On The Utility of Fine-Grained Complexity Theory

by

Manuel Sabin
(they/them)

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Shafi Goldwasser, Chair
Associate Professor Prasad Raghavendra
Assistant Professor Nikhil Srivastava

Summer 2020

On The Utility of Fine-Grained Complexity Theory

Abstract

On The Utility of Fine-Grained Complexity Theory

by

Manuel Sabin
(they/them)

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Shafi Goldwasser, Chair

The nascent field of Fine-Grained Complexity Theory has emerged and grown *rapidly* in the past decade. By studying "Hardness within P" and the connections of problems computable in, say, n^2 time versus n^3 time, this field addresses the *practical* efficiency of problems. However, while this more deeply *quantitative* approach better addresses practical hardness problem-by-problem, we lose connection to key qualitative claims of classical complexity theory such as the general ability to hide secrets (**Cryptography**), the ability to show that a world where we have access to randomness is no more powerful computationally than a deterministic one (**Derandomization**), and the ability to take a problem that is hard in the worst-case scenario and turn it into one that is hard almost always (**Hardness Amplification**).

We show that these connections can be recovered and that the problem-specific claims of Fine-Grained Complexity Theory cannot exist in a vacuum without ramifications to classical structural Complexity Theory. Namely, we show that the core *worst-case* hardness assumptions that define Fine-Grained Complexity Theory yield:

Hardness Amplification: We attain Fine-Grained problems that are hard *on average* (Ball et al., STOC '17). By achieving average-case hardness within the Fine-Grained world, we use this as a stepping stone to achieve both Cryptographic primitives and Derandomization.

Cryptography: We obtain the first Proofs of Work (PoWs) from worst-case complexity assumptions, thus finally placing these fundamental primitives on a rigorous theoretical foundation (Ball et al., CRYPTO '18). We further propose the concept of Fine-Grained Cryptography and this call has now been answered in (LaVigne et al., CRYPTO '20) where some progress is made towards achieving Public-Key Fine-Grained Cryptography.

Derandomization: We construct complexity-theoretic Pseudorandom Generators (Carmosino et al., ICALP '18). This both achieves the best known derandomizations from *uniform* assumptions as well as connects the problem-centric Fine-Grained Complexity to the resource-centric study in Complexity Theory of randomness as a resource.

To my Mom who taught me to breathe, my Dad who taught me to listen to the breath of others, and to the Queer, Trans, and POC community who gave me room to breathe.

To all those with just as much potential but not enough room to breathe.

Contents

Acknowledgments

All that you touch
You Change.
All that you Change
Changes you.

– Octavia E. Butler,
Parable of the Sower

After being accepted to UC Berkeley, I went to their Visit Days where, at one point, all of us new admits were gathered into a large auditorium and told all the great things that would await us at Berkeley if we accepted the offer. One thing, that wasn't Berkeley-specific, but excited me immensely was being told "as different as you are now from when you graduated high school, is as different as you will be from now when you graduate from here." Feeling that I had Changed dramatically since my high school years, I couldn't begin imagining how much different I could be finishing my PhD given that I already felt like I knew who I was and what I wanted at that time, and this mystery excited me greatly. I proceeded to fight Change tooth and nail for the next few years. This is likely because I didn't actually have a good sense of who I was or what I wanted. Or maybe they just Changed.

In either case, the prediction of massive Change was true and I was very right to be excited for it. This journey has been long and often painful and I have learned and unlearned much about academia and the world. So I'm going to take my *time* here as I look back on six years of Change and Growth and acknowledge those that helped nurture it:

One thing I have learned in graduate school is that research is an emergent process. Coursework and textbooks paint a picture of knowledge that is like building blocks: You start with the fundamentals and then stack concepts like building blocks so that, if you're understanding the material, you can always see the bottom and the clear connections in-between to make a clean, linear, bottom-up view of knowledge. But the social activity of "research" immediately breaks this illusion. In any deep field of study, the myriad *towers* of building blocks are so high and vast and maze-like that the bottom is always hazy to make out and each person has a different internal heuristic map of this city of blocks in their head that they use to navigate this research landscape.

In this way, research is a collectively shared vision of a landscape that we each have a different segment of a hazy picture of. Because of the vastness of this landscape of knowledge, we have to rely on the accounts of other researchers who have explored areas we haven't and we have to connect all these pieces together not through a bottom-up, linear, unified proof of the facts spoken through the language of math but instead through an ever-changing collection of narratives and social interactions through which the landscape of a field emerges. Research is tied together by connotations and stories.

Because of the social and narrative-based nature of research, it is impossible to separate the research done in a field from the metaphors, analogies, terminology, and jokes of a research community and the cultures we come from that make these metaphors, analogies, terminology, and jokes salient. And it is impossible to separate research from our writing conventions, dissemination styles, agreed upon ethics, and even what we consider interesting and research-worthy. What body-language is used to communicate mathematical concepts? What spatial and visual cues do we use to sketch an informal proof-by-pictures? And what poetics do we imbue into research questions that help us conceptualize them into the larger research landscape and research narrative? As these cultural practices propagate and solidify a field's conventions for understanding concepts, they become just as much a driver of research (and just as much as an historical accident) as the small well-worn bag of mathematical techniques we use to prove all that we've been able to so far.

This is all to say that all research that I've seen, along with my own research, has so many unacknowledged, intangible, and untraceable influences that are often invisible to the emergent process of research that I could never do justice to acknowledging them and how important to me they've been. The intellectual and emotional journey that occurred throughout my time at UC Berkeley and in the production of this work has deep influences that are technical, cultural, emotional, etc. and has far too many people and communities to thank. Know that if you think you have had some influence in my life, you probably have. And it has likely affected my research and my future research. Beyond these platitudes, I'd still like to *attempt* acknowledging those that have Changed me. Both "good" and "bad" experiences, of course, have shaped me. And I won't ever be able to do any of them justice. So, for this thesis, I think it makes the most sense to just tell stories about people I have a lot warmth for.

Coming to a massive elite research university like UC Berkeley from a commuter teaching university was and continues to be a culture shock in many ways. There are some things at UC Berkeley I got that I would never have been able to get at my undergraduate university, California State University Sacramento, but there are some thing I got at Sacramento State that I never would have gotten at UC Berkeley. The life-changing teaching in small classrooms at Sacramento State's Math Department and the differential perspective to compare elite academia and its communities to are things I could never trade. The first place to feel like a home as a young adult was the Math Lab, where I tutored, talked math, and most importantly found community. My partner compared my relationship with the Math Lab with the TV series *Cheers*: Where everybody knows your name. There are many people to name but Jacob Russell-Madonia was one of my first real math sparring partners and Janelle Currey was my first teaching sparring partner and both are dear friends. My connection to them is most clearly bolstered by the person that drew us together and was the first to allow us to truly fall in love with math: Scott Farrand.

Like every math major at my undergraduate university, I had heard Scott was *the* professor to have. Of course I didn't have any sense of what that could mean except that he was probably really really good at explaining stuff and might be a little funny, but that's because I had never seen what teaching could be before Scott. I could try to compare his teaching to mildly more familiar things like a good coach or a facilitator of play therapy or a camp counsellor or an after-school activities organizer who is familiar with social work or some other thing but since so many have never experienced his type of teaching nor had a good transformative experience with any of my attempted comparisons, all of these analogies always feel inadequate (especially as people try to grasp how Abstract Algebra could be taught in this way). A go-to quote of his I use when describing him (and many people inside and outside of academia have heard descriptions of him now) is that when he was attempting to get the class to see a pattern or make a conjecture he would sometimes encourage us to think "If I were God, how would I design the universe so that the solution to this problem would be as cool and beautiful as possible? And the answer is usually exactly that, or something better than I could have imagined." Before Scott's courses I probably liked math mostly because I was good at it. Afterwards, I was in love with math because it was an endlessly bigger-than-me land to explore where unadulterated truth could be tasted while it consistently exceeded my imagination in what I could hope for. Yet this transformative experience still barely scratches the surface of the many ways Scott has Changed my life. Not only did he Change my personal relationship with math, it was Scott who first showed me math as a *social* activity. I can't express how much these affected my academic life and research. But most importantly, Scott completely transformed my understanding of teaching and, because of his humanistic teaching style, the way I engage with and view people in general. Who I am and how I move through the world will have deep roots with Scott and he set the path I'm on in a very large way. I'm always excited when that path intersects with Scott again.

Coming from Sacramento State to UC Berkeley was daunting in many ways but my first advisor, Christos Papadimitriou, welcomed and believed in me far before he had any indication that it might be merited and long before I believed I deserved it. His kindness, support, and belief was more necessary to me than I can grasp even now. His poetic and philosophical bent set the cultural tone I was looking for in Theoretical Computer Science (TCS) and I will continue to carry it with me. As sad as it was when Christos left to Columbia University, I couldn't have had been luckier with timing as Shafi Goldwasser was just then planning to move back to Berkeley after decades of being at MIT. Shafi immediately took me seriously as a researcher. After three years of finding my feet within TCS without much time or a definitive milestone for me to look back on my progress and see *myself* as a researcher, Shafi's confidence in me as an accomplished self-sufficient researcher in the TCS community came just when it was needed. Most importantly, Christos and Shafi's inimitable flavor of research and conceptual creativity will forever be goalposts for me and my work.

My real learning of TCS, though, began with the students at Berkeley. At Visit Day I met Ben Caulfield and we decided then that, if we both accepted Berkeley's offer, we would look for housing together. We both accepted. Ben and I both entered graduate school together excited and confused and scared and curious and we both fumblingly explored academia and TCS and the Bay Area and music and life. I don't know how I would have started graduate school without Ben, and I can't imagine how different a trajectory I would have been on if I hadn't. We lived together the first two years of my life in the Bay and those wide-eyed years of newness and learning will always be a special point of time of real camaraderie. Jonah

Brown-Cohen was two years senior to me when I entered and he set the tone for the culture of the TCS community for me. Or at least it was the tone I was looking for: one that centers the social aspect of research and is fun and doesn't take itself to seriously. He was one of the people I first took note of as an example of how well-rounded an academic could be (as opposed to the harmful solipsistic genius trope of movies). One of the most important things he passed on to me was how everything has a simple way of being said and to search for ways to knock scary, complex ideas off of their pedestal. It turns out everything is simple, there's just a lot of it. In general, there are far too many people in the Berkeley Theory group from when I entered to when I left that should be mentioned here, so I won't try. But this group provided a special community of peers and laidback environment that I was able to feel comfortable and welcomed in in many ways as I waded into many unfamiliar waters.

I also had the privilege of becoming embedded in the UC San Diego and MIT Theory groups thanks to Russell Impagliazzo and Ryan Williams hosting me at their universities for a semester each, respectively. Both Russell and Ryan's research has consistently been the epitome of the flavor of research I love and emulate my own research after. And their research is only matched by their personalities, injecting humor and poeticism into the social endeavor of TCS to be some of Complexity Theory's best storytellers. Being able to take courses from, play boardgames and music with, and find food around Berkeley with them as they visited the Simons Institute were excitements that I was extremely fortunate to complement with full-semester visits to each of them. The UC San Diego and MIT Theory groups were beyond welcoming and I was always sad I couldn't bring them back to Berkeley with me as well. I feel sad that so much of my time at MIT was spent writing job applications but being able to join their Theory retreat, play music often with them, and join their end-of-the-year talent show made up for it. Again, there are far too many people among these two Theory groups to name (especially MIT's giant group) but I want to acknowledge Rio LaVigne who I shared my first attempt at research in elite academia with all the way back in my undergraduate degree when I visited Stanford for Summer. During that Research Experience for Undergraduates program Rio and I did our first research together and she showed me what research at that level could look like. It has been an unexpected pleasure how long our paths have intersected and it is always nourishing when we get a chance to reflect on the ways we've grown and struggled since our first young meeting.

Despite all of the talent and community at my home university, I actually never worked towards a publication with fellow students or professors there. While I'm partly saddened by this, the reason is because there were simply too many people to meet and things to work on at the Simons Institute, where experts from across the world would come each semester and where I found all of my collaborators and many friends. This is where I met Andrej Bogdonav, Russell Impagliazzo, Alon Rosen, and Ryan Williams, who I each visited at their home universities for a semester each. It is also where I met Marshall Ball, Marco Carmosino, and Prashant Nalini Vasudevan. Although some of these people I had already emailed with before or others where I didn't remember our meeting until we met again later, this is my current full roster of co-authors, all of which I originally met at the Simons Institute. There is far too much I could say about Andrej and the *amazing* food and math he showed me and

Prashant in Hong Kong and his limitless hospitality in showing us the area and inviting us into his home; or to say about Marshall and the limitless angles he will attack a problem with armed with his encyclopedic knowledge of techniques and his gorgeous and lucid talks that converted me to drawing my talks on an iPad; or to say about Russell and playing music to his TCS-themed parody songs and joining him in improv scenes and boardgames and the gems of technical knowledge that he constantly drops that are second only to his clarifying interpretations of results that give them story and meaning; or to say about Ryan and the music community he wills into existence where ever he goes and the amount of knowledge I've gained from his explanations (and corrections). But this is already getting too long and has much more to go, so I'll talk about some of the people that have been some of my first introductions to research and the indelible Change that will permeate through my life.

The first cutting of my teeth with research was with Alon Rosen. Alon's role in the TCS community is one my dream roles. He invites junior students to study with him over the summer and serves as a mentor and community builder amongst them, putting his research knowledge, intuitions, philosophies, and humor at his cohorts disposal and guides them as they stretch their research legs for the first time. He is fully aware that these young adults are newly exploring a foreign research domain with a foreign community in a foreign place and treats them as peers while guiding them in these formative stages. He is an embodiment of research being a social activity, and he aims to make it a fun one. Alon gave me my first setting in graduate school where research can be for the love of the game.

If Alon was my first mentor in research, Prashant Nalini Vasudevan was my first peer. It is generous to myself to use the word 'peer' here given how much knowledge and creativity Prashant had compared to myself (or my perception of myself) when I first met him, but his patience and firm intuitive grasp of so many concepts made it easy to ask him *many* "basic" questions as a fellow student. I still take advantage of his insight. We were both working with Alon when we met and have since explored large swaths of TCS and five countries together. I have met few people so curious and non-presumptive as Prashant is as he wades into unfamiliar areas of TCS along with unfamiliar cities and cultures. Prashant is a complete wanderer. Most crucially, though, he has a tender heart and has zero knee-jerk defenses for class, prestige, "work ethic," etc. As my identities are very underrepresented in academia along multiple axes, it has been a particularly painful journey but Prashant's unassuming openness throughout it has made his friendship invaluable to me. It has also taken us into many unlikely places across the world from jazz shows in a much too cramped after-hours barbershop in Hong Kong, to Queer underground wrestling events in the Oakland Bay Area, to accidentally driving on train tracks to get to a hostel in Jerusalem, to standing on a beach's wave breakers to see what the big deal with the storm warning was about, and so on and so forth and countless chocolate shops along the way. Our continually intersecting lives through time, place, and topic seems to have our journeys intertwined and I hope it stays that way.

I don't think it's possible to convey how much Marco Carmosino has influenced and been a core part of my graduate life to anyone, least of all to Marco. In the years that I've known him, I've become very familiar with how extremely generous he is with his time and knowledge and how he immediately treats younger researchers, still finding their footing in academia, as peers and guides them in their journey. But when I first met him and he showed this generosity towards me, it felt unprecedented and unrepayable. It still does. Part of this kindness does seem to have some ulterior motive of converting everyone he can into Circuit Complexity Theorists, thus jeopardizing their research fundability and job prospects, but I couldn't be more grateful for this conversion. My understanding of TCS, the TCS community, academia, and the practice of research is inextricable from Marco's influence. More crucially, his wide interests, depth of knowledge, and goals beyond TCS showed me that not only can you be well-rounded in academia but that you don't have to be siloed to a research area or even academia itself (you can also become hyper-pedantic about organization systems and music production audio quality and epistemological concerns of meta-science and so on). As I've recently begun branching in new research directions and interests, it has been with his freeing example as a guide. The most important thing he has done for me, though, has been being endlessly empathetic, supportive, and validating as I've quite painfully moved through academia as a Queer, Genderqueer, underrepresented Person of Color. Since early on, he has given me and helped me dissect theories of systemic issues, words and frameworks to contextualize my experience with, and his time and ear that has helped me navigate academia and get a bird's-eye view of my position within this system. If I stay in academia in the longterm, it will be in no small part due to Marco.

The last cluster of people I want to mention that I met through the Simons Institute have created a fully new branch of my research trajectory. I was worried about TCS's new subfield of Algorithmic Fairness *before* I attended the Simons Institute's Summer clusters on this young field but, after, it was clear that my critiques were shared and already explored in depth. Finding community outside of TCS that shared similar concerns and critical analyses has now started me down an interdisciplinary path that I am excited about and take as a responsibility. Thinking routinely through these issues with Ezra Goss, Lily Hu, and Steph Teeple has been both intellectually enriching and life-giving. Having my last big travel before COVID-19 be to Barcelona to see them and run a a workshop with them on introducing power analyses to Algorithmic Fairness couldn't have been a better way to indefinitely end my traveling. I also want to deeply thank the person who encouraged us three to run that workshop and who has been so generously helpful to us and has given me so much career path counseling: Sorelle Friedler. In general, I am indebted to Ruha Benjamin, Sorelle Friedler, Nancy Krieger, and Timnit Gebru for encouraging me to help the TCS community I am a part of to address the critiques of Algorithmic Fairness and starting me on this interdisciplinary path. These influences have already had an immeasurable effect on the direction of my life: my 3.5 year postdoc position is now with a team of lawyers and legal philosophers and my TCS projects will be secondary. Lastly, in this big fork in my research life I want to thank Natalia Bilenko and Ria Kalluri for creating community and philosophies through the Queer in AI and Radical AI networks that provided the exact landing place I needed in this transition and for being so welcoming as I embedded myself into these great communities. I look deeply forward to continuing to co-create these communities with them and for their involvement in the QTPOC Reclaiming Education and STEM (QTPRES)

conference I co-organized with Salaine Ramirez and Cal Zielenski (indefinitely postponed due to COVID-19). I am extremely excited for this new direction and the community I will find along with the community I already have.

Lastly for academics, I want to thank deeply the the staff and administration who have always been nothing but kind to me whether I was at the UC Berkeley EECS Department, the Simons Institute, IDC Herzliya, the Chinese University of Hong Kong, UC San Diego, MIT, or any place I visited even briefly. They have always been absolutely crucial to academia and to my journey in it. I especially want to thank Shirley Salanio and Aimee Tabor. Shirley has always been a very supportive and sympathetic constant. As I've felt detached from academia, Shirley's kindness has always helped me still feel tethered. Aimee guided me through my REU program at Stanford, through my application process to graduate school which was an absolute mystery to me beforehand, through fellowship applications including the NSF GRFP fellowship I won, and through my first couple years as a constant support in a bizarre journey for me. It was painful to discover her passing and I am sad that I didn't get to complete this part of my journey with her but I am so grateful she was able to start it with me. My life would likely be in a very different place without her.

If the list of people that have helped me academically is long (and this has just been a miniature version of it) then the people that have shaped me not explicitly within an academic context is much much longer. So long, in fact, that the remaining portion of this section will actually be shorter because I can not possibly try to be as individualized as I was with my academic acknowledgments. I will leave far far too many people out but I want to acknowledge some people that are inseparable from my academic journey.

Stacy Taniguchi has been my first therapist and I am now extremely zealous about the importance of therapy. Or maybe I am just extremely zealous about the importance of Stacy. And I am scared to have to find a new therapist as graduate school ends when I know I will measure them with respect to her. In general, I pride myself on introspection, reflection, and Change, but I wish I had complemented and assisted these processes with therapy much much sooner. Over the past two years, Stacy has had an immeasurable effect on my path through academia and the ways in which I move and take up space in it. It had become repeatedly clear in many many ways that someone like me was never supposed to "make it" into that level of academia. It continues to be. And I found Stacy just as I was finding words for this experience and just starting to feel like I was losing my mind as I began questioning the long-standing, prestige-based systems I was in. Stacy reframed this: "It sounds like you're using your voice." Being voiceless in a community that you're meant to be a part of and joined purely for the *social* activity of research is not sustainable and not healthy. It is still not clear if academia has room for me, but the importance of finding my voice while in academia with Stacy was more than I could ever say here. If I find a *sustainable* way for me to stay in academia, it will be in large part because of Stacy. And if I don't stay in academia, whatever sense of balance and sustainability in life I find will have roots with her.

Of course my deepest roots to who I am come from my family. I've grown into adulthood watching my siblings grow into adulthood (too quickly). I've learned so much about teaching, learning, communicating, fighting, and being a person from Michelle, Dulcinea, and Elias Sabin. And I can't even begin to unravel the influence on my life my cousin Rico Sabin has had, my first non-parent role model. Much of childhood, pre-teens, and teens (and, thus, my life in general) were shaped by Rico and our young explorations of what aspirations mean. I'd be remiss if I didn't name every single one of my family members, and so I must be remiss. But I would like to acknowledge my grandpa Habib (Harold) Houshmand, my uncle Darius Houshmand, my uncle David Sabin, and my other grandpa and the original eponymous Manuel Allen Sabin who all got to see me start this journey but are not here with me to see me end it.

This entire thesis could certainly be filled with ways to acknowledge my parents' effect on me and there's no way to summarize it here. The privilege I've had with my parents cannot be overstated and I would absolutely not be who I am or where I am without the stable environment they've always held for me with *unconditional* support and love. I'm often embarrassed by the amount of unearned privilege I have both materially and emotionally by just being born to these two who always first-and-foremost center family. Thankfully, they taught me how to question and use that privilege. They also taught me how to not be reliant on their pride even though they are proud of me nonetheless. Most importantly, they allowed me to dream free and high by always having visions of high possibility for me but no expectations on what that would look like. So much Change has happened these past six years and the freedom from expectations that allowed me to sit in that Change and Grow has meant the world. My life and entire worldview is fundamentally rooted in their raising of me in ways I'm still understanding. I mostly want to thank them for always encouraging me to ask "Why?" even when it wasn't in their best interest. Subjecting them to never-ending strings of why's to understand things to their fundamentals and constantly questioning their authority and reasoning has served me well in research and, most crucially, made my life an intentional one. Mostly, it has pushed me to tilt at windmills as I ride through this world.

There is never enough I would be able to say about Cal Zielenski. They are my friend, partner, spouse, housemate, confidant, witness, advisor, and so on and so forth. As a practice, we have tried to define what we consider special to our relationship and what we consider or want considered unique to us, and it seems to boil down to the simple but pure fact that we grew up together. We were children when we met and now we have shared a life together and have much more life to go. My personality, who I am, what I have experienced, how I have interpreted it, and so on is all inextricably tied to Cal and how they have shaped me. I can't imagine who I would be if I never met Cal because I would be different person; a different career, a different philosophy, a different home, a different set of friends, a different politics, a different history. A stranger. And I've learned so much about what it means to love and own yourself from Cal that I have zero curiosity of who that stranger would have been because I love the person I've become with Cal. I have learned and continue to learn so much from them, and my logic to conceptualize and understand systemic issues and harms is always playing catch-up to their human-centered intuitions. I am ashamed that it took the pain of being marginalized within academia for me to search out the language and frameworks to describe the issues that Cal already *knew*. Their strength and unapologetic voice continues to inspire, challenge, and teach me. They have seen me at all my stages of adolescence and adulthood so far and all my stages of academia and have held much of the pain and excitement of it throughout. Life continues to be weird and uncertain and unpredictable, especially now, but I'm very happy that our lives continue to wind through this world together.

Lastly, I want to express the deepest gratitude I have for the QTPOC community, who first showed me what community could mean. I have learned so much from you. About language, ways of knowing, critical theories, recipes, systemic frameworks, ways of connecting, the Bay Area, self-love, ways of experiencing, power, and on and on. But most importantly I learned my voice from you. And how to use it. A while ago, I thought that in my acknowledgments section I would say something to the extent that "If I have succeeded at all, it is because far too many people have been far too kind to me far too often." The QTPOC community taught me to unlearn this. While many people have been kind to me, it was not too many and it was not too often and there's much of my own kindness to myself to credit here as well. As Hannah Gadsby says "Do you understand what self-deprecating means when it comes from somebody who already exists in the margins? It's not humility. It's humiliation." I learned from the QTPOC community how it is possible to be humble while still taking up space and while still using your voice. This seems like it should be the place with the longest length and the most names, but, compared to the finiteness of academia, the list of friends, influences, partners, advocates, acquaintances, and so and and so forth is practically endless and is inter-generational and is populated just as much by collectives and memes as it is by individuals. Many would likely want to remain anonymous anyways. Just know that I see all of you. And I want each of you to find joy. And I owe so much to all of you. This world-wide community and its emergent culture and philosophies and humor and politics is second only to the individual humans that populate it. I will always be rooting for you and it will not be from the sidelines.

To finish, since this has become a long journal reflection entry anyways, I would like to acknowledge myself. I have personally devalued degrees and the centralization of prestige and the use-cases for much of research under capitalism-based academia, so it is hard for me to value this milestone accomplishment for myself. So much of what "success" I have achieved has roots in my unearned privilege and it feels gratuitous to feel proud of a Computer Science degree which already arbitrarily wields so much political and social and economic power, especially over "soft science" degrees or those that produce or organize without degrees. But, as I mentioned with Hannah Gadsby, self-deprecation for those already in the margins is humiliation, not humility. So, while knowing that others are kept from the ivory tower despite having just as much potential and just as much hard work to put in, I still find pride and accomplishment that I did put that work in. And that I have fought to include the communities that are kept out. I am proud that I entered a field, community, and culture that was fully foreign to me and learned it and became a part of, all while questioning it and Changing it and continuing to Change it. I am proud that I was not passive in this journey. I am proud to have listened more than I have talked and I am proud to have Changed. Constantly. I am proud to be at a point where I can feel comfortable saying I am proud of myself now and I am proud to have found a voice in these six years. I am proud to be a part of the QTPOC community that helped me find that voice. I want to end here acknowledging myself and, in that, acknowledging everyone that makes me. Because I am not an island. Every person and community that has shaped me I carry with me and, while I may have had enough stubbornness to push through this program without every person mentioned here, ever person mentioned here has made this journey feel more like life and not a conveyor belt. I have so much love for everyone who makes me me because I have learned to have a lot of love for myself. You have all created me. Thank you so much.

Chapter 1

Introduction

From complete invention, Cosimo, I believe, had arrived by successive approximations, at an almost entirely truthful account of the facts.

– Italo Calvino, *Baron in the Trees*

[Complexity Theory is] not good at making more answers, but we're good at making less questions.

– Christos Papadimitriou

Is it as easy to verify a problem's solutions as it is to find them? Can we hide secrets within computational hardness to achieve Cryptography? Can we find as many answers in a world where we didn't have access to randomness? The field of Complexity Theory deals in asking profound *questions* that are fundamental to the nature of computation and what they tell us about what we are able able to solve and know the solutions to in our world. However this field is notorious for not being able to *answer* them. That is, thus far, we, as a field, have been unable to provide definitive answers to our most fundamental and simply stated questions or often even many of their supposedly easier sub-questions.

For someone who is unfamiliar with the results of the field, a natural question may be what *has* Complexity Theory accomplished? Is all of its sustained effort kept steady and vigilant just to celebrate the few unconditional theorems that are sporadically achieved? While Complexity Theory would be certainly able to merit itself on its ability to formulate and formalize fundamentally new and deep questions alone, it is not clear a field would be able to maintain motivation and morale based solely on this.

Complexity Theory's sustaining force and connective tissue that fills the field in between its sparse unconditional theorems lies in its most impressive and consistent output: its ability to *connect* its many disparate questions, showing that many of these questions are equivalent or that their answers would imply partial answers to each other. Thus, Complexity Theory's strength is in showing that many of the core questions of the field, and of Theoretical Computer Science (TCS) more generally, are simply different sides of a many-sided coin.

Indeed: the computational difficulty of problems in Number Theory would imply secrets can be hidden to achieve Cryptography, e.g., [BM84, GM84]; the existence of Cryptography would disallow an entire class of natural seeming proof styles from ever showing that it is harder to verify a problem's solution than it is to find it (i.e. $\mathsf{P} \neq \mathsf{NP}$) [RR97]; if we *were* able to use these natural types of proofs to resolve easier questions, then computationally learning what certain types of circuits look like would also be easy [CIKK16]; if there are some problems that are *hard* for circuits to compute, then all problems that were easy to solve using randomness are still *easy* when our access to randomness is taken away [NW94, IW97, STV01]; and so on and so forth.

The continually emerging crown jewels of Complexity Theory results are often deep connections between deep questions of Complexity Theory. These lay the foundations, reveal the barriers, and populate the techniques of TCS in general, and when one of Complexity Theory's few unconditional results is achieved it usually from utilizing this vast web of connections.

Understanding this crucial connective component of Complexity Theory's ethos thus makes it alarming when an area of Complexity Theory grows in isolation: The nascent field of Fine-Grained Complexity Theory has emerged and grown *rapidly* in the past decade. By studying "Hardness within P" (i.e. fine-grained levels of polynomial-time hardness within the set of problems that already have polytomial-time solutions) and the connections of problems computable in, say, n^2 time versus n^3 time, this field addresses the *practical* efficiency of problems. However, while this more deeply *quantitative* approach better addresses practical hardness problem-by-problem, we lose connection to key *qualitative* claims of classical complexity theory such as the general ability to hide secrets (**Cryptography**), the ability to show that a world where we have access to randomness is no more powerful computationally than a deterministic one (**Derandomization**), and the ability to take a problem that is hard in the worst-case scenario and turn it into one that is hard almost always (**Hardness Amplification**).

That is, Fine-Grained Complexity Theory has, similar to Classical Complexity Theory, few unconditional results and instead defines itself by constructing an intricate network of connections between existing questions *yet these questions are largely all self-contained within Fine-Grained Complexity Theory with few connections to the pre-existing backdrop of Classical Complexity Theory.* This leaves Fine-Grained Complexity Theory in the uneasy state of having interesting things to say but being uncertain if those interesting things are actually true and, more importantly, not having the connective tissue relating these questions back to the rest of TCS.

As we will discuss in Section 2.3, there *have* been some deep connections from Fine-Grained Complexity Theory to Classical Complexity Theory found but, prior to the work presented here, these connections are predicated on the *collapse* of Fine-Grained Complexity – i.e. that the core conjectures of the field are false. If Fine-Grained Complexity Theory doesn't collapse then, does it stand on its own in isolation from Classical Complexity Theory?

The aim of this work is to show that the answer to this question is 'No.' We show that the thriving of Fine-Grained Complexity Theory does not exist in a vacuum without deep ramifications to the philosophical claims, applications, and core questions of Classical Complexity Theory. Namely, we show that the core *worst-case* hardness assumptions that define Fine-Grained Complexity Theory yield:

Hardness Amplification: We attain Fine-Grained problems that are hard *on average* in [BRSV17a]. By achieving average-case hardness within the Fine-Grained world, we use this as a stepping stone to achieve both Cryptographic primitives and Derandomization.

Cryptography: We obtain the first Proofs of Work (PoWs) from worst-case complexity assumptions, thus finally placing these fundamental primitives on a rigorous theoretical foundation [BRSV18]. We further propose the concept of Fine-Grained Cryptography and this call has now been answered in [LLW19] where some progress is made towards achieving Public-Key Fine-Grained Cyrptography.

Derandomization: We achieve complexity-theoretic Pseudorandom Generators [CIS18]. This both achieves the best known derandomizations from *uniform* assumptions as well as connects the problem-centric Fine-Grained Complexity to the resource-centric study in Complexity Theory of randomness as a resource.

To understand these connections we must first immerse ourselves in the Fine-Grained and Classical Worlds.

Chapter 2

The Fine Grained World: Background

Every TCS graduate student should be able to recite the "party line" answer that it has happened again and again that once a problem has been shown to be in polynomial time, people managed to come up with algorithms with small exponents

Boaz Barak, *Windows On Theory*

The study of Complexity Theory is, as Alan Cobham put it, "most directly indicated by simply asking two questions: first, is it harder to multiply than to add? and second, why?" [Cob65]. That is, Complexity Theory, at its core, simply seeks to understand the inherent difficulty of solving computational problems and the relative computational complexity between these problems. As larger inputs are given for a computational problem, how many steps are needed to reach a solution to the input – i.e. what is the minimum runtime, $f(n)$, required for *any* algorithm attempting to solve the problem as the input's size, n, grows larger? [1] Within his same seminal work [Cob65], Cobham seeks to quantitatively characterize what should computationally be considered "easy" and what should be considered "hard:" If a problem can be solved in polynomial time (i.e. $f(n) = n^k$ for some k) then it is considered "easy," all else is "hard."

This principle, the Cobham-Edmonds thesis [Cob65, Edm65], is now widely accepted and is formalized as the complexity class P: the set of all computational problems that have a polynomial time solution [Gol08]. While a polynomial runtime certainly seems more easy to handle than an exponential runtime, the polynomial itself may have a very large exponent or a large leading constant: While a runtime of $2^{.00000001\ n}$ will absolutely overtake $99999\ n^{200}$

[1]There are many possible measures of the complexity of computational problems including the amount of time, memory, random bits, quantum bits, non-uniform advice, etc. that the problem may require to be solved but, for exposition in introducing Fine-Grained Complexity Theory, we will focus on runtime here.

asymptotically for very large input sizes n, for most practical inputs the polynomial runtime is actually much more prohibitive than the exponential one.

Given this critique, there are three important justifications for this weighty assertion that a class as broad as polynomial runtime is useful to consider as the set of "easy" problems:

P is a proof of concept for "easiness." Once *some* polynomial runtime is found for a problem, more efficient algorithms very often follow with smaller exponents (e.g. quadratic) and smaller leading constants.

P makes useful *qualitative* and philosophical distinctions. Some computational problems have so little structure in them to exploit algorithmically that finding a solution can be done no faster than brute forcing every possible answer to check if it is the actual solution. Understanding if SAT is one of these problems is the core of what $\mathsf{P} \overset{?}{=} \mathsf{NP}$ is often meant to ask; if a solution can be represented with n bits, this amounts to brute forcing a domain of size 2^n. A polynomial-time algorithm for a problem then tells us something deep about the structure of a computational problem: There is significant, e.g, algebraic structure, graph structure, combinatorial structure, etc. in the problem and that this can be exploited in an algorithm to achieve something fundamentally different than a naïve brute force algorithm. This makes separations between polynomial time, quasi-polynomial time, exponential time, etc. act as *qualitative* separations instead of just quantitative ones as they create a taxonomy of hardness that can be seen as making distinctions between distinct levels of the structural impoverishment of computational problems.

P plays well with our mathematical toolkit. Polynomial time has several useful properties for creating a taxonomy of computational complexity and for the techniques used in doing so. For example, when considering a Turing Machine (TM) as a computational model to measure a problem's complexity with, the distinctions of how to define that TM all vanish when treating all of polynomial time as an equivalence class: Whether a TM has one working tape versus two or many, whether the TM has RAM, which programming language you write your algorithm in, whether you express your TM with circuits, etc. can all be translated to one another within polynomial time and so a new theory of Complexity isn't needed for each distinct model. Thus, we can have *one* theory of computational complexity that is robust polynomial time transformations and thus to arbitrary fluctuations in model design choices. This allows a rich framework for Complexity Theory such that trivial model discrepancies can be swept under the rug via a coarse-grained polynomial lens so that the only model distinctions are those that introduce a genuinely new resource such as the ability to manipulate quantum bits or considerations of memory or the use of randomness or non-deterministic machines or machines that can receive advice, etc.; indeed, the power relations amongst these different models remain among the core questions of the entire field. Further, since the "bread and butter" of Complexity Theory is in algorithmically reducing computational problems to each other, it is very important that reductions can be chained together meaningfully; this is guaranteed for polynomial-time reductions since function composition of polynomials remains polynomial. This makes complexity classes defined via polynomial time reductions amongst its problems hold together and easily manageable with our technical tools.

The latter two points here give us powerful philosophical and technical justification for our use of polynomial-time reductions and for the existence of P as a proxy for "easiness." But these justifications may provide small comfort for practitioners in the world who have a computational problem they need solved and may have more limiting resource constraints on what they can accept as "easy." Even the first justification may be vacuous for them if they need faster solutions *now* or if even quadratic time is prohibitively expensive for them. Indeed, while practical problems in DNA sequencing can be solved in "just" quadratic time, they are still, when considering the human genome consists of roughly 3 billion base pairs [BI15], prohibitively expensive on their large input sizes.

To contend not just with the philosophical insights of the coarse-grained, polynomial-time equivalence classes of Classical Complexity Theory but with the practical concerns of real-world computational problems with large inputs and limited resource constraints, a more fine-grained lens needs to be applied to Complexity Theory.

2.1 Fine-Grained Hardness: Hardness Within P

With the real-world difficulty of handling cubic or even quadratic runtimes on practical problems, Fine-Grained Complexity Theory has rapidly blossomed in the last decade not just for its novel insights but for its practical urgency. Classical Complexity Theory has used membership or non-membership of complexity classes to understand a problem's complexity and has used the showing of a problem to be NP-complete (i.e. provably one of the hardest problems within NP) as a "gold standard" for giving evidence of a problem's computational intractability. But these tactics are all too coarse-grained to distinguish the hardness of polynomial-time problems to within, say, their exact exponent since such distinctions are invisible to common tools like general polynomial-time reductions. Namely, such problems with fixed polynomial-time solutions are lumped together as "easy" under Classical Complexity Theory and the field lacks language and techniques to understand their *fine-grained hardness* beyond that broad stamp of "easy."

But how, then, do we discuss practical problems that can be solved in, say n^2 or n^3 time but have resisted all attempts to speed them up further? What if n^3 isn't fast *enough*? Fine-Grained Complexity Theory, thus, has emerged within the last decade to classify and explain the *moderate hardness* of practical computational problems already inside of P. Indeed this young field is sometimes referred to as "Hardness within P" (see [Wil15, Wil18] for great surveys of the area).

Similar to Classical Complexity Theory, a main tool of this field is in the ability to reduce computational problems to each other such that a faster solution to one problem yields a faster solution to another (and so, by contrapositive, the, say, quadratic hardness of one problem implies the quadratic hardness of another so long as the runtime of the reduction itself is subquadratically "tight" enough).[2] This creates a web of connections between problems that yields a taxonomy of computational problems and reveals key players.

Unlike Classical Complexity Theory, though, this web doesn't seem to reveal useful clusters of complexity classes and models of computation that have typically structured and defined Complexity Theory. Namely, we lose the aforementioned nice properties that Cobham described [Cob65] to justify P: for general polynomial-time you had that polynomial time reductions composed to become another polynomial-time reduction and you had that you could have a largely model-independent Complexity Theory since all core computational models could be transformed to one another in polynomial time. Both of these fail when considering a more fine-grained analysis: composing quadratic-time reductions, for example, is no longer quadratic time and transforming between basic computational models might already take longer than solving your quadratic-time problem!

In this light, Fine-Grained Complexity Theory is model-dependent and, further, has a difficult time defining complexity classes.[3] Without a functioning notion of "completeness" for a complexity class, which is most usually attained with reductions that compose, it is especially difficult to define complexity classes that may give evidence of fine-grained *hardness*. Thus, the main structures that have arisen out of Fine-Grained Complexity Theory are not the usual complexity classes defined by a complete problem or by a resource-bounded model of computation, but instead the web of connections between problems itself.

This has made the field of Fine-Grained Complexity Theory inherently *problem-centric*. This is in contrast to the interplay between problems and resources afforded by notions of completeness and complexity classes in Classical Complexity Theory (e.g. the resource-centric lens of considering nondeterminism as a resource for polynomial time computation via the complexity class NP coincides fully with the problem-centric view of considering the efficiency of a problem like SAT since SAT is complete for NP under polynomial time reductions) or the ability to connect general theories of hardness to general applications (e.g. with theories of Cryptography and Randomness). In this work we will reconnect this field to some of the resource-centric claims and the applications studied in Classical Complexity Theory. But, before that, we should consider the life that Fine-Grained Complexity Theory has enjoyed on its own. Despite the field's landscape not being navigable by complexity classes, the web of reductions has revealed some useful patterns and has illuminated some core problems of the field. Out of this picture has emerged some clear and useful islands on the Fine-Grained world.

[2]See [Wil18] for general definitions of fine-grained reductions. We do not need such a general framework in this work to describe our results and prove our theorems.

[3]A fine-grained complexity class is defined in [GIKW17], although this is done in the spirit of Descriptive Complexity Theory where classes are defined in terms of the expressiveness of the logic needed to describe the computational problems in formal language.

2.2 The Main Islands of Fine-Grained Complexity

A large component of what is studied in Fine-Grained Complexity Theory is the gathering and organizing of evidence of fine-grained hardness for practical problems. That is, problems in string processing, computational geometry, dynamic graph problems, etc. that have long hovered at state-of-the-art runtimes of n^2 or n^3 are now being given evidence that these runtimes are in fact the best you could hope for and that, if this is still too slow in practice, heuristics should be used instead of hoping for a better worst-case algorithm. This is achieved by conjecturing some minimal core set of problems to be fine-grained hard and then showing fine-grained reductions from these core problems to the the many practical problems in question: If these conjectures hold, then the hardness of this small set of explanatory problems props up the hardness of many practical problems.

Thus, in the current landscape of Fine-Grained Complexity Theory there are four main islands, each upheld by one of the four key problems: Orthogonal Vectors, $\mathsf{3SUM}$, All-Pairs Shortest Path, and k-CLIQUE. These problems define the "islands" of the field since they partition what is known in the field: There are no known reductions between them but they reduce to many other problems, thus comprising the basis for Hardness within P.

- *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.
- *k-CLIQUE*: Given an unweighted, undirected graph on n vertices, the k-CLIQUE problem is to determine whether there is a k-clique – i.e. whether there is a set of k vertices such that every pair of vertices in that set has an edge between them.

These are each long-studied problems and, after extensive attempts to improve over the (typically trivial) algorithms that are the state-of-the-art for each of these problems, the following popular fine-grained hardness conjectures have been made:

- OV Conjecture: For any $d = \omega(\log n)$, any algorithm for OV_d requires $n^{2-o(1)}$ time.
- $\mathsf{3SUM}$ Conjecture: Any algorithm for $\mathsf{3SUM}$ requires $n^{2-o(1)}$ time.
- APSP Conjecture: Any algorithm for APSP requires $n^{3-o(1)}$ time.
- k-CLIQUE Conjecture: Any algorithm for k-CLIQUE requires $n^{\omega k/3-o(1)}$ time, where ω is the matrix multiplication constant.

Remark 2.2.1. *As Fine-Grained Complexity Theory is model-dependent as discussed earlier, we will adopt the convention of the field so that all our discussion will be in the Word RAM model of computation with $O(\log(n))$-bit words. These conjectures are made in this model. Later, in the technical portions of this work we will also define, conjecture, and justify these to hold when the computation is randomized as well.*

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 (see [Wil15, Wil18] for a more thorough survey of this):

OV Consequences. It has been shown that if the the OV conjecture is true, then many string processing problems relevant to DNA sequencing and data comparison also have hardness bounds (typically sub-quadratic) [AWW14, ABW15b, BI15, 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) (which states that there is no $\epsilon > 0$ such that t-SAT can be solved in time $\widetilde{O}(2^{n(1-\epsilon)})$ for all values of t) would break. Thus, the island defined by the OV conjecture is also supported by the stronger assumption of SETH.

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

APSP Consequences. Further, 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.

k-CLIQUE Consequences. Finally, the fourth most recently emerging island seems to based on explanatory power of the k-CLIQUE problem. The fastest known algorithm for deciding if a graph has a k-CLIQUE (given its adjacency matrix) runs in time $O(n^{\omega k/3})$, and was discovered in 1985 [NP85] for k a multiple of three (for other k different ideas are needed [EG04]). The parameterized version of the famous NP-hard MAX-CLIQUE problem [Kar72], k-CLIQUE is one of the most heavily studied problems in theoretical computer science and is the canonical intractable (W[1]-complete) problem in parameterized complexity and it is known that refuting the k-CLIQUE conjecture deterministically would give a faster exact algorithm for MAX-CUT [Wil05] (see [ABW15a] for a review of the copious evidence of k-CLIQUE's hardness and consequences of its algorithm's exponent being improved). If, on the other hand, the current running time can't be improved and thus the k-CLIQUE conjecture holds, recent work shows that the conjecture implies fine-grained hardness for many practical problems such as parsing languages and RNA folding [ABW15a, BGL17, BDT16, BT17].

These four problems are thus justified as core to Fine-Grained Complexity Theory since their hardness props up the fine-grained hardness of many practical problems, yet remain islands as they are not known to directly connect to each other (indeed, a barrier has been shown for reducing OV to 3SUM or APSP as it would cause a surprising breakthrough in Classical Complexity Theory [CGI+16]). Thus, as we attempt to connect Fine-Grained Complexity Theory to the questions and applications considered in Classical Complexity Theory, we will do so with respect to these four core problems and their conjectures.

2.3 Fine-Grained Easiness: Connecting to Classical Complexity Through Fine-Grained Collapses

Either we win or we win, but we don't know which way we win.

Russell Impagliazzo

There is a precipice between two steep mountains: the city is over the void, bound to the two crests with ropes and chains and catwalks... Suspended over the abyss, the life of [the city's] inhabitants is less uncertain than in other cities. They know the net will last only so long.

Italo Calvino, *Invisible Cities*

A common practice in Complexity Theory is "hedging our bets:" Since so few unconditional results are known in the field and much of our computational beliefs are suspended by intricate webs of implications, we routinely plan for its possible collapse. By revealing which interesting theorems would become unconditionally true if long-held popular conjectures turned out to be *false*, we create "Win-Win" scenarios where either our conjectures and all of their implications hold up *or* they collapse but leave us guaranteed breakthrough theorems in their absence.

Fine-Grained Complexity Theory, since its recent inception, has been no different. Indeed, one of the seminal works of the field shows that if the OV conjecture were false (thus sinking one of Fine-Grained Complexity Theory's four core islands along with all of the fine-grained hardness for string processing problems it would have guaranteed) then we at least have a breakthrough in Classical Complexity Theory as it would imply SETH is false and thus that there are faster algorithms for the famous t-SAT problem [Wil05]. Further, showing that OV or some of the fine-grained problems on its islands have faster algorithms would yield circuit lower bounds [AHWW16, JMV15]. Thus, we see currently existing connections from Fine-Grained Complexity Theory to Classical Complexity Theory in that if fine-grained *easiness* occurs – that is, if the popular conjectures of fine-grained hardness that prop up the field fail – then breakthroughs in Classical Complexity occur.

These types of results serve two purposes. The first purpose is one of hedging bets: these results set up a Win-Win scenario so that either Fine-Grained Hardness and all of its explanatory power for the hardness of practical problems exists *or* the young field and its conjectures collapse yet leave surprising results in Classical Complexity. The second purpose is supportive: the Classical Complexity Theory consequences that would occur if Fine-Grained Complexity *did* collapse are not always ones we believe or are ones that we believe are currently far out of our reach of proving, and thus Fine-Grained Complexity Theory is not likely to collapse any time soon. This second purpose affords us even more explanatory power for understanding why some practical polynomial problems have resisted speed-ups for so long. For example, your attempts to get a slightly faster DNA sequencer have failed not because you just need to be a little more clever in your isolated string processing problem, but because if you succeeded you would have made a far-reaching breakthrough on long-studied questions of Classical Complexity Theory. It further crucially allows us to use our believed conjectures of Classical Complexity to support the core conjectures of Fine-Grained Complexity.

But what about the other direction? Are the only connections achievable such that Classical Complexity is, at worst, opportunistically banking on the fall of Fine-Grained Complexity or, at best, playing a supportive role to it? Prior to the work presented in this thesis, connections between Fine-Grained Complexity Theory and Classical Complexity Theory have all been in one direction such that Classical Complexity supports Fine-Grained Complexity (or, by contrapositive, would benefit from its collapse). But does Fine-Grained Complexity Theory have anything to say about the conceptual and philosophical questions posed by Classical Complexity Theory? What can be said *within* the Fine-Grained world of the ability to launder computational hardness to attain more hardness? Or if this hardness can be used to manufacture pseudorandomness? Or if it can be utilized for Cryptographic purposes? Does the standing of the conjectures of the Fine-Grained world have any bearing on the standing of those in the Classical one?

2.4 Connecting the Fine-Grained and Classical Worlds

Thus far, we have seen that the utility of Fine-Grained Complexity Theory has either been internal, where the field is interesting in its own right by connecting the computational hardness of practical problems, or its utility lies in its collapse, where known connections to

Classical Complexity Theory are predicated on the suspect belief that the core conjectures of the Fine-Grained world are actually false. What can we say if, as is increasingly believed, the foundations of Fine-Grained Complexity hold steady and the field continues to flourish? Is the field isolated to making claims about itself and can only draw connections within its domain of "Hardness within P" with nothing to say about the Classical Complexity Theory that precedes it and, indeed, that it is born out of?

We would like to make clear in this work that Fine-Grained Complexity Theory does not exist in a vacuum (whether or not it is supported externally by Classical Complexity) but instead that the hardness conjectures that it bases its entire field on have far-reaching ramifications to both the deep questions about the nature of computation and the conjectures of Classical Complexity Theory. More concretely, we will address the following fundamental questions of Classical Complexity:

- Hardness Amplification: Can a problem that is hard in the worst-case scenario be turned into one that is hard almost always?
- Cryptography: Can we utilize computational hardness to achieve a computational advantage over adversaries?
- Derandomization: Can we decide the truth to as many questions in a world where we didn't have access to randomness as to one where we did?

The rest of this thesis will devote time to formalizing and addressing these questions from the vantage point of Fine-Grained Complexity Theory.

– In Chapter 3 we will show that the core problems of the Fine-Grained world have fine-grained hardness amplification schemes such that average-case hardness can be achieved *within* P and we will discuss consequences of this.

– Chapter 4 will use the average-case fine-grained hardness to achieve the first Proofs of Work, a core cryptographic primitive, that are guaranteed secure based on worst-case assumptions – indeed, the assumptions will be core conjectures of Fine-Grained Complexity Theory.

– Lastly, Chapter 5 will further use the average-case fine-grained hardness we achieved in Chapter 3 to show that randomized polynomial computations can be significantly derandomized; this not only addresses the philosophical questions of the Classical world within the Fine-Grained world, but shows core Fine-Grained conjectures imply core Classical conjectures (e.g. the OV conjecture will have consequences for the ability to derandomize BPP).

Chapter 3

Average-Case Fine-Grained Hardness

There is a large gap between a problem not being easy and the same problem being difficult. A problem could have no efficient worst-case algorithm but still be solvable for "most" instances, or on instances that arise in practice.

– Russell Impagliazzo [Imp95]

"We take some hard problems and try to make them harder." "You mean easier, right?" "No, harder."

– Andrej Bogdanov being asked by airport security about his research, shortly before being taken for questioning

In this chapter we present our work from [BRSV17a], which introduces functions that can be computed in some fixed polynomial time but are hard *on average* for any algorithm that runs in slightly smaller time, assuming widely-conjectured *worst-case* hardness for problems from the study of fine-grained complexity. Unconditional constructions of such functions are known from before [GGH94], but these have been canonical functions that have not found further use, while our functions are closely related to well-studied problems and have considerable algebraic structure.

We prove our hardness results in each case by showing fine-grained reductions from solving one of three problems – namely, Orthogonal Vectors (OV), 3SUM, and All-Pairs Shortest Paths (APSP) – in the worst case to computing our function correctly on a uniformly random input.[1] The conjectured hardness of OV and 3SUM then gives us functions that

[1]Since the writing of the work [BRSV17a] that is presented here, a similar result for k-CLIQUE has been discovered in our work [CIS18] (which we will present in Chapter 5) and independently in [GR18a].

require $n^{2-o(1)}$ time to compute on average, and that of APSP gives us a function that requires $n^{3-o(1)}$ time. Using the same techniques we also obtain a conditional average-case time hierarchy of functions.

Based on the average-case hardness and structural properties of our functions, we outline the construction of a *Proof of Work* scheme and discuss possible approaches to constructing fine-grained One-Way Functions. We also show how our reductions make conjectures regarding the worst-case hardness of the problems we reduce from (and consequently the Strong Exponential Time Hypothesis) *heuristically falsifiable* in a sense similar to that of [Nao03].

3.1 Introduction

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, Wil18], 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 [BI15], 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 paper 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.

3.1.1 Our Results

We present fine-grained hardness amplification in the form of worst-case–to–average-case fine-grained reductions from three main islands of fine-grained complexity theory (see Chapter 5 for this achieved for a fourth island). We recall these three problems from Section 2.2 here to frame our work, and their relevance is discussed in Section 3.2.

- *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 $\mathcal{F}\mathsf{OV}$, $\mathcal{F}\mathsf{3SUM}$, and $\mathcal{F}\mathsf{ZWT}$ 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, $\mathsf{3SUM}$, and APSP to their respective families of polynomials (where $\mathsf{3SUM}$ reduces also to $\mathcal{F}\mathsf{ZWT}$). Specifically, we show:

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

Further, we conjecture a fourth family of polynomials, $\mathcal{F}\mathsf{TC}$, to also be average-case hard and support this with fine-grained reductions from $\mathsf{3SUM}$, and APSP, and k-SAT. The reduction from k-SAT makes $\mathcal{F}\mathsf{TC}$ 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, while $\mathcal{F}$OV, even with a broken SETH, remains hard unless *all* first-order graph problems become easy since [GIKW17] shows that all such problems reduce to OV.[2]

Our results crucially rely on the fact that the polynomials in $\mathcal{F}$OV, $\mathcal{F}$3SUM, $\mathcal{F}$ZWT, and $\mathcal{F}$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 [Lip89, FF91], or more generally to show local correctability of Reed-Muller codes [GS92].

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. Indeed, this technique is generally thought of in *resource-centric* terms that considers hardness amplification for exponential time complexity classes like E or EXP or for space-bounded classes like PSPACE that have the resources to perform this naïve interpolation to attain a multilinear extension [BFNW93]. The *problem-centric* nature of fine-grained complexity theory, then, steers us to the main insight of this work of taking a very white-box 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 allows 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

[2]Technically this requires *moderate-dimension* OV, for which our results still apply to, and we need to consider the generalization we introduce, $\mathcal{F}\mathsf{OV}^k$, to account for *all* first-order graph properties.

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.

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. In subsequent work [BRSV18], we extend this to prove a *direct product theorem* for the problem of evaluating our polynomials. This is presented in Chapter 4.

We additionally show 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 itself.

Applications. In achieving fine-grained average-case hardness, we pose the application of creating fine-grained cryptography as an open problem but outline a structural barrier to using the most natural methods of achieving them from our results. Despite not constructing fine-grained one-way functions, we show that the structure of our problems can be leveraged and combined with ideas from [Wil16] to create fine-grained cryptographic objects: we outline the construction of a *Proof of Work* scheme. This should be viewed as a *proof of concept* for attaining Proofs of Work from worst-case hardness assumptions since the schemes presented here are very basic and don't offer all the types of security one usually expects from Proofs of Work. In subsequent work [BRSV18], we utilize the significant structure of our fine-grained problems to achieve versatile and secure Proofs of Work. This is presented in Chapter 4.

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, 3SUM, and APSP conjectures falsifiable in a practical sense. Specifically, in Section 3.5.6 we show that *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.

Further applications of this work have been found and explored since the original publication of this work and we discuss some of them now in our Related Work section.

3.1.2 Related Work

Recently, and independently of our work, a sequence of papers [BK16b, GR18b, 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, GR18b, 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, GR18b, 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, GR18b, 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 [GR18b], in fact, identifies a natural class of "locally characterizable sets" that contains problems admitting low-degree polynomials akin to the ones considered here.

Our low-degree multilinear framework has spawned many works in the average-case hardness of fine-grained problems and in delegating computation – e.g. [BBB19, Gol18, GR20, GR18a] – and our posing of Fine-Grained Cryptography[3] has led to work making progress towards Fine-Grained Public-Key [LLW19].

Further, as average-case hardness is a precondition to attaining any sorts of cryptographic objects or to laundering computational hardness into pseudorandomness for the purposes of derandomization, the core results and frameworks of this paper have been used to achieve strong Proofs of Work [BRSV18] and derandomization [CIS18] from worst-case fine-grained assumptions. These are respectively discussed in Chapters 4 and 5 along with works related to them.

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 that wasn't directly inspired by the work presented here. 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. We discuss this further in Section 3.5.6.

[3]Here we mean Fine-Grained Cryptography in the sense of fixed polynomial time bounds, such as the Hardness within P [Wil15] context this dissertation is rooted in. The term "Fine-Grained Cryptography" has been used in different settings to talk about security against restricted circuit classes. This usage will not be addressed in this work.

3.1.3 Organization

For reference, our paper's results are discussed as follows:

- Worst-case–to–average-case fine-grained reductions for the evaluation of our families of polynomials $\mathcal{F}\mathsf{OV}$, $\mathcal{F}\mathsf{3SUM}$, $\mathcal{F}\mathsf{ZWT}$, and $\mathcal{F}\mathsf{TC}$ are in Section 3.3.1, Appendix A.3, Section 3.3.2, and Section 3.3.3, respectively.
- An infinite average-case time hierarchy arising from the assumption of SETH and two other hierarchies from the assumption of the k-SUM conjecture as discussed in Section 3.4.
- Definitions of fine-grained cryptography and one-way functions, a barrier to achieving them from our results, and a possible way to bypass this barrier all in Section 3.5.
- Constructions of Proofs of Work guaranteed by worst-case hardness assumptions in Section 3.5.5.
- Applications of our average-case results and Proofs of Work to the heuristic falsifiability of worst-case assumptions, including SETH, in Section 3.5.6.
- Amplifying hardness of our average-case problems and smooth trade-offs between hardness and running time in Appendix A.4.
- An isolation lemma for the Orthogonal Vectors problem that lets us perform our reductions to polynomials over much smaller fields in Appendix A.5.
- Open problems in average-case fine-grained hardness, fine-grained cryptography, and the heuristic falsifiability of assumptions in practice in Section 3.6.

3.2 Worst-Case Conjectures

We now recall the core problems of fine-grained complexity discussed in Section 2.2 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, Wil18].

(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.)

3.2.1 Main Islands of Fine-Grained Complexity

First we recall the problems of OV, 3SUM, and APSP defined in Section 3.1.1. These problems currently remain the three key problems of fine-grained complexity as they partition what is known in the field. That is, 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:

- 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.

Remark 3.2.1. *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 (as we apply to all of our conjectures for the use of worst-case to average-case reductions).*

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, BI15, 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 Section 3.1.1, SETH implies the OV conjecture.

(We note that the results in this paper 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, was shown to be hard for the class of all first-order graph problems in [GIKW17].)

Similarly, if the 3SUM conjecture is true, problems in computational geometry [GO95] and exact weighted subgraph problems [AL13] are 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.

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, BGL17, BDT16, BT17]. To explore delegating computation, [BK16b, GR18b, 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 [GR18b, Wil16], or they have too large of a degree for our worst-case to average-case reductions [BK16b]. In the original publication of this work, we left open the problem of finding a family of polynomials to represent k-CLIQUE that both are computable in time $n^{\omega k/3}$ and have degree $n^{o(1)}$. This has now been solved in [CIS18], which will be presented in Chapter 5, and independently in [GR18a].

3.2.2 Auxiliary Problems

For these practical connections, any reduction to or from the main island problems has interesting consequences (see [Wil15, Wil18] 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 [GIKW17, AHWW16]. 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 [AHWW16].[4]

3.3 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 3.2 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 [BT06a].

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 [Lip89, FF91]. 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 Appendix A.1 in order to accurately assess the running time of the algorithm involved.

Lemma 3.3.1. *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*

[4]Recent work seems to make make progress towards achieving average-case fine-grained hardness for these sorts of string processing problems using techniques different from the ones used here [GK20], thus bypassing the polynomial method barrier.

is an algorithm A running in time t such that A is an average-case solver. That is,

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

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

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

Remark 3.3.2. *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 3.3.3. *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 $\mathsf{Triangle\text{-}Collection}$, respectively, and then applying the random self-reducibility of low-degree polynomials as just described. A fourth family of polynomials arising from $\mathsf{3SUM}$ can be seen in Appendix A.3. The landscape of these reductions is seen in Figure 3.1.

3.3.1 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.

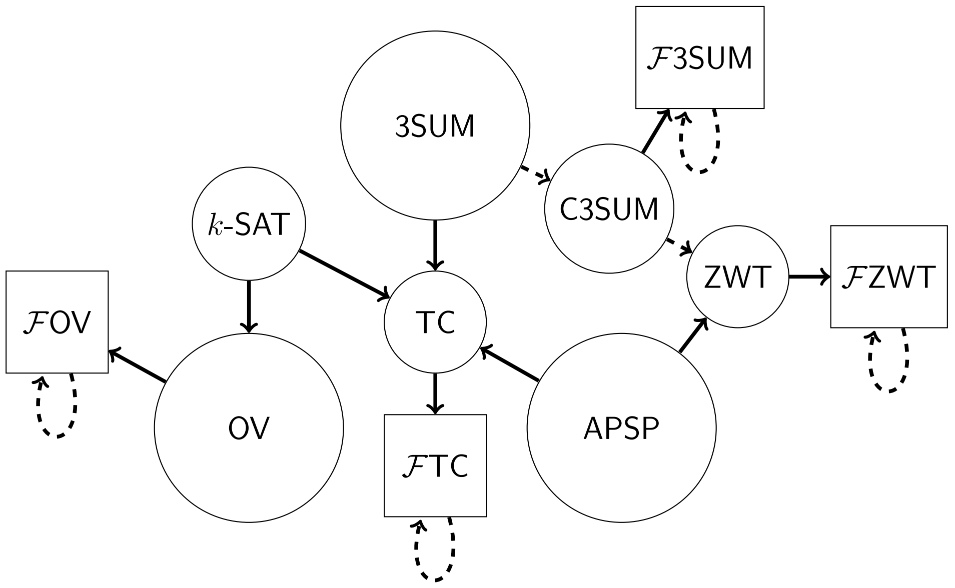

Figure 3.1: Arrows represent (fine-grained) reductions and dashed means they're randomized. Thus a dashed self-loop is a worst-case to average-case self-reduction. Our work introduces $\mathcal{F}$OV, $\mathcal{F}$3SUM, $\mathcal{F}$ZWT, and $\mathcal{F}$TC and the reductions involving them. See Appendix A.3 for C3SUM and $\mathcal{F}$3SUM.

Theorem 3.3.4. *If $\mathcal{F}$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 3.3.5 (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 3.3.5. Once p is known, A can be used along with Lemma 3.3.1 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 3.3.6. *If* OV *requires $n^{2-o(1)}$ time to decide, $\mathcal{F}$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 3.2) and thus an appropriately parametrized variant of $\mathcal{F}$OV is average-case hard for the class of all first-order graph problems as defined in [GIKW17].

3.3.2 3SUM and All-Pairs Shortest Path

Recall from Section 3.2 that both 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 $i, j \in [n]$:

$$f\mathsf{ZWT}_n(E) = \sum_{i,j,k \in [n]} \prod_{\ell \in [d]} \left(1 - \left(s_\ell\left(w_{ij}, w_{jk}\right) - s_\ell\left(\overline{w}_{ik}, 0\ldots01\right)\right)^2\right)$$

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.2. Further, $\overline{w}_{ik}$ represents the set of linear polynomials that toggle all the bits in a boolean valued w_{ij}; so $s_\ell\left(\overline{w}_{ik}, 0\ldots01\right)$ 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 3.3.4.

Theorem 3.3.7. *If $\mathcal{F}\mathsf{ZWT}$ 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 3.3.8. *If ZWT requires $n^{3-o(1)}$ time to decide, $\mathcal{F}\mathsf{ZWT}$ 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 $\mathcal{F}\mathsf{ZWT}$ 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 $\mathsf{3SUM}$ and APSP.

Corollary 3.3.9. *If* either *$\mathsf{3SUM}$ requires $n^{2-o(1)}$ time or APSP requires $n^{3-o(1)}$ time, then $\mathcal{F}\mathsf{ZWT}$ takes $n^{3-o(1)}$ time to compute on average.*

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

We now give our most encompassing worst-case–to–average-case result. Recall from Section 3.2 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 $\mathsf{Triangle\text{-}Collection}^*$. 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 $\mathsf{Triangle\text{-}Collection}^*$. 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 3.3.4.

Theorem 3.3.10. *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 3.3.11. *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 $\mathsf{polylog}(n)$ 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 3.3.12. *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 $\mathcal{F}\mathsf{ZWT}$.

3.4 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 [Wil05] and can be found in [Wil18]. 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 $d = \lceil \log^2(n) \rceil$, $f\mathsf{OV}_n^k : \mathbb{F}_p^{knd} \to \mathbb{F}_p$ is defined as:

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

Exactly as $f\mathsf{OV}_n$ does, when $f\mathsf{OV}_n^k$'s input is a k-OV instance from $\{0,1\}^{knd}$, then $f\mathsf{OV}_n^k(U_1, \ldots, U_k)$ 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 3.3.4.

Theorem 3.4.1. *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 3.4.2. *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, as discussed in Section 3.1.2, 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 Section 3.5.5 and in [BRSV18].

Remark 3.4.3. *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 a polynomial introduced in Appendix A.3,* $\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).*

3.5 Towards Fine-Grained Cryptography

Average-case hardness has frequently found use in cryptography, where it is in fact a necessity – it is almost always required that a cryptographic object be hard to defeat on average for it to be useful. In this light, it is a natural question to ask whether the worst-case to average-case reductions presented here, along with the conjectured hardness of the worst-case problems, can be used to do cryptography.

Of course, since these problems are actually computable in polynomial time, one cannot expect to use them to construct standard cryptographic objects, which have to be secure against all polynomial time adversaries. We hence consider *fine-grained* cryptography, where we only ask for security against adversaries that run in some fixed polynomial time.

A major reason for interest in such notions is that a form of cryptography might be realizable even in Impagliazzo's Pessiland (or Heuristica or even Algorithmica) [Imp95]. That is, even if we live in a world where One-Way Functions or "coarse-grained" average-case hardness do not exist, it may still be possible to construct fine-grained cryptographic objects and salvage practical cryptography. Further, even if we *do* have standard cryptography, it may be the case that fine-grained cryptography can use weaker assumptions as it only needs to be secure against moderately powerful adversaries. Though not done so very extensively and not done from the sort fine-grained complexity in [Wil15, Wil18], such notions have indeed been considered several times before – see for instance [Mer78, Hås87, Mau92, DVV16].

In this section we define a notion of fine-grained cryptography in Section 3.5.1, show a structural barrier to achieving it in Section 3.5.2, and outline as an open problem a possible approach to circumventing this barrier in Section 3.5.3. While this outline has not yet yielded fine-grained one-way functions, we use its techniques in Section 3.5.5 to give a proof of concept that Proofs of Work can be achieved from worst-case fine-grained hardness assumptions. We have since expanded on this in [BRSV18] to achieve versatile and strongly secure PoWs which we will dedicate Chapter 4 to discussing.

3.5.1 Fine-Grained One-Way Functions

To illustrate the kinds of objects, approaches, and difficulties fine-grained cryptography might involve, we consider the case of One-Way Functions (OWFs). 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 2. 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,

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

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).

3.5.2 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 [Wil18]), 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 CNF-UNSAT instances. Interestingly, the impossibility of this was recently conjectured and formalized as NSETH (the Non-deterministic Strong Exponential Time Hypothesis) in [CGI+16] along with the establishment of its barrier status: 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 deterministic certification for APSP instances (the reduction from 3SUM is randomized). So we are left with much weaker "barriers" for $\mathcal{F}$3SUM and $\mathcal{F}$ZWT samplers. Indeed, [CGI+16]'s introduction of NSETH was in a direction opposite to ours: while we explore NSETH to argue that OV is unlikely to have small co-nondeterministic complexity because of its relationship to SAT, they introduce it to argue that 3SUM and APSP are unlikely to have a relationship to SAT (as OV does) by showing them to actually have small co-nondeterministic complexities. Thus they explicitly break both "barriers" for a $\mathcal{F}$3SUM or $\mathcal{F}$ZWT sampler!

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 $\mathcal{F}\mathsf{OV}$.

3.5.3 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, 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 “barrier” of 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 3.3.1 – 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.

We next detail the MA protocol from [Wil16] to pose an open problem of efficiently converting this to an AM protocol, thus breaking the barrier to an erring sampler (and thus a fine-grained OWF) with the further hope such a sampler could be "reverse-engineered" from an AM protocol. In Section 3.5.5, show how this MA protocol is already useful to our framework by constructing Proofs of Work.

3.5.4 An MA Protocol

We briefly describe the MA protocol for $f\mathsf{OV}_n$ achieved in [Wil16], where a more general protocol and framework can be found. Recall that the objective of the prover is to convince the verifier that, for given (x, y), $f\mathsf{OV}_n(x) = y$. We will expand x into $(U, V) \in \mathbb{F}_p^{n\times d} \times \mathbb{F}_p^{n\times d}$, which is the syntax we used when defining $f\mathsf{OV}_n$ as:

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

For a fixed V, we define the supporting polynomial $f\mathsf{OV}_V : \mathbb{F}_p^d \to \mathbb{F}_p$ as:

$$f\mathsf{OV}_V(x_1, \ldots, x_d) = \sum_{j\in[n]} \prod_{\ell\in[d]} (1 - x_\ell v_{j\ell})$$

Now we may write $f\mathsf{OV}_n$ as:

$$f\mathsf{OV}_n(U, V) = \sum_{i\in[n]} f\mathsf{OV}_V(u_{i1}, \ldots, u_{id})$$

Next let $\phi_1, \ldots, \phi_d : \mathbb{F}_p \to \mathbb{F}_p$ be the polynomials of degree $(n-1)$ such that for $i \in [n]$, $\phi_\ell(i) = u_{i\ell}$ (treating i as an element of $\mathbb{F}_p$). Define the polynomial $R_{U,V}(x) = f\mathsf{OV}_V(\phi_1(x), \ldots, \phi_d(x))$. We can then write:

$$f\mathsf{OV}_n(U, V) = \sum_{i\in[n]} R_{U,V}(i)$$

The degree of $R_{U,V}$ is at most $(n-1)d$. If the verifier knew the coefficients of $R_{U,V}$, then it could evaluate it on the points $\{1, \ldots, n\}$ in time $\widetilde{O}(nd)$ (using fast techniques for batch evaluation on univariate polynomials [Fid72]) and thus compute $f\mathsf{OV}_n(U, V)$. This suggests the following protocol. The prover sends over the coefficients of a polynomial R^* that it claims to be $R_{U,V}$.

To verify this claim, the verifier checks whether $R^*(x) = R_{U,V}(x)$ for a random $x \in \mathbb{F}_p$. This can be done because even though the verifier does not know the coefficients of $R_{U,V}$, it can evaluate it at an input x in time $\widetilde{O}(nd)$ by first evaluating all the $\phi_\ell(x)$'s (these polynomials can be found using fast interpolation techniques for univariate polynomials [Hor72]), and then evaluating $f\mathsf{OV}_V$ using these values. By the Schwartz-Zippel lemma, since the field is considerably larger than the degree of R^* and $R_{U,V}$, if these are not the same polynomial, the verifier will catch this with high probability. If this check passes, the verifier uses R^* to compute $f\mathsf{OV}_n(U, V)$.

We pose the open problem of using the ideas of this protocol to construct an AM protocol that does the same. Further we pose the problem of creating an erring sampler from such a protocol, or more generally constructing a fine-grained OWF. We now show two applications of our results. In Section 3.5.5, we show how to use the existing MA scheme to realize the concept of Proofs of Work and in Section 3.5.6 we will use this to introduce the concept of Heuristic Falsifiability. While we focus on $\mathcal{F}\mathsf{OV}$, these applications can similarly be made for $\mathcal{F}\mathsf{3SUM}$, $\mathcal{F}\mathsf{ZWT}$, and $\mathcal{F}\mathsf{TC}$ (and $\mathcal{F}\mathsf{OV}^k$ from Section 3.4).

3.5.5 Proofs of Work

A Proof of Work (PoW) scheme is a means for one party (that we call the *Prover*) to prove to another (called the *Challenger*) that it has expended a certain amount of computational resources. These were introduced by Dwork and Naor [DN92] as a means to deter spam mail and denial-of-service attacks, and have also found use in cryptocurrencies [Nak08]. Here we formulate a simplified definition of a PoW scheme to illustrate how our results could be used in this context. This will be for ease of exposition and as a proof of concept; for a more detailed definition and a more robust and intricate construction, see Chapter 4. A PoW scheme consists of three algorithms:

- $\mathsf{Challenge}(1^n)$ takes a difficulty parameter n and produces a challenge c of this difficulty.
- $\mathsf{Solve}(c)$ solves the challenge c and produces a solution s.
- $\mathsf{Verify}(c, s)$ verifies that s is a valid solution to the challenge c.

The idea is that when the challenger wants the prover to work for a certain amount of time (and prove that it has done so), it runs $\mathsf{Challenge}$ with the appropriate difficulty parameter to generate a challenge c that it sends over. The prover is then required to produce a solution s to the challenge that the challenger verifies using Verify.

Intuitively, our average-case hardness results for $f\mathsf{OV}_n$ says that random inputs, thought of as challenges, must require a certain amount of work for a prover to evaluate $f\mathsf{OV}_n$ on. Then, the MA protocol in Section 3.5.4 gives a quick way for the challenger to verify a solution.

Definition 3. The triple of algorithms $(\mathsf{Challenge}, \mathsf{Solve}, \mathsf{Verify})$ is a Proof of Work (PoW) scheme if the following properties are satisfied:

- $\mathsf{Challenge}$ and Verify are computable in time $s(n)$.
- Solve is computable in time $t(n)$ and $\mathsf{Verify}(c, \mathsf{Solve}(c)) = 1$. $(t(n) > s(n))$
- There are constants $\epsilon, \epsilon' \in [0, 1]$ such that for any A and n satisfying:

$$\Pr_{c \leftarrow \mathsf{Challenge}(1^n)} [\mathsf{Verify}(c, A(c)) = 1] \geq \epsilon$$

  it is the case that A takes time at least $t'(n)$ on a ϵ' fraction of challenges of difficulty n. $(t'(n) > s(n)$. Ideally, $t'(n) \approx t(n)$.)

We want $\mathsf{Challenge}$ and Verify to be efficiently computable so that the challenger does not have to do too much work in this process. We want Solve to be more expensive but not by too much so that the prover can indeed produce a valid solution, albeit with more work than that done by the challenger. The lower bound on the difficulty of producing valid solutions ensures that the prover actually has to spend a certain amount of time on this task and that its production of a valid solution is proof that it has spent this time.

While above we use time as the computational resource of interest as we are working in the Word RAM model, these specifications can be in terms of other resources such as space, circuit size, etc. when working in other models. For different applications, there could also be other properties that might be desirable in such schemes such as resistance to parallelism, non-batchability, etc., but we shall ignore these for now.

The MA protocol for $f\mathsf{OV}_n$ along with our average-case reductions can be used to construct a PoW scheme based on the hardness of OV as follows.

- $\mathsf{Challenge}(1^n)$ samples a random input $(U, V) \in \mathbb{F}_p^{2nd}$ to $f\mathsf{OV}_n$.
- $\mathsf{Solve}((U, V))$ outputs the coefficients of $R_{U,V}$ (as defined in Section 3.5.4).
- $\mathsf{Verify}((U, V), R^*)$ checks that $R^*(x) = R_{U,V}(x)$ for a random $x \in \mathbb{F}_p$.

By the arguments in Sections 3.3.1 and 3.5.4 regarding the verifier in the MA protocol, both $\mathsf{Challenge}$ and Verify above can be computed in $\widetilde{O}(n)$ time. Solve can be computed in $\widetilde{O}(n^2)$ time either by using the explicit expression for $R_{U,V}$, or by evaluating $R_{U,V}$ on sufficiently many points using the ϕ_ℓ's and $f\mathsf{OV}_V$ and interpolating to find its coefficients.

On the other hand, if any algorithm A produces R^* that passes Verify with probability, say, $5/6$ (over the uniform distribution of (U, V)), then by the soundness of Verify it follows that A is actually producing $R_{U,V}$ with almost the same probability. As noted earlier, given the coefficients of $R_{U,V}$, $f\mathsf{OV}_n(U, V)$ can be computed in $\widetilde{O}(nd)$ time. So A can be used to compute $f\mathsf{OV}_n$ correctly on average with an additive $\widetilde{O}(nd)$ overhead.

Corollary 3.3.6 now implies that if OV is hard for sub-quadratic algorithms, then A cannot be sub-quadratic *and* correct on more than a $3/4$ fraction of inputs. This leaves at least a

$\frac{5}{6} - \frac{3}{4} = \frac{1}{12}$ fraction of inputs where A takes $\Omega(n^{2-o(1)})$ time. These numbers can be improved by repetition and by considering the better reductions from Appendix A.4.

Thus, based on the conjectured hardness of OV, the above is a PoW scheme where a party can prove to a challenger that it has run for a certain amount of time where the challenger only has to run for about a square-root of this amount of time. This construction has since been extended and strengthened in our work [BRSV18], which is the subject of Chapter 4.

Remark 3.5.1. *We note that* OV *is non-batchable due to downward self-reducibility. For instance, let* $\ell(n) = O(n^{1-\varepsilon})$ *for some* $\varepsilon > 0$*. If there is a batch evaluation algorithm that can solve* $\ell^2(n)$ *instances of size* $n/\ell(n)$ *in time at most* $O(n^{2-\delta})$ *for some* $\delta > 0$*, then* OV *on an input* (U, V) *can be solved in sub-quadratic time by partitioning the* U *and* V *into* $\ell(n)$ *sets of size* $n/\ell(n)$ *each and running this algorithm on all pairings of these partitions together.*

[BRSV18] has generalized this to show that $f\mathsf{OV}_n$ *(as well as the above PoW) is non-batchable in the average-case. This is done by showing a* direct product theorem *for* $\mathcal{F}\mathsf{OV}$*. That is, multiple instances makes the probability of success drop exponentially, and so our results are amenable to non-trivial hardness amplification techniques. This will be covered in Chapter 4.*

3.5.6 On the Heuristic Falsifiability of Conjectures

The above Proof of Work 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 challenger 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 marriage of average-case and fine-grained complexity that we accomplish: 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 conjecture on large time complexity like SETH could not be feasibly be broken in practice. 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 3.5.2. *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 than 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 only says 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* OV*'s conjecture and thus* SETH*.*

To put this observation in more concrete terms, we consider [Nao03] in which Moni Naor introduced a stronger notion of falsifiable assumptions: 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 might be 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.

3.6 Open Questions

To our knowledge, our work is the first to attain average-case results in the young field of conditional fine-grained complexity as described in [Wil15, Wil18], and there are many ripe questions to ask of average-case fine-grained complexity. Below we reproduce the list of open questions posed in the publication of [BRSV17a], which this chapter is based on. We add a note for partial answers that have been achieved since [BRSV17a]'s initial publication:

1. Can other low-degree polynomials be found for other interesting problems in the fine-grained world to match conjectured worst-case lower bounds on them – e.g. can k-CLIQUE reduce to evaluating a family of polynomials that are both computable in time $n^{\omega k/3}$ and have degree $n^{o(1)}$?

   A polynomial for k-CLIQUE has since been found in both [GR18a] and [CIS18]. Chapter 5 will specifically cover the work of [CIS18] and the explicit construction of k-CLIQUE's polynomial appears in Appendix C.1. Finding a polynomial for k-SUM that is computable in $n^{\lceil k/2 \rceil}$ to match k-SUM's efficiency is still an important open question.

2. Are there samplable distributions over which well-studied problems like 3SUM, OV, etc. are hard on average? Towards this, is it possible to reduce any of our polynomial families back to the problems they came from or to problems further down their reduction chain?

   While it still remains difficult to achieve average-case hardness for the original fine-grained problems we're concerned with themselves, [GK20] recently gives direct product theorems for string processing problems like EDIT-DISTANCE and Longest Common Subsequence and [GR18a, BBB19] achieve average-case hard distributions for the counting version of k-CLIQUE. Achieving average-case hardness for more of the problems from fine-grained complexity and doing so for their conventional decision-problem form may still be very fruitful.

3. Are there any other worst-case to average-case reduction techniques that might be interesting in the fine-grained world?

   As mentioned above, [GK20] gives fine-grained direct product theorems for fine-grained problems as does [BRSV18] which will be covered in Chapter 4. A larger toolkit of worst-case–to–average-case reductions, especially in the fine-grained world, would still be very useful.

4. Classically, generic derandomization from worst-case hardness assumptions proceeds in two steps: Achieving average-case hardness from worst-case hardness, and achieving

strong pseudorandom generators from average-case hardness. In this paper we achieve the first step in a fine-grained way. Can pseduorandom generators be achieved in a fine-grained way from our average-case hardness?

[CIS18] strongly answers this in the affirmative. This work is covered in Chapter 5 and, from core worst-case fined-grained assumptions, improves over the strongest derandomizations possible under uniform assumptions at the time of its publication. There is still a significant amount of room for improvement, however, as discussed in the Open Questions of Section 5.4.

5. 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 $2^{(1-\epsilon)n}$-time verifier)?
6. Can we construct a fine-grained OWF from worst-case hardness assumptions? Is it possible to realize it from an AM protocol for $\mathcal{F}\mathsf{OV}$ as discussed in Section 3.5.3?

   [LLW19] makes progress towards public-key fine-grained Cryptography, although using very different techniques and using average-case fine-grained assumptions. This area still has much room for exploration.
7. Standard OWFs are sufficient for secret-key cryptography as they are equivalent to Pseudorandom Generators and Pseudorandom Functions [HILL99, GGM86]. What similar equivalences hold in the fine-grained world and what fine-grained cryptography can be accomplished from fine-grained OWFs? More generally, what standard cryptographic results translate over to fine-grained cryptography and are there any that hold only in the fine-grained world?
8. Can we heuristically falsify any of the three big worst-case fine-grained conjectures (from Section 3.2)? Are there other ways in which we can develop techniques for practice to influence theory and give concrete and parameterizable theoretical evidence?

Chapter 4

Proofs of Work From Worst-Case Assumptions

While the NP *complete problems show promise for cryptographic use, current understanding of their difficulty includes only worst case analysis. For cryptographic purposes, [average-case] computational costs must be considered.*

– Whit Diffie and Martin Hellman [DH76]

The problem of good cipher design is essentially one of finding difficult problems... This is a rather unusual situation, since one is ordinarily seeking the simple and easily soluble problems in a field.

– Claude Shannon [Sha49]

In this chapter we present our work first published in [BRSV18], which expands on the ideas of Chapter 3 and uses the average-case fine-grained hardness achieved there to construct Proofs of Work (PoWs) whose hardness is based on well-studied *worst-case* assumptions from fine-grained complexity theory. The past chapter, covering the work of [BRSV17a], built PoWs that are based on the Orthogonal Vectors, 3SUM, and All-Pairs Shortest Path problems. These, however, were presented as a 'proof of concept' of provably secure PoWs and did not fully meet the requirements of a conventional PoW: namely, it was not shown that multiple proofs could not be generated faster than generating each individually. We use the considerable *algebraic structure* of these PoWs to prove that this non-amortizability of multiple proofs does in fact hold and further show that the PoWs' structure can be exploited in ways previous heuristic PoWs could not.

This creates full PoWs that are provably hard from worst-case assumptions (previously, PoWs were either only based on heuristic assumptions or on much stronger cryptographic assumptions (Bitansky et al., ITCS '16)) while still retaining significant structure to enable extra properties of our PoWs. Namely, we show that the PoWs of Chapter 3 can be modified to have much faster verification time, can be proved in zero knowledge, and more.

Finally, as our PoWs are based on evaluating low-degree polynomials originating from average-case fine-grained complexity, we prove an *average-case direct sum theorem* for the problem of evaluating these polynomials, which may be of independent interest. For our context, this implies the required non-amortizability of our PoWs.

4.1 Introduction

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.

This work, building on the techniques and the proof of concept of our results in [BRSV17a] as discussed in Chapter 3, addresses 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; this was the missing piece of [BRSV17a] in achieving PoWs according to their usual definition [DN92].

4.1.1 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 which we define in Section 4.1.2, 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 CNF-SAT formulas and thus a major insight to the P vs NP problem.

In general, achieving Cryptography from the assumption that $\mathsf{P} \neq \mathsf{NP}$ (or $\mathsf{NP} \not\subseteq \mathsf{BPP}$ or similar variants) is one of the most fundamental questions of Foundational Cryptography and many structural barriers to answering this questions have been uncovered – e.g. [AGGM06, BT06b, FF91]. In this chapter we show that cryptographic objects – i.e. PoWs – can be achieved from the strengthening of $\mathsf{NP} \not\subseteq \mathsf{BPP}$ to randomized SETH being false. Thus, more fine-grained techniques seem to bypass many of the structural barriers posed to connecting Cryptography and worst-case Complexity Theory.

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, All-Pairs Shortest Path, and k-CLIQUE 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. Heuristic PoWs like SHA-256, however, aren't even known to have natural generalizations or asymptotics much less connections to other computational problems and so a break would simply say that that specific design for that specific input size happened to not be as secure as we thought.

4.1.2 Our Results

In this chapter we introduce PoWs based on the Orthogonal Vectors (OV), 3SUM, and All-Pairs Shortest Path problems, which comprise the central problems of the field of fine-grained complexity theory.[1] Similar PoWs were introduced in [BRSV17a], although these

[1]We cover only three of the four islands in this chapter to keep the chronology of the results clear as this chapter covers [BRSV18] and a polynomial for k-CLIQUE wasn't discovered until later [GR18a, CIS18]. The

failed to prove non-amortizability of these PoWs: that many challenges take proportionally more work, as is required by the definition of PoWs [DN92, BGJ+16]. We show here that the PoWs of [BRSV17a] shown in Chapter 3 can be extended to exploit their considerable algebraic structure to show non-amortizability via a direct sum theorem and, thus, that they are genuine PoWs according to the conventional definition. Further, we show that this structure to can be used to allow for much quicker verification and zero-knowledge PoWs. We also note that our structure plugs into the framework of [BK16b] to obtain distributed PoWs robust to Byzantine failure.

While all of our results and techniques will be analogous for $\mathsf{3SUM}$ and APSP, we will use OV as our running example for our proofs and results statements. Namely, OV (defined in Section 4.2.2) is a well-studied problem that is conjectured to require $n^{2-o(1)}$ time in the *worst-case* [Wil18]. Roughly, we show the following.

Informal Theorem 4.1.1. *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 4.5.

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 $f\mathsf{OV}^k$.

polynomial for k-CLIQUE is discussed in Chapter 5 and it easily fits the framework discussed in this chapter to attain PoWs.

Informal Theorem 4.1.2. *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 $\ell(n) \cdot n^{k-o(1)}$.*

4.1.3 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.

In this chapter we build on the ideas introduced in Section 3.5.5, which covered the work of [BRSV17a]. While [BRSV17a] introduced the PoWs as a proof-of-concept that PoWs can be based on well-studied worst-case assumptions, they did not fully satisfy the definition of a PoW in that the PoWs were not shown to be non-amortizable. That is, it was not proven that many challenges could not be batch-evaluated faster than solving each of them individually. We show here that these PoWs are in fact non-amortizable by proving a direct sum theorem in Section 4.4. Further, the k-OV-based PoWs of [BRSV17a] have verification times of $\widetilde{O}(n^{k/2})$ whereas we show how to achieve verification in time $\widetilde{O}(n)$, which makes the PoWs much more realistic for use. These are both properties that are *expected* of a PoW that were not included in [BRSV17a]. Beyond that, we show that our PoWs can be proved in zero knowledge and note that our PoWs can be distributed across many worker in way that is robust to Byzantine error, both of which are properties seemingly *not achievable* from the current 'structureless' heuristic PoWs that are used.

Provably secure PoWs have been considered before in [BGJ+16] where PoWs are achieved from *cryptographic assumptions* (even stronger than an average-case assumption). 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 the work presented here, Goldreich and Rothblum [GR18a] constructed (implicitly) a PoW protocol based on the worst-case hardness k-CLIQUE: 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. Independently, [CIS18] also found a polynomial for k-CLIQUE and noted its ability to fit the PoW framework presented here.

[LLW19] [GK20]

A previous version of the work presented here appeared under the title Proofs of Useful Work [BRSV17b], where we had presented the same protocol as in this paper as a PoW scheme where the prover's work could be made "useful" by using it to perform independently useful computation. However, it was pointed out to us (by anonymous reviewers) that a naïve construction satisfied our definition of a "Useful PoW."

4.2 Proofs of Work from Worst-Case Assumptions

In this section, we first define Proof of Work (PoW) schemes, and then present our construction of such a scheme based on the hardness of Orthogonal Vectors (OV) and related problems. In Section 4.2.1, we define PoWs; in Section 4.2.2, we introduce OV and related problems; in Section 4.2.3, we describe an interactive proof for these problems that is used in our eventual construction, which is presented in Section 4.2.4. Our PoWs, while similar, will differ from those of [BRSV17a] in that we allow interaction to significantly speed the verification time by exploiting the PoWs' algebraic structure. We will show how to remove interaction in the Random Oracle model in Section 4.5.

4.2.1 Definition

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 4.2.1 (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,

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

  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 4.5.

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. We will now reduce the hardness of our PoW to the hardness of well-studied worst-case problems in fine-grained complexity theory.

4.2.2 Orthogonal Vectors

For ease of readability, we now recall the Orthogonal Vectors (OV) problem and its generalization k-OV whose hardness we use to construct our PoW scheme. The properties possessed by OV that enable this construction are also shared by other well-studied problems mentioned earlier, including $\mathsf{3SUM}$ and APSP as noted in [BRSV17a], and an array of other problems [BK16b, GR18b, 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, $\mathsf{3SUM}$, 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 [GIKW17], and so we can obtain PoWs if *any* problem statable as a first-order graph property is hard.

As in Chapter 3, 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 4.2.5) are known to the reductions.

Remark 4.2.2. *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 exposition's sake.*

Definition 4.2.3 (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, for which many conditional fine-grained hardness results are based on [Wil18], and 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 4.2.4 (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. [BRSV17a], however, defines a related problem that is shown to be average-case hard when assuming the worst-case hardness of k-OV. This problem is that of evaluating the following polynomial:

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 4.2.5 ($\mathcal{F}\mathsf{OV}^k$)**.** 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 4.2.6. *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 4.1). 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.*

4.2.3 Preliminaries

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 subsection; here, we describe the protocol for the latter (Protocol 4.1), 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, $\mathsf{3SUM}$, and APSP in [BRSV17a], 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 4.5. 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 $\phi^s_1,\ldots,\phi^s_d : \mathbb{F}_p \to \mathbb{F}_p$, where ϕ^s_ℓ represents the ℓ^{th} column of U_s – that is, for $i \in [n]$, $\phi^s_\ell(i) = u^s_{i\ell}$. Each ϕ^s_ℓ has degree at most $(n-1)$. $f\mathsf{OV}^k_{n,d,p}$ can now be written as:

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

where q is defined for convenience as:

$$
q(i_1,\dots,i_k) = \prod_{\ell\in[d]} \left(1 - \phi^1_\ell(i_1)\cdots\phi^k_\ell(i_k)\right)
$$

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 $\phi^s_\ell(i_s)$'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 $\alpha_1,\dots,\alpha_{s-1} \in \mathbb{F}_p$, define the following univariate polynomial:

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

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 4.1 below. The completeness and soundness of this interactive proof is then asserted by Theorem 4.2.7, which is proven in Section 4.3.

Theorem 4.2.7. *For any $k \geq 2$, let d and p be as in Definition 4.2.5. Protocol 4.1 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 4.1 is non-interactive when $k = 2$. We then get the following corollary for $\mathcal{F}\mathsf{OV}$.

Corollary 4.2.8. *For $k = 2$, let d and p be as in Definition 4.2.5. Protocol 4.1 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 $\widetilde{O}(n)$.*

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

4.2.4 The PoW Protocol

We now present Protocol 4.2, which we show to be a Proof of Work scheme assuming the hardness of k-OV.

Theorem 4.2.9. *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 4.2 is an (n^k, δ)-Proof of Work scheme for any function $\delta(n) > 1/n^{o(1)}$.*

Remark 4.2.10. *As is, this will be an interactive Proof of Work protocol. In the special case of $k = 2$, Corollary 4.2.8 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 [BRSV17a]. To keep verification time at $\widetilde{O}(n)$, we instead show how to remove interaction in the Random Oracle model in Section 4.5. 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.*

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 4.1 with input $(\boldsymbol{c}, z)$, Solve as prover, and Verify as verifier.
  - Verify accepts iff the verifier in the above instance of Protocol 4.1 accepts.

Protocol 4.2: Proof of Work based on the hardness of k-OV.

Remark 4.2.11. *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 4.3.4 for further details.*

We will use Theorem 4.2.7 to argue for the completeness and soundness of Protocol 4.2. 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 4.2.12. 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 4.2.13, 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 4.2. We prove Theorem 4.2.13 via a direct sum theorem in Appendix B.1, and prove a weaker version for illustrative purposes in Section 4.4.

Theorem 4.2.13. *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 4.2.9 as follows.

Proof of Theorem 4.2.9. We prove that Protocol 4.2 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 4.1, which can be done in $\widetilde{O}(n^k)$ time by Theorem 4.2.7. So in all it takes takes $\widetilde{O}(n^k)$ time.
- Verify runs the verifier in an instance of Protocol 4.1, taking $\widetilde{O}(n)$ time, again by Theorem 4.2.7.

Completeness: This follows immediately from the completeness of Protocol 4.1 as an interactive proof for $\mathcal{F}\mathsf{OV}^k$, as stated in Theorem 4.2.7, 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 4.1 for *all* the instances $\{(\boldsymbol{c}_i, z_i)\}$ with probability at least $\delta(n)$.

By the negligible soundness error of Protocol 4.1 guaranteed by Theorem 4.2.7, 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 4.2.13 says is impossible under our assumptions. So such a Solve^* cannot exist, and this proves the hardness of Protocol 4.2.

We have thus proven all the properties necessary and hence Protocol 4.2 is indeed an (n^k, δ)-Proof of Work under the hypothesised hardness of k-OV for any $\delta(n) > 1/n^{o(1)}$. $\square$

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

In this section, we prove Theorem 4.2.7 (stated in Section 4.2), which is about Protocol 4.1 being a valid interactive proof for proving evaluations of $\mathcal{F}\mathsf{OV}^k$. We use here terminology from the theorem statement and protocol description. Recall the the input to the protocol is $U_1, \dots, U_k \in \mathbb{F}_p^{nd}$ and $y \in \mathbb{F}_p$, and the prover wishes to prove that $y = f\mathsf{OV}^k_{n,d,p}(U_1, \dots, U_k)$.

Completeness. If indeed $y = f\mathsf{OV}^k_{n,d,p}(U_1, \dots, U_k)$, then the prover can make the verifier in the protocol accept by using the polynomials $(q_1, q_{2,\alpha_1}, \dots, q_{k,\alpha_1,\dots,\alpha_k})$ in place of $(q_1^*, q_{2,\alpha_1}^*, \dots, q_{k,\alpha_1,\dots,\alpha_k}^*)$. 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 $y \neq f\mathsf{OV}^k_{n,d,p}(U_1, \dots, U_k)$. 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 4.3.1 (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 $x_1, \dots, x_N \in \mathbb{F}_p$, the set of evaluations $(q(x_1), \dots, q(x_N))$ can be computed in time $O(N \log^3 N \log p)$.*

Lemma 4.3.2 (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 4.3.2. 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 4.3.3. *The coefficients of the polynomial $q_{s,\alpha_1,\dots,\alpha_{s-1}}$ 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 $s, \alpha_1, \dots, \alpha_{s-1}$.
2. For each $\ell \in [d]$, compute the evaluation of ϕ_ℓ^s on nd points, say $\{1, \dots, nd\}$.
    - 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 4.3.1, for a total of $\widetilde{O}(nd^2 \log p)$ for all the ϕ_ℓ^s's.
3. For each setting of $i_{s+1}, \dots, i_k$, compute the evaluations of the polynomial $\rho_{i_{s+1},\dots,i_k}(x)$ $= q(\alpha_1, \dots, \alpha_{s-1}, x, i_{s+1}, \dots, i_k)$, on the points $\{1, \dots, nd\}$.
    - First substitute the constants $\alpha_1, \dots, \alpha_{s-1}, i_{s+1}, \dots, i_k$ into the definition of q.
    - This requires computing, for each $\ell \in [d]$ and $s' \in [k]\backslash\{s\}$, 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 $\rho_{i_{s+1},\dots,i_k}$ pointwise over all settings of $(i_{s+1}, \dots, i_k)$.
    - There are n^{k-s} possible settings of $(i_{s+1}, \dots, i_k)$, 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 $q_{s,\alpha_1,\dots,\alpha_{s-1}}$, 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 4.3.2.

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 4.3.1. 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 4.3.4. *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.*

4.4 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 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, GR18b, 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 4.2.13 on $\mathcal{F}\mathsf{OV}$'s non-batchability (Theorem 4.2.13 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 4.2.12 in Section 4.2.

Theorem 4.4.1. *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 4.4.1 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 4.4.2. $\mathcal{F}$ 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 4.4.3. $\mathcal{F}$ 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 strict notion than the related non-adaptive random self-reducibility as defined in [FF91]. 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 4.4.4. *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 4.4.1 using Lemma 4.4.4, and then prove the lemma itself.

Proof of Theorem 4.4.1. 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 4.4.5. *$\mathcal{F}$ 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 $\{(U_{1j_1}, \ldots, U_{kj_k}) \mid j_i \in [m^{c/k}]\}$. 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 4.4.6. *$\mathcal{F}'$ 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 $k\text{-}\mathsf{OV}(U_1, \ldots, U_k) = f\mathsf{OV}^k_m(U_1, \ldots, U_k)$. 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 $g_{\boldsymbol{x},\boldsymbol{x}_1,\boldsymbol{x}_2}(t) = f\mathsf{OV}^k_m(\boldsymbol{x}+t\boldsymbol{x}_1+t^2\boldsymbol{x}_2)$. 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 $f\mathsf{OV}^k_m$ 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 4.4.4, 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 4.4.4. Given the hypothesised downward reduction (Split_d, Merge_d), robust reduction (Split_r, 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 as 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:

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

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. $\square$

4.5 Removing Interaction

All models are wrong, but some (like Random Oracle Models) are useful.

– MrXOR

In this section we show how to remove the interaction in Protocol 4.2 via the Fiat-Shamir heuristic and thus prove security of our non-interactive PoW in the Random Oracle model.

Remark 4.5.1. *Recent papers have constructed hash functions for which the Fiat-Shamir heuristic will provably succeed when the Random Oracle is replaced with these hash functions [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; [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 4.2.5 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 4.1) 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 4.2, the non-interactive proof scheme for $\mathcal{F}\mathsf{OV}^k$ is described as Protocol 4.3.

Overloading the definition, we now consider Protocol 4.2 as our PoW as before except that we now use the *non-interactive* Protocol 4.3 as the the basis of our Solve and Verify

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(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 $\tau_{k-1}$

Verifier(x, y, τ*):
    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 4.3: A Non-Interactive Proof for $\mathcal{F}\mathsf{OV}^k$

algorithms. The following theorem states that this substitution gives us a *non-interactive* PoW in the Random Oracle model.

Theorem 4.5.2. *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 4.2, when using Protocol 4.3 in place of Protocol 4.1, 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 4.2 are easily seen to follow identically as in the proof of Theorem 4.2.9 in Section 4.2. Hardness also follow identically to the proof of Theorem 4.2.9's hardness except that the proof there required the *soundness* of Protocol 4.1, the interactive proof of $\mathcal{F}\mathsf{OV}^k$ that was previously used to implement Solve and Verify. To complete the proof of Theorem 4.5.2, then, we prove the following lemma that Protocol 4.3 is also sound.

Lemma 4.5.3. *For any $k \geq 2$, if Protocol 4.1 is sound as an interactive proof, then Protocol 4.3 is sound as a non-interactive proof system in the Random Oracle model.*

Proof Sketch. Let P be a cheating prover for the non-interactive proof (Protocol 4.3) 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 4.1) 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.

$\square$

4.6 Zero-Knowledge Proofs of Work

In this section we show that the algebraic structure of the 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 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 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 perhaps elsewhere.

Definition 4.6.1. 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,

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

where $\mathrm{View}_{P,V}(x)$ 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 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 4.2.3

- 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 coeffecients of the challenge response over the subgroup size q to Verifier with the public key $(g, h = g^x)$:

$$\begin{aligned}\mathsf{E}(R^*(\cdot); S(\cdot)) &= \mathsf{E}(mr_0^*; s_0), \dots, \mathsf{E}(mr_{nd-1}^*; s_{nd-1}) \\ &= (g^{s_0}, g^{r_0^*} h^{xs_0}), \dots, (g^{s_{nd-1}}, g^{mr_{nd-1}^*} h^{xs_{nd-1}}).\end{aligned}$$

  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 $y = f\mathsf{OV}_V(\phi_1(z), \dots, \phi_d(z))$ to get g^{my}. Then, homomorphically evaluates $\mathsf{E}(R^*; S)$ on z so that $\mathsf{E}(R^*(z); S(z))$ equals

$$\begin{aligned}&\Big((g^{s_0})(g^{s_1})^z \cdots (g^{s_{nd-1}})^{z^d}, (g^{r_0^*} h^{s_0})(g^{mr_1^*} h^{s_1})^z \cdots (g^{mr_{nd-1}^*} h^{s_{nd-1}})^{z^d}\Big) \\ &= (u_1, u_2)\end{aligned}$$

  Then, Verifier accepts if and only if

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

Recall that the success probability of a subquadratic prover (in the non-zero-knowledge case) does not have negligible success probability.

Remark 4.6.2. *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 4.6.3. *Suppose OV takes n^2 time to decide for all but finitely many input lengths for any $d=\omega(\log n)$ and the DDH the holds in Schnorr groups, then the above protocol is a $\tilde{O}(n)$-simulatable (n^2,δ)-interactive Proof of Work scheme for any function $\delta(n)>1/n^{o(1)}$.*

Proof. 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

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

and

$$\begin{aligned}h^w&=h^{t+cS(z)}\\&=h^t(g^0h^{S(z)})^c\\&=a_2\left((g^{r_0^*}h^{s_0})(g^{mr_1^*}h^{s_1})^z\cdots(g^{mr_{nd-1}^*}h^{s_{nd-1}})^{z^d}g^{-f\mathsf{OV}_V(\phi_1(z),\ldots,\phi_d(z))}\right)^c.\end{aligned}$$

Hardness. Suppose a cheating prover runs in subquadratic time, then by the hardness of Protocol 4.2 with high probability $R^* \not\equiv R_{U,V}$, and so 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,

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

Note that $u_1, u_2/g^{f\mathsf{OV}_V(\phi_1(z),\ldots,\phi_d(z))}$ 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 $\log_g u_1 = \log_h u_2/g^y$. 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 $h^w = a_2(u_2/g^y)^c$. Then,

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

Therefore, because $u_1 \neq u_2/g^y$ there can be at most one c for which a Prover can convince the verifier. Such a c is chosen with negligible probability.

$\tilde{O}(nd)$-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 $a_w = \frac{h^w}{h^{cS(z)}}$.
- Output $((g, h), a_1, a_2, z, c, w)$.

Notice that do to the semantic security of ElGamal, the transcript output is computationally indistinguishable from that of an honest Prover. Moreover, the simulator runs in $\tilde{O}(nd)$ time, the time 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 $\mathsf{polylog}(n)$ each. The verfier takes $\tilde{O}(nd)$ time as well. Note that the homomorphic evaluation requires $O(d \log z^d) = O(d^2 \log z) = \mathsf{polylog}(d)$ exponentiations and $d = \mathsf{polylog}(n)$ multiplications.

$\square$

Chapter 5

Fine-Grained Derandomization: From Problem-Centric to Resource-Centric Complexity

[W]e call a function poly-random if no polynomial-time algorithm...can distinguish a computation during which it receives the true values of the function, from a computation during which it receives the outcome of independent coin flips. Notice the analogy with the Turing Test for intelligence.

– Oded Goldreich, Shafi Goldwasser, and Silvio Micali [GGM84]

Pretend to be good always, and even God will be fooled.

– Kurt Vonnegut, *God Bless You, Mr. Rosewater*

In this chapter we present our work first published in [CIS18], which uses the average-case fine-grained hardness attained in Chapter 3 to show that popular hardness conjectures about problems from the field of fine-grained complexity theory imply structural results for resource-based complexity classes. Namely, we show that if either k-Orthogonal Vectors or k-CLIQUE requires $n^{\epsilon k}$ time, for some constant $\epsilon > 1/2$, to *count* (note that these conjectures are significantly weaker than the usual ones made on these problems) on randomized machines for all but finitely many input lengths, then we have the following derandomizations:

- BPP can be decided in polynomial time *using only n^α random bits* on average over *any* efficient input distribution, for any constant $\alpha > 0$
- BPP can be decided in polynomial time with *no randomness* on average *over the uniform distribution*

As noted in Section 3.6, this answers an open question of [BRSV17a] in the positive of whether derandomization can be achieved from conjectures from fine-grained complexity theory. More strongly, these derandomizations improve over all previous ones achieved from *worst-case uniform* assumptions by succeeding on all but finitely many input lengths, as is wanted for asymptotics. Previously, derandomizations from worst-case uniform assumptions were only know to succeed on infinitely many input lengths. It is specifically the structure and moderate hardness of the k-Orthogonal Vectors and k-CLIQUE problems that makes removing this restriction possible.

Via this uniform derandomization, we connect the problem-centric and resource-centric views of complexity theory by showing that exact hardness assumptions about specific problems like k-CLIQUE imply quantitative and qualitative relationships between randomized and deterministic time. This can be either viewed as a barrier to proving some of the main conjectures of fine-grained complexity theory lest we achieve a major breakthrough in unconditional derandomization or, optimistically, as route to attain such derandomizations by working on very concrete and weak conjectures about specific problems.

5.1 Introduction

Computational complexity can be viewed through two main perspectives: problem-centric or resource-centric. Problem-centric complexity theory asks what resources are required to solve *specific problems*, while resource-centric complexity deals with the relative power of different computational models given different resource budgets such as time, memory, non-determinism, randomness, etc. (see [GI16] for a discussion). Through complete problems, these two perspectives often coincide, so that a resource-centric view acts as a fine proxy for answering questions about the complexity of specific problems – e.g. the complexity of SAT can often stand in for asking about the power of NP since SAT is complete for NP under polynomial time reductions. The rapidly progressing field of fine-grained complexity theory, however, brings attention back to the problem-centric viewpoint, raising fine distinctions even between problems complete for the same complexity class, and making connections between problems at very different levels of complexity. To what extent are these two approaches linked – i.e., to what extent can inferences about the fine-grained complexities of specific problems be made from general assumptions about complexity classes, and vice versa?

Here, we examine such links between the fine-grained complexity of specific problems such as the k-Orthogonal Vectors and k-CLIQUE problems and general results about derandomization of algorithms. Derandomization has been a very fruitful study in complexity theory, with many fascinating connections between lower bounds, showing that problems require large amounts of resources to solve, and upper bounds, showing that classes of probabilistic algorithms can be 'derandomized' by simulating them deterministically in a non-trivial fashion. In particular, the hardness-to-randomness framework shows that in many cases, the existence of any "hard" problem can be used to derandomize classes of algorithms. We reconsider this framework from the fine-grained, problem-centric perspective. We show that replacing a generic hard problem with specific hardness conjectures from fine-grained complexity leads to quantitatively and qualitatively stronger derandomization results than one gets from the analogous assumption about a generic problem. In particular, we show that starting from these assumptions, we can simulate any polynomial-time probabilistic algorithm (on any samplable distribution on inputs with a very small fraction of errors) by a polynomial time probabilistc algorithm that uses only n^α random coins, for any $\alpha > 0$. This type of derandomization previously either assumed the existence of cryptographic One-Way Functions or *exponential non-uniform* hardness of Boolean functions.

Thus, the problem-centric conjectures of fine-grained complexity cannot live in isolation from classical resource-centric consequences about the power of randomness. Viewed another way, our results can be seen as a barrier to proving some of the key hardness assumptions used by fine-grained complexity theory. That is, despite recent progress towards proving hardness for k-Orthogonal Vectors, one of fine-grained complexity's key problems, in restricted models of computation [KW17], doing so for general randomized algorithms would immediately prove *all* problems in BPP are easy on average (over, say, uniformly chosen inputs).

Previous derandomization results in the uniform setting ([IW01, GW02, TV07]) used two properties of the hard problem: random self-reducibility and downward self-reducibility. To obtain our results, we need problems that have stronger, "fine-grained" versions of both (or can be reduced to problems that do). In particular, we need problems where not only can instances of size n be reduced to smaller instances of the same problem, but that these instances are much smaller, of size n^ϵ for $\epsilon < 1$, and that the reduction is "fine-grained", in that any improvement in the time to solve the smaller instances yields a similar improvement in the time to solve the larger ones.

5.1.1 Our Results

We obtain two main theorems about the power of BPP from uniform worst-case assumptions about well-studied problems from the field of fine-grained complexity theory. Namely, we consider the k-Orthogonal Vectors (k-OV) and the k-CLIQUE problems, where we recall Section 2.2's definitions and motivations in Section 5.2.1, and show that (even weaker versions of) popular conjectures on their hardness give two flavors of average-case derandomization that improve over the classical *uniform* derandomizations.

All previous derandomizations from *uniform* assumptions on worst-case hardness only succeed on *infinitely many input lengths*. Our work is the first to use worst-case uniform assumptions to derandomize BPP *for all but finitely many input lengths*, as is wanted for asymptotics. The only other worst-case uniform assumptions known to imply such results are those so strong as to imply cryptographic assumptions or circuit lower bounds, fitting closer to the cryptographic or non-uniform derandomization literature. In contrast, our *uniform derandomizations* are from extremely weak worst-case uniform conjectures on simple, natural, combinatorial problems. Informally, we prove the following.

Informal Theorem 5.1.1. *If k-OV or k-CLIQUE requires $n^{\epsilon k}$ time, for some constant $\epsilon > 1/2$, to* ***count*** *on randomized machines in the worst-case for all but finitely many input lengths, then BPP can be decided in polynomial time* ***using only n^{α} random bits*** *on average over* ***any*** *efficient input distribution, for any constant $\alpha > 0$.*

Randomness can be removed entirely by simply brute-forcing all random bits and taking the majority of the outputs to give the following more familiar full derandomization.

Corollary. *If k-OV or k-CLIQUE requires $n^{\epsilon k}$ time, for some constant $\epsilon > 1/2$, to* ***count*** *on randomized machines in the worst-case for all but finitely many input lengths, then BPP can be decided with* ***no randomness*** *in sub-exponential time on average over* ***any*** *efficient input distribution.*

This conclusion is strictly stronger than the classic uniform derandomizations of [IW01, TV07]. The weakest uniform assumption previously known to imply such a conclusion was from those already strong enough to imply the cryptographic assumption of the existence of One-Way Functions that are hard to invert for polynomial time adversaries [BM84, GKL93, GL89, HILL99, Yao82] or those implying *non-uniform* circuit lower bounds [BFNW93].

Our second main theorem, using techniques from [KvMS12], shows how to remove all randomness *within polynomial time* if the distribution over inputs is uniform. The only stronger derandomization from uniform assumptions were, again, from those already strong enough to imply circuit lower bounds or the cryptographic assumption of the existence of One-Way Permutations that require *exponential* time to invert [BM84, GL89, Yao82].

Informal Theorem 5.1.2. *If k-OV or k-CLIQUE require $n^{\epsilon k}$ time, for some constant $\epsilon > 1/2$, to* ***count*** *on randomized machines in the worst-case for all but finitely many input lengths, then BPP can be decided in polynomial time with* ***no randomness*** *on average over* ***the uniform distribution****.*

These results should be viewed through three main points: First, that we conceptually tie problem-centric conjectures to resource-centric consequences, thus partly reconnecting the two perspectives of complexity theory that separate in the fine-grained world. Secondly, we add to the general derandomization literature by achieving quantitatively and qualitatively stronger derandomization from weak uniform assumptions. Lastly, our results can be seen, pessimistically, as demonstrating a barrier to proving even weak versions of some of fine-grained complexity theory's main conjectures lest we achieve a breakthrough in unconditional derandomization or, optimistically, as providing a path to achieve such general resource-centric results by instead considering extremely weak conjectures on very concrete, simple, and structured combinatorial problems.

5.1.2 Related Work

We now discuss previous connections between problem-centric and resource-centric complexity and previous derandomization results.

Connections Between Problem-Centric and Resource-Centric Complexity. Most connections from problem-centric to resource-centric complexity show that faster algorithms for OV or related problems give circuit lower bounds. For instance, improvements in EDIT-DISTANCE algorithms imply circuit lower bounds [AHWW16] and solving OV faster (and thus CNF-SAT [Wil05]) implies circuit lower bounds [JMV15]. These are all non-uniform results, however, whereas in this paper we are concerned with *machines* and their resource-bounds as opposed to circuits. On the uniform side, [GIKW17] recently showed that the exact complexity of k-Orthogonal Vectors is closely related to the complexity of uniform AC^0, although a connection between more powerful machine models and fine-grained assumptions was still not known until now. Further, most of these results follow from OV being *easy*. Our work shows instead that there are interesting resource-centric consequences if our fine-grained problems are *hard*.

Uniform Derandomization Framework. The uniform derandomization framework was introduced in [IW01], a breakthrough paper that showed the first derandomization from a uniform assumption ($\mathsf{EXP} \neq \mathsf{BPP}$) in the *low-end* setting: a weak assumption gives a *slow* (subexponential-time) deterministic simulation of BPP. This is in contrast to our simulation which retains small amounts of randomness but is *fast* (this is a strictly stronger result as it recovers the [IW01] derandomization as a corollary).

We build on [TV07], which simplifies the proof of [IW01] to prove that $\mathsf{PSPACE} \neq \mathsf{BPP}$ implies a non-trivial deterministic simulation of BPP. The technique of [TV07] carefully arithmetizes the PSPACE-complete problem TQBF and uses this as a hard function in the generator of [IW01]. Our proof substitutes a carefully-arithmetized k-OV problem from [BRSV17a]. Numerous other works study derandomization from uniform assumptions ([Kab01, Lu01, IKW02, GST03, SU09]), but these all focus on assumptions and consequences about *nondeterministic* classes.

All worst-case uniform derandomizations, including [TV07] and [IW01], seem to only be able to achieve simulations of BPP that succeed for infinitely many input lengths because of how their proofs use downward self-reductions. Our is the first work to achieve simulations on *all but finitely many input lengths*, because the k-OV and k-CLIQUE-inspired problems have very parallelizable downward self reductions so that we can reduce to a *single* much smaller input length rather than recurse through a chain of incrementally smaller input lengths in our downward self-reduction.

Since the original publication of the work presented here in [CIS18], a new work shows that conjecturing a randomized variant of the uniform assumption ETH (a weaker version of SETH) achieves derandomization [CRTY19]. The average-case derandomization that they achieve, which is comparable to the first "flavor" of derandomization we will achieve in this chapter, is faster than ours but, as with previous uniform derandomizations, only guarantees success on infinitely many input lengths. We note that using the ideas later sketched in Remark 5.3.9 should allow us to us to relax our assumptions on OV to the point that we can also achieve infinitely-often derandomization from assumptions on randomized variants of ETH, although these will be slower than the ones found in [CRTY19]. Our weak assumptions on problems from fine-grained complexity, however, may independently be true even if ETH fails and further is the only uniform derandomization work that isn't restricted to only guaranteeing successful derandomizations for infinitely many input lengths.

Heuristics by Extracting Randomness From the Input. A separate line of work began when [GW02] introduced the idea of using the *input itself* as a source of randomness to heuristically simulate randomized algorithms over uniformly-distributed inputs. While their assumptions contain oracles and are mostly non-uniform and average-case, they construct an algebraic problem inside P whose worst-case uniform hardness can be used in the framework of [IW01] to get an *infinitely-often* simulation of BPP in polynomial time. Our work differs in that we achieve an *almost-everywhere* simulation, that our assumptions are based on canonical fine-grained problems, and that our assumptions aren't against machines with SAT-oracles. Further, the downward self-reduction of their problem requires an expansion by minors of the determinant and so they cannot also obtain an *almost-everywhere* heuristic using our techniques without placing the determinant in NC^1 (as our modification to [IW01] exploits *embarrassingly parallel* downward self reductions).

The work of [KvMS12], generalizing [Sha11], removes the SAT-oracles needed in the assumptions of [GW02] by showing that the Nisan-Wigderson generator (see [NW94]) remains secure against *non-uniform* adversaries even if the seed is revealed to potential distinguishers. In Section 5.3.2 we will show their arguments can be made *uniform* so we can derandomize from uniform assumptions. Seed-revealing Nisan-Wigderson generators are used in [KvMS12] to obtain polynomial-time heuristics for randomized algorithms, where the uniformly distributed input is used as a seed to the generator. The derandomizations in [KvMS12] are achieved from *non-uniform* assumptions of polynomial *average-case* hardness. From *worst-case uniform* assumptions we achieve the same derandomizations.

5.2 Preliminaries

Here we recall the relevant background given in Section 2.2 from fine-grained complexity theory and motivate our assumptions, present standard tools from derandomization, and give definitions of the peculiarities that arise specifically in derandomization from uniform assumptions such as average-case tractability and infinitely-often qualifiers.

As with the previous chapters, all complexity measures of fine-grained problems will refer to time on a randomized word RAM with $O(\log(n))$-bit word length, as is standard for the fine-grained literature [Wil15, BRSV17a]. Specifically, we will consider two-sided bounded error as in [BRSV17a].

5.2.1 Fine-Grained Hardness

The problem-centric field of fine-grained complexity theory has had impressive success in showing the fixed polynomial time ("fine-grained") hardness of many practical problems by assuming the fine-grained hardness of four "key" well-studied problems, as discussed in Section 2.2. We obtain our results under hardness conjectures about two of these four key problems: the k-Orthogonal Vectors (k-OV) problem and the k-CLIQUE problem. Evidence continues to accumulate that these problems are actually hard. Thus, not only is the connection between problem-centric and resource-centric complexity of independent interest, but the strong derandomizations of this paper are now supported by plausible conjectures about concrete problems.

k-CLIQUE. Denote the matrix multiplication constant by ω. The fastest known algorithm for deciding if a graph has a k-CLIQUE (given its adjacency matrix) runs in time $O(n^{\omega k/3})$, and was discovered in 1985 [NP85] for k a multiple of three (for other k different ideas are needed [EG04]). It is conjectured that no algorithm can improve the exponent to a better constant. The parameterized version of the famous NP-hard MAX-CLIQUE problem [Kar72], k-CLIQUE is one of the most heavily studied problems in theoretical computer science and is the canonical intractable (W[1]-complete) problem in parameterized complexity (see [ABW15a] for a review of the copious evidence of k-CLIQUE's hardness and consequences of its algorithm's exponent being improved). Recent work has shown that conjecturing k-CLIQUE to require $n^{\omega k/3-o(1)}$ time, for k a multiple of three, leads to interesting hardness results for other important problems such as parsing languages and RNA folding [ABW15a, BGL17, BDT16, BT17], and it is known that refuting this conjecture deterministically would give a faster exact algorithm for MAX-CUT [Wil05]. Our results hold under a *much weaker version of the conjecture*:

Definition 5.2.1 (Weak k-CLIQUE Conjecture)**.** There exists an absolute constant $\epsilon_0 > 1/2$ such that, for all $k \in \mathbb{N}$ a multiple of three, any randomized algorithm that *counts* the number of k-CLIQUE's in an n node graph requires $n^{\epsilon_0 k}$ time.

Note that this conjecture gives leeway for the exponent of the k-CLIQUE algorithm to be improved so long as it doesn't get down to $k/2$; even finding a linear time algorithm for Boolean matrix multipliaction ($\omega = 2$) would not contradict this conjecture! Further, even if it is possible to *decide* the k-CLIQUE problem that quickly, this conjecture still holds unless it is possible to *count all of the k-CLIQUE's in that time.* (With a more careful analysis of our techniques to focus on k-CLIQUE we can actually use the even weaker conjecture of $\epsilon_0 > \omega/(\omega + 3)$, as argued in Appendix C.1).

k-Orthogonal Vectors. Although the k-CLIQUE problem is certainly *at least* as important as the k-OV problem, for concreteness we will use the k-OV problem to demonstrate our techniques throughout the paper. Proofs based on hardness of k-CLIQUE follow identically.

Definition 5.2.2 (k-Orthogonal Vectors Problem, k-$\mathsf{OV}_{n,d}$)**.** For an integer $k \geq 2$, the k-$\mathsf{OV}_{n,d}$ problem on vectors of dimension d is to determine, given k sets $(U_1, \dots, U_k)$ of n vectors from $\{0,1\}^d$ each, whether there exist $u_i \in U_i$ for each i such that over $\mathbb{Z}$,

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

If left unspecified, d is to be taken to be $d(n) = \lceil \log^2 n \rceil$.

Definition 5.2.3 (k-Orthogonal Vectors Conjecture)**.** For any $d = \omega(\log n)$, for all $k \geq 2$, any randomized algorithm for the k-$\mathsf{OV}_{n,d}$ problem requires $n^{k-o(1)}$ time.

For $k = 2$ the Orthogonal Vectors conjecture for deterministic algorithms has been extensively studied and is supported by the *Strong Exponential Time Hypothesis* (SETH) [Wil05], which states that there is no $\epsilon > 0$ such that t-SAT can be solved in time $\widetilde{O}(2^{n(1-\epsilon)})$ for all values of t. The natural generalization to k-OV is studied in [BRSV17a, GIKW17] and its deterministic hardness is also supported by SETH. While SETH has been controversial , the deterministic k-OV conjecture seems to be a much weaker assumption and is independently believable and supported: it has been shown that it holds unless *all* first-order graph properties become easy to decide [GIKW17] and the 2-OV conjecture has recently been proven unconditionally when the model of computation is restricted to branching programs [KW17]. This conjecture has also been used to support the hardness of many practical and well-studied fine-grained problems [AWW14, BI15, BK15]. As with k-CLIQUE, our main results will hold *using a much weaker version of the randomized k-OV conjecture* introduced below.

Definition 5.2.4 (Weak k-Orthogonal Vectors Conjecture)**.** For any $d = \omega(\log n)$, there exists an absolute constant $\epsilon_0 > 1/2$ such that, for all $k \geq 2$, any randomized algorithm *counting* the number of k-$\mathsf{OV}_{n,d}$ solutions requires $n^{\epsilon_0 k}$ time.

Remark 5.2.5. *For all of these conjectures we will also consider the strengthened versions that assume that all algorithms running in time less than what is required will fail **on all but finitely many input lengths**, as opposed to only on infinitely many input lengths. For most natural problems, an 'almost-everywhere' assumption like this seems natural. That is, we don't expect that the problem becomes easy for, say, even input sizes and hard on odd input sizes or other degenerate cases like this, but instead believe that the hardness comes from the structure of the problem and will simply grow (as opposed to oscillate) asymptotically.*

For the purposes of derandomization, for a given k, we will use the family of average-case fine-grained hard polynomials introduced in Chapter 3 from [BRSV17a]. We will denote them in this chapter as $\left\{f^k_{n,d,p} : \mathbb{F}_p^{knd} \to \mathbb{F}_p\right\}_{n,d,p \in \mathbb{N}}$ and recall that the variables are grouped into sets of size nd in the form of a matrix $U_i \in \mathbb{F}_p{}^{n \times d}$ where the n rows $u_i \in U_i$ are each collections of d variables:

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

As covered in Chapter 3, the worst-case hardness of evaluating these polynomials was related to the worst-case hardness of k-OV_n in [BRSV17a].

Lemma 5.2.6. *Let p be the smallest prime number larger than n^k and $d = \lceil \log^2(n)\rceil$. If $f^k_{n,d,p}$ can be computed in $O(n^{k/2+c})$ time for some $c > 0$, then k-OV_n can be* ***counted*** *in time $\widetilde{O}(n^{k/2+c})$*

Derandomization from uniform assumptions typically requires two other properties of the assumed hard problem: random self-reducibility and downward self-reducibility. We recall from Chapter 3 that $f^k_{n,d,p}$ satisfies both of these properties. We give a polynomial for k-CLIQUE and show that it also has the necessary properties in Appendix C.1.

Random Self-Reducible. $f^k_{n,d,p}$ is *random self-reducible* by the following classical lemma [Lip89, FF91] (see Chapter 3 or [BRSV17a] for a proof). Note that degree $\log^2 n$ adds negligibly to the random self-reduction time.

Lemma 5.2.7 (Random Self-Reducibility of Polynomials)**.** *If $f : \mathbb{F}_P^N \to \mathbb{F}_P$ is a degree $9 < D < P/12$ polynomial, then there exists a randomized algorithm that takes a circuit $\widehat{C}$ 3/4-approximating f and produces a circuit C exactly computing f, such that the algorithm succeeds with high probability and runs in time poly$(N, D, \log P, |\widehat{C}|)$.*

Downward Self-Reducible. We will show that $f^k_{n,d,p}$ is downward self-reducible in the sense that, if we we have a way to produce an oracle for $f^k_{\sqrt{n},d,p}$, we can quickly compute $f^k_{n,d,p}$ with it. Compare this to downward self-reducibility going from input size n to $n-1$ in previous uniform derandomizations. We exploit our more dramatic shrinkage and parallelism to later give an almost-everywhere derandomization, instead of an infinitely-often one.

Lemma 5.2.8. *If there exists an algorithm A that, on input 1^n, outputs a circuit C computing $f^k_{\sqrt{n},d,p}$, then there exists an algorithm that computes $f^k_{n,d,p}$ in time $O(n^{k/2}|C| + \mathsf{TIME}(A))$.*

Proof. Using A, we print a circuit C computing $f^k_{\sqrt{n},d,p}$ in time $\mathsf{TIME}(A)$. To solve an instance of $f^k_{n,d,p}$, we break up its input as follows.

Intuitively, we break each U_i into $\sqrt{n}$ chunks of $\sqrt{n}$ rows each which partitions the sum of $f^k_{n,d,p}$ into $\sqrt{n}^k$ sub-summands. More formally, for $j \in [\sqrt{n}]$ let $U_i^j \in \mathbb{F}^{\sqrt{n}\times d}$ be the submatrix of U_i consisting of just the $((j-1)\sqrt{n}+1)^{th}$, $((j-1)\sqrt{n}+2)^{th}, \ldots, ((j-1)\sqrt{n}+\sqrt{n})^{th}$ rows.

Now we can feed $U_1^{j_1}, U_2^{j_2}, \ldots, U_k^{j_k}$ as input to C for all $j_1, j_2, \ldots, j_k \in [\sqrt{n}]$ and sum the results that C gives. This will give the correct answer by inspection and makes $\sqrt{n}^k$ calls to C. □

5.2.2 Derandomization

We now define pseudorandom generators (PRGs) in terms of their distinguishers.

Definition 5.2.9 (Distinguishers)**.** A test $T : \{0,1\}^{m^\ell} \to \{0,1\}$ is an ϵ-*distinguisher* against $G : \{0,1\}^m \to \{0,1\}^{m^\ell}$, denoted $T \in \mathsf{DIS}(G, \epsilon)$, if:

$$\left| \Pr_{r \sim \mathcal{U}_{m^\ell}} [T(r)] - \Pr_{z \sim \mathcal{U}_m} [T(G(z))] \right| > \epsilon$$

We also will consider the seemingly weaker object of distinguishers that succeed if they are also given the seed to the PRG. These were studied in [TV07] to relate uniform derandomization to average-case hardness and in [KvMS12] to derandomize over the uniform distribution by *using the random input itself as the seed* to the PRG.

Definition 5.2.10 (Seed-Aware Distinguishers)**.** A test $T : \{0,1\}^m \times \{0,1\}^{m^\ell} \to \{0,1\}$ is an ϵ-*seed-aware distinguisher* against G, denoted $T \in \mathsf{DIS}(G, \epsilon)$, if:

$$\left| \Pr_{x \sim \mathcal{U}_m, r \sim \mathcal{U}_{m^\ell}} [T(x, r)] - \Pr_{x \sim \mathcal{U}_m} [T(x, G(x))] \right| > \epsilon$$

Standard hardness-to-randomness arguments typically derandomize using generators that are based on some 'hard' function by contrapositive: if derandomization fails, then a distinguisher for the generator can be produced. Further, from a distinguisher, we can create a small circuit for the supposedly hard function that the generator was based on. For our purposes, we require an algorithmic version of this argument for derandomization from *uniform* hardness assumptions. More specifically, we will use the following lemma which was originally proved for distinguishers [TV07, IW01] but Lemma 2.9 of [KvMS12] proves that it also holds for seed-aware distinguishers (while the proof of [KvMS12] is non-uniform, it is easy to see that it can be made constructive, in the same way that [IW01] gave a constructive version of [NW94]). Thus, $\mathsf{DIS}(G, \epsilon)$ in the lemma below can be thought to refer to either regular or seed-aware distinguishers (which justifies overloading this notation).

Lemma 5.2.11 (Algorithmic Distinguishers to Predictors ([TV07, IW01]))**.** *For every random self-reducible f, there exists a function G with stretch m bits to m^ℓ bits and a constant c such that*

- *$G(z)$ can be computed in time $(|z|^\ell)^c$, given oracle access to f on inputs of length at most $|z|$*
- *There exists a polynomial-time randomized algorithm A that, with high probability, given as input circuit $D \in \mathsf{DIS}(G, \epsilon)$ for ϵ at least inverse polynomial and an oracle for f, prints a circuit computing f exactly.*

5.2.3 Uniform Derandomization

Previous techniques for derandomizations from worst-case uniform assumptions seemed to have inherent caveats: the derandomization only succeeds on average and, even then, only for infinitely many input lengths. Our results *will remove the infinitely-often caveat* and so, in this section, we pay careful attention to infinitely-often simulation. First, we give the definitions of average-case tractability that arise in uniform derandomization.

Average-Case Tractability. We give standard definitions of average-case tractability (for an extensive survey of these notions, see [BT06a]).

Definition 5.2.12 ($t(n)$-Samplable Ensemble)**.** An ensemble $\mu = \{\mu_n\}$ is $t(n)$-samplable if there is a randomized algorithm A that, on input a number n, outputs a string in $\{0,1\}^*$ and:

- A runs in time at most $t(n)$ on input n, regardless of its internal coin tosses
- for every n and for every $x \in \{0,1\}^*$, $\Pr[A(n) = x] = \mu_n(x)$

With this notion of samplable ensemble we can now consider *heuristic algorithms* that perform well on some language $\mathcal{L} : \{0,1\}^* \to \{0,1\}$ over some μ. The pair $(\mathcal{L}, \mu)$ is a *distributional problem.*

Definition 5.2.13 (Heuristics for Distributional Problems)**.** For $t : \mathbb{N} \to \mathbb{N}$, $\delta : \mathbb{N} \to \mathbb{R}^+$, we say $(\mathcal{L}, \mu) \in \text{Heur}_{\delta(n)}\mathsf{DTIME}[t(n)]$ if there is a time $t(n)$ deterministic algorithm A such that, for all but finitely many n:

$$\Pr_{x \sim \mu_n}[A(x) \neq \mathcal{L}(x)] \leq \delta(n)$$

For a class of languages $\mathcal{C}$ we say that,
$(\mathcal{C}, \mu) \subseteq \text{Heur}_{\delta(n)}\mathsf{DTIME}[t(n)]$ if $(\mathcal{L}, \mu) \in \text{Heur}_{\delta(n)}\mathsf{DTIME}[t(n)]$ for all $\mathcal{L} \in \mathcal{C}$.

As in [BT06a], $\text{Heur}_\delta\mathsf{P}$ is defined as the union over all polynomials p of $\text{Heur}_\delta\mathsf{DTIME}(p(n))$ and $\text{Heur}\mathsf{P}$ is the intersection over all inverse polynomial $\delta(n)$ of $\text{Heur}_\delta\mathsf{P}$. $\text{Heur}\mathsf{SUBEXP}$ is defined similarly where $\mathsf{SUBEXP} = \cap_{\epsilon>0}\mathsf{DTIME}\left[2^{n^\epsilon}\right]$.

In other words, $\text{Heur}\mathsf{P}$ is the class of distributional problems that can be solved in deterministic polynomial time for any inverse polynomial error. Thus, while saying $(\mathcal{L}, \mu) \in \text{Heur}\mathsf{P}$ is not a *worst-case* guarantee on $\mathcal{L}$ being easy, $\text{Heur}\mathsf{P}$ still captures a very strong real-world notion of tractability: $\mathcal{L}$ can be easily decided up to *any* inverse polynomial probability of error over input distribution μ. It is then interesting to see over which input distributions a language can be made tractable over.

Finally, to discuss the *randomness-reduced* simulations we construct, we define BPTIME with a limited number of random coins in the natural way.

Definition 5.2.14 (Randomized Heuristics with Bounded Coins)**.** For $t : \mathbb{N} \to \mathbb{N}$, $\delta : \mathbb{N} \to \mathbb{R}^+$, and coin bound $r : \mathbb{N} \to \mathbb{N}$ we say $(\mathcal{L}, \mu) \in \mathrm{Heur}_{\delta(n)}\mathsf{BPTIME}_{[r(n)]}[t(n)]$ if there is randomized algorithm A running in time $t(n)$ and flipping $r(n)$ coins such that, for all but finitely many n:

$$\Pr_{x\sim\mu_n}\left[\Pr_{r\sim\mathcal{U}_{r(n)}}[A(x,r) \neq \mathcal{L}(x)] > 1/3\right] \leq \delta(n)$$

For example, $\mathrm{Heur}\mathsf{BPP}_{[r(n)]}$ denotes the class of distributional problems that, for every inverse polynomial error, have a polynomial time randomized algorithm using only $r(n)$ random coins.

Infinitely-Often Simulation. As opposed to an algorithm that decides a language (possibly on average) "for all but finitely many n" as in Definition 5.2.13, an infinitely-often (io-) qualifier can be added to any complexity class to specify that an algorithm need only succeed on infinitely many input lengths within the time and error bounds. Thus, to derandomize BPP into io-Heur$\mathsf{P}$ over the uniform distribution is to say that every language in BPP can be simulated on average in polynomial time by an algorithm that is only guaranteed to succeed for infinitely many input lengths. *There is no guarantee on what those input lengths are or how large the gaps could be between them.* This is obviously a very undesirable notion of 'tractability'.

Non-uniform hardness to randomness trade-offs can derandomize almost-everywhere (the desired notion of tractability for asymptotics) by assuming almost-everywhere hardness: that no algorithm works *for all sufficiently large input lengths.* That is, the 'infinitely-often' qualifier on the consequent can be *flipped* across the implication to be an 'almost-everywhere' qualifier on the assumption and vice-versa. Thus, the unrealistic 'infinitely-often' notion of tractability can be dropped by slightly strengthening the assumption to the (as argued in Remark 5.2.5) realistic 'almost-everywhere' hardness. For *non-uniform* derandomizations this is possible.

Starting with [IW01] and the techniques it introduced, all *uniform* derandomizations have been infinitely-often derandomizations *without* being able to flip the io- qualifier to an 'almost-everywhere' assumption. Our work is the first that is able to do this in the uniform derandomization framework, thus removing the 'infinitely-often' qualifier from our derandomizations.

5.3 Fine-Grained Derandomization

We will prove our main derandomization results (Theorems 5.3.4 and 5.3.10) here. Under either the (weak) k-OV or k-CLIQUE conjectures, we derandomize BPP on average, where 'on average' will have two different flavors. Although all techniques apply to k-CLIQUE, for concreteness we will use k-OV throughout this section.

We show in Section 5.3.1 that if we base pseudorandom generators on $f^k_{n,d,p}$, then an algorithm printing distinguishers for this PRG can be used to count k-OV solutions quickly.

We will then show in Section 5.3.2 how to attain these distinguisher-printing algorithms if derandomization *doesn't* work on average (for both flavors of on average). Thus, a failed derandomization using these PRGs refutes the k-OV conjecture (similarly for k-CLIQUE).

More specifically, in Section 5.3.2 we will show that the amount of randomness needed can be shrunk in polynomial time to only n^α random bits for any constant $\alpha > 0$ and still succeed in deciding the language on average *over any efficient distribution on inputs* (which implies a *fully deterministic* sub-exponential derandomization over all efficient distributions). The second flavor of derandomization will be shown in Section 5.3.2, that we can fully derandomize in *polynomial* time on average over the *uniform* distribution. Namely, we will show that if either flavor of these derandomizations fail, we will have an algorithm that prints distinguishers for infinitely many input lengths.

5.3.1 Counting k-OV from Distinguishers

In this section we show that any algorithm producing a distinguisher for $G^{f^k_{m,d,p}}$ (the generator guaranteed to exist from Lemma 5.3.1, using the hard function $f^k_{m,d,p}$) can be used to quickly count k-OV solutions.

First, Lemma 5.3.1 follows immediately by combining the distinguisher to predictor algorithm of Lemma 5.2.11 with the fact that $f^k_{m,d,p}$ is random self-reducible as in Lemma 5.2.7.

Lemma 5.3.1. *There is a randomized algorithm $A^{f^k_{m,d,p}}$ that takes any circuit D that is a distinguisher for $G^{f^k_{m,d,p}}$ and produces a circuit C exactly computing $f^k_{m,d,p}$, such that A succeeds with high probability and runs in time* $\mathsf{poly}(m, d, \log p, |D|)$

As usual, having an oracle to $f^k_{m,d,p}$, the assumed hard function, is not desirable and our algorithms dependence on it will need to be removed. We stray from the techniques of [IW01] and worst-case uniform derandomization in general, however, as we use the fact that our problems are from the fine-grained world, and thus polynomial-time computable, to simply answer our oracle queries by brute force. This difference is in part how we can remove the 'infinitely-often' qualifier that plagues all previous worst-case uniform derandomizations.

As $f^k_{m,d,p}$ is computable in time $O(m^k \mathsf{poly}(d, \log p))$, we get the following theorem *without* an oracle by running the algorithm guaranteed in Lemma 5.3.1 with each oracle call answered by the naïve brute force computation of $f^k_{m,d,p}$.

Lemma 5.3.2. *There is a randomized algorithm B that takes any circuit D that is a distinguisher for $G^{f^k_{m,d,p}}$ and produces a circuit C of size* $\mathsf{poly}(m, d, \log p, |D|)$ *exactly computing $f^k_{m,d,p}$. B succeeds with high probability and runs in time $O(m^k \mathsf{poly}(m, d, \log p, |D|))$.*

Now we show that, if we have an algorithm producing a distinguisher, then we have an algorithm counting k-OV. (In Sections 5.3.2 and 5.3.2 we will show how to attain such uniform distinguisher-printing algorithms if either of our types of derandomization fail.)

Theorem 5.3.3. *Let p be the smallest prime number larger than n^k and $d = \lceil \log^2(n) \rceil$. If there is an algorithm A that, on input 1^n, outputs a distinguisher D of $\mathsf{poly}(n)$ size for $G^{f^k_{\sqrt{n},d,p}}$, then there exists a randomized algorithm counting k-OV_n that runs in time $O(n^{k/2+c} + \mathsf{TIME}(A))$, where c only depends on $|D|$.*

Proof. Using A, we print a distinguisher circuit D for $G^{f^k_{\sqrt{n},d,p}}$. Then, by Lemma 5.3.2, we know there exists a randomized algorithm running in time $O(n^{k/2}\mathsf{poly}(\sqrt{n}, d, \log p, |D|)) = O(n^{k/2+c_1})$ that yields a circuit exactly computing $f^k_{\sqrt{n},d,p}$ of size only $\mathsf{poly}(\sqrt{n}, d, \log p, |D|) = O(n^{c_2})$, where c_1 and c_2 only depend on $|D|$. Thus, by Lemma 5.2.8, there exists an algorithm computing $f^k_{n,d,p}$ in time $O(n^{k/2+c_2} + (n^{k/2+c_1} + \mathsf{TIME}(A))) = O(n^{k/2+c} + \mathsf{TIME}(A))$ for $c = \max\{c_1, c_2\}$. Finally, this gives us an algorithm running in time $\widetilde{O}(n^{k/2+c} + \mathsf{TIME}(A))$ to count k-OV_n by Lemma 5.2.6. □

5.3.2 Printing Distinguishers from Failed Derandomization

We now show that, if either of two (shown in 5.3.2 and 5.3.2 respectively) types of derandomization fail, we can uniformly print the distinguishers needed in Section 5.3.1 and thus count k-OV solutions.

Randomness-Reduced Heuristics Over Any Efficient Distribution

Our first main result in derandomizing BPP is to reduce the amount of randomness required to arbitrarily small quantities, over any efficient distribution of inputs. This simulation trades time for reduced randomness under fine-grained hardness assumptions.

Theorem 5.3.4. *If the weak k-OV conjecture holds almost everywhere, then, for all polynomially samplable ensembles μ and for all constants $\alpha > 0$,*

$$(\mathsf{BPP}, \mu) \subseteq \mathrm{Heur}\mathsf{BPP}_{[n^\alpha]}$$

Thus, for any efficient distribution over inputs that nature might be drawing from and for any inverse polynomial error rate we specify, we can simulate BPP using only n^α random bits for any constant $\alpha > 0$ we want. By brute-forcing over all random bits and taking majority answer over this randomness-reduced computation we can always trivially create a fully deterministic simulation to achieve the following more traditional-looking derandomization result.

Corollary 5.3.5. *If the weak k-OV conjecture holds almost everywhere, then, for all polynomially samplable ensembles μ,*

$$(\mathsf{BPP}, \mu) \subseteq \mathrm{Heur}\mathsf{SUBEXP}$$

Remark 5.3.6. *While we are able to remove the infinitely-often qualifier from our derandomization, note that for each efficient distribution of inputs and each inverse polynomial error rate we are guaranteed to have a derandomizing algorithm, whereas [IW01] is able to achieve a **single** derandomizing algorithm that works for each efficient distribution. Nonetheless, our result is a strictly stronger derandomization as it implies, for instance,* $\mathsf{EXP} \neq \mathsf{BPP}$ *(shown in Section C.2) which is the assumption used to achieve the derandomization of [IW01].*

In contrast to typical full derandomizations which brute-force all seeds to a pseudorandom generator and take majority answer, we now show that choosing a *single random seed* and using the generator's output as our randomness yields randomness-reduced simulations so long as the generator is efficient enough. Typically, the generator is not fast enough for this application; 'quick' complexity-theoretic PRGs are usually given exponential time in their seed length as they construct pseudorandom strings via queries to problems that have *exponential* hardness.

Definition 5.3.7 (Randomness-Reduced Simulations)**.** Let $A : \{0,1\}^N \times \{0,1\}^{N^\ell} \to \{0,1\}$ be a randomized algorithm that uses N^ℓ random bits and let $G : \{0,1\}^{N^\alpha} \to \{0,1\}^{N^\ell}$ be a function.
Then for constant $\alpha > 0$, define the randomness-reduced simulation to be a randomized algorithm $B : \{0,1\}^N \times \{0,1\}^{N^\alpha} \to \{0,1\}$ using only N^α random bits as $B(x,r) = A(x, G(r))$.

We now show that if B does not work as a randomness-reduced heuristic, we can uniformly print a distinguisher for the function G.

Lemma 5.3.8 (Failed Randomness-Reduction to Distinguishers)**.** *Let A, B, and G be as in Definition 5.3.7 such that for language $\mathcal{L} : \{0,1\}^N \to \{0,1\}$,*

$$\Pr_{r \sim \mathcal{U}_{N^\ell}} [A(x,r) \neq \mathcal{L}(x)] \leq 1/10$$

That is, that A as a good randomized algorithm deciding $\mathcal{L}$ for all $x \in \{0,1\}^N$. Yet, also assume that, for μ samplable in time N^{a_1} and $\delta(n) = 1/N^{a_2}$, it holds that

$$\Pr_{x \sim \mu_N} \left[\Pr_{r \sim \mathcal{U}_{N^\alpha}} [B(x,r) \neq \mathcal{L}(x)] > 1/3 \right] \geq \delta(N)$$

That is, B as a (randomness-reduced) randomized algorithm does not decide $\mathcal{L}$ on average over μ.
Then $1^N \mapsto \mathsf{DIS}(G, 1/5)$ is in randomized time N^c $\mathsf{TIME}(G)$ for c depending on a_1 and a_2.

Proof. Assume the antecedents of Lemma 5.3.8. This means that, with probability at least $\delta(N)$, choosing x from μ will result in an x such that

$$\Pr_{r \sim \mathcal{U}_{N^\alpha}} [B(x,r) \neq \mathcal{L}(x)] > 1/3$$

If we find an $x' \in \{0,1\}^N$ that induces this "bad" performance of B on $\mathcal{L}$, the test $T(r) = A(x', r)$ will be in $\mathsf{DIS}(G, 1/5)$. That is, since x' makes B perform poorly while A still performs well on *all* x's and since $B(x', z) = A(x', G(z))$, the distinguishing gap of T is

$$\left| \Pr_{r \sim \mathcal{U}_{N^\ell}}[T(r)] - \Pr_{z \sim \mathcal{U}_{N^\alpha}}[T(G(z))] \right| = \left| \Pr_{r \sim \mathcal{U}_{N^\ell}}[A(x', r)] - \Pr_{z \sim \mathcal{U}_{N^\alpha}}[B(x', G(z))] \right|$$

$$> |1/10 - 1/3| > 1/5$$

To find such an x', we simply sample-and-test as in [IW01]: sample $\widetilde{O}(1/\delta(N))$ possible $x_i \in \{0,1\}^N$'s from μ and, for each of them, define the test $T_i(r) = A(x_i, r)$. For each T_i, in polynomial time we can estimate the fraction of $r \in \{0,1\}^{N^\ell}$ that T_i accepts on and, by making calls to G, we can estimate the fraction of $G(z)$ for $z \in \{0,1\}^{N^\alpha}$ that T_i accepts on in polynomial time times $\mathsf{TIME}(G)$. Thus we can estimate the distinguishing gap for each T_i.

With high probability one of these T_i is a 1/5-distinguisher for G and our estimation of its distinguishing gap reveals it. Print this circuit. □

Randomness-Reduced Simulations from k-OV. To finish defining a randomness-reduced simulation, we need to use a specific pseudorandom generator G that, for input length N, stretches N^α coins to N^ℓ. Thus, consider the family of simulations B_k using the standard generators $G^{f^k_{\sqrt{n},d,p}}$ of Lemma 5.2.11 that map $\sqrt{n}^s$ bits to $\sqrt{n}^b$ bits, for some fixed s and any b we choose, using $f^k_{\sqrt{n},d,p}$ as our hard function, for $d = \log^2 n$ and p the smallest prime number larger than n^k. Set $b = s\ell/\alpha$ and $\sqrt{n} = N^{\alpha/s}$. Note that $\mathsf{TIME}(G^{f^k_{\sqrt{n},d,p}}) = \mathsf{poly}(N)\ n^{k/2} = \mathsf{poly}(N)$ by naïvely evaluating $f^k_{\sqrt{n},d,p}$ at each oracle call, giving an efficient randomness-reduced simulation. Further, since $N = \mathsf{poly}(n)$, $\mathsf{TIME}(G^{f^k_{\sqrt{n},d,p}})$ also equals $n^{k/2+c}$ for some constant c not depending on k (this will be useful in quickly counting k-OV_n using downward self-reducibility in the following proof). Thus, given an N^ℓ-coin machine A, we have the N^α-coin machine $B_k(x, r) = A\left(x, G^{f^k_{\sqrt{n},d,p}}(r)\right)$.

We now prove our main Theorem 5.3.4 using this simulation and the above lemma.

Proof of Theorem 5.3.4. We proceed by contradiction. Assume that the weak k-OV_n conjecture holds for all but finitely many input lengths, where $\epsilon_0 = 1/2 + \gamma$ for some constant $\gamma > 0$, but that there exists $\mathcal{L} \in \mathsf{BPP}$, a polynomially samplable distribution μ, constant α, and an inverse polynomial function $\delta(N)$ such that *any* polynomial-time randomness-reduced algorithm with coin bound N^α fails in deciding $\mathcal{L}$ on average over μ within $\delta(N)$ error for infinitely many input lengths N.

Since $\mathcal{L} \in \mathsf{BPP}$ there is a randomized algorithm A deciding $\mathcal{L}$ with probability of error at most 1/10 over its randomness yet, since *any* polynomial-time randomness-reduced algorithm fails to decide $\mathcal{L}$ on average, B_k, the randomness-reduced simulation of A described above, fails on average infinitely often, for *any* constant k. Thus, the antecedents of Lemma 5.3.8

are satisfied and we can uniformly print $D \in \mathsf{DIS}(G^{f^k_{\sqrt{n},d,p}}, 1/5)$ in time n^{c_1} $\mathsf{TIME}\left(G^{f^k_{\sqrt{n},d,p}}\right) = n^{c_1}\ n^{c_2} n^{k/2}$.

This uniform printing of D allows us to apply Theorem 5.3.3 to count k-OV_n in time $O(n^{k/2+c_3} + n^{k/2+c_1+c_2}) = O(n^{k/2+c}) = O(n^{\left(\frac{1}{2}+\frac{c}{k}\right)k})$ for *any* k, where $c = \max\{c_1 + c_2, c_3\}$. Setting k such that $\frac{c}{k} < \gamma$ yields our contradictions: on the infinitely many input lengths where B_k fails to derandomize $\mathcal{L}$, the algorithm counts k-OV faster than $n^{\epsilon_0 k}$ time. □

Remark 5.3.9. *Note the bottleneck of why our "weak" conjectures require $\epsilon_0 > 1/2$ is* not *because of our use of the technique of [BRSV17a] of spending time $\widetilde{O}(n^{k/2+c})$ for any $c > 0$ to find a prime for $f^k_{n,d,p}$; indeed, since choosing random primes is much faster than finding the single large prime, we could instead use the randomness in our contrapositive derandomization arguments to choose many random primes so that an evaluation of the polynomial over each prime will be able reconstruct the true count via the Chinese Remainder Theorem. The bottleneck is then actually because of the* downward self-reducibility, *as seen in the above proof, where* $\max\{n^{\epsilon_0 k+c_3}, n^{(1-\epsilon_0)k+c_1+c_2}\}$ *must occur and so $\epsilon_0 = 1/2$ minimizes the overall runtime of the reduction (where ϵ_0 can be set to the desired degree of severity of the downward self-reduction). Using a downward self-reduction of n to $n-1$ as is done in [IW01], however, is possible and would allow us to weaken our assumption all the way to just requiring $\epsilon > 0$, however we would lose our the key ability of our massive downward self-reduction to get rid of the* io- *qualifier in our derandomization.*

Fast Heuristics for BPP Over the Uniform Distribution

Here we present our second flavor of derandomization: a fully deterministic heuristic for BPP when inputs are sampled according to the uniform distribution.

Theorem 5.3.10. *If the weak k-OV conjecture holds almost everywhere, then*

$$(\mathsf{BPP}, \mathcal{U}) \subseteq \text{HeurP}$$

This strictly improves previous uniform derandomizations over the uniform distribution. Specifically, [GW02] can be seen to achieve our derandomization identically from a worst-case uniform assumption if combined with techniques from [KvMS12] *except only on infinitely many input lengths.*

We proceed by showing that if a PRG fails to give a good heuristic for BPP over the uniform distribution, a seed-aware distinguisher for the PRG can be produced uniformly and efficiently, which can then be used to count k-OV solutions quickly using Theorem 5.3.3.

Definition 5.3.11 (Input-As-Seed Heuristics)**.** Let $A : \{0,1\}^N \times \{0,1\}^{N^\ell} \to \{0,1\}$ be a polynomial-time randomized algorithm using N^ℓ random bits. Let $G : \{0,1\}^N \to \{0,1\}^{N^\ell}$ be a deterministic function. Define the heuristic $B : \{0,1\}^N \to \{0,1\}$ that uses its input as G's seed as $B(x) = A(x, G(x))$.

Now recall the Main Lemma of [KvMS12] giving the consequences of failed heuristics in the *non-uniform* setting:

Lemma 5.3.12 (Main Lemma of [KvMS12])**.** *Let $A : \{0,1\}^N \times \{0,1\}^{N^\ell} \to \{0,1\}$ and $\mathcal{L} : \{0,1\}^N \to \{0,1\}$ be functions such that*

$$\Pr_{x \sim \mathcal{U}_N, r \sim \mathcal{U}_{N^\ell}} [A(x,r) \neq \mathcal{L}(x)] \leq \rho$$

*Let B be the seed-as-input heuristic for A using function G. Then, if B does **not** succeed on a $(3\rho + \epsilon)$ fraction of the inputs of a given length, there **exists** an $r' \in \{0,1\}^{N^\ell}$ such that the test $T_{r'}(x,r) = A(x,r) \oplus A(x,r')$ is in $\mathsf{DIS}(G,\epsilon)$.*

The proof of the above lemma uses *non-uniformity* to obtain a good r' for distinguishing, but we can instead *uniformly* obtain good strings r' via a sample-and-test procedure. There is some loss in the accuracy of the resulting simulation, but this can be made an arbitrarily small inverse polynomial via standard error reduction.

Intuitively, if B is a *bad* heuristic for $\mathcal{L}$, then we could use $B(x) = A(x, G(x))$ as a seed-aware distinguisher for G by comparing $B(x)$ to $\mathcal{L}(x)$. Unfortunately we cannot afford to print distinguishers with $\mathcal{L}$-oracles. But since we are guaranteed that A is a *good* heuristic for $\mathcal{L}$, we can obtain a deterministic circuit close to $\mathcal{L}$ from A, by fixing a string of good random bits r'. If we compare $B(x)$ and the fixed-coin algorithm $A(x,r')$, they will also tend to disagree, giving the necessary distinguishing gap. We can find and fix a good r' uniformly by showing that they are dense and then sampling and testing for goodness. Formally:

Lemma 5.3.13 (Failed Heuristics to Distinguishers)**.** *Let A, $\mathcal{L}$, G, and B be as in Lemma 5.3.12 above. Then, if B does **not** succeed on a $(5\rho + \epsilon)$ fraction of the inputs of a given length, the map $1^N \mapsto \mathsf{DIS}(G,\epsilon)$ is uniform and in randomized polynomial time, for infinitely many N.*

Proof. By the assumption that A succeeds on a ρ fraction of input-coin pairs, we know that *many* fixings of the coins of A produce an algorithm close to $\mathcal{L}$. We can obtain in polynomial time and with high probability a string $r' \in \{0,1\}^{N^\ell}$ such that

$$\Pr_{x \sim \mathcal{U}_N} [A(x,r') \neq \mathcal{L}(x)] \leq 2\rho$$

by repeatedly sampling and testing such fixings of A's coins. This is the first of several "constructive averaging arguments" used in this proof. As above, define the tests,

$$T_{r'}(x,r) = A(x,r) \oplus A(x,r')$$

which output '1' when, for input x, A on fixed coins r' disagrees with A run on input coins r. To output a distinguisher, we simply find an appropriate string r' and then print the circuit $T_{r'}$. This only takes polynomial time. To show that with high probability $T_{r'} \in \mathsf{DIS}(G,\epsilon)$, consider two cases.

1. $T(x, G(x))$: We have by definition of $B = A(x, G(x))$ and the assumption that B is *not* a good heuristic that $A(x, G(x))$ will tend to disagree with $\mathcal{L}$. So $A(x, G(x))$ will also tend to disagree with $A(x, r')$. Thus, the test will be biased towards printing 1 when run with generator output.

2. $T(x, \mathcal{U}_{N^\ell})$: The algorithm $A(x, \mathcal{U}_{N^\ell})$ will tend to agree with $A(x, r')$ by our constructive averaging above. So the test will tend to output 0 when run with true random bits.

These cases correspond to the first and second terms of the distinguishing gap equation (Definition 5.2.10), which we substitute the test into below:

$$\left| \Pr_{x \sim \mathcal{U}_N} [A(x, G(x)) \neq A(x, r')] - \Pr_{x \sim \mathcal{U}_N, R \sim \mathcal{U}_{N^\ell}} [A(x, r) \neq A(x, r')] \right|$$

An upper bound on term 2 is straightforward by union bound and Fréchet inequality: it is at most 3ρ. For term 1, observe that we have these bounds on the relative Hamming distance from $A(x, G(x))$ to $\mathcal{L}$, and from $A(x, r')$ to $\mathcal{L}$:

$$\begin{aligned} d(A(x, G(x)),\ \mathcal{L}) &\geq 5\rho + \epsilon && \text{by assumption on B} \\ d(A(x, r'),\ \mathcal{L}) &\leq 2\rho && \text{by constructive averaging} \end{aligned}$$

So, by triangle inequality, we obtain a lower bound of $3\rho + \epsilon$ on the first term of the distinguisher equation. Therefore, if our constructive averaging succeeds in finding a good r', we have $T_{r'} \in \mathsf{DIS}(G, \epsilon)$. The time to print $T_{r'}$ is just the polynomial time to find a good r' by repeatedly sampling and running A to test, plus the time to print A as a circuit. Thus the *size* of the distinguisher printed is only $\widetilde{O}(\mathsf{TIME}(A))$ — it does not depend on the runtime of the constructive averaging argument. □

Fully Deterministic Heuristics from k-OV. Here we specify a family of heuristics B_k, by specifying the generator G, that stretches a seed of length N to N^ℓ, as the generators $G^{f^k_{\sqrt{n},d,p}}$ of Lemma 5.2.11. These map $\sqrt{n}^s$ bits to $\sqrt{n}^b$ bits, for some fixed s and any b we choose, using $f^k_{\sqrt{n},d,p}$, for $d = \log^2 n$ and p the smallest prime number larger than n^k. Set $b = s\ell$ and $\sqrt{n} = N^{1/s}$. All comments about the runtime of the randomness-reduced heuristic in Section 5.3.2 also apply to this fully deterministic heuristic. Thus, given an N^ℓ-coin machine A, we have the *deterministic* machine $B_k(x) = A\left(x, G^{f^k_{\sqrt{n},d,p}}(x)\right)$.

We now prove our main Theorem 5.3.10 using this simulation and the above lemma.

Proof of Theorem 5.3.10. We proceed by contradiction. Assume that the weak k-OV_n conjecture holds for all but finitely many input lengths, where $\epsilon_0 = 1/2 + \gamma$ for some constant $\gamma > 0$, but that there exists $\mathcal{L} \in \mathsf{BPP}$ and an inverse polynomial function $\delta(N)$ such that *any* polynomial-time deterministic algorithm fails in deciding $\mathcal{L}$ on average over μ within $\delta(N)$ error for infinitely many input lengths N.

Namely, since $\mathcal{L} \in \mathsf{BPP}$ there is a randomized algorithm A deciding $\mathcal{L}$ with probability of failing over its coins at most $\rho(N)$ for any inverse polynomial (by standard error reduction), yet it must be the case that our B_k simulation of A described above fails on average infinitely often as well, for *any* constant k. Thus, choose inverse polynomial $\epsilon(N)$ and $\rho(N)$ such that $5\rho + \epsilon \leq \delta(N)$ and this failure satisfies the preconditions of Lemma 5.3.13 and thus we can uniformly print $D \in \mathsf{DIS}(G^{f^k_{\sqrt{n},d,p}}, \epsilon)$ in time $n^{k/2+c_1}$.

Again this allows us to apply Theorem 5.3.3, which counts k-OV in time $O(n^{k/2+c_2} + n^{k/2+c_1}) = O(n^{\left(\frac{1}{2}+\frac{c}{k}\right)k})$ for *any* k, where $c = \max\{c_1, c_2\}$. Setting k such that $\frac{c}{k} < \gamma$ yields our contradiction: on the infinitely many input lengths where B fails to derandomize $\mathcal{L}$, the algorithm counts k-OV_n faster than $n^{\epsilon_0 k}$ time. □

5.4 Open Questions

- We derandomize under hardness conjectures about two of four 'key' problems in fine-grained complexity: k-OV and k-CLIQUE. What about k-SUM and APSP? APSP doesn't seem to have a natural hierarchy and so doesn't fit our framework (although it does reduce to ZERO-$\mathsf{TRIANGLE}$ which generalizes to ZERO-k-CLIQUE and should easily work in our framework using polynomials similar to those in [BRSV17a]). k-SUM however is actually computable in $O(n^{\lceil k/2 \rceil})$ time and so our downward self-reducibility techniques are not fast enough to break this conjecture in the contrapositive. The ideas mentioned in Remark 5.3.9 would allow us to derandomize from k-SUM, but these would be io- derandomizations. The clearest path we see to getting derandomization without reintroducing the io- qualifier is to find a polynomial for k-SUM that is also computable in $\widetilde{O}(n^{\lceil k/2 \rceil})$ time (unlike the one discussed in Chapter 3).

- Our derandomizations hold under (randomized) SETH, since SETH implies the k-OV conjecture. Can a better derandomization be obtained directly from SETH, the stronger assumption? A stumbling block here is the random self-reduction, an ingredient in all known uniform derandomization techniques: If t-SAT has a straightforward and efficient random-self-reduction, PH collapses [FF93, BT06b]. So derandomizing from SETH directly could require new ideas, or a strange random self-reduction. An *inefficient* random self-reduction for t-SAT shouldn't collapse PH except to say that t-SAT has a mildly exponential MA proof which is already known to be true [Wil16], although most random self-reductions we know are through arithmetization which seems to always have 'low' degree to the point that such a polynomial would still collapse PH. A recent work achieves *nearly* polynomial derandomization from the weaker assumption of (randomized) ETH [CRTY19]. Can full polynomial derandomizations be achieved from SETH?

- Is a strong "derandomization to hardness" converse possible for these heuristic simulations of BPP? In appendix C.2, we show a weak converse: our simulation is impossible without separting $\mathsf{DTIME}[n^{\omega(1)}]$ from BPP. But this is a very different statement from the k-OV or k-CLIQUE conjectures. In [KvMS12], they show that herusitic simulations of BPP with inverse-subexponential error rates imply circuit lower bounds, by generalizing techniques of [KI04]. Do the efficient inverse-polynomial error heuristics we obtain imply any circuit lower bounds?

Bibliography

[Abb17] Amir Abboud. Personal communication, 2017.

[ABW15a] Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. If the current clique algorithms are optimal, so is valiant's parser. In Venkatesan Guruswami, editor, *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 98–117. IEEE Computer Society, 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 Jon M. Kleinberg, editor, *Proceedings of the 38th Annual ACM Symposium on Theory of Computing, Seattle, WA, USA, May 21-23, 2006*, pages 701–710. ACM, 2006.

[AHWW16] 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. In Daniel Wichs and Yishay Mansour, editors, *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 375–388. ACM, 2016.

[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, 2013.

[AWW14] Amir Abboud, Virginia Vassilevska Williams, and Oren Weimann. Consequences of faster alignment of sequences. In Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias, editors, *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I*, volume 8572 of *Lecture Notes in Computer Science*, 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 *Proceedings of the 17th Annual ACM Symposium on Theory of Computing, May 6-8, 1985, Providence, Rhode Island, USA*, pages 421–429, 1985.

[BBB19] Enric Boix-Adserà, Matthew Brennan, and Guy Bresler. The average-case complexity of counting cliques in erdős-rényi hypergraphs. In David Zuckerman, editor, *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, Maryland, USA, November 9-12, 2019*, pages 1256–1280. IEEE Computer Society, 2019.

[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. In Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, volume 55 of *LIPIcs*, pages 81:1–81:13. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016.

[BFNW93] László Babai, Lance Fortnow, Noam Nisan, and Avi Wigderson. BPP has subexponential time simulations unless EXPTIME has publishable proofs. *Computational Complexity*, 3:307–318, 1993.

[BGJ$^+$16] Nir Bitansky, Shafi Goldwasser, Abhishek Jain, Omer Paneth, Vinod Vaikuntanathan, and Brent Waters. Time-lock puzzles from randomized encodings. In Madhu Sudan, editor, *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science, Cambridge, MA, USA, January 14-16, 2016*, pages 345–356. ACM, 2016.

[BGL17] Karl Bringmann, Allan Grønlund, and Kasper Green Larsen. A dichotomy for regular expression membership testing. In Chris Umans, editor, *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 307–318. IEEE Computer Society, 2017.

[BHvM09] Armin Biere, Marijn Heule, and Hans van Maaren. *Handbook of satisfiability*, volume 185. ios press, 2009.

[BI15] Arturs Backurs and Piotr Indyk. Edit distance cannot be computed in strongly subquadratic time (unless SETH is false). In Rocco A. Servedio and Ronitt Rubinfeld, editors, *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 51–58. ACM, 2015.

[BK15] Karl Bringmann and Marvin Künnemann. Quadratic conditional lower bounds for string problems and dynamic time warping. In Venkatesan Guruswami, editor, *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 79–97. IEEE Computer Society, 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.

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

[BRSV17a] 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.

[BRSV17b] Marshall Ball, Alon Rosen, Manuel Sabin, and Prashant Nalini Vasudevan. Proofs of useful work. *IACR Cryptology ePrint Archive*, 2017:203, 2017.

[BRSV18] Marshall Ball, Alon Rosen, Manuel Sabin, and Prashant Nalini Vasudevan. Proofs of work from worst-case assumptions. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part I*, volume 10991 of *Lecture Notes in Computer Science*, pages 789–819. Springer, 2018.

[BT06a] Andrej Bogdanov and Luca Trevisan. Average-case complexity. *Foundations and Trends in Theoretical Computer Science*, 2(1), 2006.

[BT06b] Andrej Bogdanov and Luca Trevisan. On worst-case to average-case reductions for NP problems. *SIAM J. Comput.*, 36(4):1119–1159, 2006.

[BT17] Arturs Backurs and Christos Tzamos. Improving viterbi is hard: Better runtimes imply faster clique algorithms. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 311–321. PMLR, 2017.

[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.

[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 Madhu Sudan, editor, *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science, Cambridge, MA, USA, January 14-16, 2016*, pages 261–270. ACM, 2016.

[CIKK16] Marco L. Carmosino, Russell Impagliazzo, Valentine Kabanets, and Antonina Kolokolova. Learning algorithms from natural proofs. In Ran Raz, editor, *31st Conference on Computational Complexity, CCC 2016, May 29 to June 1, 2016, Tokyo, Japan*, volume 50 of *LIPIcs*, pages 10:1–10:24. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 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.

[Cob65] Alan Cobham. The intrinsic computational difficulty of functions. In Yehoshua Bar-Hillel, editor, *Logic, Methodology and Philosophy of Science: Proceedings of the 1964 International Congress (Studies in Logic and the Foundations of Mathematics)*, pages 24–30. North-Holland Publishing, 1965.

[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.

[CRTY19] Lijie Chen, Ron Rothblum, Roei Tell, and Eylon Yogev. On exponential-time hypotheses, derandomization, and circuit lower bounds. *Electronic Colloquium on Computational Complexity (ECCC)*, 26:169, 2019.

[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.

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Trans. Inf. 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.

[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.

[DVV16] Akshay Degwekar, Vinod Vaikuntanathan, and Prashant Nalini Vasudevan. Fine-grained cryptography. In *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part III*, pages 533–562, 2016.

[Edm65] Jack Edmonds. Paths, trees, and flowers. *Canadian Journal of mathematics*, 17:449–467, 1965.

[EG04] Friedrich Eisenbrand and Fabrizio Grandoni. On the complexity of fixed parameter clique and dominating set. *Theor. Comput. Sci.*, 326(1-3):57–67, 2004.

[FF91] Joan Feigenbaum and Lance Fortnow. On the random-self-reducibility of complete sets. In *Proceedings of the Sixth Annual Structure in Complexity Theory Conference, Chicago, Illinois, USA, June 30 - July 3, 1991*, pages 124–132. IEEE Computer Society, 1991.

[FF93] Joan Feigenbaum and Lance Fortnow. Random-self-reducibility of complete sets. *SIAM J. Comput.*, 22(5):994–1005, 1993.

[Fid72] Charles M. Fiduccia. Polynomial evaluation via the division algorithm: The fast fourier transform revisited. In *Proceedings of the 4th Annual ACM Symposium on Theory of Computing, May 1-3, 1972, Denver, Colorado, USA*, pages 88–93, 1972.

[GGH94] Mikael Goldmann, Per Grape, and Johan Håstad. On average time hierarchies. *Inf. Process. Lett.*, 49(1):15–20, 1994.

[GGM84] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions (extended abstract). In *25th Annual Symposium on Foundations of Computer Science, West Palm Beach, Florida, USA, 24-26 October 1984*, pages 464–479. IEEE Computer Society, 1984.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 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.

[GIKW17] Jiawei Gao, Russell Impagliazzo, Antonina Kolokolova, and R. Ryan Williams. Completeness for first-order properties on sparse structures with algorithmic applications. In Philip N. Klein, editor, *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 2162–2181. SIAM, 2017.

[GK20] Elazar Goldenberg and Karthik C. S. Hardness amplification of optimization problems. In Thomas Vidick, editor, *11th Innovations in Theoretical Computer Science Conference, ITCS 2020, January 12-14, 2020, Seattle, Washington, USA*, volume 151 of *LIPIcs*, pages 1:1–1:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[GKL93] Oded Goldreich, Hugo Krawczyk, and Michael Luby. On the existence of pseudorandom generators. *SIAM J. Comput.*, 22(6):1163–1175, 1993.

[GL89] Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In David S. Johnson, editor, *Proceedings of the 21st Annual ACM Symposium on Theory of Computing, May 14-17, 1989, Seattle, Washigton, USA*, pages 25–32. ACM, 1989.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984.

[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.

[Gol08] Oded Goldreich. *Computational Complexity - a conceptual perspective*. Cambridge University Press, 2008.

[Gol18] Oded Goldreich. On doubly-efficient interactive proof systems. *Foundations and Trends in Theoretical Computer Science*, 13(3):158–246, 2018.

[GR18a] Oded Goldreich and Guy N. Rothblum. Counting t-cliques: Worst-case to average-case reductions and direct interactive proof systems. In Mikkel Thorup, editor, *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 77–88. IEEE Computer Society, 2018.

[GR18b] Oded Goldreich and Guy N. Rothblum. Simple doubly-efficient interactive proof systems for locally-characterizable sets. In Anna R. Karlin, editor, *9th Innovations in Theoretical Computer Science Conference, ITCS 2018, January 11-14, 2018, Cambridge, MA, USA*, volume 94 of *LIPIcs*, pages 18:1–18:19. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018.

[GR20] Oded Goldreich and Guy N. Rothblum. Worst-case to average-case reductions for subclasses of P. In Oded Goldreich, editor, *Computational Complexity and Property Testing - On the Interplay Between Randomness and Computation*, volume 12050 of *Lecture Notes in Computer Science*, pages 249–295. Springer, 2020.

[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.

[GST03] Dan Gutfreund, Ronen Shaltiel, and Amnon Ta-Shma. Uniform hardness vs. randomness tradeoffs for arthur-merlin games. In *18th Annual IEEE Conference on Computational Complexity (Complexity 2003), 7-10 July 2003, Aarhus, Denmark*, pages 33–47, 2003.

[GW02] Oded Goldreich and Avi Wigderson. Derandomization that is rarely wrong from short advice that is typically good. In José D. P. Rolim and Salil P.

Vadhan, editors, *Randomization and Approximation Techniques, 6th International Workshop, RANDOM 2002, Cambridge, MA, USA, September 13-15, 2002, Proceedings*, volume 2483 of *Lecture Notes in Computer Science*, pages 209–223. Springer, 2002.

[Hås87] Johan Håstad. One-way permutations in NC^0. *Information Processing Letters*, 26(3):153–155, 1987.

[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. *Inf. Process. Lett.*, 1(4):157–163, 1972.

[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.

[IKW02] Russell Impagliazzo, Valentine Kabanets, and Avi Wigderson. In search of an easy witness: exponential time vs. probabilistic polynomial time. *J. Comput. Syst. Sci.*, 65(4):672–694, 2002.

[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, 1989.

[Imp95] Russell Impagliazzo. A personal view of average-case complexity. In *Proceedings of the Tenth Annual Structure in Complexity Theory Conference, Minneapolis, Minnesota, USA, June 19-22, 1995*, pages 134–147, 1995.

[IW97] Russell Impagliazzo and Avi Wigderson. $P = BPP$ if E requires exponential circuits: Derandomizing the XOR lemma. In Frank Thomson Leighton and Peter W. Shor, editors, *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, El Paso, Texas, USA, May 4-6, 1997*, pages 220–229. ACM, 1997.

[IW01] Russell Impagliazzo and Avi Wigderson. Randomness vs time: Derandomization under a uniform assumption. *J. Comput. Syst. Sci.*, 63(4):672–688, 2001.

[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.

[JMV15] Hamid Jahanjou, Eric Miles, and Emanuele Viola. Local reductions. In *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part I*, pages 749–760, 2015.

[Kab01] Valentine Kabanets. Easiness assumptions and hardness tests: Trading time for zero error. *J. Comput. Syst. Sci.*, 63(2):236–252, 2001.

[Kar72] Richard M. Karp. Reducibility among combinatorial problems. In Raymond E. Miller and James W. Thatcher, editors, *Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York.*, The IBM Research Symposia Series, pages 85–103. Plenum Press, New York, 1972.

[KI04] Valentine Kabanets and Russell Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. *Computational Complexity*, 13(1-2):1–46, 2004.

[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.

[KvMS12] Jeff Kinne, Dieter van Melkebeek, and Ronen Shaltiel. Pseudorandom generators, typically-correct derandomization, and circuit lower bounds. *Computational Complexity*, 21(1):3–61, 2012.

[KW17] Daniel M. Kane and R. Ryan Williams. The orthogonal vectors conjecture for branching programs and formulas. *CoRR*, abs/1709.05294, 2017.

[Lev86] Leonid A. Levin. Average case complete problems. *SIAM J. Comput.*, 15(1):285–286, 1986.

[Lin18] Andrea Lincoln. Personal communication, 2018.

[Lip89] Richard J. Lipton. New directions in testing. In Joan Feigenbaum and Michael Merritt, editors, *Distributed Computing And Cryptography, Proceedings of a DIMACS Workshop, Princeton, New Jersey, USA, October 4-6, 1989*, volume 2 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 191–202. DIMACS/AMS, 1989.

[LLW19] Rio LaVigne, Andrea Lincoln, and Virginia Vassilevska Williams. Public-key cryptography in the fine-grained setting. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part III*, volume 11694 of *Lecture Notes in Computer Science*, pages 605–635. Springer, 2019.

[LO87] Jeffrey C Lagarias and Andrew M. Odlyzko. Computing π (x): An analytic method. *Journal of Algorithms*, 8(2):173–191, 1987.

[Lu01] Chi-Jen Lu. Derandomizing arthur-merlin games under uniform assumptions. *Computational Complexity*, 10(3):247–259, 2001.

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

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

[Nak08] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008.

[Nao03] Moni Naor. On cryptographic assumptions and challenges. In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, volume 2729 of *Lecture Notes in Computer Science*, 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.

[P̆10] 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.

[RR97] Alexander A. Razborov and Steven Rudich. Natural proofs. *J. Comput. Syst. Sci.*, 55(1):24–35, 1997.

[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.

[RZ04] Liam Roditty and Uri Zwick. On dynamic shortest paths problems. In *European Symposium on Algorithms*, pages 580–591. Springer, 2004.

[Sha49] Claude E. Shannon. Communication theory of secrecy systems. *Bell Syst. Tech. J.*, 28(4):656–715, 1949.

[Sha11] Ronen Shaltiel. Weak derandomization of weak algorithms: Explicit versions of yao's lemma. *Computational Complexity*, 20(1):87–143, 2011.

[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.

[STV01] Madhu Sudan, Luca Trevisan, and Salil P. Vadhan. Pseudorandom generators without the XOR lemma. *J. Comput. Syst. Sci.*, 62(2):236–266, 2001.

[SU09] Ronen Shaltiel and Christopher Umans. Low-end uniform hardness versus randomness tradeoffs for AM. *SIAM J. Comput.*, 39(3):1006–1037, 2009.

[Sud97] Madhu Sudan. Decoding of reed solomon codes beyond the error-correction bound. *J. Complexity*, 13(1):180–193, 1997.

[TV07] Luca Trevisan and Salil P. Vadhan. Pseudorandomness and average-case complexity via uniform reductions. *Computational Complexity*, 16(4):331–364, 2007.

[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.

[WB86] Lloyd R Welch and Elwyn R Berlekamp. Error correction for algebraic block codes, December 30 1986. US Patent 4,633,470.

[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 (invited talk). In Thore Husfeldt and Iyad A. Kanj, editors, *10th International Symposium on Parameterized and Exact Computation, IPEC 2015, September 16-18, 2015, Patras, Greece*, volume 43 of *LIPIcs*, pages 17–29. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015.

[Wil16] Richard Ryan Williams. Strong ETH breaks with merlin and arthur: Short non-interactive proofs of batch evaluation. In Ran Raz, editor, *31st Conference on Computational Complexity, CCC 2016, May 29 to June 1, 2016, Tokyo, Japan*, volume 50 of *LIPIcs*, pages 2:1–2:17. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016.

[Wil18] Virginia Vassilevska Williams. On some fine-grained questions in algorithms and complexity. In *Proceedings of the ICM*, 2018.

[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.

Appendix A

Appendix for Average-Case Fine-Grained Hardness

A.1 Evaluating Low Degree Polynomials

Here we recall and prove Lemma 3.3.1.

Lemma 3.3.1. *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:*

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

Then there is a randomized algorithm B that runs in time $O(ND^2 \log^2 p + D^3 + tD)$ such that for any $\boldsymbol{x} \in \mathbb{F}_p^N$:

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

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 [WB86] 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 [WB86] 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:

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

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 $O(D\log^2 p)$ [WB86]. Hence, given A with the above properties, the running time of B is:

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

□

A.2 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:

$$c_1(x, y) = x_1y_1$$

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):

$$\begin{aligned} s_\ell &= (1 - x_\ell)(1 - y_\ell)c_{\ell-1} + (1 - x_\ell)y_\ell(1 - c_{\ell-1}) + x_\ell(1 - y_\ell)(1 - c_{\ell-1}) + x_\ell y_\ell c_{\ell-1} \\ c_\ell &= x_\ell y_\ell(1 - c_{\ell-1}) + x_\ell(1 - y_\ell)c_{\ell-1} + (1 - x_\ell)y_\ell c_{\ell-1} + x_\ell y_\ell c_{\ell-1} \end{aligned}$$

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.3 CONVOLUTION-3SUM

Here we give another average-case fine-grained hard problem based on the hardness of 3SUM. While Section 3.3.2 already has a polynomial whose average-case hardness is based on the 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 [Pǐ10] 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 3.3.2 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 3.3.2, we get the following theorem for $\mathcal{F}\mathsf{3SUM} = \{f\mathsf{3SUM}_n\}$.

Theorem A.3.1. *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 A.3.2. *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.

A.4 A Tighter Reduction for $\mathcal{F}$OV

We show that sub-quadratic algorithms cannot compute $f\mathsf{OV}_n$ on even a $1/\mathsf{polylog}(n)$-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 3.3 (as Lemma 3.3.1) 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 3.3 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 $1/\mathsf{polylog}(n)$ 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 A.4.1 ([CPS99])**.** *For any polynomial q over $\mathbb{F}_p$, 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 A.4.2. *Let S be as in Lemma B.1.3 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, 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 $\mathsf{S}_f := \{(i, f(i)) \in \mathrm{Graph}(f) \cap S\}$, 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 A.4.3 ([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 A.4.4. *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 Arith(n) *is a time bound on arithmetic operations over prime fields size $O(n)$.*

Finally, we present a more explicitly parametrized variant of $\mathcal{F}\mathsf{OV}$, denoted $\mathcal{G}\mathsf{OV} = \{g\mathsf{OV}_{n,d,p}\}_{n,d,p\in\mathbb{Z}^3}$, 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 A.4.5. *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 A.4.6. *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 $> 1 - \frac{4}{\delta m} = 1 - 1/\mathsf{polylog}(n)$, 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 B.1.4, 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 B.1.6, 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.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 A.5.1. *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 A.5.2. *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 A.5.3. *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 A.5.2. 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 A.5.1. 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 A.5.3 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 $O(\log^3 n(nd\log n + n^c d^{c'})) = \widetilde{O}(n^c d^{c'})$. $\square$

Appendix B

Appendix for Proofs of Work From Worst-Case Assumptions

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 4.2.13.

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 B.1.1. 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 B.1.2. *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 [BRSV17a] described in Appendix A.4 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 B.1.3 ([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 B.1.4. *Let S be as in Lemma B.1.3 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 B.1.5 ([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 B.1.6. *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 B.1.7. *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 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 4.4.5, 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 $1 - \frac{4}{\delta r} = 1 - 1/\mathsf{polylog}(m)$, 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 B.1.4, 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 B.1.6, 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 $\varepsilon < 4\log m\delta/d$. □

Appendix C

Appendix for Fine-Grained Derandomization

C.1 A Polynomial For k-CLIQUE

We now introduce a polynomial for k-CLIQUE that will, under the weak k-CLIQUE conjecture, achieve the same results as k-OV does. We will also discuss how our results hold under an even *weaker* version of the k-CLIQUE conjecture. This polynomial is independently found in [GR18a]. We first define the k-partite k-CLIQUE problem.

Definition C.1.1 (k-partite k-CLIQUE problem)**.** For an integer $k \geq 3$, the k-CLIQUE$_n$ problem is to, given k graphs on n nodes such that all the edges are only *between* the graphs, decide if there is a k-CLIQUE among them. The input is given as $\binom{k}{2}$ biadjacency matrices between the k graphs.

The k-partite k-CLIQUE problem is equivalent to the common k-CLIQUE problem on one graph (i.e. they reduce to each other in $O(n^2)$ time). We now introduce a polynomial that can count k-CLIQUE's.

For a given k, consider the family of polynomials $\left\{g^k_{n,p} : (\mathbb{F}_p^{n\times n})^{\binom{k}{2}} \to \mathbb{F}_p\right\}_{n,p\in\mathbb{N}}$. Overloading notation, let $\binom{k}{2} = \{(i,j) : 1 \leq i < j \leq k\}$. Then,

$$g^k_{n,p}(A^{(1,2)}, A^{(1,3)}, \ldots, A^{(k-1,k)}) = \sum_{v_1,\ldots,v_k\in[n]} \prod_{(i,j)\in\binom{k}{2}} A^{(i,j)}_{v_i,v_j}$$

Note that boolean input corresponds to biadjacency matrices that comprise a k-partite k-CLIQUE instance and that the polynomial counts the number of k-CLIQUE's so long as p is prime larger than n^k. Now, instead of following the technique of [BRSV17a] to find such a prime in time $\widetilde{O}(n^{k/2+c})$ for any $c > 0$, we can use the randomness in our contrapositive derandomization arguments to choose many random primes so that an evaluation of the polynomial over each prime will be able reconstruct the count via the Chinese Remainder

Theorem. Since choosing random primes is much faster than finding the single large prime, this along with following remark will allow us to make a weaker k-CLIQUE conjecture.

Remark C.1.2. *$g^k_{n,p}$ is guaranteed to be $n^{\omega k/3-o(1)}$ hard under k-CLIQUE's hardness for k a multiple of three but a naïve evaluation takes $\widetilde{O}(n^k)$ time. However, interpreting each matrix of field elements as the biadjacency matrix of a* ***weighted*** *graph, the polynomial can be evaluated by the methods of [NP85] as shown in [Lin18]. This faster evaluation solves an open problem of [BRSV17a] of finding a polynomial whose computability is* ***tight*** *to its hardness for k-CLIQUE. Importantly, this will speed up the oracle calls in our downward self-reduction in our derandomization arugments, allowing for a weaker k-CLIQUE conjecture.*

We now show that $g^k_{n,p}$ is random self-reducible and downward self-reducible as needed in our results. Random self-reducibility is automatic as with $f^k_{n,d,p}$ from Lemma 5.2.7 (note that our degree is the constant $\binom{k}{2}$ and so adds negligibly to the random self-reduction time), and we will show $g^k_{n,p}$ reduces to $g^k_{n^\delta,p}$ similarly to $f^k_{n,d,p}$ (we choose δ different than 1/2 since we can now evaluate the polynomial quick enough to make weaker conjectures). Namely, we will show the following lemma.

Lemma C.1.3. *If there exists an algorithm A that, on input 1^n, outputs a circuit C computing $g^k_{n^\delta,p}$, then there exists an algorithm that computes $g^k_{n,p}$ in time $O(n^{(1-\delta)k}|C| + \mathsf{TIME}(A))$, for any $\delta > 0$.*

Proof. Using A, we print a circuit C computing $g^k_{n^\delta,p}$ in time $\mathsf{TIME}(A)$. To solve an instance $A^{(i,j)}$, $(i,j) \in \binom{k}{2}$, of $g^k_{n,p}$, we break up its input as follows.

Let $P = \left\{\{(j-1)n^\delta+1, (j-1)n^\delta+2, \dots, (j-1)n^\delta+n^\delta\} : j \in n^{1-\delta}\right\}$ be a partitioning of $[n]$ into $n^{1-\delta}$ sets of size n^δ each. Then we can see $g^k_{n,p}$ breaks into sub-summands as follows.

$$g^k_{n,p}(A^{(1,2)}, A^{(1,3)}, \dots, A^{(k-1,k)}) = \sum_{P_1,\dots,P_k \in P} \left(\sum_{v_1 \in P_1, \dots, v_k \in P_k} \prod_{(i,j) \in \binom{k}{2}} A^{(i,j)}_{v_i,v_j} \right) \tag{C.1}$$

We claim the inner sum can be computed by $g^k_{n^\delta,p}$ if given the right inputs. Namely, lets say we have $P_1, \dots, P_k \in P$. Now we can make new matrices $B^{(i,j)} \in \mathbb{F}_p^{n^\delta \times n^\delta}$, $(i,j) \in \binom{k}{2}$ as new input for $g^k_{n^\delta,p}$ for C to solve for us:

To create $B^{(i,j)}$ we consider P_i and P_j and fill $B^{(i,j)}$'s entries in as $A^{(i,j)}$ restricted to the submatrix on row indices P_i and column indices P_j. By inspection, these $B^{(i,j)}$ passed as input to $g^k_{n^\delta,p}$ will yield the inner summand for $P_1, \dots, P_k$.

Thus, feeding these inputs to C for all $P_1, \dots, P_k$ and summing the results that C gives will give the evaluation of $g^k_{n,p}$ on $A^{(i,j)}$, $(i,j) \in \binom{k}{2}$. This takes $n^{(1-\delta)k}$ calls to C. □

Remark C.1.4. *Since constructing circuit C (from broken derandomization) for the above lemma takes time $\widetilde{O}(n^{\delta\omega k/3+c_2})$ time by using the fast/tight evaluation of $g^k_{n^\delta,p}$ from Remark*

C.1.2, then evaluating $g_{n,p}^k$ using the lemma will take $\widetilde{O}(n^{(1-\delta)k+c_1} + n^{\delta\omega k/3+c_2})$ time in total, for constants c_1 and c_2. Setting $\delta = 3/(\omega+3)$ is optimal and yields an $\widetilde{O}(n^{\frac{\omega}{\omega+3}k+c})$ algorithm for k-CLIQUE for some constant c. Thus the k-CLIQUE conjecture can be made with $\epsilon_0 > \omega/(\omega+3)$ instead of $\epsilon > 1/2$.

C.2 Heuristics Imply Separations

We now show that, if a deterministic simulation of BPP succeeds infinitely-often and on average, then BPP doesn't contain any deterministic time class larger than P. Compare this to Corollary 7 of [IW01].

Theorem C.2.1. *If $(\mathsf{BPP},\mathcal{U}) \subseteq \textsf{io-Heur}_{1/3}\mathsf{P}$ then, for all $t(n) = n^{\omega(1)}$ time-constructible:*

$$\mathsf{DTIME}[t(n)] \nsubseteq \mathsf{BPP}$$

Intuitively, if we were able to get good enough derandomization to say that $\mathsf{BPP} \subseteq \mathsf{P}$, then we would know that $\mathsf{DTIME}[n^{\omega(1)}] \nsubseteq \mathsf{BPP}$ by the time-hierarchy theorem. Thus, proving a time-hierarchy theorem that is robust to the io- and Heur qualifiers suffices to concluding $\mathsf{DTIME}[n^{\omega(1)}] \nsubseteq \mathsf{BPP}$ from a $(\mathsf{BPP},\mathcal{U}) \subseteq \textsf{io-HeurP}$ derandomization. We expand ideas from [IW01] to prove the following sufficient lemma for simplicity.

Lemma C.2.2 (Robust Time Hierarchy Theorem)**.** *For all time-constructible $t(n)$:*

$$\textsf{io-Heur}_{1/3}\mathsf{DTIME}[t(n)] \subset \mathsf{DTIME}[t(n)^3]$$

Proof. We give a deterministic machine M that runs in time $t(n)^3$ but whose language it decides is not also in $\textsf{io-Heur}_{1/3}\mathsf{DTIME}[t(n)]$:

Let $\ell(n) = \log t(n)$. On input $x = u||v$, where u is the first $n - \ell(n)$ bits and v is the last $\ell(n)$ bits, M simulates all Turing machines of description length $.5\ell(n)$ on every n-bit input that begins with u for $t(n)$ steps. This creates $2^{.5\ell(n)} = \sqrt{t(n)}$ strings of length $t(n)$ which are the truth tables of each $.5\ell(n)$-size Turing machines that runs in $t(n)$ steps on all of the $2^{\ell(n)} = t(n)$ inputs that begin with u.

It is easy to see with Chebyshev that a random string of length $t(n)$ agrees with any fixed $t(n)$-length string on at least $2/3$ of its values only with probability $1/O(t(n))$. Since there are only $\sqrt{t(n)} < O(t(n))$ strings that are our truth tables though, by union bound there must exist a $t(n)$-length string that disagrees with *all* of our truth tables on at least $1/3$ of each of their values. Of course this string might have high complexity to generate but, since our analysis here only involved a Chebyshev bound, a pairwise independent hash family will fool this analysis and have the same conclusion.

Namely, considering a random string from the pairwise independent hash family $\mathcal{H} = \{\langle r, \cdot\rangle : r \in \{0,1\}^{\ell(n)}\}$ is sufficient for the analysis and so we have that there must exist a specific $\langle r_u, \cdot\rangle$ that disagrees with all of the truth tables on at least $1/3$ of their values. Thus, since $\mathcal{H}$ is relatively small and its functions are easy to compute, M can find that r_u by brute force and outputs its final binary value as $\langle r_u, v\rangle$.

Thus, this whole process of M on $x = u||v$ of simulating $\sqrt{t(n)}$ Turing Machines on $t(n)$ inputs for $t(n)$ time and checking all $t(n)$ r's for the one that fools all of the truth tables adequately enough takes at most $t(n)^3$ time for large enough n. However, by construction, all time $t(n)$ Turing machines fail in deciding this language on at least 1/3 of its inputs for all sufficiently large n.

$\square$