Proceedings of the 17th Annual ACM Symposium on Theory of Computing, May 6-8, 1985, Providence, Rhode Island, USA*. ACM, 1985.

[She12] Alexander A Sherstov. Strong direct product theorems for quantum communication and query complexity. *SIAM Journal on Computing*, 41(5):1122–1165, 2012.

[SKR+11] Douglas Stebila, Lakshmi Kuppusamy, Jothi Rangasamy, Colin Boyd, and Juan Manuel González Nieto. Stronger difficulty notions for client puzzles and denial-of-service-resistant protocols. In Aggelos Kiayias, editor, *Topics in Cryptology - CT-RSA 2011 - The Cryptographers' Track at the RSA Conference 2011, San Francisco, CA, USA, February 14-18, 2011. Proceedings*, volume 6558 of *Lecture Notes in Computer Science*, pages 284–301. Springer, 2011.

[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.

[Sud97] Madhu Sudan. Decoding of reed solomon codes beyond the error-correction bound. *J. Complexity*, 13(1):180–193, 1997.

[Sud16] Madhu Sudan, editor. *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science, Cambridge, MA, USA, January 14-16, 2016*. ACM, 2016.

[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.

[VV85] Leslie G. Valiant and Vijay V. Vazirani. NP is as easy as detecting unique solutions. In *Proceedings of the 17th Annual ACM Symposium on Theory of Computing, May 6-8, 1985, Providence, Rhode Island, USA*, pages 458–463, 1985.

[VW09] Virginia Vassilevska and Ryan Williams. Finding, minimizing, and counting weighted subgraphs. In *In Proceedings of the Fourty-First Annual ACM Symposium on the Theory of Computing*, pages 455–464, 2009.

[Wil05] Ryan Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theor. Comput. Sci.*, 348(2-3):357–365, 2005.

[Wil15] Virginia Vassilevska Williams. Hardness of easy problems: Basing hardness on popular conjectures such as the strong exponential time hypothesis. In *Proc. International Symposium on Parameterized and Exact Computation*, pages 16–28, 2015.

[Wil16] Ryan Williams. Strong ETH breaks with merlin and arthur: Short non-interactive proofs of batch evaluation. In *31st Conference on Computational Complexity, CCC 2016, May 29 to June 1, 2016, Tokyo, Japan*, pages 2:1–2:17, 2016.

[WW10] Virginia Vassilevska Williams and Ryan Williams. Subcubic equivalences between path, matrix and triangle problems. *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, 00(undefined):645–654, 2010.

[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.