Theoretical and Practical Efficiency of Cryptographic Primitives

by

Scott Knibbe Ames

Submitted in Partial Fulfillment of the
Requirements for the Degree
Doctor of Philosophy

Supervised by Professor Muthuramakrishnan Venkitasubramaniam

Department of Computer Science
Edmund A. Hajim School of Engineering and Applied Sciences

University of Rochester
Rochester, New York

2020
# Table of Contents

Biographical Sketch ........................................ vii

Acknowledgments ........................................... x

Abstract ................................................... xi

Contributors and Funding Sources ....................... xiv

List of Figures ............................................ xvi

1 Introduction ............................................. 1
   1.1 Round Complexity of Precise Zero Knowledge .... 5
      1.1.1 Our Results ...................................... 7
   1.2 UOWHF from Regular One Way Functions ......... 9
      1.2.1 Our Contribution .................................. 13
      1.2.2 Comparison with Previous Work ............... 15
   1.3 Ligero .................................................. 16
      1.3.1 Our Results ...................................... 19
# TABLE OF CONTENTS

## 2 Preliminaries

2.1 Security Parameter ........................................... 25
2.2 Basic Cryptography ....................................... 30
2.3 One-Way Functions .................................... 31
2.4 Indistinguishability ....................................... 34
2.5 Pseudorandom Generators ....................................... 36
2.6 Interactive Turing Machines ....................................... 36
2.7 Universal Hash Function Families ..................... 41
2.8 Commitment Schemes ....................................... 41

## 3 Round Complexity of Precise Zero Knowledge

3.1 A Framework for Partial Black-Box Zero-Knowledge Simulators ....................................... 45
3.1.1 The Facilitator $F$ ....................................... 45
3.1.2 Partial Black-Box Precise Zero Knowledge .............. 48
3.1.3 A Generic Lower Bound ....................................... 49
3.2 Lower Bound for Arbitrary Precision ..................... 57
3.3 A Stronger Lower Bound ....................................... 60
3.4 Constructions of Precise Zero-Knowledge Arguments for Arbitrary Precision ....................................... 65

## 4 UOWHFs from Regular One-Way Functions

4.1 The Generalized Randomized Iterate ...................... 70
  4.1.1 The Reusable Generalized Randomized Iterate ....... 72
# Table of Contents

4.1.2 A Technical Lemma .............................................. 73
4.2 Constructions of Universal One-Way Hash Functions .............. 79
  4.2.1 A Construction with Linear Key Size .......................... 79
  4.2.2 A Construction with Logarithmic Key Size ..................... 85
4.3 Pseudorandom Generator Construction ............................... 89
  4.3.1 A New PRG from Regular OWFs with Logarithmic Seed Size ................................................. 91
4.4 Hardness Amplification of Regular One-way Functions ........... 98
  4.4.1 Target Set Lemma .................................................. 103

5 Ligero ................................................................. 110
  5.1 Preliminaries ......................................................... 110
    5.1.1 Collision-Resistant Hashing and Merkle Trees .............. 110
    5.1.2 Zero-Knowledge Arguments ................................... 112
    5.1.3 Interactive PCPs ................................................ 112
    5.1.4 Secret Sharing .................................................. 114
5.2 From MPC to ZKIPCP ................................................. 116
    5.2.1 Our MPC Model .................................................. 116
    5.2.2 ZKIPCP for NP - The General Case ............................ 120
5.3 A Direct ZKIPCP Construction ....................................... 126
    5.3.1 Testing Interleaved Linear Codes ............................. 127
    5.3.2 Testing Linear Constraints over Interleaved Reed-Solomon Codes ............................................. 135
### TABLE OF CONTENTS

5.3.3 Testing Quadratic Constraints over Interleaved Reed-Solomon Codes .................................................. 138
5.3.4 IPCP for Arithmetic Circuits .................................................. 141
5.3.5 IPCP for Boolean Circuits .................................................. 146
5.3.6 Achieving Zero Knowledge .................................................. 148
5.3.7 The Final ZKIPCP .................................................. 152
5.4 From ZKIPCP to ZK .................................................................. 158
  5.4.1 The Interactive Variant .................................................. 159
  5.4.2 The Noninteractive Variant .................................................. 160
  5.4.3 Sublinear Zero-Knowledge Argument .................................................. 163
  5.4.4 Multi-Instance Amortization .................................................. 166
5.5 Implementation and Results .................................................. 168
5.6 Case $e < d/3$: Proof of Lemma 5.17 .................................................. 173
5.7 Generalizing IPCP Tests .................................................. 175
  5.7.1 Generalized Interleaved Linear Code Testing .................. 176
  5.7.2 Affine Interleaved Linear Code Testing .................. 177
  5.7.3 Generalized Affine Interleaved Linear Code Testing .................. 179
  5.7.4 Generalized Affine Linear Constraint Testing over Interleaved Reed-Solomon Codes .................. 182
  5.7.5 Generalized Testing Quadratic Constraints over Interleaved Reed Solomon Codes .................. 185

6 Conclusion ............................................................................. 189
Bibliography 192
Biographical Sketch

Scott Ames was born in Los Gatos, California, USA. He attended University of Rochester, and graduated with a Bachelor of Science degree in computer science in 2011. He began doctoral studies in Computer Science at the University of Rochester in 2011. He received a Master of Science degree from the University of Rochester in 2013. He pursued his research in Computer Science under the direction of Muthuramakrishnan Venkitasubramaniam.
Publications


Acknowledgments

First and foremost I want to thank my advisor Muthu for all the work he’s done helping me and for being extraordinarily kind and patient. It meant a great deal to me. I want to thank my sister for all her support. I also thank my parents for all their help.
Abstract

In this thesis we investigate the limits of efficiency of cryptographic constructions both asymptotically and concretely. In particular, we identify round complexity, communication complexity, and query complexity for different cryptographic primitives. To frame a clean theoretical question, we restrict the topic to black box constructions. Informally, a black box construction of a primitive $Q$ from a primitive $P$ that requires the existence of a pair of algorithms: (1) a construction algorithm that implements $Q$ while making only black box invocations of the primitive $P$, and (2) a reduction algorithm that breaks the security of $P$ whenever it is given black box access to an adversary that breaks the security of primitive $Q$ and the primitive $P$. This thesis presents three main results:

1. Precise Zero-Knowledge Proofs: We identify the round complexity for implementing precise zero-knowledge proofs for an arbitrary precision function with a black box simulation by presenting tight upper and lower bounds. The traditional notion of zero knowledge only guarantees that the class of probabilistic polynomial time
verifiers learn nothing more than the validity of the statement being proved. The notion of precise zero knowledge requires a stronger “knowledge tightness” guarantee where the verifier does not learn more than what can be computed in time more closely related to the actual time it spent in its interaction with the prover. More precisely, a zero-knowledge proof is said to have precision $p$ if the simulator uses at most $p(|x|, t)$ steps to output a view in which $V$ takes $t$ steps on common input an instance $x$.

2. **Universal One-Way Hash Functions**: We present the first improvement in designing universal one-way hash-functions (used in digital signatures) based on regular one-way functions that makes only a linear number of black-box calls to the underlying one-way function. Previously, the best construction required $\tilde{O}(n^7)$ invocations of the underlying function.

3. **Succinct Zero-Knowledge Arguments**: We present Ligero, the first concretely efficient fully black-box construction of a sublinear zero-knowledge argument system based on collision-resistant hash-functions for all of $\textbf{NP}$. This can be made noninteractive via the Fiat-Shamir heuristic to yield a concretely efficient succinct noninteractive succinct zero-knowledge argument of knowledge without trusted setup (referred to as a zk-STARK). Previous works either relied on public-key assumptions (that have shown to be vulnerable to
quantum attacks) or required on an expensive trusted setup phase. Subsequent to our work, a lot of effort has been made to improve the concrete efficiency of zk-STARKs, and to date, Ligero remains the best implementation with respect to prover complexity for most applications.
Contributors and Funding Sources

This work was supported by a dissertation committee consisting of Professor Muthuramakrishnan Venkitasubramiam (advisor), Professor Lane Hemaspaandra, Professor Joel Seiferas, and Professor Daniel Stefankovic of the Department of Computer Science and Professor Tom Howard of the Department of Electrical and Computer Engineering.

This material is based upon work supported by National Science Foundation award CNS-1526377 and a Google Research Grant. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of above-named organizations.

Chapter 3 is based on an unpublished manuscript additionally co-authored by Muthuramakrishnan Venkitasubramian. Chapter 4 is based on a paper additionally co-authored by Rosario Gennaro and Muthuramakrishnan Venkitasubramian. Chapter 5 is based on a paper additionally co-authored by Carmit Hazay, Yuval Ishai, and Muthuramakrishnan Venkitasubramian.
# List of Figures

4.1 Shoup Reconstruction Algorithm ........................................ 109

5.1 Protocol $\Pi_{ZKIPCP}$ .................................................. 121

5.2 TestInterleaved($\mathcal{F}, L[n, k, d], m, t; U$) ...................... 129

5.3 TestLinearConstraintsIRS($\mathcal{F}, L = RS_{\mathcal{F}, n, k, l}, m, t, \zeta, A, b; U$) ... 136

5.4 TestQuadraticConstraintsIRS($\mathcal{F}, L = RS_{\mathcal{F}, n, k, l}, m, t, \zeta, a, b; U$) ... 139

5.5 Prover and verifier running times for verifying a single instance of different circuit sizes. .......................... 168

5.6 Proof lengths for proving a single instance of different circuit sizes. .................................................. 168

5.7 Amortized prover and verifier running times for verifying multiple instances of circuit with 2048 gates and SHA-256 circuit. .................................................. 169

5.8 Amortized proof lengths for multiple instances of circuit with 2048 gates and SHA-256 circuit. ......................... 169

5.9 GeneralTestInterleaved($\mathcal{F}, L[n, k, d], m, t, \sigma; U$) ......... 176

5.10 AffineTestInterleaved($\mathcal{F}, L[n, k, d], m, t; U, u')$ ............. 178
5.11 GeneralAffineTestInterleaved($F, L[n, k, d], m, t, \sigma; U, u')$ \hspace{1cm} 179
5.12 GeneralAffineTestLinearConstrsIRS($F, RS_{F,n,k,n}, m, t, \zeta, A, b, \sigma; U$) \hspace{1cm} 184
5.13 GeneralTestQuadraticConstrsIRS($F, L = RS_{F,n,k,n}, m, t, \zeta, a, b, \sigma; U^x, U^y, U^z$) \hspace{1cm} 186
Chapter 1

Introduction

In this thesis we investigate the limits of efficiency of cryptographic primitives both theoretically and in terms of concrete implementation.

Even if a particular algorithm has good time complexity, it may be the case that constant factors involved in any practical implementation are unacceptably high. Many cryptographic protocols have “unsecure” noncryptographic alternatives, such as sending information unencrypted instead of encrypted, or trusting a cloud server not to violate your privacy instead of making it impossible for the server to do so. If practical implementations are not efficient enough, then the benefit of ensuring security will be outweighed by the increased time cost, so we may have to use the unsecure implementation, thus forfeiting the benefits of using cryptography.

However, theoretical performance is not sufficient. Even if a particular implementation has good concrete performance on all available benchmarks, if its theoretical performance is bad then it is possible that the
implementation will take unexpectedly long under some circumstances, perhaps even longer than a human lifetime if the worst-case performance is not polynomial time.

In chapter 3 we present results from our unpublished paper [AV15] on precise zero-knowledge proofs. Precise zero knowledge, first introduced by Micali and Pass [MP06], guarantees that the view of any verifier $V$ can be simulated in time closely related to the actual (as opposed to worst-case) time spent by $V$ in the generated view. Informally, a zero-knowledge proof system has a precise simulator $S$ with precision $p$ if and only if the amount of time $S$ takes to produce its output is bounded by $p(n, t)$, where $n$ is the length of the common input to the prover and verifier and $t$ is the number of steps that the verifier took in the view outputted by $S$. [MP06] show how to construct, for any $f(n) \in \omega(\log n)$, $f(n)$-round zero-knowledge proofs with constant precision and, for any $g(n) \in \omega(1)$, $g(n)$-round zero-knowledge proofs with polynomial precision for all of $\text{NP}$. We present constructions of zero-knowledge proofs for arbitrary precision that have optimal round complexity. In our negative result we show that for any $c$ if a language $L$ has a $\frac{c \log n}{\log g(n)}$-round precise zero-knowledge proof with precision $p(n, t) = cg(n)t + n^{c-1}$, then $L \in \text{BPP}$ and no interaction is necessary. In our positive result we show that for any $L \in \text{NP}$ and for any $r(n) \in \omega(\frac{\log n}{\log g(n)})$ there exists an $r(n)$-round precise zero-knowledge argument with the precision $p(n, t) = cg(n)t + cn$ for some $c$. For the specific case of constant-precision precise zero knowledge, we strengthen
our lower bound to apply to simulators that are only required to do a simulation that is somewhat similar to the true distribution of views of the verifier interacting with the honest prover. We introduce a general framework to prove such lower bounds and prove each of our results in this framework.

In chapter 4 we explore the efficiency of UOWHFs in terms of key length, the efficiency of PRGs, and the efficiency of hardness amplification of weakly one-way functions. To do this we present the Generalized Randomized Iterate of a (regular) one-way function $f$ and show that it can be used to build Universal One-Way Hash Function (UOWHF) families with $O(n^2)$ key length. We then show that Shoup’s technique for UOWHF domain extension can be used to improve the efficiency of the previous construction. We present the Reusable Generalized Randomized Iterate which consists of $k \geq n + 1$ iterations of a regular one-way function composed at each iteration with a pairwise independent hash function, where we only use $\log k$ such hash functions, and we “schedule” them according to the same scheduling of Shoup’s domain extension technique. The end result is a UOWHF construction from regular one-way functions with an $O(n \log n)$ key. These were the first such efficient constructions of UOWHF from regular one-way functions of regularity that cannot efficiently be computed. Finally we show that the Shoup’s domain extension technique can also be used in lieu of derandomization techniques to improve the efficiency of PRGs and of hardness amplification of regular
weakly one-way functions.

In chapter 5 we explore enforcing that a server correctly computes what it was requested to compute in a way that scales sublinearly with the size of the circuit to be computed. We design and implement a simple zero-knowledge proof protocol for \( \mathbf{NP} \) whose communication complexity is proportional to the square root of the verification circuit size. The protocol can be constructed based on any collision-resistant hash function. It can also be made noninteractive in the random oracle model, yielding concretely-efficient zk-SNARKs that do not require a trusted setup or public-key cryptography.

Our protocol is attractive not only for very large verification circuits but also for moderately large circuits that arise in applications. For instance, for verifying a SHA-256 preimage in zero-knowledge with \( 2^{-40} \) soundness error, the communication complexity is roughly 44KiB (or less than 34KiB under a plausible conjecture), the prover running time is 140 ms, and the verifier running time is 62 ms. This proof is roughly four times shorter than the proof length achieved by ZKB++ \([\text{CDG}^+17]\), a protocol with similar properties.

The communication complexity of our protocol is independent of the circuit structure and depends only on the number of gates. For \( 2^{-40} \) soundness error, the communication becomes smaller than the circuit size for circuits containing roughly 3 million gates or more. Our efficiency advantages become even bigger in an amortized setting, where several
instances need to be proven simultaneously.

1.1 Round Complexity of Precise Zero Knowledge

Zero-knowledge interactive proofs, introduced by [GMR89] are paradoxical constructs that allow an entity (called the prover) to convince another entity (called the verifier) of the validity of a mathematical statement $x \in L$, while providing no additional knowledge to the verifier. The zero-knowledge property is formalized by requiring that the view of a malicious probabilistic polynomial-time verifier $V^*$ in an interaction with a prover can be reconstructed “indistinguishably” by a probabilistic polynomial-time simulator $S$, without interacting with anyone, on input just $x$. Since the simulator is allowed be to an arbitrary probabilistic polynomial-time machine, the traditional notion of zero knowledge only guarantees that the class of probabilistic polynomial-time verifiers learn nothing. [MP06] introduced the notion of precise zero knowledge with a stronger so-called “knowledge tightness” guarantee. Namely, in contrast to traditional zero knowledge, precise zero knowledge considers the knowledge of an individual verifier on a per execution basis—it requires that the view of any verifier $V$ in which $V$ takes $t$ computational steps can be reconstructed in time closely related to $t$—for example $2t$ steps. More generally, a zero-knowledge proof is said to have precision $p$
if the simulator uses at most \( p(|x|, t) \) steps to output a view in which \( V \) takes \( t \) steps on common input an instance \( x \). In essence, this notion guarantees that the verifier does not learn more than what can be computed in time more closely related to the actual time it spent in its interaction with the prover.

Micali and Pass show in [MP06] that precise zero-knowledge proof systems with black-box simulators only exist for “trivial” languages (namely, only for \( L \in \text{BPP} \)). Loosely speaking, an interactive proof is black-box zero knowledge if there exists a (universal) simulator \( S \) that uses the (possibly dishonest) verifier \( V' \) as a black box in order to perform the simulation: the simulator only uses \( V' \) as a subroutine and does not use the description of \( V' \) directly. Hence, to construct precise zero-knowledge proofs, the simulator is required to have non-black-box access to the verifier. In the same work, Micali and Pass also show how to construct precise zero-knowledge proofs in a model where the simulator is allowed to learn the running time distribution of the verifier. More precisely, they show how to construct \( \omega(1) \)-round precise zero-knowledge proofs with polynomial precision, i.e. \( p \) is polynomial in both \( n \) and \( t \) and \( \omega(\log n) \)-round precise zero-knowledge proofs with linear precision, i.e. \( p(n, t) = ct + n^c \) for some constant \( c \). A natural question that is raised by their work, is the following:

*What is the optimal round complexity for achieving precise zero-knowledge proofs with linear precision or polynomial precision?*
More generally, what is the round complexity of achieving precise zero-knowledge proofs for an arbitrary precision function?

In the literature, most round complexity lower bounds for zero-knowledge proofs such as [GK96, CKPR01, BL02, APV05] consider only black box zero-knowledge proofs and as pointed out above, black box precise zero-knowledge proofs only exist for trivial languages. Furthermore, obtaining lower bounds for non-black-box simulators is a notoriously hard problem and previous results that obtain such lower bounds [BLV06] rely on nonstandard complexity assumptions.

1.1.1 Our Results

We resolve the above question completely in a setting where the simulator is non-black-box only in a weak sense, namely, the simulator is only allowed to learn the running time of the verifier and is “black box” otherwise. More precisely, we show the following theorem:

**Theorem 1.1** (Informal). There exists a constant c such that for any arbitrary polynomial-time computable function $f(n)$ that is bounded above by a polynomial there exists an $r(n)$ round “partial” black-box precise zero-knowledge proof with precision $p(n, t) = ct f(n) + cn$ if and only if $r(n) \in \omega \left( \frac{\log n}{\log(f(n))} \right)$.

To obtain our positive result of constructing a precise zero-knowledge argument with an arbitrary precision function $f$, we show that a slight
variant of the simulation provided by Micali and Pass yields the required simulator. We provide the proof of our positive result in Section 3.4.

The lower bound, however, requires careful formulation. First, we provide a framework in which we can analyze simulators that only have access to the runtime distribution of the verifiers. Towards this, we introduce a universal “mediator” machine that we call the facilitator that interacts with the simulator. The role of the facilitator is to provide access to the verifier in a black-box way, to relay the running time of the verifier to the simulator, and to allow the simulator to set a time limit on the execution of an instance of the verifier. This framework is discussed in Section 3.1. Then we prove a generic lower bound in this framework. In essence, this lower bound encapsulates a part of the proof that most lower bounds on round complexity rely on. More precisely, any such lower bound employs the following steps:

1. Construct a malicious verifier $V^*$ that essentially follows the code of the honest verifier but makes some decisions such as delaying (or aborting) with some probability. In essence, the distribution of the messages of $V^*$ is identical to that of the messages generated by $V$.

2. Show that any simulator that simulates $V^*$ in a black-box way must be able to generate transcripts with non-negligible probability without rewinding the verifier $V^*$.

3. Construct a cheating prover $P^*$ that can convince a verifier on any
input $x$ with the same probability that the simulator outputs a transcript on which it did not rewind the verifier (or at least did not rewind it past the point where it got a message).

4. Conclude from the soundness of the interactive proof that the simulator cannot simulate the verifier on false statements and thus can be used to decide the language $L$. Since the simulator is a probabilistic polynomial-time machine, we have that $L \in \text{BPP}$.

We strengthen our negative result by proving that even if we want to make a protocol that is a precise zero-knowledge proof for a limited class of verifiers which is only allowed to delay the messages of the honest verifier and is not permitted to abort or alter these messages. We achieve our positive result without this assumption: it is secure even against malicious verifiers that are allowed to alter their messages.

In Section 3.2, we give our lower bound for arbitrary precision and in Section 3.4 we provide our constructions of precise zero-knowledge arguments. Next we strengthen our lower bound for the specific case of linear precision where we relax the simulation to only require a constant distinguishing deviation gap.

### 1.2 UOWHF from Regular One Way Functions

One of the central results in modern cryptography is that the existence of one-way functions implies the existence of digital signatures (as defined
in [GMR88]). This result was first established in [NY89] for one-way permutations via the notion of universal one-way hash functions (UOWHFs). Later Rompel in [Rom90] proved that UOWHFs can be built from any one-way function. The notion of UOWHF is interesting on its own, apart from its connection to digital signatures. UOWHFs are compressing functions (i.e. the output is shorter than the input) and have a target collision resistance property: a function family $G$ is a UOWHF if no efficient adversary $A$ succeeds in the following game with nonnegligible probability:

- $A$ chooses a target input $z$;
- a randomly-chosen function $g \in G$ is selected;
- $A$ finds a collision for $g(z)$, i.e. an input $z' \neq z$ such that $g(z) = g(z')$.

A seemingly-weaker notion is second preimage resistance where the target input $z$ is randomly chosen (instead of being chosen by $A$). It is, however, well-known how to convert a second preimage resistant function family into a UOWHF.

The security of these constructions is proven by reductions: given an adversary $A$ that wins the above UOWHF game, we build an “inverter” $I$ that is able to solve a computationally hard problem, e.g. invert a one-way function. A crucial feature of these reductions is their efficiency, i.e. the relationship between the running time of $D$ (or $A$) and $I$ and the resulting degradation in the security parameters. For the case of UOWHFs one of
the most important efficiency measures is the size of the key needed to run the algorithm.

Unfortunately the construction of UOWHFs based on general one-way functions do not perform very well by that metric. If $n$ is the security parameter, the original Rompel construction yielded a key of size $\tilde{O}(n^{12})$, which was later improved to $\tilde{O}(n^7)$ in [HHR+ 10]. Conversely under the (much stronger) assumption of the existence of one-way permutations, [NY 89] achieve linear key size. Apart from the above works, we are aware of only one work (by [SY 90]) that constructs UOWHFs from regular one-way functions (i.e. functions that have constant size preimage sets). Their construction achieves $O(n \log n)$ key size, but is very complicated and, more importantly, requires knowledge of the regularity of the regular one-way function used.

We reinvestigate the construction of UOWHFs from regular one-way functions. We obtain a very simple construction with $O(n \log n)$ key size, which does not require knowledge of the regularity of the regular one-way function used. This is the first such efficient construction of a UOWHF from regular one-way functions of unknown regularity.

Somewhat surprisingly our UOWHF construction is obtained via a simple “tweak” on a well-known algorithm for pseudorandom number generation from regular one-way functions: the Randomized Iterate [GKL 93, HHR 06]. Another surprising connection established by this paper is that Shoup’s domain extension technique [Shoo 00] can be used
CHAPTER 1. INTRODUCTION

to improve the seed size in both the PRG and UOWHF. Collision-resistant hashing is an ubiquitous tool in Cryptography and, in practice, a stronger notion of collision resistance is used where the adversary is given just $H \in \mathcal{H}$ and must find $z, z'$ that collide (we will refer to this notion as full collision resistance, as opposed to the target collision-resistance property achieved by UOWHFs).

This is problematic because there is strong evidence that this stronger notion cannot be achieved by assuming just one-way functions. [Sim98] proves that there is no black box construction\(^1\) of a fully collision-resistant hash function from one-way permutations. While a non-black-box construction based on one-way functions remains theoretically possible, such construction would probably be very inefficient, since efficient constructions based on general assumptions tend to be black box constructions.

Furthermore, the analysis of practical and widely adopted supposedly collision-resistant functions has reminded us of the importance of constructing efficient candidates for collision-resistant functions that are also provably-secure relative to a well-established computationally-hard problem. The above explains why researchers and practitioners alike are looking at UOWHFs to replace full collision-resistant hashing in practical applications (such as certifications – see for example the work of Halevi and Krawczyk on randomized hashing [HK06]).

\(^{1}\)Informally, a black-box construction accesses the underlying one-way function only via input queries, without any knowledge of its internal structure.
Current efficient candidates for UOWHFs have either no proof of security or make stronger assumptions than the existence of one-way permutations.\textsuperscript{2} Achieving a truly-efficient UOWHF construction based on one-way functions would offer practitioners a target collision-resistant function which can be used in practice and gives the peace of mind of a strong security guarantee.

In order to achieve this goal, our construction slightly relaxes the assumption to regular one-way functions, yielding a dramatic improvement to an $O(n \log n)$ key size. We are following the same approach as [HHR\textsuperscript{06}] for pseudorandom generators: looking at the more limited case of regular one-way functions not only to improve the efficiency, but also to explore techniques that might benefit constructions in the general case (which is what happened in the PRG case).

1.2.1 Our Contribution

We present a new algorithm (we call it the Generalized Randomized Iterate or GRI), which, depending on its parameters, can be used to build either PRGs or UOWHFs based on regular one-way functions.

First proposed in [GKL\textsuperscript{93}] the original Randomized Iterate construction involves composing the regular one-way function with different $n$-
wise independent (later improved to simply pairwise-independent in [HHR06]) hash functions at each iteration. More specifically, if $f$ is a regular one-way function, and $h_1, \ldots, h_m$ are pairwise-independent hash functions from $\{0, 1\}^n$ to $\{0, 1\}^n$, the $m^{th}$ randomized iterate of $f$ using the $h_i$ is defined as $f^k = f \circ h_k \circ f \circ h_{k-1} \circ \ldots \circ f \circ h_1 \circ f$. In [GKL93, HHR06] it is shown that this function is hard to invert at each stage and therefore can be used to construct PRGs in conjunction with a generic hardcore predicate (such as the Goldreich-Levin bit [GL89]).

We generalize the Randomized Iterate to use compressing pairwise-independent hash functions $h_i$ at each stage. Somewhat surprisingly, we then show that the resulting family (see Definition 4.1) is second preimage resistant.

Notice that in the above applications the universal hash functions $h_i$ are part of the secret key of the resulting algorithm (the seed for the PRG, the index key for the UOWHF). Therefore it is desirable to have constructions in which the number of functions can be minimized.

The Randomized Iterate PRG construction in [HHR06] has an $O(n^2)$ seed, but it was also shown how an $O(n \log n)$ seed could be achieved by using generic derandomization techniques. First we point out that this approach does not immediately work in the UOWHF case, as in order to reduce the key size, the derandomization procedure requires an additional property.\footnote{The actual derandomization algorithm (the Nisan-Zuckerman PRG for space-}
We then explore another fascinating and somewhat-unexpected connection. We observe that instead of using derandomization techniques, the structure of the Generalized Randomized Iterate can be improved by using Shoup’s domain extension technique for UOWHFs [Sh00]. We define the Reusable Generalized Randomized Iterate RGRI: Using Shoup’s approach we prove that it is possible to “recycle” some of the hash functions in the Generalized Randomized Iterate, to $O(\log m)$ for $m$ iterations (instead of $m$). The net result is that we achieve a UOWHF with $O(n \log n)$ key size.

Finally we point out that the RGRI also yields an $O(n \log n)$-seed PRG from regular one-way function, and can be also used for hardness amplification of regular one-way functions, obtaining alternative proofs of results already appearing in [HHR06].

1.2.2 Comparison with Previous Work

We already mention the previous works on UOWHFs based on general assumptions [NY89, Rom90, HHR+10, SY90] and how they compare to our work.

As discussed above our UOWHF construction uses in a crucial way tools that were developed for pseudorandom generators. In this sense, our work follows the path of recent papers on inaccessible entropy bounded computations) used in [HHR06] has this property, but a generic PRG for space-bounded computation might not.
Those works elegantly show that the known constructions of PRGs and UOWHFs can be interpreted as similar manipulation techniques on different forms of computational entropy (pseudoentropy for PRGs and inaccessible entropy for UOWHFs). While less general, our work shows a more direct and specific connection: a single algorithm (the Generalized Randomized Iterate) which is sufficiently “flexible” to be used either as a PRG or as a UOWHF.

\section*{1.3 Ligero}

Verifying outsourced computations is important for situations where there is a potential incentive for the party performing the computation to report incorrect answers. We present a concretely-efficient interactive proof protocol for $\mathbf{NP}$ with communication complexity proportional to the square root of the size of the circuit verifying the $\mathbf{NP}$ relation. Our proof system is, in fact, a zero-knowledge argument of knowledge, and it only requires the verifier to send public coins to the prover. The latter feature implies that it can be made noninteractive via the Fiat-Shamir transform \cite{FS86}, yielding an efficient implementation of zero-knowledge succinct noninteractive arguments of knowledge (zk-SNARKs \cite{BCCT13}) without a trusted setup. To put our work in the proper context, we provide some relevant background. The last half-decade has seen tremendous progress in designing and implementing efficient systems for verifi-
able computation (see [WB15, BBC+17] for recent surveys). These efforts can be divided into three broad categories according to the underlying combinatorial machinery.

**Doubly-efficient Interactive Proofs:** This line of work, initiated by Goldwasser, Kalai, and Rothblum [GKR15] (following a rich line of work on interactive proofs with computationally-unbounded provers [GMR85, LFKN90, Sha90]), provides sublinear communication and efficiently-verifiable proofs for low-depth polynomial-time computations.  

4 See [CMT12, Tha13, VSBW13, RRR16] and references therein for a survey of works along this line.

**Probabilistically-Checkable Proofs (PCPs) and Their Interactive Variants:** Originating from the works of [BFLS91, Kil92, Mic94], recent works [BCGT13, BCG+16, BBC+17] have shown how to obtain efficient sublinear arguments for $\text{NP}$ from PCPs [BFLS91, AS98, ALM+98]. Classical PCPs have been extended to allow additional interaction with the prover, first in the model of interactive PCPs (IPCPs) [KR08] and then in the more general setting of interactive oracle proofs (IOP) [BCS16], also known as probabilistically-checkable interactive proofs (PCIP) [RRR16]. Arguments obtained via PCPs and IOPs have the advantages of not relying on public-key cryptography, not requiring a trusted setup, and offering

---

4 The GKR technique has been recently extended to the case of $\text{NP}$ statements by Zhang et al. [ZGK+17]. However, the communication complexity of the resulting proofs still grows with the verification circuit depth, and moreover their efficient instantiation requires the use of public-key cryptography.
conjectured security against quantum attacks. However, current implementations along this line are still quite far from having good concrete efficiency.

**Linear PCPs:** This line of work, initiated by Ishai, Kushilevitz, and Ostrovsky [IKO07] (in the interactive or designated verifier setting) and by Groth [Gro10] (in the noninteractive, public verification setting of SNARKs) obtains sublinear arguments for \textbf{NP} with preprocessing by combining linear PCPs with homomorphic public-key cryptography. In a linear PCP the verifier can obtain a small number of linear combinations of a proof vector. Linear PCPs are simpler to construct than classical PCPs and have served as the basis for some of the first implementations of verifiable computation protocols [SMBW12]. A very efficient construction of linear PCPs for \textbf{NP}, which serves as the basis for most current SNARK implementations, including the ones used in zero cash [BCG+14], was given by Gennaro, Gentry, Parno, and Raykova in [GGPR13]. (The view of these SNARKs as being based on linear PCPs is due to Bitansky et al. [BCI+13] and Setty et al. [SBV+13].) Two practical disadvantages of the protocols along this line are that they are quite slow on the prover side (due to a heavy use of public-key cryptography), and their soundness in the noninteractive setting crucially relies on the existence of a long and “structured” common reference string that needs to either be generated by a trusted party or by an expensive distributed protocol.

Our goal in this work is to combine the best features of previous ap-
proaches to the extent possible:

Obtain a simple, concretely-efficient, sublinear communication zero-
knowledge argument system for \textbf{NP}, without any setup, complex
\textit{PCP} machinery, or expensive public-key operations.

As discussed above, all prior works fall short of meeting the above
goal on one or more counts.

1.3.1 Our Results

The main topic of this chapter is a zero-knowledge argument protocol for
\textbf{NP} with the following features.

- It is \textit{sublinear}, in the sense that the asymptotic communication com-
  plexity is roughly the square root of the verification circuit size.

- It is simple to describe and analyze in a self-contained way.

- It only employs symmetric-key primitives (collision-resistant hash
  functions) in a black box way. Moreover, the protocol can be made
  noninteractive in the random oracle model by using the Fiat-Shamir
  transform \cite{FS86}, thus providing a lightweight implementation of
  (publicly-verifiable) zero-knowledge \textit{SNARKs}.

- It does not require any trusted setup, even in the noninteractive
case.
• It is concretely efficient. We demonstrate its concrete efficiency via an implementation.

• In the multi-instance setting where many instances for the same NP verification circuit are required, we obtain improved amortized communication complexity with sublinear verification time.

Our protocol can be seen as a lightweight instance of the second category of protocols discussed above. However, instead of directly applying techniques from the PCP literature, we combine efficient protocols for secure multiparty computation (MPC) with a variant of the general transformation of Ishai, Kushilevitz, Ostrovsky, and Sahai (IKOS) [IKOS07] that transforms such MPC protocols to zero-knowledge interactive PCPs (ZKIPCP).

More concretely, we instantiate the MPC component with an optimized variant of the protocol of Damgård and Ishai [DI06] (similar to the one described in appendix C of [IPS09]) and transform it into a ZKIPCP by applying a more efficient variant of the IKOS transformation in the spirit of the IPS compiler [IPS08]. The main difference with respect to the original IKOS transformation is that we restrict the topology of the MPC network in a way that leads to a better tradeoff between soundness error and communication complexity.

A key feature of the underlying MPC protocol is that its total communication complexity between the parties is independent of the number of
parties and is roughly equal to the size of the circuit being evaluated. Let-
ting the number of parties be approximately the square root of the circuit
size results in per-party communication that is also roughly the square
root of the circuit size. This translates into a ZKIPCP with similar pa-
rameters. See Section 5.3 for a self-contained presentation of the ZKIPCP
obtained via the above approach.

The recent work of Giacomelli, Madsen and Orlandi [GMO16] and its
improvement due to Chase et al. [CDG+17] already demonstrated that
the IKOS transformation can lead to concretely-efficient zero-knowledge
arguments, but where the communication is bigger than the verification
circuit size. We obtain a sublinear variant of this result by modifying both
the IKOS transformation and the underlying MPC machinery.

To summarize, using the above approach we obtain a simple proof of
the following theorem with good concrete efficiency:

**Theorem 1.2** (Informal). Assume the existence of collision-resistant hash
functions. Then there is a public-coin zero-knowledge argument for
proving the satisfiability of a circuit $C$ with communication complexity
$\tilde{O}(\sqrt{|C|})$.

**Concrete efficiency.** We now give more detailed information about the
concrete efficiency of our implementation. The following numbers apply
either to interactive zero-knowledge protocols based on collision-resistant
hash functions or to noninteractive zk-SNARKs in the random oracle
model obtained via the Fiat-Shamir transform. We refer the reader to Section 5.5 for more details and give only a few representative figures below.

The communication complexity of proving the satisfiability of an arithmetic circuit with $s > 30000$ gates over a finite field $\mathbb{F}$ of size $|\mathbb{F}| \geq 2^{128}$ with soundness error $2^{-40}$ consists of roughly $70\sqrt{s}$ elements. For the case of $2^{-80}$ error, the communication is roughly $120\sqrt{s}$.

In the case of boolean circuits, the communication complexity becomes smaller than the circuit size for circuits with more than roughly 3 million gates. One concrete benchmark that has been used in prior works is verifying a SHA-256 preimage in zero knowledge. For this benchmark, the communication complexity of our protocol with $2^{-40}$ soundness error is less than 34KB, the prover running time is 140 ms, and the verifier running time is 62 ms. This is roughly 4 times less communication than a similar proof of ZKB++ [CDG+17], an optimized variant of ZK-Boo [GMO16]. Requiring $2^{-80}$ soundness error doubles the communication (as in [GMO16, CDG+17]).

Our protocol easily extends to a multi-instance setting and provides additional benefits there. In this setting, we can handle $N$ instances of a circuit of size $s$ with soundness error $2^{-\kappa}$ at an amortized communication cost per instance smaller than $s$ when $N = \Omega(\kappa^2)$. Moreover, the amortized verification time in the multi-instance setting is sublinear, involving a total of $O(s \log s + N \log N)$ field operations. Finally, the prover’s run-
ning time grows linearly with the number of instances but still remains practically feasible for reasonable number of instances. For the SHA-256 circuit, we show that the amortized communication over 4096 instances is 2KB with amortized prover time of 151 ms and verification time of 500 µs. This amortization is relevant to natural applications, e.g., in the context of cryptocurrencies [DFKP13, BCG+14].

**Related work.** In a concurrent and independent work [BBHR18], Ben-Sasson et al. use different techniques to construct concretely efficient IOPs that imply “transparent” proof systems, referred to as zk-STARKs, of the same type we obtain here. These zk-STARK constructions significantly improve over the previous ones from [BBC+17]. A preliminary comparison with the concrete efficiency of our construction suggests that our construction is generally more attractive in terms of prover computation time and also in terms of proof size for smaller circuits (say, of size comparable to a few SHA-256 circuits), whereas the construction from [BBHR18] is more attractive in terms of verifier computation time and proof size for larger circuits. We leave a more thorough comparison between the two approaches to future work.

A related topic is interactive proofs of proximity with sublinear-time verifiers [RVW13] that ensure approximate correctness. Namely, the verifier accepts every input in the language with high probability, and rejects every input that is far from the language. In this work, Rothblum et al. prove that all languages in $NC$ have interactive proofs of prox-
imity with roughly square root (in the input size) query and commu-
nication complexities, and polylogarithmic round complexity. In another
work [CDD+16], Cascudo et al. introduce a new primitive called interac-
tive proximity testing that can be used to verify whether a string is close
to a given linear code.
Chapter 2

Preliminaries

In this chapter we define important mathematical terms for the following chapters.

2.1 Security Parameter

When we construct cryptographic systems, we typically build them using other cryptographics systems. For example, one might construct an efficient hash function with certain properties that compresses its inputs by two bits given an efficient implementation of a hash function with the same properties that compresses its input by one bit. One also might construct something that acts very similarly to a random function (with reasonable caveats) given a length-preserving function that is efficient to compute but is impossible to efficiently invert. To prove them secure, we show that any machine that can break the properties of the new system can be efficiently converted into one which can break the properties of at
least one of the component systems. By the law of contrapositives, this proves that if the component systems are secure then the new system must be secure.

There is a subtle problem that can arise with such constructions, especially when composed together. To illustrate the problem and its solution we imagine hypothetical cryptographic systems we will call $A$s, $B$s, and $C$s. Their exact properties are unimportant: the problem is generic enough to apply to nearly all nontrivial cryptographic constructions. The only thing we will require is that they can be applied to inputs of any length, that it is mathematically defined how to “break” them in some sense we will not specify, and that given unbounded time it is always possible to break them. We will suppose we can use $A$s to build $B$s and that we can use $B$s to build $C$s.

In more detail, suppose we have an instance of a $A$ and we know that breaking it on inputs of length $n$ takes about $2^{\Omega(n)}$ steps. Then when we use it we just have to estimate an upper bound on how much time our enemies will be able to spend on breaking our system, and choose an $n$ such that this time is higher than that. We’d like to preserve our ability to do this kind of computation when we use our $A$ to construct a $B$. Typically such constructions will be proven correct for any instance of the base system, but the problem we will illustrate remains even if we are constructing a new system based on a particular instance of the old system.
Suppose we prove the security of the $B$ by defining a polynomial-time oracle Turing machine $M$ and proving it has the property that if its oracle $E$ is a machine that can break our $B$ construction on inputs of length $2n^3 \lg n + 1$ with probability $p$ then $M^E$ can break the underlying $A$ with probability $p^2$ on inputs of length $n$. The $B$ is still secure, but less so in the sense that the time it takes to break our $B$ on inputs of length $n$ is less than the time it takes to break our $A$ on inputs of length $n$. In order to get the same level of security that our $A$ has on inputs of length $n$, our $B$ has to have an input length of $2n^3 \lg n + 1$. Further suppose that doing the things we want to do with our $B$ on inputs of length $n$ takes $n^2$ steps. Then the situation is worse than one might think: in order to get the same level of security that our $A$ has on inputs of length $n$, our $B$ has to use inputs of length $2n^3 \lg n + 1$, so the time it will take to do what we want to do with our $B$ at that level of security is actually $(2n^3 \lg n + 1)^2$, which is $\Theta(n^6 \lg^2 n)$.

This effect is even stronger if we use our $B$ to create a $C$ and the security of the $C$ is further diluted from the security of the $B$ in the same way that the security of our $B$ is diluted from the security of our $A$. Breaking the $C$ will still take an amount of time that is superpolynomial in the input length to the $C$, but it will be a much smaller superpolynomial amount of time than the amount of time it takes to break the $A$ as a function of the input length to the $A$. The minimum input length we need for our $C$ in order to make it secure against our estimate of the computational
power of our enemies is therefore notably (but polynomially) higher than the input length we need to make our $A$ secure against the same enemies running corresponding attacks for the same amount of time. The key insight required to solve this problem (in the way we will soon describe) is that it is more useful to analyze the performance of our $B$ and our $C$ in terms of the input length of the $A$ than to analyze them in terms of their own input lengths. This is the idea behind the paradigm of the security parameter.

The parameters of a system are defined as functions of the security parameter: given the security parameter (usually represented by the symbol $\kappa$) we can compute the input lengths required for all the subcomponents of our system, as well as any other useful parameters used in their construction. In our imaginary construction, we would have that when the security parameter is $\kappa$ the input length to the $A$ is $\kappa$ and the input length to the $B$ is $2\kappa^3 \log \kappa + 1$. In more complex constructions, we would need to define additional parameters, but all such parameters would be defined as a function of the security parameter, and we would effectively be proving that when the system is instantiated with those parameters that it is about as secure as the component systems instantiated on inputs of length $\kappa$. In fact, it would be more correct to say that it is about as secure as the components instantiated with a security parameter of $\kappa$, because the component systems might also be constructed based on the security of their own components. If the input length of our $A$ is $n = f(\kappa)$ for some function $f$
then the input length of our \( B \) would be 
\[
2n^3 \lg n + 1 = 2f(\kappa)^3 \lg f(\kappa) + 1.
\]

When constructing a cryptographic system that uses multiple component cryptographic systems, we could express the input length of the new system as a function of the input lengths of each of its components separately. It almost always makes the most sense to let the security parameter be the same for all of its components. One exception is if we are highly confident of the security of one subsystem and less confident about the security of another subsystem. Since we do not know exactly how much time it will take to break the subsystem we know less about, it makes sense to instantiate it on larger inputs so that even if our estimates of how much time it takes to break it are too high, the actual time it will take to break it will still be high enough that it is infeasible for our enemies.

This could happen if the security of one subsystem is proven against probabilistic polynomial-time adversaries and the security of the other is proven against unbounded time adversaries. In cryptography it is typically easy to define a Turing machine that achieves optimal probability of success at some operation if we don’t care about how much time the machine takes. However, it is often the case that we don’t know the best possible performance of efficient attacks of a system, only that the success probability decreases superpolynomially for sufficiently-large inputs for any given attack, and what “sufficiently-large” means and the exact rate the success probability decreases depends on the properties of the
machine you are using to attempt to break the system. Under such circumstances it would be appropriate to separate the security parameter into two components, but this is still typically not done.

2.2 Basic Cryptography

**Basic notations.** We denote the security parameter by \( \kappa \). We use the abbreviation PPT to denote probabilistic polynomial-time and denote by \([n]\) the set of elements \( \{1, \ldots, n\} \) for some \( n \in \mathbb{N} \), and by \([a]_i\) the \( i^{th} \) element \( a_i \) from the set \( a \). For an \( \text{NP} \) relation \( \mathcal{R} \), we denote by \( \mathcal{R}_x \) the set of witnesses of \( x \) and by \( \mathcal{L}_\mathcal{R} \) its associated language. That is, \( \mathcal{R}_x = \{w \mid (x, w) \in \mathcal{R}\} \) and \( \mathcal{L}_\mathcal{R} = \{x \mid \exists w \text{ s.t. } (x, w) \in \mathcal{R}\} \). In order for \( \mathcal{R} \) to be an \( \text{NP} \) relation in the first place it is, of course, necessary that the witness length be polynomially-bounded, or more precisely, that there is a \( k \in \mathbb{N} \) such that \( \mathcal{R} \subseteq \bigcup_{i=0}^{\infty} \{0, 1\}^i \times \bigcup_{j=0}^{k} \{0, 1\}^j \).

**Definition 2.1.** A function \( f : \mathbb{N} \to \mathbb{R} \) is negligible if and only if it is asymptotically less than one divided by any given polynomial. More precisely, \( f \) is negligible if and only if it is true that for all \( k \in \mathbb{N} \) there exists an \( N \in \mathbb{N} \) such that for all \( n \in \mathbb{N} \) such that \( n > N \) it is true that \( f(n) \leq n^{-k} \).
2.3 One-Way Functions

Definition 2.2. Let \( f : \{0,1\}^* \rightarrow \{0,1\}^* \) be a deterministic polynomial-time computable function. \( f \) is a one-way function if and only if for every probabilistic polynomial-time machine \( A \), there exists a negligible function \( \epsilon \) such that

\[
\Pr[x \leftarrow \{0,1\}^n; y = f(x) : f(A(1^n, y)) = y] \leq \epsilon(n).
\]

Definition 2.3 (Regular One-Way Functions). Let \( f : \{0,1\}^* \rightarrow \{0,1\}^* \) be a one-way function. \( f \) is a regular one-way function if and only if there exists a function \( \alpha : \mathbb{N} \rightarrow \mathbb{N} \) such that for every \( n \in \mathbb{N} \) and every \( x \in \{0,1\}^n \) the following is true:

\[
|f^{-1}(f(x))| = \alpha(n).
\]

We do not assume that the regularity \( \alpha \) of a function \( f \) can be computed in probabilistic polynomial time. We often assume one-way functions are length preserving, i.e. for all \( n \in \mathbb{N} \) it is true that \( f(\{0,1\}^n) \subseteq \{0,1\}^n \).

Definition 2.4. Let \( f : \{0,1\}^* \rightarrow \{0,1\}^* \) be a one-way function and let \( b : \{0,1\}^* \rightarrow \{0,1\} \) be a deterministic polynomial-time computable function. \( b \) is a hardcore predicate of \( f \) if and only if for every probabilistic polynomial-time machine \( A \) there exists a negligible func-
tion $\epsilon$ such that for all $n \in \mathbb{N}$

$$\Pr[x \leftarrow \{0,1\}^n; y = f(x) : A(1^n, y) = b(x)] \leq \frac{1}{2} + \epsilon(n).$$

If $f$ is a one-way function then [GL89] prove that the one-way function $f'$ defined as $f'(x, r) = (f(x), r)$ (where $|x| = |r|$ if $|x| + |r|$ is even and $|x| = |r| + 1$ otherwise) admits the hardcore predicate $b(x, r) = (\sum x_i \land r_i) \mod 2$ where $x_i$ is the $i^{th}$ bit of $x$ and $r_i$ is the $i^{th}$ bit of $r$. We call this predicate the GL bit of $f'$.

**Definition 2.5 (Universal One-Way Hash Function).** Let $\ell \in \mathbb{N}$. Let $G = \{g_{n,k}\}_{n \in \mathbb{N}, k \in \mathcal{K}_n}$ be a family of functions such that $g_{n,k} : \{0,1\}^{n+\ell} \rightarrow \{0,1\}^n$ and there is a deterministic polynomial-time Turing machine $E$ with the property that for all $n \in \mathbb{N}$ and for all $k \in \mathcal{K}_n$ and for all $x \in \{0,1\}^{n+\ell}$ it is true that $E(1^n, k, x) = g_{n,k}(x)$. We also require that $\mathcal{K}_n$ can be efficiently sampled. $G$ is a universal one-way hash function family (UOWHF) if and only if it is true that for any probabilistic polynomial-time Turing machine $A$ there is a negligible function $\epsilon$ such that for all $n \in \mathbb{N}$ it is true that

$$\Pr[(x_1, \sigma) \leftarrow A(0, 1^n);$$

$$k \leftarrow \mathcal{K}_n;$$

$$x_2 \leftarrow A(1, x_1, \sigma, k):$$

$$x_1 \neq x_2 \land g_{n,k}(x_1) = g_{n,k}(x_2)] \leq \epsilon(n).$$
Universal one-way hash function families [NY89] as defined above enjoy the property of target collision resistance. Next, we define the seemingly weaker notion of second preimage resistance where the adversary cannot find a collision for randomly chosen input and key. It is well-known how to construct UOWHFs from second preimage resistant families.

**Definition 2.6 (Second Preimage Resistance).** Let \( G = \{g_{n,k}\}_{n \in \mathbb{N}, k \in \mathcal{K}_n} \) be a family of functions such that \( g_{n,k} : \{0, 1\}^{n+\ell} \rightarrow \{0, 1\}^n \) and there is a deterministic polynomial-time Turing machine \( E \) with the property that for all \( n \in \mathbb{N} \) and for all \( k \in \mathcal{K}_n \) and for all \( x \in \{0, 1\}^{n+\ell} \) it is true that \( E(1^n, k, x) = g_{n,k}(x) \). We also require that \( \mathcal{K}_n \) can be efficiently sampled. \( G \) is a second preimage resistant hash function family if and only if it is true that for any probabilistic polynomial-time Turing machine \( A \) there is a negligible function \( \epsilon \) such that for all \( n \in \mathbb{N} \) it is true that

\[
\Pr[x_1 \leftarrow \{0, 1\}^{n+\ell}, \\
k \leftarrow \mathcal{K}_n; \\
x_2 \leftarrow A(x_1, k) : \\
x_1 \neq x_2 \land g_{n,k}(x_1) = g_{n,k}(x_2)] \leq \epsilon(n).
\]


2.4 Indistinguishability

**Definition 2.7** (Probability Ensemble). A probability ensemble is a set of probability distributions. We use the notation \( \{q(n, x, y)\}_{n \in \mathbb{N}, x \in A, y \in f(x)} \) to mean the set of probability distributions parameterized by \( x \in A \) and \( y \in f(x) \) for some \( f, q, \) and \( A \) that are defined in context. The first term in the subscript often defines a natural number (usually \( n \)), used to define properties of distribution \( n \) in the infinite sequence. The following terms are parameters, each taken from a set. The value of a parameter may depend on the values of earlier parameters, including \( n \). The term inside the braces is a probability distribution parameterized by any valid assignment of the parameters as they are defined in the subscript.

**Definition 2.8** (Indistinguishable Ensembles). Let \( e_1 = \{A_{n,x}\}_{n \in \mathbb{N}, x \in X} \) and \( e_2 = \{B_{n,y}\}_{n \in \mathbb{N}, y \in Y} \) be probability ensembles. Let \( g : \mathbb{N} \rightarrow \mathbb{Q}^+ \). \( e_1 \) and \( e_2 \) are computationally indistinguishable with deviation gap \( g \) if and only if it is true that for all probabilistic polynomial-time Turing machines \( D \) there is a negligible function \( \epsilon \) such that for all \( n \in \mathbb{N} \) and for all \( x \in X \) and for all \( y \in Y \) it is true that

\[
\left| \Pr[z \leftarrow A_{n,x} : D(z) = 1] - \Pr[z \leftarrow B_{n,y} : D(z) = 1] \right| \leq g(n) + \epsilon(n).
\]

When we say that two ensembles are computationally indistinguishable without specifying a deviation gap, we mean that they are computa-
tionally indistinguishable within deviation gap $o$.

**Definition 2.9.** Let $n \in \mathbb{N}$ and let $X$ and $Y$ be probability distributions over a finite domain $\Omega \subseteq \{0,1\}^n$. The statistical distance between $X$ and $Y$ is defined to be

$$SD(X, Y) = \frac{1}{2} \sum_{w \in \Omega} |\Pr[X = w] - \Pr[Y = w]|.$$ 

$X$ and $Y$ are $\epsilon$-close if and only if $SD(X, Y) \leq \epsilon$.

**Definition 2.10.** Let $e_1 = \{X_n\}_{n \in \mathbb{N}}$ and $e_2 = \{Y_n\}_{n \in \mathbb{N}}$ be probability ensembles. $e_1$ and $e_2$ are statistically close (denoted $e_1 \approx_s e_2$) if and only if it is true that for all $n \in \mathbb{N}$ it is true that $SD(X_n, Y_n) \leq \epsilon(n)$.

**Definition 2.11 (RF$_n$).** For $n \in \mathbb{N}$, we define $RF_n$ to be the uniform distribution over all functions $f : \{0,1\}^n \rightarrow \{0,1\}^n$.

**Definition 2.12 (Pseudorandom Function).** A set $\{f_{n,k}\}_{n \in \mathbb{N}, k \in \{0,1\}^n}$ is a pseudorandom function family if and only if it can be computed in worst-case polynomial time and for all oracle probabilistic polynomial-time Turing machines $M$ it holds that there is a negligible function $\epsilon$ such that for all $n \in \mathbb{N}$ it holds that

$$|\Pr[s \leftarrow \{0,1\}^n : M^{f_{n,s}}(1^n) = 1] - \Pr[A \leftarrow RF_n : M^A(1^n) = 1]| \leq \epsilon(n).$$

We sometimes ignore the fact that pseudorandom functions (PRFs) cannot directly be queried on strings longer than $n$ bits once instantiated.
with a random seed of length \( n \). There exist standard techniques to extend the domain of PRFs as needed. We skip these details in our proofs.

## 2.5 Pseudorandom Generators

**Definition 2.13.** Let \( \ell : \mathbb{N} \to \mathbb{N}^+ \) be a deterministic polynomial-time computable function. Let \( G_n : \{0, 1\}^n \to \{0, 1\}^{n+\ell(n)} \) be a deterministic polynomial-time computable function. \( \{G\}_{n \in \mathbb{N}} \) is a pseudorandom generator if and only if for every probabilistic polynomial-time Turing machine \( A \), there exists a negligible function \( \epsilon \) such that

\[
\left| \Pr[x \leftarrow \{0, 1\}^n; y = G(x) : A(1^n, y) = 1] - \Pr[y \leftarrow \{0, 1\}^{n+\ell(n)} : A(1^n, y) = 1] \right| \leq \epsilon(n).
\]

## 2.6 Interactive Turing Machines

An interactive Turing machine (ITM) is a Turing machine with a constant number of work tapes, a random tape, a read-only input tape, and a write-only output tape augmented with features to allow interaction. These features are: an identity \( \sigma \in \{1, 2\} \), an identity tape with one cell, an outbox tape, and an inbox tape. When two interactive Turing machines \( (A, B) \) are run on an input \( (x, y) \), we let them share the identity tape, and let the inbox tape of each be the same as the outbox tape of the other.
A gets $x$ as input and $B$ gets $y$ as input. We may also specify a common input given to both. The only cell of the identity tape initially contains the character 1, and the communication tapes are initially empty. During the execution only the machine whose identity is stored on the identity tape is active. The active machine runs until it sets the content of identity tape to be the identity of the other machine, or until it halts. If the active machine halts, the interaction halts. We require that if the active machine sets the content of the identity tape to be the identity of the other machine that it also has written a special end of message symbol $\$ to its outbox exactly once since the step when the active machine most recently became active, and we require that it not write this symbol on its outbox tape under any other circumstances. We can permit it to writes this symbol on its outbox under other circumstances if it uses “escape sequences” but we omit the details. When we say that an ITM sends a string $x$ we mean that it writes $x\$ to its outbox then writes the opposite of its identity on its identity tape. The active machine may read its inbox, and may write to its outbox. It may not move its head left on either communication tape, but it may choose to keep the head stationary. The inactive machine does not change its internal state or move its heads until it becomes active. If an interactive Turing machine has an oracle, then we augment it with two extra tapes: an oracle query tape, and an oracle response tape, and the definition of the machine must specify a special oracle query finite state and a distinct special oracle response finite state. Whenever the machine
enters the oracle query finite state, the next configuration of the machine is one where all the heads are in the same place, the finite state is the oracle response finite state, and the contents of the oracle response tape are replaced with the output of the oracle when queried on the contents of the oracle query tape. We denote by $\langle A(x), B(y) \rangle (z)$ the random variable representing the “joint view” of the interaction: $x, y$, the common input $z$, the contents of the random tape of $A$ and $B$, the final contents of the each output tape, the final contents of each communication tape, and the list of queries and responses made to each oracle, if present. When $v$ is a joint view of an interaction and $i \in \{1, 2\}$, we denote by $\text{view}_i(v)$ the part of the joint view visible to the machine with identity $i$: the contents of its input tape, output tape, random tape, both communication tapes, and the list of all oracle queries and responses if it has an oracle. When $v$ is a random variable representing the joint view of an interaction we define the random variable $\text{view}_i(v)$ in the natural way.

When we use the notation $\langle A(x), B(y) \rangle$, $A$ will always have identity 1 and $B$ will always have identity 2. In the context of interactive proofs, the machine with identity 1 is the prover and the machine with identity 2 is the verifier. The prover will usually be given an $\text{NP}$ witness as an auxiliary input to help in the proof.

An ITM $A$ is a polynomial-time interactive Turing machine if and only if there is a $c \in \mathbb{N}$ and a $k \in \mathbb{N}$ such that for all inputs $x \in \{0, 1\}^n$ the total number of steps $A(x)$ takes in the entire interaction is at most $cn^k + c$. This
upper bound must hold regardless of the machine $A$ is interacting with, as long as the other machine always takes a finite number of steps.

**Definition 2.14** (Interactive Argument). A pair of polynomial-time ITMs $(P, V)$ are an interactive argument for a language $L \in \text{NP}$ with witness relation $W$ if and only if the following two conditions hold:

- (validity) for all $n \in \mathbb{N}$ and for all $(x, w) \in W$ such that $|x| = n$ it holds that $\Pr[v \leftarrow \text{view}_2(\langle P(w), V \rangle(x)) : v \text{ accepts}] = 1$, and
- (soundness) for every polynomial-time interactive Turing machine $P^*$ there is a negligible function $\epsilon$ such that for all $n \in \mathbb{N}$ and $x \in \{0, 1\}^n$ such that $x \not\in L$ it is true that $\Pr[v \leftarrow \text{view}_2(\langle P^*, V \rangle(x)) : v \text{ accepts}] \leq \epsilon(n)$.

An interactive proof is like an interactive argument, except that the honest prover may take exponential time and the soundness property must hold against even exponential-time dishonest provers. We will not consider interactive proofs that do not have a polynomial-time prover.

**Definition 2.15** (Witness-Indistinguishable Interactive Argument). Let $L$ be a language in $\text{NP}$ with witness relation $W$. An interactive argument $(P, V)$ for $L$ is witness indistinguishable if and only if it is true that for all probabilistic polynomial-time Turing machines $A$ it is true that there is a negligible function $\epsilon$ such that for all $n \in \mathbb{N}$ and for all $x \in L \cap \{0, 1\}^n$
and for all $w_1, w_2$ such that $\{(x, w_1), (x, w_2)\} \subseteq W$ it is true that

$$\left| \Pr[v \leftarrow \text{view}_2(\langle P(w_1), V \rangle(x)) : A(v) = 1] - \Pr[v \leftarrow \text{view}_2(\langle P(w_2), V \rangle(x)) : A(v) = 1]\right| \leq \epsilon(n).$$

**Definition 2.16 (Argument of Knowledge).** An interactive argument $(P, V)$ for $L \in \mathbf{NP}$ with witness relation $W$ is an argument of knowledge if and only if there is a polynomial-time interactive Turing machine $E$ such that for any nonuniform polynomial-time interactive Turing machine $P^*$ there is a negligible function $\epsilon$ such that for all $n \in \mathbb{N}$ and for all $x \in L \cap \{0, 1\}^n$ it is true that

$$\Pr[(x, E^{P^*(x)}(x)) \in W(x)] \geq \Pr[v \leftarrow \text{view}_2(\langle P^*, V \rangle(x)) : v \text{ accepts}] - \epsilon(n).$$

When we give $E$ oracle access to $P^*$ we mean that $E$ may simulate instances of $P^*$ but may only use the messages that $P^*$ sends. $E$ does not have access to the description of $P^*$, the random tape of $P^*$, the nonuniform advice string of $P^*$, or the number of steps $P^*$ takes to compute the messages it sends.
2.7 Universal Hash Function Families

Definition 2.17. Let \( \ell \in \mathbb{N} \). Let \( \mathcal{H} \) be a family of functions where each function \( h \in \mathcal{H} \) goes from \( \{0,1\}^{n+\ell} \) to \( \{0,1\}^n \). \( \mathcal{H} \) is an efficient family of pairwise-independent hash functions if and only if (i) the functions \( h \in \mathcal{H} \) can be described with a number of bits bounded by a polynomial in \( n \), (ii) there is a polynomial-time algorithm to evaluate \( h \in \mathcal{H} \) on an arbitrary input, and (iii) for all \( x \neq x' \in \{0,1\}^{n+\ell} \) and for all \( y, y' \in \{0,1\}^n \)

\[
\Pr_{h \in \mathcal{H}}[h(x) = y \land h(x') = y'] = 2^{-2n}.
\]

Efficient families of pairwise-independent hash functions are also known as a universal hash function families.

2.8 Commitment Schemes

Commitment schemes are used to enable one party, known as the \textit{sender} Sen, to commit itself to a value while keeping said value secret from the \textit{receiver} Rec (this property is called \textit{hiding}). Furthermore, in a later stage when the commitment is “opened”, it is guaranteed that the opening can yield only a single value determined in the committing phase (this property is called \textit{binding}): Sen cannot trick Rec into thinking that the string it committed to is different than it is. Some commitment schemes are \textit{statistically hiding}: the hiding property holds against computationally
unbounded (nonuniform) adversaries, but the binding property is only required to hold against computationally-bounded adversaries. We refer the reader to \cite{Goldreich01} for more details.

**Definition 2.18 (Commitment Schemes).** Let $\text{Com} = \langle \text{Sen}, \text{Rec} \rangle$ be a pair of polynomial-time interactive Turing machines and let $V$ and $D$ be polynomial-time Turing machines. Com is a commitment scheme with decommitment algorithm $D$ and verification algorithm $V$ if and only if the following three properties hold.

**Validity:** If the honest sender and honest receiver interact, then the sender sends the decommitment string, the verifier $V$ must always output 1. More formally, for any $n \in \mathbb{N}$ and for any $x \in \{0, 1\}^n$ it must be true that

$$\Pr[v \leftarrow \text{view}_1((\langle \text{Sen}(x), \text{Rec}(1^n) \rangle)); (x', d) \leftarrow D(v); x' = x \land V(x, d) = 1] = 1.$$ 

**Statistically Hiding:** For every polynomial-time interactive Turing machine $\text{Rec}^*$ there is a negligible function $\epsilon$ such that for any $n \in \mathbb{N}$ and any $\{s_1, s_2\} \subseteq \{0, 1\}^n$ the statistical distance between the following two distributions is at most $\epsilon(n)$:

- $\{\text{view}_2((\langle \text{Sen}(s_1), \text{Rec}^*(1^n) \rangle))\}$
• \( \{ \text{VIEW}_2(\langle \text{Sen}(s_2), \text{Rec}^*(1^n) \rangle) \} \).

**Computationally Binding:** For any polynomial-time ITM sender \( \text{Sen}^* \) and any probabilistic polynomial-time Turing machine \( C \) there is a negligible function \( \epsilon \) such that for all \( n \in \mathbb{N} \) and for all \( x \in \{0, 1\}^n \) the following is true:

\[
\Pr[v \leftarrow \text{VIEW}_1(\langle \text{Sen}^*(x), \text{Rec}(1^n) \rangle); \\
(y_1, d_1) \leftarrow D(v) ; \\
(y_2, d_2) \leftarrow C(v) : \\
y_1 = x \land V(y_1, d_1) = 1 \land y_2 \neq x \land V(y_2, d_2) = 1] \leq \epsilon(n).
\]
Chapter 3

Round Complexity of Precise Zero Knowledge

In this chapter, we give a precise zero-knowledge argument for any language in $\text{NP}$ which achieves a certain round complexity in relation to the precision of its simulator, and prove that round complexity to be optimal (except for the case of languages in $\text{BPP}$, for which no interaction is necessary). To prove optimality, we first formally define a framework of what simulators of precise zero-knowledge arguments are allowed to do, especially in terms of how they interact with simulated (possibly dishonest) verifiers. Our simulators are “mostly” black box in the sense that they are allowed to “step” the execution of a verifier as it computes its next message, and to rewind verifiers to previous configurations. This means simulators can stop waiting for an instance of the verifier that is taking too long and instead interact with a different instance. However, we do not allow simulators direct access to a description of the (possibly dishonest) verifier to be simulated. Note that the definition of the simulator
may not depend on the verifier to be simulated. Our results hold even for simulators that are only required to function for dishonest verifiers that may delay messages of the honest verifier but may not change them.

3.1 A Framework for Partial Black-Box Zero-Knowledge Simulators

To facilitate proving lower bounds, we introduce a “mediator” interactive Turing machine that we call the facilitator $F$. This machine $F$ encapsulates the access to the malicious verifier $V^*$ of the simulator, acting as a remote server that can run multiple instances of a verifier on the behalf of the simulator.

3.1.1 The Facilitator $F$

We consider a universal facilitator $F$. $F$ is an oracle interactive Turing machine. We require that the tape alphabet contain all of ASCII in order to make the message labels (“STEP”, “CLONE”, etc) from $S$ easier for the reader to understand and remember in the proofs. This requirement can, of course be reduced to a binary alphabet (plus a blank symbol) using standard techniques, and will not affect the results because all such messages are of constant length and would thus remain so once translated into a binary tape alphabet (plus a blank symbol).

The messages $S$ sends to $F$ must be of a certain format. If they are not
then $F$ aborts the computation. Each message from $S$ must encode a pair of strings as $(\text{sid}, j)$, or $F$ aborts. The string $\text{sid}$ must be a nonnegative integer encoded in binary in the standard, “big endian” way with no preceding 0 characters unless the integer represented is exactly zero in which case it must be represented as 0, not as the empty string. The $\text{sid}$ indicates to $F$ the instance of $V$ that $S$ wants to work with. For each session $F$ keeps an instance of both $V$ and $V^*$. The string $j$ must be of the form “STEP”, “CLONE $z$”, “SEND $z$”, “INIT $z$”, or “FINALIZE” for some string $z$.

We assume that $V^*$ is a verifier that sends the same messages as $V$ would but may choose to delay those messages. It may help to think of $V$ as a subroutine of $V^*$.

1. When $F$ receives an instruction of the form $(\text{sid}, \text{INIT } z)$, if $\text{sid}$ is strictly greater than the current number of sessions then $F$ aborts. This ensures that sessions are numbered from 0 to $q - 1$ for some integer $q$. If $F$ has received an earlier INIT instruction using the same random seed $z$ then $F$ aborts. Next, $F$ erases the old session with session identifier $\text{sid}$ (if any) and creates a new instance of $V^*(x)$ and sets the input tape to $x$ and the identity tapes to 1. The random tape is initialized with $z$. $F$ then replies with an empty string.

2. When $F$ receives an instruction of the form $(\text{sid}, \text{SEND } z)$ it aborts if the identity tape of session $\text{sid}$ is 2 or if $V^*$ instance $\text{sid}$ has halted,
otherwise it appends $z$ to the inbox of $V^*$ instance $\mathbf{sid}$. $F$ sets the identity tape of this $V^*$ instance to 2. $F$ sends an empty message back to $S$.

3. When $F$ receives an instruction of the form $(\mathbf{sid}, \text{STEP})$, it aborts if $V^*$ instance $\mathbf{sid}$ has halted. $F$ then checks if the identity tape of $V^*$ session $\mathbf{sid}$ has a 1 on it. If so, $F$ aborts. Otherwise, $F$ simulates one step of the $V^*$ instance for session $\mathbf{sid}$. After one step is complete, $F$ checks the identity tape of $V^*$ again. If the identity tape reads 2 then $F$ sends an empty string back to $S$. If not, then $F$ reads the most recent message in the outbox of $V^*$. Let $m$ be this message. $F$ sends the message $(\text{MESSAGE}, m)$.

4. When $F$ receives an instruction of the form $(\mathbf{sid}, \text{CLONE } z)$ it erases the old session with session identifier $z$ (if any) then copies the entire runtime configuration of session $\mathbf{sid}$ into session $z$. $F$ then sends the empty string to $S$.

5. When $F$ receives an instruction of the form $(\mathbf{sid}, \text{FINALIZE})$, if $F$ has received a message of this form before then $F$ aborts. $F$ checks whether $V^*$ instance $\mathbf{sid}$ has halted. If not then $F$ aborts. If so, $F$ sends the view of that $V^*$ instance to $S$. Once $F$ has received an instruction of this form it ignores all future messages regardless of content and reply with the empty string.

We also require that the final output of $S$ must be the reply to a
FINALIZE message that $S$ sent to $F$.

In our analysis we prove our lower bounds by analyzing the number of STEP messages that $S$ sends. We ignore the time that $S$ takes to do all other operations, including writing the $\text{sid}$ on its communication tape and our results will still hold, even if $S$ is allowed to spend an arbitrarily-high but polynomial amount of time deciding what messages to send to $F$.

### 3.1.2 Partial Black-Box Precise Zero Knowledge

**Definition 3.1 (Precise Zero Knowledge).** Let $L$ be a language in $\text{NP}$ with witness relation $R_L$, let $(P, V)$ be an interactive proof (argument) system for $L$ where $V$ is a polynomial-time interactive Turing machine and $p : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ a function monotonically increasing in both its arguments. Let $F$ be the universal partial black-box precise zero-knowledge facilitator. $(P, V)$ is a partial black-box precise zero-knowledge proof (argument) system for $L$ with precision $p$ if and only if there is a uniform polynomial-time interactive Turing machine $S$ such that for every nonuniform polynomial-time interactive Turing machine $V^*$ which functions the same as $V$ except that it may delay the messages of $V$ the following two conditions hold:
For every $x \in L$,

$$\text{Pr}\left[\text{view} \leftarrow \langle S, F(V^*) \rangle(x); \right.$$

$$\text{out} = \text{OUT}_1(\text{view}) :$$

$$\text{STEPS}_1(\text{view}) \leq p(|x|, \text{STEPS}_2(\text{out}))\right] = 1.$$

2. The following two ensembles are computationally indistinguishable:

- $\{\text{VIEW}_2(\langle P(w), V^* \rangle(x))\}_{n \in \mathbb{N}, x \in L \cap \{0,1\}^n, w \in R_L(x)}$
- $\{\text{OUT}_1(\langle S, F(V^*) \rangle(x))\}_{n \in \mathbb{N}, x \in L \cap \{0,1\}^n}$.

For our second result we prove the lower bound even if the simulated distribution and the distribution of the honest interaction of the prover and verifier are only indistinguishable for constant deviation gaps. An interactive proof is $\delta$-weakly-precise zero-knowledge with precision $p$ if and only if it is a precise zero-knowledge proof with precision $p$ when only $\delta$ deviation gap is required of the simulator’s output.

### 3.1.3 A Generic Lower Bound

The main idea behind this lemma is that if the simulator can, with non-negligible probability, output accepting transcripts without rewinding then we can construct a dishonest prover which convinces the honest verifier to accept with nonnegligible probability whenever $x \in L$. Said
prover is dishonest because it still attempts to convince the honest verifier to accept even when \( x \notin L \), though we cannot fairly blame it for doing so as it does not know whether this is the case during the interaction. This allows us to construct a (noninteractive) Turing machine that runs polynomially-many independent instances of the interaction of the dishonest prover and the honest verifier and accepts if it produces at least one accepting transcript of this interaction.

For Lemma 3.2 we require the notion of a “nonrewinding” interaction. To help explain this concept at a high level, we briefly consider an alternative definition of the universal facilitator that does not allow the simulator to make copies of \( V^* \) instances and does allow the simulator to rewind \( V^* \) instances to a previous configuration. In this alternative model of precise zero knowledge, a nonrewinding interaction is one in which the simulator outputs its interaction with a \( V^* \) instance that the simulator did not rewind. Note that the simulator may rewind other instances of \( V^* \) and still produce a nonrewinding interaction. Our proofs are simpler overall if we use a facilitator that permits copying instead of rewinding, even though it requires a different and more complicated version of the concept of a nonrewinding interaction. We now cease consideration of the alternative universal facilitator and present a variation of the definition of a nonrewinding interaction corresponding to the universal facilitator as we have defined in 3.1. At a high level, a nonrewinding interaction is one in which the simulator may make copies of instances of \( V^* \) and may even
send different messages to different copies of the same instance, but must output an interaction with an instance of $V^*$ that the simulator completed without using this ability in a meaningful way. Naturally we will have to define what we mean by “in a meaningful way”.

Since we will be analyzing simulators in terms of number of STEP messages sent to $F$, not the number of computation steps of the simulator itself, we can assume that the simulator never overwrites a session that has been initialized with another. This also means that we can ignore the cost of writing the sid of a particular session in the message the simulator sends to $F$.

We now describe the nonrewinding condition informally. Whether a session is nonrewinding is a property of how it was produced by the simulator, not a property of the messages sent between the simulated prover and verifier. The simulator may make multiple copies of a (simulated) interaction between prover and verifier without breaking the nonrewinding condition for the session, but if the simulator sends at least one message to more than one copy, all of them are no longer nonrewinding sessions. If the simulator makes at least one copy and sends messages to both the original and a copy then the nonrewinding condition is broken. If the simulator initializes two or more sessions with the same initial randomness, then the nonrewinding condition is broken. Otherwise a session is nonrewinding.

More formally, a session $i$ is nonrewinding in the interaction
\langle S, F(V^*) \rangle(x) \text{ if and only if for all sessions } j \text{ with the same } (V^*) \text{ random tape contents as } i, \text{ the communication tapes of } i \text{ are each a prefix of the corresponding communication tapes of } j \text{ or the communication tapes of } j \text{ are each a prefix of the corresponding communication tape of } i. \text{ Simulators cannot initialize multiple sessions with the same random string, but simulators may make use CLONE messages to make copies of a session that has been initialized with a given string before sending any other messages regarding that session. A joint view of an interaction between the simulator and verifier is NONREWINDING if and only if the simulator outputs a NONREWINDING transcript.}

Lemma 3.2. Let \( L \) be a language in NP with a partial black-box precise zero-knowledge argument system \((P, V)\) with \( r \) rounds and simulator \( S \). Suppose there is a \( c \in \mathbb{N} \) and a polynomial-time interactive Turing machine \( V^* \) such that for all \( n \in \mathbb{N} \) and all strings \( x \in \{0,1\}^n \) it holds that \( S \) never makes more than \( n^c \) sessions on input \( x \) and

\[
\Pr[j \leftarrow \langle S, F(V^*) \rangle(x) : j \text{ is NONREWINDING}] \geq \frac{1}{n^c}.
\]

Then \( L \in \text{BPP} \).

Proof. This proof involves several related machines, so to reduce ambiguity we minimize our usage of pronouns referring to them when describing how said machines relate to one another. This policy yields statements that are more awkward and less likely to be misinterpreted than alter-
native phrasings that do not follow this policy. We consider this a net improvement.

At a high level, the idea behind this proof is that if $S$ produces an accepting transcript while receiving exactly $r$ messages in total from all of the $V^*$ instances $S$ creates (via $F$), then it must be true that $S$ managed to send convincing prover messages to a single nonrewinding session of $V^*$. We construct an interactive Turing machine $P^*$, which we will use as a dishonest prover. $P^*$ on arbitrary input $x$ simulates $\langle S, F(V^*) \rangle(x)$ with a slight variation on its normal execution. $P^*$ guesses the session $S$ will use to produce $S$’s verifier messages in $S$’s output and $P^*$ forwards these messages to the verifier $P^*$ is interacting with, and forwards the responses back to $F$ as if the simulated verifier made the response that the external verifier made. When $x \in L$, this dishonest verifier convinces $V$ with nonnegligible probability. Since $(P, V)$ is a sound interactive proof, it holds that if $x \notin L$ then $P^*$ will probably fail to convince the honest verifier. To prove $L \in \text{BPP}$ we construct a probabilistic polynomial-time Turing machine $D$ which simulates multiple sessions of an interaction between the dishonest prover and the honest verifier and accepts if and only if the verifier accepts at least once, then modify $D$ slightly to be correct on all strings $x$ instead of all but finitely-many of them.

Note that the behavior of the $V^*$ instance and the external verifier have identical behavior. We conclude from this that $P^*$ convinces the external verifier with nonnegligible probability. Then, from the soundness
property of the interactive proof \((P, V)\), we show that \(P^*\) cannot convince an external verifier if it started on an input \(x \notin L\). This will enable us to construct a probabilistic polynomial-time distinguisher which decides the language \(L\) by simply emulating the interaction of \(P^*\) with an honest verifier \(V\).

More formally, \(P^*\) on input \(x\) internally emulates an execution of \(⟨S, F(V^*)⟩(x)\) while externally interacting with an external verifier which we will for convenience assume to be the honest verifier \(V\). \(P^*\) randomly selects an integer \(q\) uniformly at random from zero to \(n^c - 1\) (inclusive) representing the session that \(P^*\) will effectively replace with the external verifier. If a session is marked then \(P^*\) will treat it in a special way we describe later. We now describe how sessions are marked and unmarked.

- If \(S\) sends a message of the form \((\text{sid}, \text{INIT} \ z)\) and \(\text{sid} = q\) then \(P^*\) marks the session \(\text{sid}\).

- If \(S\) sends a message of the form \((i, \text{CLONE} \ j)\) message (requesting that \(F\) clone session \(i\) over session \(j\)) then \(P^*\) marks session \(j\) if and only if session \(i\) is marked.

- If \(S\) sends a message of the form \((i, \text{SEND} \ j)\) and session \(i\) is marked then \(P^*\) sends \(j\) to the external verifier and unmarks all other sessions.

- If \(S\) sends a message of the form \((i, \text{STEP})\) and session \(i\) is marked and the associated \(V^*\) instance invokes \(V\) to get the next message of
V, \( P^* \) intercepts this invocation and makes the \( V^* \) instance behave as if the next message of \( V \) were the most recent message that \( P^* \) got from the external verifier.

Next we analyze the success probability, the probability that \( P^* \) convinces the external verifier \( V \).

First we recall from our framework that a dishonest verifier \( V^* \) is only allowed to delay responses from the honest verifier \( V \). Therefore for any \( x \in L \), the probability that \( V^* \) outputs an accepting transcript when interacting with an honest prover is 1. From the (precise) zero-knowledge property of \( (P, V, S) \) it holds that there is a negligible function \( \epsilon_1 \) such that for every \( x \in L \), \( \text{out}_1(\langle S, F(V^*) \rangle(x)) \) is the (verifier) view of an accepting interaction with probability at least \( 1 - \epsilon_1(|x|) \). This means that

\[
\Pr[j \leftarrow \langle S, F(V^*) \rangle(x) : j \text{ is nonrewinding} \quad \text{and } S \text{ outputs an accepting transcript}] \geq \frac{\epsilon_1(n)}{n^c}.
\]

So for sufficiently large \( n \) we have that

\[
\Pr[j \leftarrow \langle S, F(V^*) \rangle(x) : j \text{ is nonrewinding} \quad \text{and } S \text{ outputs an accepting transcript}] \geq \frac{1}{2n^c}.
\]
From our framework we have that if $S$ outputs an accepting transcript without rewind then all the messages of the transcript must have come from the nonrewinding session.

Recall that the session chosen by $P^*$ to be marked is chosen independently of the behavior of the simulator. This means that the probability that $S$ outputs an accepting transcript and is nonrewinding in the internal emulation by $P^*$ conditioned on $P^*$ marking the right session is at least $\frac{1}{2^{nc}}$.

Next we observe that for the marked session, the distribution of the verifier messages $S$ receives from the external verifier interacting with $P^*$ is identical to the distribution of messages $S$ would have received from $V^*$ in the internal emulation of $F$. This means that the internal emulation by $P^*$ when $P^*$ chooses the correct session proceeds identically to an interaction between $S(x)$ and $F(x,V^*)$, conditioned on there being an accepting transcript in $S$ in a view that is nonrewinding. Therefore, conditioned on $P^*$ marking the right session and the internal emulation having an accepting transcript in the simulator’s output from a view that is nonrewinding, it must hold that $P^*$ convinces the external verifier $V$. Since $P^*$ chooses the right session with probability $\frac{1}{n^c}$, we conclude that on any input $x \in L \cap \{0,1\}^n$, $P^*$ convinces the external verifier $V$ with probability at least $\frac{1}{2^{nc}}$ for all sufficiently large $n$.

To prove that $L \in \text{BPP}$ we will construct a decider based on $P^*$. $D(x)$ simulates $\langle P^*, V \rangle(x)$ and accepts if and only if the result is an accepting
transcript.

Since $P^*$ is a probabilistic polynomial-time interactive Turing machine and the interactive proof has negligible soundness error against probabilistic polynomial-time interactive Turing machine provers, for some negligible function $\epsilon_2$ we have that $\forall x \not\in L. \Pr[\text{out}_2(\langle P^*, V^* \rangle(x)) = 1] \leq \epsilon_2(|x|)$. From the preceding argument, we have that if $x \in L$ then $\Pr[\text{out}_2(\langle P^*, V^* \rangle(x)) = 1] \geq \frac{1}{2n^c}$. Therefore $D$ accepts all $x \in L$ with probability $\frac{1}{2n^c}$ and rejects all $x \not\in L$ with probability at least $1 - \epsilon_2(n)$. To construct a probabilistic polynomial-time machine that decides the language with probability 2/3, we define $D_2$ as the machine that repeats the experiment by $D n^{3c}$ times with independent randomness and accepts if any round of the experiment accepts. Since we only analyzed the success probability in terms of sufficiently-large inputs, we augment this final machine with a finite lookup table containing the information of whether strings shorter than this length are in $L$ to produce our final decider, $D_3$, which has all the required properties of this lemma.

3.2 Lower Bound for Arbitrary Precision

Let $g : \mathbb{N} \rightarrow \mathbb{Q}$ be an arbitrary polynomial-time computable function. For any $c \in \mathbb{N}$ we prove the lower bound for precision $p(n, t) = cg(n)t + n^{c-1}$ by constructing a malicious verifier $V^*$ and showing that there exists a set of transcripts $T$ such that $V^*$ and $T$ satisfy the conditions of Lemma 3.2.
We now describe our malicious verifier as a family of polynomial-time interactive Turing machines $V_{c,g}^*$ indexed by the constant $c$ and function $g$ as the precision function. $V_{c,g}^*(x)$ is a nonuniform interactive Turing machine that internally incorporates a simulated instance of $V$ and proceeds as we will now describe. $V_{c,g}^*$ sets the random tape of its simulated $V$ instance with $V_{c,g}^*$’s random tape. $V_{c,g}^*$ will use its nonuniform advice string to make choices that we will think of as random despite the fact that they are not. When $V_{c,g}^*$ receives a message $m$ in its inbox it sends it to the simulated $V$ and runs the simulation until it gets a response $m'$. Next, $V_{c,g}^*$ delays by $n^c$ steps. After this, $V_{c,g}^*$ samples a “random” bit that takes the value 1 with probability $1 - (cg)^{-1}$. If the bit is 1, then $V^*$ delays by an additional $crgn^c$ steps. $V_{c,g}^*$ then sends the message $m'$. This “random” bit is selected using the bits of $V_{c,g}^*$’s nonuniform advice string. $V_{c,g}^*$ uses the nonuniform advice string as the seed for a PRF. To get the “random” bits it uses to sample the biased coin it flips, $V_{c,g}^*$ computes the output of the PRF when applied to the concatenation of the random tape of $V_{c,g}^*$ and both communication tapes of $V_{c,g}^*$.

**Claim 3.3.** $V^*$ described above satisfies the conditions of Lemma 3.2.

**Proof.** Recall that our malicious verifier delays by an additional $crgn^c$ steps with probability $1 - \frac{1}{c^g}$ in every round. Therefore, on the honest
interaction with $P$ we have that

$$\Pr[t \leftarrow \langle P(w), V_{c,g}^* \rangle(x) : \text{steps}_2(t) < r n^c] \geq \frac{1}{(c g)^r}.$$  

From the precise zero-knowledge property of the interactive proof, we have that there is a negligible function $\epsilon_1$ such that

$$\Pr[t \leftarrow \langle S, F(V_{c,g}^*) \rangle(x) : \text{steps}_1(t) < cr g n^c + n^{c-1}] \geq \frac{1}{(c g)^r} - \epsilon_1(|x|).$$

Next, we show that if $S$ spends at most $cr g n^c + n^{c-1}$ steps, there is a nonnegligible probability that $S$ received exactly $r$ simulated messages from $V_{c,g}^*$.

When $S$ sends a message to $V_{c,g}^*$, without loss of generality $S$ waits for at least $n^c$ steps because $S$ receives no information about whether $V^*$ will delay by more than $n^c$ steps until that time. If, for each message $S$ sends to $V_{c,g}^*$, $S$ must wait $n^c$ steps for a response, then $S$ can send at most $c r g$ messages to $V_{c,g}^*$ instances. Let $E$ be the random event that occurs when $S$ gets exactly $r$ simulated $V_{c,g}^*$ messages from $F$. We want a lower bound on $\Pr[E]$. Let $E_i$ be the random variable that takes the value 1 if and only if $S$ receives a response to the $i^{th}$ prover message it sends to $V_{c,g}^*$ (via $F$). It holds that $\Pr[E] = \Pr[E_1 + \cdots + E_{c r g} = r]$. Since $V_{c,g}^*$ responds on a message within $n^c$ steps with probability $\left(\frac{1}{c g}\right)$, it holds
that $\Pr[E_i = 1] = \left(\frac{1}{cg}\right)$ for every $i$. Therefore from the binomial theorem,

$$\Pr[E] = \binom{crg}{r} \left(\frac{1}{cg}\right)^r \left(1 - \frac{1}{cg}\right)^{crg - r}$$

$$= \binom{crg}{r} \left(\frac{crg - 1}{r - 1}\right) \cdots \left(\frac{crg - r + 1}{1}\right) \left(\frac{1}{cg}\right)^r \left(1 - \frac{1}{cg}\right)^{crg - r}$$

$$\geq (cg)^r \left(\frac{1}{cg}\right)^r \left(1 - \frac{1}{cg}\right)^{crg - r}$$

$$\geq e^{-r}.$$

If $S$ gets exactly $r$ simulated $V^*$ messages from $F$ and $S$ succeeds, then $S$ cannot have rewinded. Therefore we can conclude that

$$\Pr[j \leftarrow \langle S, F(V^*_c)\rangle(x) : j \text{ is NONREWINDING}] \geq \frac{1}{2(ecg)^r}$$

which is nonnegligible. $\blacksquare$

**Corollary 3.4.** Let $c \in \mathbb{N}$. Let $g(n)$ be a polynomial. Let $L$ be a language in $\mathbf{NP}$ with a partial black-box precise zero-knowledge argument system $(P, V)$ with $r(n) \in O\left(\frac{\log n}{\log(g(n))}\right)$ rounds and simulator $S$ with precision $p(n, t) = cg(n)t + n^{c-1}$. Then $L \in \mathbf{BPP}$.

### 3.3 A Stronger Lower Bound

In this section we prove a lower bound for even nonaborting verifiers with simulators that have a constant deviation gap.
We now define the dishonest verifier $V^*_c$. As before, $V^*_c$ will use its nonuniform advice tape as the seed to a pseudorandom function. In each round, $V^*_c$ invokes its PRF on the concatenation of the contents of its random tape and each of its communication tapes. $V^*_c$ uses the output in order to sample the geometric distribution with an expected value of $n^c$, then delay by that many steps. Finally, $V^*_c$ simulates the honest verifier $V$ to determine the message it would send, and $V^*_c$ forwards this message to the machine $V^*_c$ is interacting with.

We will now show that $V^*_c$ satisfies the conditions of Lemma 3.2 for simulators with constant deviation gap $\delta$.

**Lemma 3.5.** Let $\delta$ be a constant. Let $c = 2\delta$. Let $q$ be a constant. Let $r(n) \leq q \log n$. Let $L$ be a language in $\text{NP}$ with interactive argument $(P, V)$ with $r$ rounds with precise simulator $S$ with precision $p(n, t) = (c - 1)t + n^{c-1}$ with deviation gap $\frac{1}{\delta}$. Then $V^*_c$ as described above satisfies that for every $x \in L \cap \{0,1\}^n$,

$$\Pr[j \leftarrow \langle S, F(V^*_c) \rangle(x) : j \text{ is nonrewinding}] \geq \frac{n^{-8\delta^2 q}}{4\delta}.$$

**Proof.** Let $V^*_c$ be the dishonest verifier defined above. From our framework we know that the view that $S$ outputs is the view of a $V^*_c$ session that $S$ simulates via $F$. Let $x$ be an arbitrary string in $L$ and let $n = |x|$.

The expected time that $V^*_c$ spends on each message is $n^c$, so by
Markov’s inequality we have that

\[
\Pr[t \leftarrow \langle P(w), V_c^* \rangle(x) : \text{steps}_2(t) < crn^c \cap t \text{ accepts}] \geq 1 - \frac{1}{c}.
\]

From the precise zero-knowledge property of the interactive proof we have that there is a negligible function \( \epsilon_1 \) such that

\[
\Pr[t \leftarrow \langle S, F(V_c^*) \rangle(x) : \text{steps}(t) < crn^c \cap t \text{ accepts}] \geq 1 - \frac{1}{c} - (1 - \frac{1}{\delta}) - \epsilon_1(n)
\]

\[
\geq \frac{1}{\delta} - \frac{1}{c} - \epsilon_1(n)
\]

\[
\geq \frac{1}{2\delta} - \epsilon_1(n).
\]

Immediately we have that

\[
\Pr[t \leftarrow \langle S, F(V_c^*) \rangle(x) : \text{steps}_1(t) < c(c - 1)rn^c + n^{c-1} \cap \text{OUT}_1(t) \text{ accepts}] \geq \frac{1}{2\delta} - \epsilon_1(n).
\]

For sufficiently-large \( n \) we have that

\[
\Pr[t \leftarrow \langle S, F(V_c^*) \rangle(x) : \text{steps}_1(t) < c(c - 1)rn^c + n^{c-1} \cap \text{OUT}_1(t) \text{ accepts}] \geq \frac{1}{3\delta}.
\]
If $S$ receives exactly $r$ messages and produces an accepting transcript then $S$ must be nonrewinding.

We round up the “time limit” on the number of steps $S$ takes from $c(c - 1)rn^c + n^{c-1}$ to $c^2rn^c$. Next we show that if $S$ spends at most $c^2rn^c$ steps then it (somewhat) likely received exactly $r$ messages from $V_c^*$. For each $STEP$ command that $S$ sends to $F$, the probability that it gets a response from the relevant instance of $V_c^*$ is $\frac{1}{n^c}$. Recall that $S$ runs for at most $c^2rn^c$ steps with probability at least $\frac{1}{3^c}$ and if that happens it sends at most $c^2rn^c$ $STEP$ messages. Suppose that $V_c^*$ used a random oracle instead of a PRF. Let $E_i$ be the indicator random variable that takes the value 1 if and only if $S$ receives a $V_c^*$ response on its $i^{th}$ $STEP$ message. Note that all $E_i$ are jointly independent because $V^*$ uses a random oracle to determine how long to wait. Let $E$ be the random variable denoting the sum over all $i$ of $E_i$. We want a lower bound on $\Pr[E = r]$. From the
binomial theorem, we have the following for sufficiently large $n$:

$$\Pr[E = r] = \binom{c^2 r n^c}{r} \left( \frac{1}{n^c} \right)^r \left( 1 - \frac{1}{n^c} \right)^{c^2 r n^c - r}$$

$$= \frac{c^2 r n^c}{r} \cdot \frac{c^2 r n^c - 1}{r - 1} \cdots \frac{c^2 r n^c - r + 1}{1} \left( \frac{1}{n^c} \right)^r \left( 1 - \frac{1}{n^c} \right)^{c^2 r n^c - r}$$

$$\geq \left( c^2 n^c \right)^r \left( \frac{1}{n^c} \right)^r \left( 1 - \frac{1}{n^c} \right)^{c^2 r n^c - r}$$

$$= c^{2r} \left( 1 - \frac{1}{n^c} \right)^{c^2 r n^c - r}$$

$$\geq c^{2r} e^{-2c^2 r}$$

$$= (2\delta)^{2r} e^{-8\delta^2 r}$$

$$= (2\delta)^{2r} e^{-2(2\delta)^2 r}$$

$$\geq (2\delta)^{2r} e^{-8\delta^2 q \log n}$$

$$\geq e^{-8\delta^2 q \log n}$$

$$\geq n^{-8\delta^2 q}.$$ 

Now we can conclude that

$$\Pr[j \leftarrow (S, F(V^*_c))(x) : j \text{ is nonrewinding}] \geq \frac{n^{-8\delta^2 q}}{3\delta}.$$ 

If we let $V^*_c$ use a random seed to its PRF instead of using a random
oracle, we have that there is a negligible function $\epsilon_2$ such that
\[
\Pr[j \leftarrow \langle S, F(V^*_c) \rangle(x) : j \text{ is nonrewinding}] \geq \frac{n^{-8\delta^2 q}}{3\delta} - \epsilon_2(n).
\]

For sufficiently-large $n$,
\[
\Pr[j \leftarrow \langle S, F(V^*_c) \rangle(x) : j \text{ is nonrewinding}] \geq \frac{n^{-8\delta^2 q}}{4\delta}.
\]

We observe that this is nonnegligible.

\section*{3.4 Constructions of Precise Zero-Knowledge Arguments for Arbitrary Precision}

In this section, we show how to construct precise zero-knowledge arguments for any language in $\textbf{NP}$ given a witness-indistinguishable argument for an $\textbf{NP}$-complete language. More specifically, we show how to produce a precise zero-knowledge argument with precision $p(n, t) = \ctg(n) + cn$ where $g(n) \in O(n^c)$.

Suppose that $(P, V)$ is a witness-indistinguishable argument for an $\textbf{NP}$-complete language $L$ with witness relation $W$. We will now describe how to construct a protocol $(P', V')$, which we will show is a precise zero-knowledge argument with the above precision. Let $f$ be any one-way function.

We will now (improperly) describe a precise argument of knowledge
for Hamiltonian path without formally defining precise arguments of knowledge. The only significant difference the definition requires from arguments of knowledge is that the extractor must have a runtime limited more precisely by the runtime of the prover it is extracting a witness from. The idea is reasonably straightforward but it would require a great deal of belaboring the details to formally define the framework for a precise argument of knowledge because our facilitator framework is only set up for simulators interacting with verifiers, not for extractors interacting with provers. We only use the precise argument of knowledge as a component to construct a precise zero-knowledge argument. Therefore we omit the formal definition of a precise argument of knowledge.

**Lemma 3.6.** Let $g(n)$ be a polynomial. Let $m(n) \in \omega(\frac{\log n}{\log g(n)})$. Assuming the existence of a constant-round statistically-hiding commitment scheme and a statistically-sound witness-indistinguishable argument for Hamiltonian path with linear precision, there is a statistically-sound witness-indistinguishable precise argument of knowledge for Hamiltonian path with precision $p(t, n) = ctg(n) + cn$ and round complexity $m(n)$.

**Proof.** First we observe that Blum’s interactive protocol for Hamiltonian path [Blu86] is witness indistinguishable and statistically sound when instantiated with statistically-hiding commitments, and has special soundness. Let $(P', V')$ be the protocol which sequentially repeats this protocol $m(n)$ times, and $V'$ accepts if and only if $V$ accepts in every repetition. We will show that this new protocol has the desired properties.
First observe that witness indistinguishability and statistical soundness are preserved under sequential repetition. All we have to show is that it is a precise argument of knowledge with the desired precision.

We now define a protocol for extracting a witness from any successful prover $P^*$. For each instance of the underlying argument we will attempt to rewind it $g(n) - 1$ times. Before attempting to rewind, we first simulate the honest interaction of $P^*$ and $V$ on this argument and record the number of steps $t_i$ it takes $P^*$ to produce the response to $V$’s challenge string. Next we rewind the simulation to just before $V$ sent its challenge string to $P^*$ and instead send a new random challenge. If $P^*$ responds within $t_i$ steps, then we use the special soundness property of the underlying argument to extract a witness from $P^*$. If no rewind attempt succeeds for any round then the simulator aborts.

We now will compute the probability that $S$ succeeds in rewinding a particular round. $S$ can fail if its challenge strings are not distinct, and $S$ can fail if all challenge strings after the first take longer for $P^*$ to produce a response to than the first challenge string it received. If the length of the challenge string is $\omega(\log n)$ then the probability that there are any duplicate challenges is negligible. Suppose there are $g(n)$ distinct challenge strings. Consider the number of steps that $P^*$ will run on each one. The only way that $S$ will fail to rewind is if the first challenge string was the fastest among $g(n)$ different challenges. The probability of this is at most $\frac{1}{g(n)}$. Combining this we get that the probability that $S$ fails to rewind is at
most $\frac{1}{2g(n)}$, and the probability that $S$ fails to rewind on all $m(n)$ rewind slots is at most $\frac{1}{(2g(n))^{m(n)}}$. By observation, when $m(n) = \omega\left(\frac{\log n}{\log g(n)}\right)$, this probability is negligible. Therefore $S$ will succeed in extracting a witness from a successful prover $P^*$ except with negligible probability.

Using this lemma we are ready to describe and prove our main positive result.

**Theorem 3.7.** Let $g(n)$ be a polynomial. Let $m(n) \in \omega\left(\frac{\log n}{\log g(n)}\right)$. Let $f(x)$ be a one-way function. Assuming the existence of a $m$-round witness-indistinguishable statistically-sound precise argument of knowledge with precision $p(t, n) = cg(n)t + cn$ for Hamiltonian path, there is a $O(m)$-round statistically-sound precise zero-knowledge argument for Hamiltonian path with precision $p(t, n) = cg(n)t + cn$.

**Proof.** There are two phases to the protocol. In phase one, $V$ uniformly samples $r_1 \leftarrow \{0, 1\}^n$ and $r_2 \leftarrow \{0, 1\}^n$ and computes $c_1 = f(r_1)$ and $c_2 = f(r_2)$ and sends $(c_1, c_2)$ to $P$. Next $V$ performs a witness-indistinguishable statistically-sound precise argument of knowledge for the claim that at least one of $c_1$ and $c_2$ is in the image of $f$. After this has completed, phase two begins. In phase two, $P$ performs a witness-indistinguishable statistically-sound precise argument of knowledge for the claim that $x \in L$ or one of $(c_1, c_2)$ is in the image of $f$. This completes the protocol.

First we will show that it is a precise zero-knowledge argument. To simulate the behavior of the simulator on an interaction with a dishonest
verifier $V^*$, $S$ extracts the witness $V^*$ used in phase one to get a preimage of one of the $c$ values. Once $S$ knows this preimage, it can simulate phase two using it as the witness for the witness-indistinguishable argument of knowledge of $P$.

We also need to show that it is sound. Suppose there is a cheating prover $P^*$. We can extract from $P^*$ the witness it uses to complete its zero-knowledge argument. If this witness is likely to be a preimage of one of the $c$ values then we can use the cheating prover to produce an efficient machine that inverts $f$. All we have to do is replace $r_1$ or $r_2$ with the string $y$ and with high probability it will compute $P^*$. If the witness we extract from $P^*$ is likely to be a witness for Hamiltonian path, then $P^*$ is a probabilistic polynomial-time heuristic that correctly computes the Hamiltonian path on this input. Since we are assuming the existence of one-way functions, we are also assuming that $\text{NP} \not\subseteq \text{BPP}$, so $P^*$ cannot be right with high probability. When it is right, the protocol is still fine because the prover is just telling the verifier what it already can compute on its own using said heuristic. □
Chapter 4

UOWHFs from Regular One-Way Functions

In Section 4.1 we introduce the Generalized Randomized Iterate and its reusable variant. We also prove a main technical lemma that is at the heart of the efficiency claim for our UOWHF construction which appears in Section 4.2. We present our alternative constructions of a $O(n \log n)$-seed PRG, and the hardness-amplification result in Section 4.4.

4.1 The Generalized Randomized Iterate

A well-known fact about one-way functions is that if you iterate them, you may not end up with a function that is difficult to invert. Indeed, while a permutation $f$, when iterated $f^{(i)} = f \circ \ldots \circ f$ (i.e. $f$ composed with itself $i$ times), remains one-way, this is not true for general one-way functions as a single application could concentrate the outputs on a very small fraction of the inputs of $f$, where $f$ might even be easy to invert.
[GKL93] introduced the Randomized Iterate construction where a randomization step is added between two application of \(f\), in its iteration. As shown in [HHR06], when using pairwise-independent hashing to implement this randomization step, the randomized iterate is hard to invert.

We introduce the Generalized Randomized Iterate (GRI) and we show how it can be used to construct both pseudorandom generators and target collision-resistant hashing. We then show an efficient form of the (Generalized) Randomized Iterate, where some of the hash functions are “recycled” during the iteration. This Reusable Generalized Randomized Iterate is the core of our efficient construction of UOWHFs.

**Definition 4.1.** Let \(f : \{0, 1\}^n \to \{0, 1\}^n\) and let \(H\) be an efficient family of pairwise-independent hash functions from \(\{0, 1\}^{n+\ell}\) to \(\{0, 1\}^n\). For input \(x \in \{0, 1\}^n, z \in \{0, 1\}^{\ell k}, (h_1, \ldots, h_m) \in H^m\) and \(m \geq k\), define the \(k\)th generalized randomized iterate \(g^k : \{0, 1\}^n \times \{0, 1\}^{\ell k} \times H^m \to \{0, 1\}^n\) recursively as:

\[
g^k(x, z, (h_1, \ldots, h_m)) = h_k(f(g^{k-1}(x, z, (h_1, \ldots, h_m)))) || z_{[(k-1)\ell+1...k\ell]}
\]

where \(g^0(x, z, h_1, \ldots, h_m) = x\), "||" denotes concatenation and \(z_{[a...b]}\) is the substring of \(z\) from position \(a\) (inclusive) to position \(b\) (inclusive), where the first character is at index \(1\).

In other words at each iteration of the Generalized Randomized Iterate, first \(f\) is applied to the output of the previous iteration, then a
block of $\ell$ bits from $z$ are appended to the output, and then a pairwise-independent hash function is applied. Note that at each iteration a new hash function is used.

While we are defining GRI for any value of $\ell$, we are going to be interested in two cases:

- $\ell = 0$, in which case $z$ is the empty string, and the pairwise-independent hash functions map $n$ bits to $n$ bits. This case is equivalent to the Randomized Iterate from [GKL93, HHR06] and as shown there it can be used to build PRGs;

- $\ell = 1$, in which case $z$ is $k$ bits long, and the hash functions compress by one bit. We will show in Section 4.2 that this function is a second preimage resistant function (from which a UOWHF can be easily built).

### 4.1.1 The Reusable Generalized Randomized Iterate

We now introduce the Reusable Generalized Randomized Iterate (RGRI), which is a version of the Randomized Generalized Iterate that uses fewer hash functions. While the GRI described in the previous section use new distinct hash functions at each iteration, we “recycle” some of these hash functions during the process. More specifically we sample $m$ hash functions $(h_1, \ldots, h_m)$ from $\mathcal{H}$ and then in the $i^{th}$ iteration of the RGRI we use the function $h_{\phi(i)}$ where $\phi(i)$ is the function that on input $i$, outputs the
highest power of 2 that divides $i$. If we have $k$ iterations it is sufficient to set $m = \lceil \log k \rceil + 1$. This “scheduling” of the hash functions is identical to the way Shoup recycles random masks in his construction of a domain extender for TCR functions [Sho00].

**Definition 4.2.** Let $f : \{0,1\}^n \rightarrow \{0,1\}^n$ and let $\mathcal{H}$ be an efficient family of pairwise-independent hash functions from $\{0,1\}^{n+\ell}$ to $\{0,1\}^n$. For input $x \in \{0,1\}^n$, $z \in \{0,1\}^{\ell k}$, $(h_1, \ldots, h_m) \in \mathcal{H}^m$ and $m \geq \lceil \log k \rceil + 1$, define the $k^{th}$ Reusable Generalized Randomized Iterate $\tilde{g}^k : \{0,1\}^n \times \{0,1\}^{\ell k} \times \mathcal{H}^m \rightarrow \{0,1\}^n$ recursively as:

$$
\tilde{g}^k(x, z, (h_1, \ldots, h_m)) =
\begin{cases}
    h_{\phi(k)}(f(\tilde{g}^{k-1}(x, z, h_1, \ldots, h_m))) || z_{[(k-1)\ell+1...k\ell]} & k > 0 \\
    x & \text{otherwise}
\end{cases}
$$

where $\phi(n)$ is one greater than the highest power of 2 that divides $n$.

### 4.1.2 A Technical Lemma

We now prove a preliminary lemma which is crucial in allowing us to achieve logarithmic key size for our UOWHF construction. This lemma abstracts the property of the “Shoup domain extension” technique we use to construct the RGRI: intuitively the lemma proves a preliminary result that will allows us later to claim that the distribution induced by the RGRI is not that far from the distribution induced by the GRI with distinct (i.e. nonreused) hash functions.

The goal of the lemma is to count how many input pairs lead to two specific values $a_0, a_1$ as outputs of the RGRI.
Lemma 4.3. Fix two arbitrary values \((a_0, a_1) \in (\{0,1\}^n)^2\) and an integer \(i\). The number of pairs \([ (x_0, z_0, h_1, \ldots, h_m), (x_1, z_1, h_1, \ldots, h_m) ]\) such that

\[
\tilde{g}^i(x_0, z_0, h_1, \ldots, h_m) = a_0 \quad \text{and} \quad \tilde{g}^i(x_1, z_1, h_1, \ldots, h_m) = a_1
\]

is bounded by \(2^{2\ell k} \cdot |\mathcal{H}|^m\).

Note that in the lemma we are counting the pairs with possibly distinct inputs \(x, z\) but same hash functions \(h_i\).

Proof: To prove the lemma we use a “key reconstruction” strategy introduced by Shoup in [Shoo00]. The algorithm in Figure 4.1 on input \(i \in [0..k], z_0, z_1 \in \{0,1\}^{\ell k}\) and \(a_0, a_1 \in \{0,1\}^n\) generates a pair of inputs \((x_0, \overline{h})\) and \((x_1, \overline{h})\) such that the output of the \(i^{th}\) iterate is \(a_0\) and \(a_1\), i.e.

\[
\tilde{g}^i(x_0, z_0, h_1, \ldots, h_m) = a_0 \quad \text{and} \quad \tilde{g}^i(x_1, z_1, h_1, \ldots, h_m) = a_1.
\]

We prove that this algorithm outputs all possible input pairs \((x_0, \overline{h})\) and \((x_1, \overline{h})\) with some probability. To complete the proof of the claim we show that the total number of distinct outputs by the algorithm is \(|\mathcal{H}|^m\) (the lemma follows since there are \(2^{2\ell k}\) possible values of \(z_0, z_1\)).

The high-level idea of the Shoup reconstruction strategy described in Figure 4.1 follows. Consider the simple case of the randomized iterate function \(g^k\) (where a different hash function is used after each iterate). Since, we use different hash functions at every iterate, we choose
all the hash functions \( h_1, \ldots, h_{i-1}, h_{i+1}, \ldots, h_m \) arbitrarily, except the one in the \( i^{th} \) iterate (i.e. \( h_i \)). Using \( x_0, z_0, h_1, \ldots, h_{i-1} \) and \( x_1, z_1, h_1, \ldots, h_{i-1} \) we compute \( y_0, y_1 \) as the outputs of \( f \circ g^{i-1} \). We then choose \( h_i \) so that 
\[ h_i(y_0||z_0,[(i-1)\ell+1...i\ell]) = a_0 \quad \text{and} \quad h_i(y_1||z_1,[(i-1)\ell+1...i\ell]) = a_1 \]
simultaneously holds. This is possible since \( \mathcal{H} \) is a pairwise-independent family. Furthermore, the number of such functions \( h_i \) is equal to \( |\mathcal{H}|/2^n \). Observe that every input pair satisfying the conditions is output by the strategy for some random choices and every random choice yields different outputs satisfying the conditions. Therefore, the total number of pairs satisfying the conditions equals the total number of random choices made by the strategy and that is \( 2^{2n}2^{2\ell i}|\mathcal{H}|^{m-1} \times |\mathcal{H}|/2^{2n} = 2^{2\ell i}|\mathcal{H}|^m \).

However this procedure does not work for the reusable randomized iterate since the hash functions are recycled. Instead, we consider segments and perform a “right to left” sweep from the \( i^{th} \) iterate to the first iterate, ensuring that each segment is locally consistent. More precisely, in each segment, for a particular \( a \), the algorithm selects hash functions and string \( x \) such that if \( x \) is fed as input to the \( j^{th} \) iterate, then the output of the computation at the \( i^{th} \) iterate \((i > j)\) is \( a \). For the segments to compose, we need to ensure that the hash functions selected by different segments do no conflict with each other and that is the technical part of the proof. To extend the algorithm to achieve consistency for two inputs it suffices to observe that for all \( x_0 \neq x_1 \) and arbitrary values \( a_0, a_1 \), there exists an \( h \) such that \( h(x_0) = a_0 \) and \( h(x_1) = a_1 \). The formal description
of the algorithm is presented in Figure 4.1.

First, we prove correctness and then compute the number of colliding pairs.

**Sub-Claim 4.4.** If the algorithm in Figure 4.1 outputs \((x_0, \overline{h}), (x_1, \overline{h})\), then it holds that \(\tilde{g}^i(x_0, z_0, \overline{h}) = a_0\) and \(\tilde{g}^i(x_1, z_1, \overline{h}) = a_1\).

**Proof:** Every iteration of the algorithm considers the segment from the \(j^{th}\) iterate to the \(i^{th}\) iterate and achieves the following: if \(x^j_0\) (and \(x^j_1\)) is fixed as the partial input to the \(j^{th}\) iterate then \(a_0\) (and \(a_1\)) is the output of the \(i^{th}\) iterate. This follows from the fact that \(h_{\phi(i)}\) is assigned a value at step 2(d) after knowing what the output of the \(i - 1^{th}\) iterate is computed. It only remains to show that two iterations do not assign values to the same hash function. The algorithm assigns a value to a hash function in steps 2(b), 2(d) and 4. By construction, step 2(b) and 4 only assign values to hash functions that have not been defined yet (indicated by the flag being false). It suffices to ensure that there are no conflicts in the assignment made at step 2(d). This is ensured by maintaining the invariant that \(h_{\phi(i)}\) is undefined before executing 2(d) in any iteration.

Observe that, in every iteration, \(\phi(j) > \phi(i)\) and for all \(c\) such that \(j < c < i\), \(\phi(c) < \phi(i)\). Therefore, before step 2(d) is reached in any iteration, the only hash functions that are defined are those with indices \(c\) such that \(\phi(c) < \phi(j)\). □
Sub-Claim 4.5. The number of distinct pairs output by the Shoup Reconstruction algorithm is bounded by $|\mathcal{H}|^m$.

Proof: From Subclaim 4.4, we know that every pair output of the algorithm satisfies the condition that $a_0$ and $a_1$ are the output of the $i^{th}$ iterate. Furthermore, every pair (that satisfies the condition) occurs as an output for some choice made by the algorithm and each choice made by the algorithm yields distinct outputs. Therefore, it suffices to compute the total number of choices made by the algorithm. To compute the number of pairs, observe that, for every choice made for $x^i_0$ and $x^i_1$ (such that $x^i_0 \neq x^i_1$) in step (b), the number of hash functions $h$ such that $h(y_0||z_{0,(i-1)\ell+1...i\ell}) = a_0$ and $h(y_1||z_{1,(i-1)\ell+1...i\ell}) = a_1$ is $\frac{|\mathcal{H}|^2}{2^{2n}}$, by the pairwise-independence property. We treat the choices made for $x^i_0$, $x^i_1$ as a choice made for $h_{\phi(i)}$ set in step 2(d). Thus, the number of choices for the hash function in step 2(d) is at most $2^{2n} \times \frac{|\mathcal{H}|^2}{2^{2n}} = |\mathcal{H}|$. The only other choices are the hash functions picked in step 2(b) and 4. Since they can take any value, they have $|\mathcal{H}|$ many choices. Corresponding to every hash function the algorithm makes $|\mathcal{H}|$ many choices. Therefore the total number of pairs is bounded by $|\mathcal{H}|^m$. 

This concludes the proof of Lemma 4.3.

The following corollary is proven by using the same counting argument and the same “reconstruction strategy” of Lemma 4.3 (intuitively, the
bound results from the fact that you can choose $x$ in $2^n$ ways, $z$ in $2^{lk}$ ways, $m - 1$ hash functions uniformly at random in $\mathcal{H}$, and the hash function $h_i$ via pairwise-independence among $|\mathcal{H}|/2^{2n}$ possible candidates).

**Corollary 4.6.** Fix arbitrary values $a_0, a_1 \in \{0, 1\}^n$, $y \in \{0, 1\}^{n+\ell}$ and an integer $i$. The number of inputs $(x, z, h_1, \ldots, h_m)$ such that

$$\tilde{g}^i(x, z, h_1, \ldots, h_m) = a_0 \quad \text{and} \quad h_i(y) = a_1$$

is bounded by $2^{\ell k - n} \cdot |\mathcal{H}|^m$. Moreover there exists a polynomial-time algorithm that samples such an input uniformly at random.

**Remark:** We point out that the “reconstruction” property outlined in Lemma 4.3 is exactly what is needed in order to prove the security of our UOWHF with $O(n \log n)$ key based on the RGRI.

This is in contrast to the case of PRG [HHR06] where any PRG for space-bounded computation would work to “derandomize” the seed from $n^2$ to $n \log n$. We can show that the particular space-bounded PRG used in [HHR06] satisfies a lemma similar to Lemma 4.3, and therefore could be used to reduce the size of the key of our UOWHF. For simplicity we just show the construction based on Shoup’s technique.
4.2 Constructions of Universal One-Way Hash Functions

In this section, we show how to construct second preimage resistant functions from regular one-way functions. We start with a simple construction (that already improves the efficiency from previous work) of quadratic key size. We then provide a more efficient and essentially-optimal solution with $O(n \log n)$ key size. Note that our functions compress a single bit (higher compression can be achieved by standard modes of iteration). Note also that UOWHFs can be easily built from second preimage resistant families.

4.2.1 A Construction with Linear Key Size

Definition 4.7. Let $f : \{0,1\}^n \to \{0,1\}^n$ and let $\mathcal{K} = \{0,1\}^n \times \mathcal{H}^{n+1}$ where $\mathcal{H}$ is an efficient family of pairwise-independent hash functions from $\{0,1\}^{n+1}$ to $\{0,1\}^n$. Define the function $g(z,k)$ with input space $z \in \{0,1\}^{n+1}$ and keyspace $k = (x,h_1,\ldots,h_{n+1}) \in \mathcal{K}$ as follows:

$$g(z,(x,h_1,\ldots,h_{n+1})) = g^{n+1}(x,z,h_1,\ldots,h_{n+1})$$

where $g^i$ is the Generalized Randomized Iterate with $\ell = 1$.

Theorem 4.8. Suppose $f$ is a $2^r$-regular one-way function. Then $g$ defined according to Definition 4.7 is a second preimage resistant function family.
**Proof Overview:** To understand how our construction works, let us assume (as a simplifying assumption) that we can uniformly sample pairs \((a_1, a_2)\) such that \(f(a_1) = f(a_2)\). Let us refer to such pairs as siblings for \(f\).

Given such a pair it is possible to set up the hash functions in the above construction so that if the adversary finds a collision, then we invert the one-way function on a point \(y\). Intuitively this is done as follows: given a random input \(z\) for the UOWHF, we choose the hash functions (i.e. the key \(k\)) so that \(g^i(z, k) = a_1\) and \(h_i(y|b) = a_2\) for a random index \(i\) and a random bit \(b\). We then run the adversary on \(z, k\) and if the adversary finds a collision \(z'\), with nonnegligible probability the collision “goes through” \(a_2\) at index \(i\), i.e. \(g^i(z', k) = a_2\) allowing us to find a preimage of \(y\).

The intuition here is that given any input \(z\) and key \(k\), at each iterate the input going into the one-way function has most \(2^r\) collisions w.r.t \(f\). For a collision to occur at a particular iterate, it must be the case that some image element \(y\) of the one-way function \(f\) must occur at the previous iterate and the hash function takes \(y\) and an input bit into one of the \(2^r\) collisions in the next iterate. Since there are at most \(2^{n-r}\) range elements, in expectation over hash functions, the number of possible inputs at the previous iterate that are mapped into the \(2^r\) collisions are small, in fact \(O(1)\). Thus the hash functions selected above will succeed with high probability.

But how do we get to sample \(a_1, a_2\), i.e. siblings for \(f\) in the first place? For this we use the adversary again. Indeed, when an adversary finds a
collision to input $z$ (say $z'$), it must be that at some iterate, the inputs into the intermediate hash functions are different and the outputs to the next iterate are strings $a_1$ and $a_2$ such that $f(a_1) = f(a_2)$, i.e. siblings for $f$.

It remains to argue that sampling $a_1$ and $a_2$ by first querying the adversary is good enough, and this is established using a collision probability analysis. We now proceed to a formal proof.

**Proof:** Assume for contradiction that there exists an adversary $A$ and polynomial $p(\cdot)$ such that for infinitely-many lengths $n$, the probability with which $A$ finds a collision on a random input $z \in \{0, 1\}^n$ and key $k = (x, h) \in \mathcal{K}$ is at least $\epsilon \geq \frac{1}{p(n)}$. We assume for simplicity that $A$ is deterministic. Fix a particular $n$ for which this happens. Using $A$, we construct a machine $M$ that inverts $f$ with probability that is polynomially related to $\epsilon$ and thus arrive at a contradiction.

The machine $M$ on arbitrary input $y \in \{0, 1\}^n$ internally incorporates the code of $A$ and proceeds as follows:

1. Sample a random input $z$ and key $k = (x, h)$. Internally run $A$ on input $(z, k)$. If $A$ fails to return a collision, halt outputting $\bot$. Otherwise, let $z'$ be the output of $A$.

2. Let $i$ be the smallest index such that $f(g^{i-1}(z, k))|z_i \neq f(g^{i-1}(z', k))|z'_i$ and $f(g^i(z, k)) = f(g^i(z', k))$ (since $g(z, k) = g(z', k)$ such an $i$ must exists). Let $a_1 = g^i(z, k)$ and $a_2 = g^i(z', k)$. It follows now that $f(a_1) = f(a_2)$. For any two colliding inputs such as $z$ and
z' with key k, we call this i the colliding-index.

3. Choose \( z^*, k^* = (x^*, h^-_1, \ldots, h^-_{n+1}) \) and a random bit \( b \) such that \( g^i(z^*, k^*) = a_1 \) and \( h^-_i(y||b) = a_2 \). This can be done using the pairwise-independence property of \( \mathcal{H} \). More precisely, choose \( z^*, x^* \) and all the hash functions except \( h^-_i \) at random and set \( h^-_i \) so that both the conditions hold. Run A on input \( (z^*, k^*) \). If A fails to return a collision or such a hash function \( h^-_i \) cannot be sampled,\(^1\) halt outputting \( \bot \). Otherwise, let \( z'' \) be the output of A.

4. If \( f(g^{i-1}(z'', k^*)) \neq y \), halt outputting \( \bot \). Otherwise, output \( g^{i-1}(z'', k^*) \).

It follows from the construction that if \( M \) outputs \( w \), then \( f(w) = y \). We now proceed to compute the success probability of \( M \). But first, we require the another definition. Define sets \( N(i, a_1, a_2) \) to contain all input-key pairs \((z, k)\) such that the following hold true: \( f(a_1) = f(a_2) \) and \( g^i(z, k) = a_1 \), and A on input \((z, k)\) returns \( z' \) such that \( g^i(z', k) = a_2 \) and \( i \) is the colliding-index. We first express the success probability of \( M \) using these sets.

**Claim 4.9.** The probability with which \( M \) succeeds in inverting \( f \) is

\[
2^{n+r-1} \sum_{i,a_1,a_2} \frac{|N(i, a_1, a_2)|^2}{(2^{2n+1} |\mathcal{H}|^{n+1})^2}.
\]

\(^1\)This occurs when \( a_1 \neq a_2 \) and \( f(g^{i-1}(z^*, k^*)) = y \) and \( z_i = b \)
**Proof:** Given a tuple \((z^*, k^*, i, a_1, a_2)\) such that \((z^*, k^*) \in N(i, a_1, a_2)\), define the following events:

**Event E1:** The randomly-chosen input-key pair \((z, k)\) by \(M\) in step 1 is in \(N(i, a_1, a_2)\). Since the input and key are chosen uniformly at random, it holds that
\[
\Pr[E_1] = 1/2^{n+1} \times 1/2^n \times 1/|\mathcal{H}|^{n+1} \times |N(i, a_1, a_2)| = |N(i, a_1, a_2)|/2^{2n+1}|\mathcal{H}|^{n+1}.
\]

**Event E2:** If \(A\) on input \((z^*, k^*)\) returns \(z'\)—where \(k^* = (x, h_1, \ldots, h_n)\)—this event denotes that \(M\)'s random choice \(b = z'_i\) and \(M\)'s input is \(y\) such that \(g^{i-1}(z'_i, k^*) = y\). Therefore, \(h_i(y||b) = h_i(y||z'_i) = a_2\). The probability that \(b = z'_i\) is 1/2. Therefore, since \(f\) is a \(2^r\)-regular OWF, \(\Pr[E_2] = 1/2 \cdot 2^r/2^n = 2^{r-1}/2^n\).

**Event E3:** \(M\) chooses \(z^*, k^*\) in Step 3. From the pairwise-independence property of \(\mathcal{H}\), it follows that\(^2\) \(\Pr[E_3] = 1/(22^{n+1}|\mathcal{H}|^{n+1}) = 1/2|\mathcal{H}|^{n+1}\).

It follows from the description that for any tuple \((z^*, k^*, i, a_1, a_2)\) such that \((z^*, k^*) \in N(i, a_1, a_2)\), if \(E_1, E_2\) and \(E_3\) occurs, \(M\) inverts \(y\). Note that \(E_1, E_2\) and \(E_3\) are independent. Therefore, for a fixed tuple \((z^*, k^*, i, a_1, a_2)\) such
\(^2\)\(z, x\) and all the hash functions except \(h_i\) are randomly-chosen. There are \(2^{2n+1}|\mathcal{H}|^{m-1}\) such tuples. \(h_i\) is chosen so that two of its values are fixed. Since \(\mathcal{H}\) is a pairwise-independent family of hash functions, there are exactly \(|\mathcal{H}|^{2n}\) such functions. Finally, one of these tuples is chosen uniformly at random.
that \((z^*, k^*) \in N(i, a_1, a_2)\) the probability that \(E_1, E_2\) and \(E_3\) occurs is
\[
\frac{|N(i, a_1, a_2)|}{2^{2n+1}|\mathcal{H}|^{n+1}} \times \frac{2^{r-1}}{2^n} \times \frac{1}{2|\mathcal{H}|^{n+1}}.
\]

It follows from the definition of the sets \(N(\cdot, \cdot, \cdot)\), that for every \((z, k)\) there exists at most one tuple \((i, a_1, a_2)\) such that \((z, k) \in N(i, a_1, a_2)\).

Therefore, the success probability of \(M\) can be expressed as the sum of the success probability of \(M\) on each tuple \((z^*, k^*, i, a_1, a_2)\) such that \((z^*, k^*) \in N(i, a_1, a_2)\). More precisely, the success probability of \(M\) is
\[
\sum_{i, a_1, a_2} \sum_{(z^*, k^*) \in N(i, a_1, a_2)} \frac{|N(i, a_1, a_2)|}{2^{2n+1}|\mathcal{H}|^{n+1}} \times \frac{2^{r-1}}{2^n} \times \frac{1}{2|\mathcal{H}|^{n+1}}
\]
\[
= \sum_{i, a_1, a_2} \frac{|N(i, a_1, a_2)|^2}{2^{2n+1}|\mathcal{H}|^{n+1}} \times \frac{2^{r-1}}{2^n} \times \frac{1}{2|\mathcal{H}|^{n+1}}
\]
\[
= 2^{n+r-1} \sum_{i, a_1, a_2} \frac{|N(i, a_1, a_2)|^2}{(2^{2n+1}|\mathcal{H}|^{n+1})^2}.
\]

We now relate this expression to the success probability of \(A\).

**Claim 4.10.** If \(A\) succeeds with probability \(\epsilon\) then
\[
\sum_{(i, a_1, a_2)} \frac{|N(i, a_1, a_2)|^2}{(2^{2n+1}|\mathcal{H}|^{n+1})^2} \geq \frac{e^2}{n2^{n+r}}.
\]

**Proof:** Since for every pair \((z, k)\), there exists at most one tuple \((i, a_1, a_2)\) such that \((z, k) \in N(i, a_1, a_2)\) and by definition if \((z, k) \in N(i, a_1, a_2)\) then
A succeeds on input \((z, k)\), we have that the success probability of \(A\) is
\[
\frac{1}{2^{2n+1} |\mathcal{H}|^{n+1}} \times \sum_{(i, a_1, a_2)} |N(i, a_1, a_2)| = \epsilon.
\]

Let us consider the sum in the left-hand side and use the Cauchy-Schwartz inequality to obtain a bound on the sum of the squares of each term. It suffices to consider the sum over all tuples \((i, a_1, a_2)\) such that \(N(i, a_1, a_2)\) is not empty. In particular, they are not empty only if \(f(a_1) = f(a_2)\). Therefore, the total number of such tuples is at most \(n2^{n+r}\). Using the Cauchy-Schwartz inequality, we have that
\[
\sum_{(i, a_1, a_2)} \frac{|N(i, a_1, a_2)|^2}{(2^{2n+1} |\mathcal{H}|^{n+1})^2} \geq \frac{\epsilon^2}{n2^{n+r}}.
\]

Now, we conclude the proof of the theorem. Applying Claim 4.10 to Claim 4.9, we obtain that the success probability of \(M\) is at least
\[
2^{n+r-1} \times \frac{\epsilon^2}{n2^{n+r}} = \frac{\epsilon^2}{2n},
\]
which is nonnegligible. Therefore, \(M\) inverts \(f\) with nonnegligible probability and we arrive at a contradiction.

### 4.2.2 A Construction with Logarithmic Key Size

We now show how to construct a more efficient second preimage resistant family from regular one-way functions, by showing that if \(f\) is a regular one-way function then the Reusable Generalized Randomized Iterate is
second preimage resistant.

**Definition 4.11.** Let $f : \{0,1\}^n \rightarrow \{0,1\}^n$ and let $\mathcal{K} = \{0,1\}^n \times \mathcal{H}^m$ where $\mathcal{H}$ is an efficient family of pairwise-independent hash functions from $\{0,1\}^{n+1}$ to $\{0,1\}^n$ and $m = O(\log n)$. Define the function $g(z,k)$ with input space $z \in \{0,1\}^{n+1}$ and key space $k = (x,h_1,\ldots,h_m) \in \mathcal{K}$ as follows:

$$g(z,(x,h_1,\ldots,h_m)) = \tilde{g}^{n+1}(x,z,h_1,\ldots,h_m)$$

where $\tilde{g}^i$ is the Reusable Generalized Randomized Iterate with $\ell = 1$.

**Theorem 4.12.** Suppose $f$ is a $2^r$-regular one-way function. Then $g$ defined according to Definition 4.11 is a second preimage resistant function family.

**Proof:** Assume for contradiction that there exists an adversary $A$ and polynomial $p$ such that for infinitely many lengths $n$, the probability with which $A$ finds a collision on a random input $z \in \{0,1\}^n$ and key $k = (x,h) \in \mathcal{K}$ is $\epsilon \geq \frac{1}{p(n)}$. As before, we assume for simplicity that $A$ is deterministic.

Fix a particular $n$ for which this happens. Using $A$, we construct a machine $M$ that inverts $f$ with probability that is polynomially-related to $\epsilon$ and thus arrive at a contradiction. The machine $M$ on arbitrary input $y \in \{0,1\}^n$ internally incorporates the code of $A$ and proceeds as follows:

1. Sample a random input $z$ and key $k = (x,h)$. Internally simulate $A$
on input \((z,k)\). If \(A\) fails to return a collision, halt and output \(\bot\).

Otherwise, let \(z'\) be the output of \(A\).

2. Let \(i\) be the colliding-index. Let \(a_1 = g^i(z,k)\) and \(a_2 = g^i(z',k)\).

3. Choose \(z^*, k^* = (z^*, h^*_1, \ldots, h^*_m)\) and a random bit \(b\) such that
\(g^i(z^*, k^*) = a_1\) and \(h^*_\phi(i)(y||b) = a_2\). This can be done in polynomial time following Corollary 4.6. Internally run \(A\) on input \((z^*, k^*)\). If \(A\) fails to return a collision, halt outputting \(\bot\). Otherwise, let \(z''\) be \(A\)'s output.

4. If \(f(g^{i-1}(z'', k^*)) \neq y\), halt outputting \(\bot\). Otherwise, output \(g^{i-1}(z'', k^*)\).

As before, we define sets \(N(i, a_1, a_2)\) that satisfy the same condition with the exception that we rely on \(\tilde{g}^i\) instead of \(g^i\). The next claim relates these sets to the success probability of \(M\).

**Claim 4.13.** The probability with which \(M\) succeeds in inverting the one-way function \(f\) is
\[
2^{n+r-1} \sum_{i,a_1,a_2} |N(i, a_1, a_2)|^2 / \left(2^{2n+1} |\mathcal{H}|^m \right)^2.
\]

**Proof:** Consider the events \(E_1, E_2,\) and \(E_3\) exactly as before. We now have that given a tuple \((z,k,i,a_1,a_2)\):

- the probability that \(E_1\) occurs is
\[
\frac{1}{2^{n+1}} \times \frac{1}{2^n} \times \frac{1}{|\mathcal{H}|^m} \times |N(i,a_1,a_2)| = \frac{|N(i,a_1,a_2)|}{2^{2n+1} |\mathcal{H}|^m}
\]
the probability that $E_2$ occurs is $\frac{2^{r-1}}{2^n}$ as before;

- the probability that $E_3$ occurs given $E_1$ and $E_2$ occurs is $\frac{1}{(2^{n+1}\frac{|\mathcal{H}|^m}{2^n})} = \frac{1}{2|\mathcal{H}|^m}$. This follows from Corollary 4.6 for $\ell = 1$ and $k = n+1$ (which are the parameters used in this construction).

Again, we have that every $(z,k)$ belongs to at most one set $N(i,a_1,a_2)$. Therefore, the success probability of $M$ is

$$\sum_{i,a_1,a_2} \sum_{(z,k) \in N(i,a_1,a_2)} \frac{|N(i,a_1,a_2)|}{2^{2n+1}\mathcal{H}|^m} \times \frac{2^{r-1}}{2^n} \times \frac{1}{2|\mathcal{H}|^m}$$

$$= 2^{n+r-1} \sum_{i,a_1,a_2} \frac{|N(i,a_1,a_2)|^2}{(2^{2n+1}|\mathcal{H}|^m)^2}.$$

This proves the claim. □

The next claim follows identically to Claim 4.10.

**Claim 4.14.** If $A$ succeeds with probability $\epsilon$ then $\sum_{(i,a_1,a_2)} \frac{|N(i,a_1,a_2)|^2}{(2^{2n+1}|\mathcal{H}|^m)^2} \geq \frac{\epsilon^2}{n^{2n+r}}$.

As before, applying Claim 4.13 to Claim 4.14, we obtain that the success probability of $M$ is at least $\frac{\epsilon^2}{2n}$ and thus we arrive at a contradiction. □
4.3 Pseudorandom Generator Construction

The idea of iterating a one-way permutation \( f \) on itself to obtain a PRG originates from the work of Blum, Micali, and Yao [BM82, Yao82]. Since \( f \) is a permutation, the function \( f^{(i)} = f \circ \ldots \circ f \) (\( f \) iterated on itself \( i \) times) is also one-way. This means that the hardcore bit of every intermediate step is unpredictable. Iterating \( n + 1 \) times on a random input of length \( n \) and outputting all the hardcore bits would then yield a PRG that stretches by 1 bit. We refer to this as the BMY construction.\(^3\)

This approach, unfortunately, does not work for general one-way functions as a single application could concentrate the outputs on a very small fraction of the inputs of \( f \), where \( f \) might even be easy to invert. The Randomized Iterate construction by Goldreich, Krawczyk, and Luby [GKL93], extended the BMY construction by adding a randomization step between two application of \( f \), and showed a construction of pseudorandom generators based on any regular one-way function. The randomization step considered in [GKL93] was applying a uniformly chosen \( n \)-wise independent hash function. Haitner, et. al [HHR06] simplified the construction to use just pairwise hashing. Below we present their definition of the Randomized Iterate, rephrased using our notion of GRI.

**Definition 4.15** (\( k^{th} \) Randomized Iterate of \( f \)). Let \( f : \{0,1\}^n \rightarrow \{0,1\}^n \)

\(^3\)If \( f \) is a permutation over \( n \)-bit strings a more efficient construction is to set the generator \( G \) as \( G(x) = f(x).b(x) \) However, this uses the property that \( f \) is a permutation in a crucial way (since if \( x \) is uniform then \( f(x) \) is also uniform).
CHAPTER 4. UOWHFS FROM REGULAR ONE-WAY FUNCTIONS

and let $\mathcal{H}$ be an efficient family of pairwise-independent hash functions from $\{0,1\}^n$ to $\{0,1\}^n$. For input $x \in \{0,1\}^n$, $h_1,\ldots,h_m \in \mathcal{H}$ and $m \geq k$, define the $k^{th}$ Randomized Iterate $f^k : \{0,1\}^n \times \mathcal{H}^m \rightarrow \{0,1\}^n$ as

$$f^k(x, h_1, \ldots, h_m) = f(g^k(x, h_1, \ldots, h_m))$$

where $g^k(x, h_1, \ldots, h_m)$ is the GRI defined by $x, h_1, \ldots, h_m$ for $\ell = 0$.

Notice that we redefine the Randomized Iterate in terms of the GRI with $\ell = 0$. This means that the input $z$ is empty.

Following [GKL93], Haitner et al. in [HHR06] build a PRG as follows. The seed is composed of $x$ (the input to the function $f$), the random string $r$ needed for the Goldreich-Levin hardcore predicate, and the description of the hash functions. The string $r$ and the hash functions can also be output, therefore to stretch the $n$-bit input by one bit we need $n + 1$ iterations. The generator is described below, where with $\overline{h}$ we denote a vector of $n + 1$ randomly-chosen hash functions from $\mathcal{H}$, and with $b$ we denote the GL hardcore predicate. The generator $G$ therefore goes from $\{0,1\}^{2n} \times \mathcal{H}^{n+1}$ to $\{0,1\}^{2n+1} \times \mathcal{H}^{n+1}$ and is defined as

$$G(x, r, \overline{h}) = (b(f^0(x, \overline{h}), r), \ldots, b(f^n(x, \overline{h}), r, r, \overline{h})).$$

The proof of security of this generator from [HHR06], assumes that $f$ is a regular one-way function and $\mathcal{H}$ is an efficient family of pairwise-
independent hash functions. The crux of the security argument is to prove that, for every $k$, the last iterate is hard to invert, since this justifies outputting the hardcore bit. In other words, given $y = f^k(x, h_1, \ldots, h_k)$ for a uniformly-chosen $x \in \{0, 1\}^n$, and given the hash functions $h_1, \ldots, h_k$, it is hard to compute the input to the last iterate in the computation of $y$, i.e. the value $x'$ such that $f(h_k(x')) = y$. The proof is done by reduction to the hardness of inverting $f$, and it relies on estimating the collision probability of the randomized iterate.

As we discussed in the introduction this generator has $O(n^2)$ seed size (since pairwise-independent hash functions can be described with $O(n)$ bits and we use $n + 1$ distinct hash functions). In [HHR06] it is shown how to reduce the seed to $O(n \log n)$ using derandomization techniques based on PRGs that fool bounded-space computations. In the next section we show how to obtain a PRG with $O(n \log n)$ seed length using a different form of derandomization. e.g. the Shoup “recycling” technique from [Shoo00] which is at the basis of our reusable randomized iterate.

### 4.3.1 A New PRG from Regular OWFs with Logarithmic Seed Size

We first define a reusable version of the randomized iterate in terms of the reusable generalized randomized iterate (for the case $\ell = 0$).

**Definition 4.16** ($k^{th}$ Reusable Randomized Iterate of $f$). Let $f : \{0, 1\}^n \rightarrow \{0, 1\}^n$ and let $\mathcal{H}$ be an efficient family of pairwise-independent hash
functions from \(\{0,1\}^n\) to \(\{0,1\}^n\). For input \(x \in \{0,1\}^n\), \(h_1, \ldots, h_m \in \mathcal{H}\) and \(m \geq \lceil \log k \rceil + 1\), define the \(k^{th}\) reusable randomized iterate \(\tilde{f}^k: \{0,1\}^n \times \mathcal{H}^m \rightarrow \{0,1\}^n\) as:

\[
\tilde{f}^k(x, h_1, \ldots, h_m) = f(g^k(x, h_1, \ldots, h_m))
\]

where \(g^k\) is the RGRI defined by \(x, h_1, \ldots, h_m\) with \(\ell = 0\).

As in the previous case, we construct a pseudorandom generator from the reusable randomized iterate by extracting the Goldreich-Levin predicate from each iteration. The seed consists of the input to the function \(f\), the random string \(r\) needed for the GL bit, and the description of the hash functions (\(r\) and the hash functions will also be part of the output). Therefore, the generator goes from \(\{0,1\}^{2n} \times \mathcal{H}^m\) to \(\{0,1\}^{2n+1} \times \mathcal{H}^m\) and is defined as

\[
G(x, r, \overline{h}) = (b(\tilde{f}^0(x, \overline{h}), r), \ldots, b(\tilde{f}^n(x, \overline{h}), r, r, \overline{h})).
\]

As we pointed out about it is sufficient to have \(m = \lceil \log n \rceil + 1\).

**Theorem 4.17.** Let \(f: \{0,1\}^n \rightarrow \{0,1\}^n\) be a regular one-way function and \(\mathcal{H}\) be an efficient family of pairwise-independent length-preserving hash functions. Define \(G: \{0,1\}^{2n} \times \mathcal{H}^m \rightarrow \{0,1\}^{2n+1} \times \mathcal{H}^m\) as

\[
G(x, r, \overline{h}) = (b(\tilde{f}^0(x, \overline{h}), r), \ldots, b(\tilde{f}^n(x, \overline{h}), r, r, \overline{h})
\]
where $b$ is the Goldreich-Levin hardcore predicate. $G$ is a pseudorandom generator.

**Proof:** As in [HHR06] the security of the PRG follows from the fact that the last iterate of the reusable randomized iterate is hard to invert. Once that statement is proven, it implies that all the Goldreich-Levin bits are actually pseudorandom, and the security of the PRG follows by standard hybrid arguments (we are going to omit these steps, since they are standard by now, but the interested reader can find them in the [HHR06] proof).

For ease of notation we define $\hat{f}^k(x, h_1, \ldots, h_m) = (\tilde{f}^k(x, h_1, \ldots, h_m), h_1, \ldots, h_m)$, i.e. the output of the reusable generalized randomized iterate together with the hash functions used in the computation.

The proof of the theorem follows from Lemma 4.18 below which states that the last iterate of the reusable randomized iterate is hard to invert, i.e. it is hard to compute $\tilde{f}^{k-1}(x)$ given $\hat{f}^k(x, h_1, \ldots, h_m)$ for any $k$. Although the approach to the proof of this lemma is similar “in spirit” to the proof of an analogous lemma in [HHR06], it requires a totally different set of techniques since we recycle the hash functions in our construction.

**Lemma 4.18.** Let $f$ be a length-preserving regular one-way function, $\mathcal{H}$ be an efficient family of length-preserving pairwise-independent hash functions. Let $m = \lceil \log k \rceil + 1$. Then, for every probabilistic polynomial-time
CHAPTER 4. UOWHFS FROM REGULAR ONE-WAY FUNCTIONS

machine $A$, there exists a negligible function $\nu$ such that for any $n \in \mathbb{N}$ it holds that if $x \leftarrow \{0,1\}^n$, $h_1, \ldots, h_m \leftarrow \mathcal{H}$ and $y = \tilde{f}^k(x, h_1, \ldots, h_m)$, then

$$\Pr[A(1^n, y, h_1, \ldots, h_m) = (\tilde{f}^{k-1}(x, h_1, \ldots, h_m))] \leq \nu(n).$$

**Proof:** We prove this result using the approach taken by [HHR06]. Suppose there exists an adversary that can find $(\tilde{f}^{k-1}(x, h_1, \ldots, h_k))$. Then we construct an adversary $B$ that inverts $f$. On a high level, the adversary $B$ on arbitrary input $y = f(x)$ for a random $x$, samples $m$ pairwise-independent hash functions $(h_1, \ldots, h_m)$ uniformly at random, runs $A$ on input $(1^n, y, h_1, \ldots, h_m)$ and outputs what $A$ outputs. For this to work, we need it to be true that the chosen hash functions $(h_1, \ldots, h_m)$ are “consistent” with $y$, i.e. there is an input $x$ for which $y = \tilde{f}^k(x, h_1, \ldots, h_m)$ and has the “right” distribution, i.e. probability of choosing $(h_1, \ldots, h_m)$ conditioned on $y$ being the output, is close to the probability of choosing it uniformly at random.

Let $A$ invert with probability $\epsilon$. Following [HHR06], we proceed in two steps. First we show that there exists a subset $T$ of the image of $\tilde{f}^k$ such that

- $T$ is “heavy”: i.e. the probability that by choosing a random $x \in \{0,1\}^n$ and $h_1, \ldots, h_m \in \mathcal{H}$ the value $\tilde{f}^k(x, h_1, \ldots, h_m)$ falls in $T$ with probability at least $\frac{\epsilon}{2}$;

- $A$ inverts every element in $T$ with probability at least $\frac{\epsilon}{2}$. 

We omit the proof of this step, as it is a standard Markov argument and can be found in [HHR06]. Then we show that for every “heavy” subset $T$ of the image of $\widehat{f}^k$, $B$ has a “good” probability of hitting an element of this subset, which will complete the proof. We show this in the following lemma, using a collision probability argument.

**Lemma 4.19.** For every set $T \subseteq \text{Im}(\widehat{f}^k)$, if

$$\Pr[x \leftarrow \{0, 1\}^n; h_1, \ldots, h_m \in \mathcal{H} : \widehat{f}^k(x, h_1, \ldots, h_m) \in T] \geq \delta$$

then

$$\Pr[x \leftarrow \{0, 1\}^n; h_1, \ldots, h_m \in \mathcal{H} : (f(x), h_1, \ldots, h_m) \in T] \geq \frac{\delta^2}{k+1}.$$ 

Using Lemma 4.19, and setting $\delta = \frac{\epsilon}{2}$ we have that $B$ hits the set $T$ with probability $\frac{\epsilon^2}{4(k+1)}$. Since $A$ inverts successfully in $T$ with probability $\frac{\epsilon}{2}$ we have that $B$ succeeds in inverting the function $f$ with probability $\frac{\epsilon^3}{8(k+1)}$. Therefore if $f$ is one-way, $\epsilon$ must be negligible. This concludes the proof of Lemma 4.18.

We now move to the proof of Lemma 4.19.

**Proof:** Informally, the lemma states that a large fraction of inputs induces a large fraction of the outputs on the $k^{th}$ iterate. This is what makes it possible for $B$ to succeed by sampling $y$ and the hash functions $h_1, \ldots, h_m$.
independently. To prove this lemma, we follow [HHR06] and use collision probability: if a large fraction of inputs to $\hat{f}^k$ would induce only a small set of outputs then necessarily $\hat{f}^k$ must have many collisions (i.e. two inputs that produce the same output). We prove that this is not the case by proving that the collision probability of $\hat{f}^k$ is low. We define the collision probability of $\hat{f}^k$ as

$$\Pr[(x_0, h_0), (x_1, h_1) \leftarrow \{0, 1\}^n \times \mathcal{H}^m : \hat{f}^k(x_0, h_0) = \hat{f}^k(x_1, h_1)].$$

Claim 4.20 below proves an upper bound on the collision probability with the value $\frac{k+1}{|\mathcal{H}|^m |\text{Im}(f)|}$.

On the other hand a lower bound on the collision probability for inputs in $T$ is obviously $\frac{\delta^2}{|T|}$, since two inputs fall both in $T$ with probability $\geq \delta^2$ and the probability that they collide is at least $\frac{1}{|T|}$. Therefore we get that

$$\frac{\delta^2}{|T|} \leq \frac{k+1}{|\mathcal{H}|^m |\text{Im}(f)|}$$

which implies

$$\frac{\delta^2}{k+1} \leq \frac{|T|}{|\mathcal{H}|^m |\text{Im}(f)|} = \Pr[x \leftarrow \{0, 1\}^n; h_1, \ldots, h_m \in \mathcal{H} : (f(x), h_1, \ldots, h_m) \in T]$$

as desired. Below we state and prove Claim 4.20 which concludes the proof of Lemma 4.19. \[\blacksquare\]
Claim 4.20.

$$\Pr[(x_0, h_0), (x_1, h_1) \leftarrow \{0,1\}^n \times \mathcal{H}^m : \hat{f}^k((x_0, h_0)) = \hat{f}^k(x_1, h_1)] \leq \frac{k + 1}{|\mathcal{H}|^m |\text{Im}(f)|}.$$  

Proof: We find the number of pairs $(x_0, h_0), (x_1, h_1)$ that collide at the $i^{th}$ iterate for the first time and then sum over all $i$ to compute the total number of colliding pairs. Observe that, for any collision, the hash functions $h_0$ and $h_1$ have to be the same. Consider $a_0, a_1 \in \{0,1\}^n$ such that $f(a_0) = f(a_1)$. Using Lemma 4.3 (for $\ell = 0$) we know that the number of pairs $(x_0, h)$ and $(x_1, h)$ such that the output of the $i^{th}$ iterate for $h$ are $a_0$ and $a_1$ respectively is at most $|\mathcal{H}|^m$. For a $2^r$ regular function $f$, the number of pairs for which $f(a_0) = f(a_1)$ is exactly $2^{n+r}$. Therefore, the number of pairs that collide at the $i^{th}$ iterate for the first time is bounded by $2^{n+r} |\mathcal{H}|^m$. Thus, the collision probability is at most

$$\sum_i \frac{2^{n+r} |\mathcal{H}|^m}{2^{2n} |\mathcal{H}|^m} = \sum_i \frac{2^{n+r}}{2^{2n} |\mathcal{H}|^m} = \sum_i \frac{1}{|\mathcal{H}|^m 2^n} = \frac{k + 1}{|\mathcal{H}|^m 2^n} = \frac{k + 1}{|\mathcal{H}|^m |\text{Im}(f)|},$$

completing the proof of Claim 4.20. ☐
CHAPTER 4. UOWHFS FROM REGULAR ONE-WAY FUNCTIONS

4.4 Hardness Amplification of Regular One-way Functions

The idea of iterating a one-way permutation \( f \) on itself to obtain a PRG originates from the work of Blum, Micali and Yao in [BM82] and [Yao82]. Since \( f \) is a permutation, the function \( f^{(i)} = f \circ \ldots \circ f \) (\( f \) iterated on itself \( i \) times) is also one-way. This means that the hardcore predicate of every intermediate step is unpredictable. Iterating \( n + 1 \) times on a random input of length \( n \) and outputting all the hardcore bits would then yield a PRG that stretches by 1 bit. We refer to this as the BMY construction.\(^4\)

This approach unfortunately does not work for general one-way functions. For the special case of regular one-way functions, [GKL93] showed how to extend the BMY construction by adding a randomization step using an \( n \)-wise independent hash function between every two applications of \( f \). [HHR06] simplified the construction to use just pairwise-hashing and further derandomized the construction by showing how to generate the \( n \) hash functions required at the randomization steps using just \( n \log n \) bits thus obtaining a PRG of seed length \( O(n \log n) \).

In [HHR06], Haitner et. al, showed that the same randomized iterate can also be used for hardness amplification to obtain strong one-way function from any regular weakly one-way function even with unknown

\(^4\)If \( f \) is a permutation over \( n \)-bit strings a more efficient construction is to set the generator \( G \) as \( G(x) = f(x)b(x) \). However, this uses in a crucial way the property that \( f \) is a permutation (since if \( x \) is uniform then \( f(x) \) is also uniform).
regularity. They also showed that similar derandomization yielded corresponding efficiency gains.

Using the Reusable Generalized Randomized Iterate, we obtain analogous PRG constructions and hardness amplification with same efficiency. We now describe our results.

**Theorem 4.21.** Let \( f : \{0,1\}^n \to \{0,1\}^n \) be a regular one-way function and \( \mathcal{H} \) be an efficient family of pairwise-independent length-preserving hash functions. Define \( G : \{0,1\}^{2n} \times \mathcal{H}^m \to \{0,1\}^{2n+1} \times \mathcal{H}^m \) as

\[
G(x,r,h) = (b(\tilde{f}^0(x,h),r),\ldots,b(\tilde{f}^n(x,h),r),r,h)
\]

where \( \tilde{f}^k(x,h_1,\ldots,h_m) = f(g^k(x,h_1,\ldots,h_m)) \) and \( g^k \) is the RGRI defined by \( x,h_1,\ldots,h_m \) with \( \ell = 0 \) and \( b \) is the Goldreich-Levin hardcore predicate. \( G \) is a pseudorandom generator.

**Theorem 4.22.** Let \( f \) be a \( \frac{1}{p(n)} \)-weak one-way function for some polynomial \( p(\cdot) \). \(^5\) Let \( k = 4np(n) \) and \( m = \lceil \log k \rceil \). For input \( x \in \{0,1\}^n \), \( \bar{h} = [h_1,\ldots,h_m] \in \mathcal{H}^m \), define \( g(x,\bar{h}) = (\tilde{f}^k(x,\bar{h}),\bar{h}) \) where \( \tilde{f}^k \) is the reusable randomized iterate of \( f \). \( g \) is a (strong) one-way function.

In [HHR06], Haitner et. al (HHR), show that the randomized iterate can be used for hardness amplification of regular weakly one-way functions of unknown regularity. They also show that the derandomized ver-

\(^5\)A function \( f \) is an \( \epsilon \)-weak one-way function, if no adversary can succeed in inverting \( f \) with probability better than \( 1 - \epsilon \).
sion of the randomized iterate also yields similar results. In this section, we show that the RGRI (as used in the PRG construction from Section 4.3) also gives an analogous hardness amplification result (matching the parameters of the derandomized construction from HHR). We follow their proof closely to obtain our result. First, we give a brief overview of their proof here.

Let \( f \) be an \( \alpha \)-weak regular one-way function. Define \( g \) as follows:

\[
g(x, h_1, \ldots, h_m) = (g^k(x, h_1, \ldots, h_{m-1}), h_1, \ldots, h_{m-1})
\]

where \( g^k \) is the GRI of \( f \) (from Definition 4.1). HHR prove that that \( g \) is a strong one-way function.

Assume for contradiction that there is some probabilistic polynomial time adversary \( A \) which can invert the randomized iterate of \( f \) with non-negligible probability. First, they prove that any weak OWF has a non-trivial “failing set” for any adversary. Informally speaking, the adversary will only be able to invert successfully with negligible probability on its failing set. Then, using \( A \) they construct an algorithm \( M^A \) which inverts the last iteration of the randomized iterate of \( f \). They use \( M^A \) as a subroutine to construct an adversary that inverts \( f \) with no failing set of density \( \delta(n) \), completing the proof by contradiction.

At a high level, the machine \( M^A \) takes as input \((y, i, h_1, \ldots, h_i)\) and tries to invert the \( i^{th} \) iterate. It does so by first sampling ran-
dom hash functions $h_{i+1}, \ldots, h_m$ and computing the output of the randomized iterate assuming $y$ is the output of the $j^{th}$ iterate, i.e. computes $g^{k-i}(h_{i+1}(y), h_{i+2}, \ldots, h_m)$ to obtain $w$. Then it runs $A$ on input $(w, h_1, \ldots, h_m)$ to obtain an inverse $x$ and outputs $g^i(x, h_1, \ldots, h_{i-1})$. If $A$ inverts successfully and provides an inverse that yields an inverse of $y$, then $M^A$ succeeds. They prove the following claim regarding $M^A$.

**Claim 4.23** (Claim 5 of [HHR06]). For any $n \in \mathbb{N}$ and $S \subseteq \{0, 1\}^n$ with
\[
\Pr[x \leftarrow \{0, 1\}^n : f(x) \in S] \geq \epsilon(n)/2,
\]
there exists an $i \in [k]$ such that
\[
\Pr \left[ x \leftarrow \{0, 1\}^n ; h_1, \ldots, h_i \leftarrow \mathcal{H} : M^A(f^{i+1}(x, h_1, \ldots, h_i), h_1, \ldots, h_i) = f^i(x, h_1, \ldots, h_i) \land f^i(x, h_1, \ldots, h_i) \in S \right] \geq \frac{\epsilon_A(n)^2}{9k(n)^2}
\]
where $\epsilon_A(n)$ is the probability that $A$ inverts $g$.

They combine the above claim with following lemma (which is analogous to our Lemma 4.19 regarding RGRI) to obtain the result (see [HHR06] for more details).

**Lemma 4.24** (Lemma 6.3 of [HHR06]). For any $n \in \mathbb{N}$ and $T \subseteq \{0, 1\}^n$ and any algorithm $M$ with
\[
\Pr \left[ x \leftarrow \{0, 1\}^n ; h_1, \ldots, h_i \leftarrow \mathcal{H} : M(f^{i+1}(x, h_1, \ldots, h_i), h_1, \ldots, h_i) = f^i(x, h_1, \ldots, h_i) \land f^i(x, h_1, \ldots, h_i) \in S \right] = \epsilon_M
\]
it must be true that

\[
\Pr \left[ x \leftarrow \{0, 1\}^n; h_1, \ldots, h_i \leftarrow \mathcal{H} : M(f(x), h_1, \ldots, h_i) = f^{-1}(f(x)) \land f(x) \in S \right] \geq \frac{\epsilon^2 M}{i+1}.
\]

To prove that the RGRI using \( f \) achieves the same result, it suffices to prove an analogous version of Claim 4.23 and then we combine with Lemma 4.19 to obtain the same result. We provide a proof sketch of Claim 4.23 below. In fact, the proof of Claim 4.23 relies only on the following two facts and we prove the analogous versions of these facts for the RGRI and obtain our result.

1. For any \( y \neq y' \in \text{Im}(f) \),

\[
\Pr[h_1, \ldots, h_k \leftarrow \mathcal{H} : f^{k+1}(f^{-1}(y), h_1, \ldots, h_k) = f^{k+1}(f^{-1}(y'), h_1, \ldots, h_k)] \leq \frac{k}{|\text{Im}(f)|}
\]

2. For any \( S \subseteq \{0, 1\}^n \) with \( \Pr[x \leftarrow \{0, 1\}^n : f(x) \in S] \geq \frac{\epsilon}{2} \),

\[
\Pr[x \leftarrow \{0, 1\}^n; h_1, \ldots, h_k \leftarrow \mathcal{H} : \exists i \in [k] \text{ such that } f^{i+1}(x, h_1, \ldots, h_k) \in S] \geq 1 - \nu(n)
\]

for some negligible function \( \nu(n) \). \(^6\)

For RGRI, fact 1 follows immediately from Claim 4.20. Therefore, it only remains to prove fact 2. We call this the target set theorem and

---

\(^6\)In [HHR06], they show something stronger, i.e. for every \( x \in \{0, 1\}^n \), the probability of missing \( S \) is at most \( 2^{-2} \). However, we state and prove this weaker property as it suffices to achieve the same result.
prove it in the next section. Thus, we obtain the following hardness-amplification theorem for regular one-way functions.

**Theorem 4.25.** Suppose $f$ is a length-preserving weak one-way function that is regular. Let $m \geq \lceil \log k \rceil + 1$ and $\tilde{f}^k$ is defined as in Definition 4.16. Then there exists a polynomial $k$ such that $\hat{f}^k$ defined as

$$\hat{f}^k(x, h_1, \ldots, h_m) = (\tilde{f}^k(x, h_1, \ldots, h_m), h_1, \ldots, h_m)$$

is a strong one-way function.

### 4.4.1 Target Set Lemma

The target set lemma states that for any target set that isn’t too small, the probability that the inputs to all iterates in the RGRI of a regular one-way function is likely to miss the target set is small.

**Lemma 4.26 (Target Set Lemma).** Let $f$ be a length-preserving regular function, where every element of the image of $f$ has exactly $2^{r(n)} = 2^r$ preimages. Let $S(n)$ be a set of strings s.t. $\Pr[x \leftarrow U_n : f(x) \in S(n)] \geq \frac{1}{p(n)}$ for some polynomial function $p$. Let $f^i(x, h_1, \ldots, h_m)$ be the RGRI of $f$ (not including the hash functions in its output). Then there exists $k = O(\log n)$
(where the constant factor depends only on $p$ and $f$) such that

$$\Pr[x \leftarrow \{0, 1\}^n : h_1, \ldots, h_k \leftarrow \mathcal{H} : \exists i \in [k] \text{ such that } f^{i+1}(x, h_1, \ldots, h_k) \in S(n)] \geq 1 - \nu(n)$$

for some negligible function $\nu$.

**Proof:** For a set of hash functions, we consider the “bad set” to be the set of inputs which lead to $f^k$ never hitting $S$. We will make our main argument by induction on the number of hash functions in our randomized iterate, and use Chebyshev’s inequality to show that the expected size of the bad set is very small.

Let $B(\cdot) = \bar{S}$ and $B(h_1, \ldots, h_i) = \{y \in \{0, 1\}^n \mid \forall j \in \{0, \ldots, m\} \ f^j(f^{-1}(y), h_1, \ldots, h_i) \notin S\}$. The core of the proof is in the following claim.

**Claim 4.27.** Let $E$ denote the event that the density of $B(h_1, \ldots, h_{i-1})$ is at most $\beta > 2^{-n-r}$ when $h_1, \ldots, h_{i-1}$ are chosen at random. Then there exist negligible functions $\nu_1$ and $\nu_2$ such that the probability that the density of $B(h_1, \ldots, h_i)$ is more than $\beta^2(1 + \nu_1(n))$ for random hash functions $h_1, \ldots, h_i$ conditioned on event $E$ occurring is at most $\nu_2(n)$.

Recall that

$$\frac{B(h_1, \ldots, h_n)}{2^{n-r}} = \frac{\bar{S}}{2^{n-r}} \geq 1 - \frac{1}{p(n)}.$$ 

By applying the lemma iteratively and using the union bound, we can
conclude that the for any $k$,

$$\Pr[h_1, \ldots, h_k \leftarrow \mathcal{H} : \frac{\mathcal{B}(h_1, \ldots, h_k)}{2^n - r} \geq \left(1 - \frac{1}{p(n)}\right)^{2^k} (1 + v_1(n))^{2^k - 1}] \leq \frac{k}{v_2(n)}.$$  

For $k = O(\log p(n))$ we obtain that $\left(1 - \frac{1}{p(n)}\right)^{2^k} (1 + v_1(n))^{2^k - 1}$ is negligible and $\frac{k}{v_2(n)}$ is also negligible. Therefore, we can conclude that

$$\Pr[x \leftarrow \{0, 1\}^n; (h_1, \ldots, h_k) \leftarrow \mathcal{H}^k :$$

$$\exists i \in [2^k] \text{ such that } f^i(x, h_1, \ldots, h_k) \in S(n)] \geq 1 - v(n)$$

for some negligible function $v$. It only remains to prove the claim.

**Proof:** [Proof of Claim 4.27] For all $a \in \text{Im}(f)$ and random hash functions $h_1, \ldots, h_i$ conditioned on event $E$, let $X_a$ be the indicator random variable which is 1 when $a$ is in the bad set of $(h_1, \ldots, h_i)$ and 0 otherwise. Note that the sum of all $X_a$ is the size of the $B$ of $(h_1, \ldots, h_i)$. In preparation for Chebyshev’s inequality, we will now compute the expected value and variance of each $X_a$.

If $a \in B(h_1, \ldots, h_{i-1})$, then $E[X_a] = V[X_a] = 0$. For the rest of the process of computing $E[X_a]$ and $V[X_a]$ we will assume this does not happen. By the construction of RGRI, we have that for any $a$, and fixed hash functions $h_1, \ldots, h_{i-1}$ (satisfying event $E$), we have that $B(h_1, \ldots, h_{i-1})$
has density at most $\beta$. Furthermore, for any $a \in \mathcal{B}(h_1, \ldots, h_{i-1})$, it holds that $a \in \mathcal{B}(h_1, \ldots, h_i)$ if and only if $h_i(f(f^{2i-1}(f^{-1}(a), h_1, \ldots, h_i))) \in \mathcal{B}(h_1, \ldots, h_{i-1})$. Since $h_i$ is chosen from a pairwise independent hash function family (and hence also 1-wise independent), this happens with probability at most $\beta$. Therefore, for any such $a$ it holds that

$$E[X_a] = \Pr[X_a = 1] \leq \beta$$ and
$$V[X_a] = E[X_a^2] - E[X_a]^2 \leq \beta$$

Therefore we have that

$$\mu = E[\sum X_a] \leq 2^{n-r} \beta^2$$

since with probability at most $\beta$, it holds that $a \in \mathcal{B}(h_1, \ldots, h_{i-1})$.

We also need to compute $V[\sum X_a]$. If the set of all $X_a$ were pairwise-independent, or at least pairwise-noncorrelated, then we could say that

$$V[\sum X_a] = \sum V[X_a].$$

While the set of all $X_a$ is not pairwise-independent, we will still be able to bound the variance as they are conditionally independent. In fact, we show at the end of the proof that

$$\sigma^2 = V[\sum X_a] \leq 2^{i+2}(\beta^2 2^{n-r}).$$
Using Chebyshev’s inequality we have that

$$\Pr[\sum X_a > \mu + \delta \sigma] \leq \frac{1}{\delta^2}.$$  

Setting $$\delta = \sigma \nu(n)/2^k$$ and observing that $$i < k$$ we obtain that

$$\Pr \left[ \left| B(h_1, \ldots, h_i) \right| > \beta^2 (1 + \nu(n)) \right] \leq \frac{2^{2k-2i-4}}{\nu(n)^2 \beta^2 2^{n-r}}.$$  

Observe that the right-hand side is negligible in $$n$$ as long as $$\beta > 2^{-\frac{n-m}{\log n}}$$ and $$\nu(n) < 2^{-\frac{n-m}{\log n}}$$ since $$r$$ is at most $$n - \omega(\log n)$$ and $$k$$ is polynomial in $$n$$. It only remains the bound the variance of the sum of $$X_a$$.

First note that $$V[\sum_a X_a] = \sum_a V[X_a] + \sum_{a \neq b} \text{Cov}[X_a, X_b]$$. We need to compute the covariance of pairs of indicator variables. In the event that $$a$$ or $$b$$ is in $$B(h_1, \ldots, h_{i-1})$$, the covariance will be zero, so for the purposes of computing the covariance, we will skip this case and only consider it again once we sum all of the covariances together. Let $$F_{a,b}$$ be the event that occurs when

$$f^{2i-1}(f^{-1}(a), h_1, \ldots, h_{i-1}) = f^{2i-1}(f^{-1}(a), h_1, \ldots, h_{i-1}).$$

Conditioned on $$F_{a,b}$$ not occurring, by the pairwise-independence property of the family $$\mathcal{H}$$, we have that the probability that $$a$$ or $$b$$ is in $$B(h_1, \ldots, h_i)$$ is independent over a random $$h_i$$. $$\Pr[X_a = 1]$$ in this case translates to $$\Pr[h_i(f(f^{2i-1}(f^{-1}(a), h_1, \ldots, h_{i-1}))) \in B(h_1, \ldots, h_{i-1})]$$.  

To bound \( \text{Cov}[X_a, X_b] \), it suffices to bound the probability of \( F_{a,b} \). Using Claim 4.20, it follows that \( \Pr[F_{a,b}] \leq \frac{2^{i+1}}{2^{n-r}} \). Note here that we are not conditioning on the event \( E \). Since we are using the probability in the union bound, it will not affect our computation.

It follows that

\[
\text{Cov}[X_a, X_b] \leq \Pr[F_{a,b}] V[X_a] \leq \frac{2^{i+1}}{2^{n-r}} \beta.
\]

Now we are ready to bound the variance of the sum. Note that

\[
\sum_a V[X_a] = 2^{n-r} \ast (0 \ast (1 - \beta) + (\beta - \beta^2) \ast \beta) \leq 2^{n-r} \beta^2.
\]

Therefore we have

\[
V[\sum_a X_a] \leq \sum_a V[X_a] + \sum_a \sum_{b \neq a} \Pr[F_{a,b}] \ast V[X_a]
\]

\[
\leq 2^{n-r} \beta^2 + 2^{n-r} 2^{n-r} \frac{2^{i+1}}{2^{n-r}} \beta^3
\]

\[
\leq \beta^2 2^{i+2+n-r}.
\]

This completes the claim. ■

Given the claim, the target set lemma follows. ■
Input: $i, z_0, z_1, a_0, a_1$

1. Set Flags $F_0, \ldots, F_{m-1}$ to false // Flags indicate which hash functions are assigned

2. while $i \neq 0$
   
   (a) $j \leftarrow (i - 2^\phi(i))$ // The new condition will be at position $j$
   
   (b) Randomly choose $(x^j_0, x^j_1)$ from $(\{0,1\})^n$. For all $j < c < i$, if $F_{\phi(c)} = \text{false}$, randomly choose $h_{\phi(c)}$ from $\mathcal{H}$ and set $F_{\phi(c)} \leftarrow \text{true}$.
   
   (c) Compute
   
   
   
   
   $x^j_0 \xrightarrow{f} ||z_0, [i \ell + 1...i\ell+1]|| \xrightarrow{h_{\phi(i+1)}} f \xrightarrow{\ldots} h_{\phi(i-1)} \xrightarrow{f} y_0$
   
   $x^j_1 \xrightarrow{f} ||z_1, [i \ell + 1...i\ell+1]|| \xrightarrow{h_{\phi(i+1)}} f \xrightarrow{\ldots} h_{\phi(i-1)} \xrightarrow{f} y_1$
   
   (d) Randomly choose $h \in \mathcal{H}$ conditioned on
   
   
   $h(y_0||z_0, [(i-1)\ell + 1...i\ell]) = a_0$ and $h(y_1||z_1, [(i-1)\ell + 1...i\ell]) = a_1$.
   
   Set $h_{\phi(i)} \leftarrow h, F_{\phi(i)} \leftarrow \text{true}$.
   
   (e) $i \leftarrow j$, $a_0 \leftarrow x^j_0$, $a_1 \leftarrow x^j_1$

3. endwhile

4. For all $c$, if $F_{\phi(c)} = \text{false}$, pick $h_{\phi(c)}$ uniformly from $\mathcal{H}$ and set $F_{\phi(c)}$ to true.

5. output $(x_0 = x^i_0, (h_1, \ldots, h_m)), (x_1 = x^i_1, (h_1, \ldots, h_m))$.

Figure 4.1: Shoup Reconstruction Algorithm
Chapter 5

Ligero

5.1 Preliminaries

5.1.1 Collision-Resistant Hashing and Merkle Trees

Let \( \{H_\kappa\}_{\kappa \in \mathbb{N}} = \{H : \{0, 1\}^{p(\kappa)} \to \{0, 1\}^{p'(\kappa)}\}_{\kappa} \) be a family of hash functions, where \( p(\cdot) \) and \( p'(\cdot) \) are polynomials so that \( p'(\kappa) \leq p(\kappa) \) for sufficiently large \( \kappa \in \mathbb{N} \). For a hash function \( H \leftarrow \mathcal{H}_\kappa \) a Merkle hash tree [Mer89] is a data structure that allows to commit to \( \ell = 2^d \) messages by a single hash value \( h \) such that revealing any message requires only to reveal \( O(d) \) hash values.

A Merkle hash tree is represented by a binary tree of depth \( d \) where the \( \ell \) messages \( m_1, \ldots, m_\ell \) are assigned to the leaves of the tree; the values assigned to the internal nodes are computed using the underlying hash function \( H \) that is applied on the values assigned to the children, whereas the value \( h \) that commits to \( m_1, \ldots, m_\ell \) is assigned to the root.
of the tree. To open the commitment to a message $m_i$, the ITM that made the commitment reveals $m_i$ together with all the values assigned to nodes on the path from the root to $m_i$, and the values assigned to the siblings of these nodes. We denote the algorithm of committing to $\ell$ messages $m_1, \ldots, m_\ell$ by $h := \text{Commit}_M(m_1, \ldots, m_\ell)$ and the opening of $m_i$ by $(m_i, \text{path}(i)) := \text{Open}_M(h, i)$. Verifying the opening of $m_i$ is carried out by essentially recomputing the entire path from the relevant leaves up to the root and comparing the final outcome (i.e., the root) to the value given at the commitment phase.

The binding property of a Merkle hash tree is due to collision-resistance. Intuitively, this says that it is infeasible to efficiently find a pair $(x, x')$ so that $H(x) = H(x')$, where $H \leftarrow \mathcal{H}_\kappa$ for sufficiently large $\kappa$. In fact, one can show that Merkle hashing is collision resistant given the collision resistance of $\{H_\kappa\}_{\kappa \in \mathbb{N}}$. Formally, we say that a family of hash functions $\{H_\kappa\}_\kappa$ is collision resistant if for any probabilistic polynomial-time adversary $A$ the following experiment outputs 1 with negligible probability: (i) a hash function $H$ is sampled from $\mathcal{H}_\kappa$; (ii) the adversary $A$ is given $H$ and outputs $x, x'$; (iii) the experiment outputs 1 if and only if $x \neq x'$ and $H(x) = H(x')$.

In the random oracle model, the Merkle tree can be computed by replacing the function $H$ with a random oracle $\rho$ where statistical binding follows due to the hardness of finding a collision in this model. We denote this algorithm by $\text{Commit}_M^{\text{RO}}$. 
5.1.2 Zero-Knowledge Arguments

**Definition 5.1 (Zero Knowledge).** Let \((\mathcal{P}, \mathcal{V})\) be an interactive proof system for some language \(L\). We say that \((\mathcal{P}, \mathcal{V})\) is *computational zero-knowledge with respect to an auxiliary input* if for every polynomial-time interactive Turing machine \(\mathcal{V}^*\) there exists a probabilistic polynomial-time algorithm \(S\), running in time polynomial in the length of its first input, such that

\[
\{\text{view}_{\mathcal{V}^*}(\langle \mathcal{P}(w), \mathcal{V}^*(z)\rangle(x))\}_{x \in L, w \in \mathcal{R}, z \in \{0,1\}^*} \approx \{S(x, z)\}_{x \in L, z \in \{0,1\}^*}
\]

(when the distinguishing gap is considered as a function of \(|x|\)). Specifically, the left term denotes the view of \(\mathcal{V}^*\) after it interacts with \(\mathcal{P}\) on common input \(x\) whereas, the right term denote the output of \(S(x, z)\). We also require that \(\mathcal{V}\) run in polynomial time in terms of the length of the common input \(x\).

Our zero-knowledge protocols in fact satisfy the additional *proof of knowledge* property, which is important for some applications.

5.1.3 Interactive PCPs

An interactive probabilistically-checkable proof [KR08] (IPCP) is a combination of a traditional PCP with an interactive proof. An IPCP is a special case of interactive oracle proofs (IOP) [BCS16] (also known as
probabilistically-checkable interactive proofs [RRR16]). We will be interested in zero-knowledge interactive PCPs [GIMS10] in which the verifier reads a small number of bits from the PCP and exchanges a small number of bits with the prover \( P \). We formalize this notion below.

**Definition 5.2** (Interactive PCP). Let \( R(x,w) \) be an \( \text{NP} \) relation corresponding to an \( \text{NP} \) language \( L \). A pair of polynomial-time ITMs \((P,V)\) is an interactive PCP (IPCP) system for \( R \) with parameters \( (q,l,\epsilon) \) if and only if the following properties hold.

1. **Syntax:** On common input \( x \) and prover input \( w \), the prover \( P \) computes in time \( \text{poly}(|x|) \) a bit string \( \pi \) (referred to as the PCP). The prover \( P \) and verifier \( V \) then interact, where the verifier has oracle access to \( \pi \).

2. **Completeness:** If \( (x, w) \in R \) then

\[
\Pr[\text{out}_2(\langle P(w), V^{\pi}\rangle(x)) = 1] = 1.
\]

3. **Soundness:** For every \( x \notin L \), every (unbounded) interactive Turing machine \( P^* \) and every \( \tilde{\pi} \in \{0,1\}^* \),

\[
\Pr[\text{out}_2(\langle P^*(w), V^{\tilde{\pi}}\rangle(x)) = 1] \leq \epsilon(|x|).
\]

4. **Complexity:** In the interaction \( \langle P(w), V^{\pi}\rangle(x) \) at most \( l(|x|) \) bits are communicated and \( V \) reads at most \( q(|x|) \) bits of \( \pi \).
A public-coin IPCP is one where every message sent by the verifier is a uniformly and independently random string of bits.

Our zero-knowledge variants of IPCP achieve perfect zero-knowledge against an honest verifier.

**Definition 5.3** (Zero-knowledge IPCP). Let \((P, V)\) be an interactive PCP for \(R\). We say that \((P, V)\) is an (honest verifier, perfect) zero-knowledge IPCP (or ZKIPCP for short) if there exists an expected polynomial time algorithm \(S\) such that for any \((x, w) \in R\) the output of \(S(x)\) is distributed identically to the view of \(V\) in the interaction \(\langle P(w), V^\pi \rangle(x)\).

### 5.1.4 Secret Sharing

A secret-sharing scheme allows distribution of a secret among a group of \(n\) players, each of whom in a *sharing phase* receive a share (or piece) of the secret. In its simplest form, the goal of secret sharing is to allow only subsets of players of size at least \(t + 1\) to reconstruct the secret. More formally, a \(t + 1\)-out-of-\(n\) secret-sharing scheme comes with a sharing algorithm that on input \(s\) (representing the secret) outputs \(n\) shares \(s_1, \ldots, s_n\) and a reconstruction algorithm that takes as input \((s_i)_{i \in S}, S)\) where \(|S| > t\) and outputs either a secret \(s'\) or \(\perp\). In this work, we will use the Shamir’s secret-sharing scheme \([Sha79]\) with secrets in \(F = GF(2^k)\) or in \(GF(p)\) for some large prime \(p\). We present the sharing and reconstruction algorithms below:
CHAPTER 5. LIGERO

Sharing algorithm \( \text{Share}(s,n,t) \): For any input \( s \in \mathbb{F} \), pick a random polynomial \( p(\cdot) \) of degree \( t \) in the polynomial-field \( \mathbb{F}[x] \) with the condition that \( p(0) = s \) and output \( p(1), \ldots, p(n) \).

Reconstruction algorithm \( \text{Reconst}((s'_i)_{i \in S}, n, t) \): For any input \( (s'_i)_{i \in S} \) where none of the \( s'_i \) are \( \perp \) and \( |S| > t \), compute a polynomial \( g(x) \) such that \( g(i) = s'_i \) for every \( i \in S \). This is possible using Lagrange interpolation where \( g \) is given by

\[
g(x) = \sum_{i \in S} s'_i \prod_{j \in S / \{i\}} \frac{x - j}{i - j}.
\]

Finally the reconstruction algorithm outputs \( g(0) \).

Packed Secret Sharing The concept of packed secret sharing was introduced by Franklin and Yung [FY92] in order to reduce the communication complexity of secure multiparty protocols, and is an extension of standard secret sharing. In [FY92] the authors considered Shamir’s secret sharing modified so that the number of secrets \( s_1, \ldots, s_\ell \) is now \( \ell \) instead of a single secret, evaluated by a polynomial \( p(\cdot) \) on \( \ell \) distinct points. To ensure privacy in case of \( t \) corrupted parties, the random polynomial must have degree at least \( t + \ell - 1 \). We use packed secret sharing in our underlying MPC protocol to save on communication complexity. We denote a packed secret-sharing scheme for \( \ell \) secrets by the pair of algorithms \( (\text{Share}_\ell, \text{Reconst}_\ell) \) and formalize the privacy guarantee of this primitive as
follows.

**Definition 5.4** (Perfect $t$-Private Packed Secret Sharing). A packed secret-sharing scheme $(\text{Share}_\ell, \text{Reconst}_\ell)$ is perfect $t$-private if and only if for every set $T \subseteq [n]$ such that $|T| \leq t$ and for any two sets of $\ell$ secrets $(s_1, \ldots, s_\ell)$ and $(s'_1, \ldots, s'_\ell)$ where $s_i, s'_i \in F$ for all $i \in [\ell]$ it holds that

$$\text{Share}_T(s_1, \ldots, s_\ell, n, t) \equiv \text{Share}_T(s'_1, \ldots, s'_\ell, n, t)$$

where $\text{Share}_T$ is the restriction of the outputs of $\text{Share}_\ell$ to the elements in $T$.

### 5.2 From MPC to ZKIPCP

#### 5.2.1 Our MPC Model

As mentioned in the introduction, the efficiency of our constructions can be distilled to identifying the right MPC model and designing an efficient protocol in this model. In this regards we deviate from the original work of [IKOS07], which provided a general transformation from any honest majority MPC protocol that can compute arbitrary probabilistic polynomial-time functionalities. In particular, our model is more in line with the watchlist mechanism of [IPS08]. We begin with the description of the MPC model and the protocol specifications that we will need to design our zero-knowledge protocol. In Section 5.3, we use MPC protocols
based on the works [Di06, CC06, IPS08, IPS09].

In our model, we consider a sender client $S$, $n$ servers $s_1, \ldots, s_n$ and a receiver client $R$. The sender has input $x$ and a witness $w$ with respect to some NP relation $R$. The receiver and the servers do not receive any input: the servers obtain random shares from the sender and evaluate the computed circuit. Upon receiving $(x, w)$ from the sender, the functionality computes $R(x, w)$ and forwards the result to the receiver $R$. We consider the specific network where the communication is restricted to a single message between $S$ and the servers at the beginning of the protocol and a single message from the servers to the receiver $R$ at the end of the protocol. Moreover, the only way the servers may communicate with each other is with a broadcast. In our actual MPC protocol, the servers will never utilize such a broadcast. Nevertheless, our transformation from MPC to zero knowledge can be extended to allow for the servers to invoke a broadcast. For simplicity, we will restrict the servers to not communicate with each other at all in our actual transformation.

We consider the security of our underlying protocols in both the “honest but curious” (passive) and the malicious (active) models. In the former model, one may break the security requirements into the following correctness and privacy requirements.

**Definition 5.5** (Correctness). We say that $\Pi$ realizes a deterministic $n + 1$-party functionality $(x, r_1, \ldots, r_n)$ with perfect (resp., statistical) correctness if for all inputs $(x, r_1, \ldots, r_n)$, the probability that the output of some
player is different from the output of \( f \) is 0 (resp., negligible in \( \kappa \)), where the probability is over the independent choices of the random inputs \( r_1, \ldots, r_n \).

**Definition 5.6** (\( t_p \)-Privacy). Let \( 1 \leq t_p < n \). We say that \( \Pi \) realizes \( f \) with perfect \( t_p \)-privacy if there is a probabilistic polynomial-time simulator \( S \) such that for any inputs \( (x, r_1, \ldots, r_n) \) and every set of corrupted players \( T \subseteq [n] \), where \( |T| \leq t_p \), the joint view \( \text{View}_T(x, r_1, \ldots, r_n) \) of players in \( T \) is distributed identically to \( S(T, x, \{r_i\}_{i \in T}, f_T(x, r_1, \ldots, r_n)) \).

With respect to our MPC model defined above, we consider privacy in the presence of a static passive adversary that corrupts the receiver \( R \) and at most \( t_p \) servers. Our zero-knowledge property will reduce to this security guarantee.

In the malicious model, in which corrupted players may behave arbitrarily, security cannot be generally broken into correctness and privacy as above. However, for our purposes we only need the protocols to satisfy a weaker notion of security in the malicious model than is implied by the standard general definition. Specifically, it suffices that \( \Pi \) be \( t_p \)-private as above, and it must satisfy the following notion of correctness in the malicious model which we reduce the soundness property to.

**Definition 5.7** (Statistical \( t_r \)-Robustness). We say that \( \Pi \) realizes \( f \) with statistical \( t_r \)-robustness if it is perfectly correct in the presence of an “honest but curious” adversary as in Definition 5.5, and furthermore for any
(unbounded) active adversary that adaptively corruptions a set $T$ of at most $t_r$ players, and for any inputs $(x, r_1, \ldots, r_n)$, the following robustness property holds: if there is no $(r_1, \ldots, r_n)$ such that $f(x, r_1, \ldots, r_n) = 1$, then the probability that $R$ outputs 1 in an execution of $\Pi$ in which the inputs of the honest players are consistent with $(x, r_1, \ldots, r_n)$ is negligible in $\kappa$ where $\kappa$ is a statistical parameter that the protocol $\Pi$ receives as input.

We prove our main theorems about our two-party zero-knowledge protocol in the presence of a static active adversary that corrupts the prover at the start of the interaction. Nevertheless, our proof relies on the security of the underlying MPC protocol (utilized in the “MPC in the head” paradigm) being robust against an active adversary that adaptively corrupts a subset of the servers in the underlying MPC protocol. Concretely, with respect to our MPC model defined above, we consider robustness in the presence of an adaptive active adversary that corrupts the sender $S$ and at most $t_r$ servers.

Finally, when used in the “MPC in the head” paradigm, we need the notion of consistent views between servers and the receiver that we define below.

**Definition 5.8 (Consistent views).** We say that a pair of views $V_i, V_j$ is consistent (with respect to the protocol $\Pi$ and some public input $x$) if the outgoing messages implicit in $V_i$ are identical to the incoming messages reported in $V_j$ and vice versa.
5.2.2 ZKIPCP for NP - The General Case

Next, we provide our construction of an IPCP from an MPC protocol satisfying the requirements specified in Section 5.2.1. We note that while the construction presented in this section works for any MPC in the model as described in the previous section, we will simplify our MPC model.

Two Phases: The protocol we consider will proceed in two phases: In phase 1, the servers receive inputs from the sender and only perform local computation. After phase 1, the servers obtain a public random string $r$ of length $l$ sampled via a coin-flipping oracle and broadcasted to all servers. The servers use this in phase 2 for their local computation. At the end of the local computation, each server sends a single output message to the receiver $R$.

No Broadcast: The servers never communicate with each other. Each server simply receives inputs from the sender at the beginning of phase 1, then receives a public random string in phase 2, and finally delivers a message to $R$.

Formally, let $L$ be an $\text{NP}$ language with $\text{NP}$ relation $R$, let $x$ an $\text{NP}$ statement that is the common input and let $w$ be the private input of the prover. We will now design a ZKIPCP protocol $\Pi_{ZKIPCP}$ (Figure 5.1) that meets Definition 5.2 based on any MPC protocol $\Pi$ that is defined according to our model described above.

We are now ready to prove the following theorem.
\begin{itemize}
\item **Input:** The prover $P$ and the verifier $V$ share a common input statement $x$ and a circuit description $C$ that realizes $R$. $P$ additionally has input $w$ such that $(x, w) \in R$.
\item **Oracle $\pi$:** The prover runs the MPC protocol $\Pi$ “in its head” as follows. It picks a random input $r_S$ and invokes $S$ on $(x, w; r_S)$ and a random input $r_i$ for every server $s_i$. The prover computes the views of the servers up to the end of phase 1 in $\Pi$, denoted by $(V_1, \ldots, V_n)$, and sets the oracle to the string $(V_1, \ldots, V_n)$.
\item **The interactive protocol.**
\begin{enumerate}
\item $V$ picks a random challenge $r$ of length $l$ and sends it to the sender.
\item Upon receiving the challenge $r$, prover $P$ sends the view $V$ of $R$. (As the prover possesses all the information about the servers, and the verifier always receives the broadcast message from each server, these broadcast messages can be sent directly from the prover to the verifier.)
\item $V$ computes the output of $R$ from the view and checks if $R$ does not abort. It then picks a random subset $Q$ of $[n]$ of size $t_p$ uniformly at random (with repetitions), and queries the oracle on $Q$.
\item $V$ obtains from the oracle the views of the servers in $Q$.
\item $V$ rejects if the views of the servers are inconsistent with the view of $R$. Otherwise, it accepts.
\end{enumerate}
\end{itemize}

Figure 5.1: Protocol $\Pi_{ZKIPCP}$

**Theorem 5.9.** Let $f$ be the following functionality for a sender $S$ and $n$ servers $s_1, \ldots, s_n$ and receiver $R$. Given a public statement $x$ and an additional input $w$ received from $S$, the functionality delivers $R(x, w)$ to $R$. Suppose that $\Pi$ is a two-phase protocol in the MPC model specified in Section 5.2.1 that realizes $f$ with statistical $t_r$-robustness (in the malicious
model) and perfect $t_p$-privacy (in the “honest but curious” model), where $t_r < \left\lceil \frac{n}{2} \right\rceil - 1$. Then protocol $\Pi_{ZKIPCP}$ described above is a ZKIPCP for $\mathbf{NP}$ relation $\mathcal{R}$, with soundness error $(1 - \frac{t_r}{n})t_r + \delta(\kappa)$ where $\delta(\kappa)$ is the robustness error of $\Pi$.

**Proof:** Our proof follows by establishing completeness, soundness and zero knowledge as required in Definition 5.2 and Definition 5.3.

**Completeness:** Completeness follows directly from the correctness of the underlying MPC protocol.

**Soundness:** Consider a statement $x \notin \mathcal{L}_\mathcal{R}$. We will show that no prover $\mathcal{P}^*$ can convince $\mathcal{V}$ to accept a false statement with more than negligible probability. We will argue soundness by following an approach similar to [IKOS07] where we first identify an inconsistency graph and then invoke the properties of the underlying MPC. More precisely, we consider an inconsistency graph $G$ based on the $n$ views $V_1, \ldots, V_n$ and the view of the receiver $R$ which contains the messages from servers $s_1, \ldots, s_n$ to $R$. Here, the servers and the receiver correspond to nodes in $G$ and inconsistency between every pair of nodes is defined as in Definition 5.8. Then there are two cases depending on the graph $G$:

**Case 1:** There are more than $t_r$ edges in $G$. In this case, we will argue that with high probability the set of servers opened by the verifier will hit one of these edges. Recall that the view of $R$ is provided to the verifier. Therefore, for any edge in $G$ between $R$ and $V_i$, if the corresponding server
s_i falls in Q, then the verifier rejects. The probability that all t_p servers chosen by the verifier misses all inconsistent edges is at most \((1 - \frac{t}{n})^{t_p}\).

**Case 2: There are fewer than t_r edges in G.** In this case, we will argue that by the statistical t_r-robustness of the underlying MPC protocol \(\Pi\), the verifier will reject except with probability \(\delta(\kappa)\). More precisely, for every cheating ITM \(P^*\) in the zero-knowledge proof we will demonstrate an adversarial strategy \(A\) attacking the underlying MPC protocol such that the probability with which \(V\) accepts a false statement when interacting with \(P^*\) on a false statement will be bounded by the probability that \(R\) outputs 1 in an execution of the underlying MPC protocol with adversary \(A\).

More precisely, consider an adversary \(A\) that is participating in the MPC protocol with \(n\) servers, a sender and a receiver. Internally, \(A\) incorporates the code of \(P^*\) while emulating the roles of the oracle and \(V\). When the protocol begins, \(P^*\) sets the oracle to the list of views of the servers as in phase 1 of \(\Pi\). These views simply contain the inputs sent to the servers (as all computations are local). After obtaining the views of the servers, \(A\) corrupts the sender in the external MPC execution, and, acting as the sender, \(A\) sends, as input to server \(s_i\), the value that was internally-generated by \(P^*\) as the view of that server, namely \(V_i\). Next, recall that in the MPC protocol the servers receive a random string from the coin-flipping oracle (in our protocol the verifier picks \(r\) as the challenge in Step 1). \(A\) internally forwards this string \(r\) to \(P^*\) as the message provided by the verifier.
Next, $\mathcal{A}$ proceeds with the internal execution by selecting $t_p$ indices for the verifier’s challenge. The oracle reveals the views of these servers. If $\mathcal{V}$ rejects in the internal execution because any of these views are inconsistent, then $\mathcal{A}$ aborts. Otherwise, $\mathcal{A}$ continues with the external execution.

Recall that in phase 2, each server sends a single message to $R$. Then just before the servers send these messages, $\mathcal{A}$ computes the inconsistency graph $G$. Recall that an edge is present between a server $s_i$ and the receiver $R$ in this graph if the view of $s_i$ is inconsistent with the view of $R$ and randomness $r$. Let $T$ be the set of servers of size $t^*$ that are connected to an edge in $G$. If $t^* > t_r$, then $\mathcal{A}$ aborts. Otherwise, $\mathcal{A}$ (adaptively) corrupts the servers in $T$ and replaces their (honestly-generated) messages sent to $R$ by what was internally-reported in the view of $R$, namely, the messages sent by $P^*$ to the verifier in the proof.

It follows from this description that the acceptance condition of the verifier in the internal emulation with $\mathcal{A}$ is identical to the output of $R$ in the external MPC execution. Since the underlying MPC protocol is $t_r$-robust and the number of parties corrupted by $\mathcal{A}$ is bounded by $t_r$, we have that $R$ outputs 0 except with probability $\delta(\kappa)$. We conclude that the verifier in the internal emulation by $P^*$ accepts the proof of a false statement except with probability at most $\delta(\kappa)$. Next, we observe that the view of the verifier emulated by $\mathcal{A}$ in the internal emulation is identically distributed to the view of an honest verifier in an interaction with $P^*$. Therefore, we can conclude that an honest verifier accepts a false
statement with probability at most $\delta(\kappa)$.

Applying a union bound, we conclude that the verifier accepts a false statement with probability at most $(1 - \frac{\kappa}{n})^{t_p} + \delta(\kappa)$.

**Zero Knowledge:** The zero-knowledge property follows from the $t_p$-privacy of the underlying MPC protocol $\Pi$. We construct a simulator $S$ that invokes the simulator for the MPC protocol, denoted by $S_{\Pi}$. $S_{\Pi}$ simulates an adversary $A$ that statically corrupts the receiver $R$ and adaptively corrupts the $t_p$ servers which have their views opened to check for consistency, where the server’s corruptions take place at the end of the computation. In this simulation, $S_{\Pi}$ is required to produce the view of $R$ upon receiving a challenge $r$. Next, upon obtaining the query $Q$ from the verifier, $S$ instructs $S_{\Pi}$ to output the views of these $t_p$ servers. □

**Communication Complexity:** The main source of complexity is in revealing the view of $R$ in the third message and revealing the view of the $t_p$ servers in the last message. If, for $i \in t_p$, the maximum size of the view of each server $s_i$ is $v_{\text{size}}$, and the size of the view of $R$ is $v_R$, then the total communication complexity from the prover is $t_p \cdot v_{\text{size}} + v_R$. In Section 5.3 we adjust the parameters of our protocol subject to the constraint that $v_{\text{size}}v_R = O(|C|)$. To minimize the communication complexity, if we set $t_p \cdot v_{\text{size}}$ and $v_R$ to be roughly equal then we obtain the optimum complexity of our approach.
5.3 A Direct ZKIPCP Construction

In this section we give a self-contained description of our zero-knowledge interactive PCP protocol. This protocol is a slightly-optimized version of the protocol obtained by applying our variant of the general “MPC to Zero Knowledge” transformation from [IKOS09] (see Section 5.2) to the honest-majority MPC protocol from [DI06].

**Coding notation.** For a code \( C \subseteq \Sigma^n \) and vector \( v \in \Sigma^n \), denote by \( d(v, C) \) the minimal distance of \( v \) from \( C \), namely the number of positions in which \( v \) differs from the closest codeword in \( C \), and by \( \Delta(v, C) \) the set of positions in which \( v \) differs from such a closest codeword (in case of ties, take the lexicographically first closest codeword), and by \( \Delta(V, C) = \bigcup_{v \in V} \{\Delta(v, C)\} \). We further denote by \( d(V, C) \) the minimal distance between a vector set \( V \) and a code \( C \), namely \( d(V, C) = \min_{v \in V} \{d(v, C)\} \).

Our ZKIPCP protocol uses Reed-Solomon codes, defined next.

**Definition 5.10 (Reed-Solomon Code).** For positive integers \( n, k \), a finite field \( \mathbb{F} \), and a vector \( \eta = (\eta_1, \ldots, \eta_n) \in \mathbb{F}^n \) of distinct field elements, the code \( \text{RS}_{\mathbb{F}, n, k, \eta} \) is the \([n, k, n - k + 1]\) linear code over \( \mathbb{F} \) that consists of all \( n \)-tuples \((p(\eta_1), \ldots, p(\eta_n))\) where \( p \) is a polynomial of degree \( < k \) over \( \mathbb{F} \).

**Definition 5.11 (Encoded message).** Let \( L = \text{RS}_{\mathbb{F}, n, k, \eta} \) be a Reed-Solomon code and \( \zeta = (\zeta_1, \ldots, \zeta_\ell) \) be a sequence of distinct elements of \( \mathbb{F} \) for \( \ell \leq k \). For \( u \in L \) we define the message \( \text{Dec}_\zeta(u) \) to be \((p_u(\zeta_1), \ldots, p_u(\zeta_\ell))\), where \( p_u \) is the polynomial (of degree \( < k \)) corresponding to \( u \). For \( U \in \)
Chapter 5. Ligero

Let $L^m$ with rows $u^1, \ldots, u^m \in L$, we let $\text{Dec}_\zeta(U)$ be the length $m\ell$ vector $x = (x_{11}, \ldots, x_{1\ell}, \ldots, x_{m1}, \ldots, x_{m\ell})$ such that $(x_{i1}, \ldots, x_{i\ell}) = \text{Dec}_\zeta(u^i)$ for $i \in [m]$. Finally, when $\zeta$ is clear from the context, we say that $U$ encodes $x$ if $x = \text{Dec}_\zeta(U)$.

At a very high level, our ZKIPCP protocol proves the satisfiability of an arithmetic circuit $C$ of size $s$ in the following way. The prover arranges (a slightly redundant representation of) the $s$ wire values of $C$ on a satisfying assignment in an $O(\sqrt{s}) \times O(\sqrt{s})$ matrix, and encodes each row of this matrix using a Reed-Solomon code. The verifier challenges the prover to reveal linear combinations of the entries of the codeword matrix, and checks their consistency with $t$ randomly-selected columns of this matrix, where $t$ is a security parameter. In the following we describe the ZKIPCP construction in a bottom-up fashion, first addressing the case of IPCP (without zero knowledge) and then introduce the modifications required for making it zero knowledge.

5.3.1 Testing Interleaved Linear Codes

We start by describing and analyzing a simple interactive prover-assisted protocol for simultaneously testing the membership of multiple vectors in a given linear code $L$. It will be convenient to view $m$-tuples of codewords in $L$ as codewords in an interleaved code $L^m$. We formally define this notion below.
**Definition 5.12** (Interleaved code). Let $L \subseteq \mathbb{F}^n$ be an $[n,k,d]$ linear code over $\mathbb{F}$. We let $L^m$ denote the $[n,mk,d]$ (interleaved) code over $\mathbb{F}^m$ whose codewords are all $m \times n$ matrices $U$ such that every row $U_i$ of $U$ satisfies $U_i \in L$. For $U \in L^m$ and $j \in [n]$, we denote by $U[j]$ the $j$th symbol (column) of $U$.

To test the membership of $U$ in $L^m$, $V$ challenges $P$ to reveal a random linear combination of the rows $U_i$, and then checks that the revealed codeword is consistent with a randomly selected set of $t$ columns of $U$. \(^1\) The complete test is described in Figure 5.2.

The following lemma follows directly from the linearity of $L$.

**Lemma 5.13.** If $U \in L^m$ and $P$ is honest, then $V$ always accepts.

Our soundness analysis will rely on the following lemma.

**Lemma 5.14.** Let $e$ be a positive integer such that $e < d/4$. Suppose $d(U^*, L^m) > e$. Then, for a random $w^*$ in the row-span of $U^*$,

$$\Pr[d(w^*, L) \leq e] \leq (e + 1)/|\mathbb{F}|.$$  

**Proof:** Let $L^*$ be the row span of $U^*$. We consider two cases.

\(^1\)This test is implicitly used in the verifiable secret sharing subprotocol of efficient MPC protocols from the literature, and in particular in the protocols from [D106, IPS09] we build on. Its soundness requires the MPC protocol to be *adaptively-secure* to accommodate $P$’s ability to make the locations of inconsistencies depend on $V$’s random challenge. When the MPC adversary is adaptive, it can potentially corrupt all parties observing such inconsistencies. Indeed, the construction of zero-knowledge proofs from statistically-secure MPC from [IKOS09] relies on the adaptive security of the underlying MPC protocol.
Oracle: An alleged $L^m$-codeword $U$. Depending on the context, we may view $U$ either as a matrix in $\mathbb{F}^{m \times n}$ in which each row $U_i$ is an alleged $L$-codeword, or as a sequence of $n$ symbols $(U[1], \ldots, U[n])$, $U[j] \in \mathbb{F}^m$.

Interactive Testing:

1. $\mathcal{V}$ picks a random linear combination $r \in \mathbb{F}^m$ and sends $r$ to $\mathcal{P}$.
2. $\mathcal{P}$ responds with $w = r^T U \in \mathbb{F}^n$.
3. $\mathcal{V}$ queries a set $Q \subseteq [n]$ of $t$ random symbols $U[j], j \in Q$.
4. $\mathcal{V}$ accepts if and only if $w \in L$ and $w$ is consistent with $U[Q]$ and $r$. That is, for every $j \in Q$ we have  
   $$\sum_{i=1}^m r_j \cdot U_{ij} = w_j.$$ 

Figure 5.2: TestInterleaved($\mathbb{F}, L[n, k, d], m, t; U$)

Case 1: There exists $v^* \in L^*$ such that $d(v^*, L) > 2e$. In this case, we show that

$$\Pr_{w^* \in R L^*} [d(w^*, L) \leq e] \leq 1/|\mathbb{F}|. \quad (5.1)$$

Using a basis for $L^*$ that includes $v^*$, a random $w^* \in L^*$ can be written as $\alpha v^* + x$, where $\alpha \in \mathbb{F}$ and $x$ is distributed independently of $\alpha$. We argue that, conditioned on any choice of $x$, there can be at most one choice of $\alpha$ such that $d(\alpha v^* + x, L) \leq e$, which implies (5.1). This follows by observing that if $d(\alpha v^* + x_0, L) \leq e$ and $d(\alpha' v^* + x_0, L) \leq e$ for $\alpha \neq \alpha'$, then by the triangle inequality it must be true that $d((\alpha - \alpha') v^*, L) \leq 2e$, 

$$\sum_{i=1}^m r_j \cdot U_{ij} = w_j.$$
contradicting the assumption that \(d(v^*, L) > 2e\).

**Case 2:** For every \(v^* \in L^*, d(v^*, L) \leq 2e\). We show that in this case 
\[
\Pr_{w^* \in R^{L^*}}[d(w^*, L) \leq e] \leq \frac{e+1}{|F|}.
\]
Let \(U^*_i\) be the \(i\)-th row of \(U^*\) and let \(E_i = \Delta(U^*_i, L)\). Note that, since \(2e < \frac{d}{2}\), each \(U^*_i\) can be written uniquely as \(U^*_i = u_i + \chi_i\) where \(u_i \in L\) and \(\chi_i\) is nonzero exactly in its \(E_i\) entries.

Let \(E = \bigcup_{i=1}^m E_i\). Since \(d(U^*, L^m) > e\), we have \(|E| > e\). We show that for each \(j \in E\) (except with probability \(\frac{1}{|F|}\) over the random choice of \(w^*\) from \(L^*\)) either \(j \in \Delta(w^*, L)\) or \(d(w^*, L) > e\). The claim follows from this.

Suppose \(j \in E_i\). As before, we write \(w^* = aU^*_i + x\) for \(a \in R \not \subseteq F\) and \(x\) distributed independently of \(a\). Condition on any possible choice \(x_0\) of \(x\).

Define a bad set
\[
B_j = \{a : j \notin \Delta(aU^*_i + x_0, L) \land d(aU^*_i + x_0, L) \leq e\}.
\]
We show that \(|B_j| \leq 1\). Suppose for contradiction that there are \(\{a, a'\} \subseteq F\) such that \(a \neq a'\) and for \(z = aU^*_i + x_0\) and \(z' = a'U^*_i + x_0\) we have \(d(z, L) \leq e\), \(d(z', L) \leq e\), \(j \notin \Delta(z, L)\), and \(j \notin \Delta(z', L)\). Since \(d > 4e\), for any \(z^*\) in the linear span of \(z\) and \(z'\) we have \(j \notin \Delta(z^*, L)\). Since \(U^*_i\) is in this linear span, we have \(j \notin \Delta(U^*_i, L)\). This contradicts the assumption that \(j \in E_i\). Therefore we have that \(|B_j| \leq 1\).

We have shown that for each \(j \in E\), conditioned on every possible choice of \(x\), either \(j \in \Delta(w^*, L)\) or \(d(w^*, L) > e\) except with probability \(\frac{1}{|F|}\) over the choice of \(a\). It follows that the same holds for a random choice.
of $x$. Taking a union bound over the first $e + 1$ elements of $E$ we get that
\[
\Pr_{w^* \in R^L}[d(w^*, L) \leq e] \leq \frac{e + 1}{|F|}
\] as required. \hfill \Box

We now prove the soundness of the testing procedure when the given oracle is far from $L^m$.

**Theorem 5.15.** Let $e$ be a positive integer such that $e < \frac{d}{4}$. Suppose $d(U^*, L^m) > e$. Then for any malicious $\mathcal{P}$ strategy, the oracle $U^*$ is rejected by $V$ except with probability at most $(1 - \frac{e}{n})^t + \frac{e + 1}{|F|}$.

**Proof:** Letting $w^* = r^TU^*$, it follows from Lemma 5.14 that
\[
\Pr[\mathcal{V} \text{ accepts } U^*] \leq \Pr[\mathcal{V} \text{ accepts } |d(w^*, L) > e] \\
+ \Pr[d(w^*, L) \leq e] \\
\leq \frac{(n-e-1)}{\binom{n}{t}} + \frac{e + 1}{|F|} \\
\leq (1 - \frac{e}{n})^t + \frac{e + 1}{|F|}
\]
as required. \hfill \Box

In Section 5.7.1 we present a simple generalization of the testing algorithm that uses $\sigma$ linear combinations to amplify soundness.

**A Tighter Analysis**

We show in this section that the requirement $e < \frac{d}{4}$ in Theorem 5.15 can be relaxed to $e < \frac{d}{3}$ or possibly even $e < \frac{d}{2}$ with essentially the same soundness error bound. In fact, it suffices for this to hold for Reed-Solomon
codes. Such a version of Theorem 5.15 would yield up to roughly 25% improvement in the size of our zero-knowledge arguments. Below we reduce such a version of Theorem 5.15 to a lemma about the distance of points on an affine line from a Reed-Solomon code.

We start by showing that for any linear code over a sufficiently-large field, when \( e < \frac{d}{3} \) we can restrict the attention to case 1 from the proof of Lemma 5.14.

**Lemma 5.16.** Let \( L \) be an \([n, k, d]\) linear code over \( \mathbb{F} \). Let \( e \) be a positive integer such that \( e < \frac{d}{3} \) and \( |\mathbb{F}| \geq e \). Suppose \( d(U^*, L^m) > e \). Then there exists \( v^* \in L^* \) such that \( d(v^*, L) > e \), where \( L^* \) is the row span of \( U^* \).

**Proof:** Assume for contradiction that \( d(v^*, L) \leq e \) for all \( v^* \in L^* \). Suppose \( v_0^* \in L^* \) maximizes the distance from \( L \). Since \( d(U^*, L^m) > e \), there must be a row \( U_i^* \) such that \( \Delta(U_i^*, L) \setminus \Delta(v_0^*, L) \neq \emptyset \). Let \( v_0^* = u_0 + \chi_0 \) and \( U_i^* = u_i + \chi_i \) for \( (u_0, u_i) \in L^2 \) and \( \chi_0, \chi_i \) of weight at most \( e \). We argue that there exists \( \alpha \in \mathbb{F} \) such that for \( \hat{v} = v_0^* + \alpha U_i^* \) we have \( d(\hat{v}, L) > d(v_0^*, L) \), contradicting the choice of \( v_0^* \). Specifically, since \( d(v_0^*, L) \leq e \) and \( d(U_i^*, L) \leq e \) (where both follow due to our assumption above), there is a vector \( w \in L \) such that the distance of \( w \) from \( \hat{v} \) is \( 2e \), namely \( w = u_0 + \alpha u_i \). Furthermore, since \( d(\hat{v}, L) < e \) there is a vector \( w' \in L \) such that the distance of \( w' \) from \( \hat{v} \) is less than \( e \). Since \( d > 3e \), it must be true that \( w = w' \) (otherwise \( d(w, w') \) would have been less than \( d \)). However, for any \( j \in \Delta(v_0^*, L) \cup \Delta(U_i^*, L) \) there is at most one choice of \( \alpha \) such that the \( j^{th} \) component of \( \chi_0 + \alpha \chi_i \) goes to zero. By a union bound, there
are at most $2e$ such values of $\alpha$. Considering any other $\alpha$, we arrive at a contradiction. □

Given Lemma 5.16, as we argue below, to obtain an equivalent guarantee to Lemma 5.14, it will suffice to show that in any affine subspace of $\mathbb{F}^n$, either all points are $e$-close to $L$ or almost all are not. This reduces to showing the same for 1-dimensional spaces, which is what we claim in the following lemma. The proof of this lemma is presented in Section 5.6.

**Lemma 5.17.** Let $L = \text{RS}_{F,n,k,\eta}$ be a Reed-Solomon code with minimal distance $d = n - k + 1$. Let $e$ be a positive integer such that $e < \frac{d}{3}$. For every $(u, v) \in (\mathbb{F}^n)^2$, define an affine line $\ell_{u,v} = \{u + \alpha v : \alpha \in \mathbb{F}\}$. Either (1) for every $x \in \ell_{u,v}$ we have $d(x, L) \leq e$, or (2) for at most $d$ points $x \in \ell_{u,v}$ we have $d(x, L) \leq e$.

We do not have a counterexample even when we relax $e < \frac{d}{2}$ and even when $L$ is a general linear code. Furthermore, even if the lemma holds with a relaxed version of condition (2) (such as if we replace $d$ with $n^2$), this relaxed version is still almost as good in the context of the efficiency of our ZKIPCP.

The following stronger version of Lemma 5.14 follows from Lemma 5.16 by extending Lemma 5.17 from lines to general affine subspaces. This extension follows from the fact that if a subspace has a point that is far from $L$, then we can partition the subspace (minus the point) into lines containing this point. Assuming the conjecture for lines, each
line should be almost entirely far from the code, so the subspace should be almost entirely far from the code.

In more detail, from Lemma 5.16, we know that if \(d(U^*, L^m) > e\) then there is a \(v^* \in L^*\) that is more than \(e\)-far from \(L\). Following the proof of Lemma 5.14, we only need to argue case 2, where we further have that \(v^*\) is at most \(2e\)-far from \(L\). Now, as before, we can express the points in \(L^*\) as \(x + \alpha v^*\) where \(\alpha \in \mathbb{R} \setminus \mathbb{F}\) and \(x\) is distributed independently of \(\alpha\). For any fixed \(x\), we have that there exists an \(\alpha\) such that \(x + \alpha v^*\) is more than \(e\)-far from \(L\). Now from part (2) in Lemma 5.17, we can conclude that there are at most \(d\) values of \(\alpha\) for which \(x + \alpha v^*\) is at most \(e\)-far from \(L\). Since this is true for each \(x\) (for each line), it is true for the entire space \(L^*\). More formally, we have the following lemma.

**Lemma 5.18.** Let \(L = \text{RS}_{\mathbb{F}, n, k, \eta}\) be a Reed-Solomon code with minimal distance \(d = n - k + 1\) and let \(e\) be a positive integer such that \(e < \frac{d}{3}\). Suppose \(d(U^*, L^m) > e\). Then, for a random \(w^*\) in the row span of \(U^*\),

\[
\Pr[d(w^*, L) \leq e] \leq \frac{d}{|\mathbb{F}|}.
\]

Lemma 5.18 implies the following stronger version of Theorem 5.15.

**Theorem 5.19.** Let \(e\) be a positive integer such that \(e < \frac{d}{3}\). Suppose \(d(U^*, L^m) > e\). Then for any malicious \(P\) strategy, the oracle \(U^*\) is rejected by \(V\) except with probability at most \((1 - \frac{e}{d})^l + \frac{d}{|\mathbb{F}|}\).
5.3.2 Testing Linear Constraints over Interleaved Reed-Solomon Codes

In this section we describe an efficient procedure for testing that a message encoded by an interleaved Reed-Solomon code satisfies a given set of linear constraints. This generalizes a procedure from [Gro09, IPS09] for testing that such an encoded message satisfies a given set of replication constraints. In the following we assign a message in $\mathbb{F}^\ell$ to a codeword in $\mathbb{F}^n$ by considering a fixed set of $\ell$ evaluation points of the polynomial defined by the codeword. Note that while each codeword has a unique message assigned to it, several different codewords can be “decoded” into the same message, as the degree of the polynomial corresponding to the codeword can be higher than $\ell - 1$. On the other hand, if the degree of the polynomial corresponding to the codeword is restricted to be smaller than $\ell$, the encoding becomes unique.

We now describe a simple testing algorithm for checking that the message $x$ encoded by $U$ satisfies a given system of linear equations $Ax = b$, for $A \in \mathbb{F}^{m\ell \times m\ell}$ and $b \in \mathbb{F}^{m\ell}$. (We will always apply this test with a sparse matrix $A$ containing $O(m\ell)$ nonzero entries.) The test simply picks a random linear combination $r \in \mathbb{F}^{m\ell}$ and checks that $(r^T A)x = r^T b$. Note that if $Ax \neq b$, the test will only pass with probability $1/|\mathbb{F}|$ probability. To make the test sublinear, we let the prover provide a polynomial encoding $(r^T A)x$ and check its consistency with $r^T b$ and with $U$ on $t$ randomly chosen symbols. To further simplify the description and analysis of the
testing algorithm, we assume that \( U \) is promised to be \( e \)-close to \( L^m \). Our final IPCP will run TestInterleaved from Section 5.3.1 to ensure that if the promise is violated, this is caught with high probability. The complete test is described in Figure 5.11.

**Oracle:** An alleged \( L^m \)-codeword \( U \) that should encode a message \( x \in \mathbb{F}^{m\ell} \) satisfying \( Ax = b \).

**Interactive testing:**

1. \( V \) picks a random vector \( r \in \mathbb{F}^{m\ell} \) and sends \( r \) to \( P \).
2. \( V \) and \( P \) compute
   
   \[
   r^T A = (r_{11}, \ldots, r_{1\ell}, \ldots, r_{m1}, \ldots, r_{m\ell})
   \]

   and, for \( i \in [m] \), let \( r_i(\cdot) \) be the unique polynomial of degree \( < \ell \) such that \( r_i(\zeta_c) = r_{ic} \) for every \( c \in [\ell] \).

3. \( P \) sends the \( k + \ell - 1 \) coefficients of the polynomial defined by \( q(\cdot) = \sum_{i=1}^m r_i(\cdot) \cdot p_i(\cdot) \), where \( p_i(\cdot) \) is the polynomial of degree \( < k \) corresponding to row \( i \) of \( U \).
4. \( V \) queries a set \( Q \subset [n] \) of \( t \) random symbols \( U[j], j \in Q \).
5. \( V \) accepts if the following conditions hold:
   
   (a) \( \sum_{c \in [\ell]} q(\zeta_c) = \sum_{i \in [m], c \in [\ell]} r_{ic} b_{ic} \).
   (b) For every \( j \in Q \), \( \sum_{i=1}^m r_i(\eta_j) \cdot U_{i,j} = q(\eta_j) \).

Figure 5.3: TestLinearConstraintsIRS(\( \mathbb{F}, L = RS_{F,n,k,m}, m, t, \zeta, A, b; U \))

The following lemma easily follows by inspection.
Lemma 5.20. If \( U \in L^m \), \( U \) encodes \( x \) such that \( Ax = b \), and \( P \) is honest, \( V \) always accepts.

Soundness is argued by the following lemma.

Lemma 5.21. Let \( e \) be a positive integer such that \( e < \frac{d}{2} \). Suppose that a (badly formed) oracle \( U^* \) is \( e \)-close to a codeword \( U \in L^m \) encoding \( x \in \mathbb{F}^{m\ell} \) such that \( Ax \neq b \). Then, for any malicious \( P \) strategy, \( U^* \) is rejected by \( V \) except with at most \( \frac{1}{|\mathbb{F}|} + (\frac{e+k+\ell}{n})^t \) probability.

Proof: Let \( q \) be the polynomial generated in Step 3 following the honest \( P \) strategy on input \( U \). Since we assume that \( Ax \neq b \), it holds that \( \Pr_r[r^T Ax = r^T b] = \frac{1}{|\mathbb{F}|} \). Namely, except with probability \( \frac{1}{|\mathbb{F}|} \) over the choice of \( r \) in Step 1, the polynomial \( q \) fails to satisfy the condition in Step 5a. This is due to the fact that \( \sum_{c \in [\ell]} q(\zeta_c) = (r^T A)x \) and \( \sum_{i \in [m], c \in [\ell]} r_{ic} b_{ic} = r^T b \).

Next, we analyze the probability that a malicious \( P \) strategy is rejected conditioned on \( q \) failing as above. Let \( q' \) be the polynomial sent by the prover. If \( q' = q \), then \( V \) rejects in Step 5a with probability \( \frac{1}{|\mathbb{F}|} \). Otherwise, using the fact that \( q \) and \( q' \) are of degree at most \( k + \ell - 2 \), we have that the number of indices \( j \in [n] \) for which \( q(\eta_j) = q'(\eta_j) \) is at most \( k + \ell - 2 \). Let \( Q' \) be the set of indices on which they agree. Then \( V \) rejects in Step 5b whenever \( Q \) selected in Step 2 contains an index \( i \notin Q' \cup E \), where \( E = \Delta(U^*, L^m) \). This happens with probability at least

\[
1 - \frac{\left(\frac{e+k+\ell-2}{n}\right)}{\left(\frac{e+k+\ell}{n}\right)^t} \geq 1 - \left(\frac{e+k+\ell}{n}\right)^t.
\]
The lemma now follows by a simple union bound.

5.3.3 Testing Quadratic Constraints over Interleaved Reed-Solomon Codes

In this section we describe a simple test for verifying that vectors $x, y, z \in \mathbb{F}^{m\ell}$, respectively encoded by $U^x, U^y, U^z \in L^m$, satisfy the constraints $x \odot y + a \odot z = b$ for some known $a, b \in \mathbb{F}^{m\ell}$, where $\odot$ denotes pointwise product. Letting $L = \text{RS}_{\mathbb{F}, n, k, \eta}$, $U^a = \text{Enc}(a)$, and $U^b = \text{Enc}(b)$, this test reduces to checking that $U^x \odot U^y + U^a \odot U^z - U^b$ encodes the all zeros message $0^{m\ell}$ in the (interleaved extension of) $\hat{L} = \text{RS}_{\mathbb{F}, n, 2k-1, \eta}$. This could be done using the general membership test for interleaved linear codes Test-Interleaved from Section 5.3.1, since the set of codewords in $\hat{L}$ that encodes the all zeros message is a linear subcode of $\hat{L}$. In Figure 5.4 we present this test in a self-contained way, exploiting the promise that $U^x, U^y, U^z$ are close to $L^m$ for a tighter analysis.

The following lemma follows again directly from the description.

**Lemma 5.22.** Let $U = \left[ \begin{array}{c|c|c|c} U^x & | & U^y & | & U^z & | & U^w \end{array} \right]^T$ where $U^w, U^x, U^y, U^z \in L^m$. If $U^x, U^y, U^z$ encode vectors $x, y, z \in \mathbb{F}^{m\ell}$ satisfying $x \odot y + a \odot z = b$ and $P$ is honest, $V$ always accepts.

Soundness is argued by the following lemma.

**Lemma 5.23.** Let $e$ be a positive integer such that $e < \frac{d}{2}$. Let $U^x, U^y, U^z$ be badly-formed oracles and let $U^* \in \mathbb{F}^{3m \times n}$ be the matrix obtained
Oracle: Purported $L^4m$-codeword $U$, where $U = \begin{bmatrix} U^x & U^y & U^z \end{bmatrix}^T$, $U^x, U^y, U^z \in L^m$ and $U^x, U^y, U^z$ that allegedly encode messages $x, y, z \in \mathbb{F}^m$ satisfying $x \odot y + a \odot z = b$.

Interactive testing:

1. Let $U^a = \text{Enc}_\xi(a)$ and $U^b = \text{Enc}_\xi(b)$.
2. $V$ picks a random linear combinations $r \in \mathbb{F}^m$ and sends $r$ to $P$.
3. $P$ sends the $2k - 1$ coefficients of the polynomial $p_0$ defined by
   $$p_0 = \sum_{i=1}^{m} r_i \cdot p_i,$$
   where $p_i = p_i^x \cdot p_i^y + p_i^a \cdot p_i^z - p_i^b$,
   and where $p_i^x, p_i^y, p_i^z$ are the polynomials of degree less than $k$ corresponding to row $i$ of $U^x, U^y, U^z$, and $p_i^a, p_i^b$ are the polynomials of degree less than $\ell$ corresponding to row $i$ of $U^a, U^b$.
4. $V$ picks a random index set $Q \subseteq [n]$ of size $t$, and queries $U[j], j \in Q$.
5. $V$ accepts if the following conditions hold:
   (a) $p_0(\xi_c) = 0$ for every $c \in [\ell]$.
   (b) For every $j \in Q$, it holds that
   $$\sum_{i=1}^{m} r_i \cdot \left[ U^x_{i,j} \cdot U^y_{i,j} + U^a_{i,j} \cdot U^z_{i,j} - U^b_{i,j} \right] = p_0(\eta_j).$$

Figure 5.4: TestQuadraticConstraintsIRS($\mathbb{F}, L = \text{RS}_{\mathbb{F}, n, k, \eta}, m, t, \xi, a, b; U$)

by vertically juxtaposing the corresponding $m \times n$ matrices. Suppose $d(U^*, L^{3m}) \leq e$, and let $(U^x, U^y, U^z)$, respectively, be the (unique) codewords in $(L^m)^3$ that are closest to $(U^x^*, U^y^*, U^z^*)$. Suppose $U^x, U^y, U^z$ encode $x, y, z$ such that $x \odot y + a \odot z \neq b$. Then, for any malicious $P$ strategy,
\((U^x, U^y, U^z)\) is accepted by \(V\) with probability at most
\[
\frac{1}{|F|} + \left( \frac{e + 2k}{n} \right)^t.
\]

**Proof:** Let \(p_0\) be the polynomial generated in Step 3 following the honest \(P\) strategy on \((U^x, U^y, U^z)\). Since \((x, y, z)\) do not satisfy the constraint \(x \odot y + a \odot z = b\), the polynomial \(p_0\) fails to satisfy the condition in Step 5a with probability \(\frac{1}{|F|}\) over the choice of \(r\) in Step 2. We have \(p_0 = \sum_{i=1}^{m} r_i \cdot p_i\) and there must exist an \(i\) and \(\zeta_c\) such that \(p_i(\zeta_c) \neq 0\).

Next, we analyze the probability that a malicious \(P\) strategy is rejected conditioned on \(p_0\) failing as above. Let \(p'_0\) be the polynomial sent by the prover in Step 3. If \(p'_0 = p_0(\cdot)\), then \(V\) rejects in Step 5a with probability \(\frac{1}{|F|}\). Otherwise, using the fact that \(p_0\) and \(p'_0\) are of degree at most \(2k - 2\), we have that the number of indices \(j \in [n]\) for which \(p_0(\eta_j) = p'_0(\eta_j)\) is at most \(2k - 2\). Let \(Q'\) be the set of indices on which \(p_0\) and \(p'_0\) agree. Then \(V\) rejects in Step 5b whenever \(Q\) selected in Step 4 contains an index \(i \notin Q' \cup E\), where \(E = \Delta(U^x, L^3m)\). This fails to happen with probability at most
\[
\frac{(e + 2k - 2)}{n^t} \leq \left( \frac{e + 2k}{n} \right)^t.
\]

The lemma now follows by a union bound. \(\square\)
5.3.4 IPCP for Arithmetic Circuits

In this section, we provide our IPCP for arithmetic circuits. Fix a large finite field \( \mathbb{F} \). Let \( C : \mathbb{F}^{n_i} \to \mathbb{F} \) be an arithmetic circuit. Without loss of generality, we will assume that the circuit contains only ADD and MUL-TIPLY gates with fan-in two. We show how a prover can convince a verifier that \( C(w) = 1 \).

Protocol IPCP\((C, \mathbb{F})\).

- **Input:** The prover \( \mathcal{P} \) and the verifier \( \mathcal{V} \) share as common input the arithmetic circuit \( C : \mathbb{F}^{n_i} \to \mathbb{F} \) and the input statement \( x \). Let \( s \) be the number of gates in the circuit. \( \mathcal{P} \) additionally has input \( \overline{\alpha} = (\alpha_1, \ldots, \alpha_{n_i}) \) such that \( C(\overline{\alpha}) = 1 \).

- **Oracle \( \pi \):** Let \( m \) and \( \ell \) be integers such that \( m \cdot \ell > n_i + s \). Then \( \mathcal{P} \) generates an extended witness \( w \in \mathbb{F}^{m \ell} \) where the first \( n_i + s \) entries of \( w \) are

\[
(\alpha_1, \ldots, \alpha_{n_i}, \beta_1, \ldots, \beta_s)
\]

where \( \beta_i \) is the output of the \( i^{th} \) gate when evaluating \( C(\overline{\alpha}) \). \( \mathcal{P} \) defines a system of constraints that contains the following constraint for every multiplication gate \( g \) in the circuit \( C \):

\[
\beta_a \cdot \beta_b - \beta_c = 0.
\]
For every addition gate $g$, there is a constraint

$$\beta_a + \beta_b - \beta_c = 0$$

where $\beta_a$ and $\beta_b$ are the input values to $g$ and $\beta_c$ is the output value of $g$ in the extended witness. For the output gate there is a constraint $\beta_a + \beta_b - 1 = 1$ if the final gate is an addition gate, and $\beta_a \cdot \beta_b - 1 = 0$ if it is a multiplication gate. $P$ constructs vectors $x, y$ and $z$ in $\mathbb{F}^{m\ell}$ where the $j^{th}$ entry of $x, y$ and $z$ contains the values $\beta_a, \beta_b, \beta_c$ corresponding to the $j^{th}$ multiplication gate in $w$. $P$ and $V$ construct matrices $P_x, P_y$ and $P_z$ in $\mathbb{F}^{m\ell \times m\ell}$ such that $x = P_xw$, $y = P_yw$, and $z = P_zw$.

Finally, $P$ constructs matrix $P_{\text{add}} \in \mathbb{F}^{m\ell \times m\ell}$ such that the $j^{th}$ position of $P_{\text{add}}w$ equals $\beta_a + \beta_b - \beta_c$ where $\beta_a, \beta_b, \beta_c$ correspond to the $j^{th}$ addition gate of the circuit in $w$. Let $(U^w, U^x, U^y, U^z) \in (L^m)^4$ encode $(w, x, y, z)$ where $L = \text{RS}_{F, n, k, \eta}$. $P$ sets the oracle $\pi$ to $U \in L^{4m}$ which is set as the vertical juxtaposition of the matrices $(U^w, U^x, U^y, U^z) \in (L^m)^4$.

All the linear constraints can be expressed as one large linear con-
Chapter 5. Ligeró

Constraint matrix:

\[ A = \begin{bmatrix} I_{3m\ell \times 3m\ell} \quad -P \\ 0_{3m\ell \times 3m\ell} \quad P_{\text{add}} \end{bmatrix}, \quad P = \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix}, \quad b = 0^{4m\ell}. \]

- The Interactive Protocol:

\( V \) and \( P \) run the following tests.

1. TestInterleaved(\( F, L, 4m, t; U \))
2. TestLinearConstraintsIRS(\( F, L, 4m, t, \zeta, A, b; U \))
3. TestQuadraticConstraintsIRS(\( F, L, m, t, \zeta, (\mathbf{1})^{m\ell}, 0^{m\ell}; U \))

Since all the tests open the same number of columns \( t \) in \( U_w, U_x, U_y, U_z \), \( V \) will simply open \( t \) columns of \( U \). \( V \) rejects if it rejects in any of the tests above.

The completeness of our IPCP follows from the following lemma.

**Lemma 5.24.** If \( P \) is honest and \( (U^w, U^x, U^y, U^z) \in (L^m)^4 \) encode vectors \( (w, x, y, z) \in (F^{m\ell})^4 \) satisfying

- \( x = P_xw \),
- \( y = P_yw \),
- \( z = P_zw \),
• $x \odot y + (-1)^{m\ell} \odot z = 0^{m\ell}$, and

• $P_{\text{add}} w = 0^{m\ell}$

then $V$ always accepts.

The proof follows directly from Lemma 5.13, Lemma 5.20, and Lemma 5.22. Next, soundness is argued by the following lemma.

**Lemma 5.25.** Let $e$ be a positive integer such that $e < \frac{d}{3}$ and suppose that there does not exist any $\bar{\alpha}$ such that $C(\bar{\alpha}) = 1$. Then, for any maliciously-formed oracle $U^*$ and any malicious prover strategy, the verifier accepts except with probability at most $\frac{d+2}{|F|} + (1 - \frac{e}{n})^t + 2(\frac{e+2k}{n})^t$.

**Proof:** At a high level, soundness will essentially follow from the soundness of the individual tests and the overall soundness error will follow by a direct application of a union bound over the soundness of these tests. In more detail, let $U$ be the vertical juxtaposition of $(U^w, U^x, U^y, U^z)$. Then we argue soundness by considering the following cases and applying a union bound:

**Case** $d(U, L_4^m) > e$: Since $e < \frac{d}{3}$, we can conclude from Theorem 5.19 that the verifier accepts in TestInterleaved executed in Step 1 with probability $(1 - \frac{e}{n})^t + \frac{d}{|F|}$.

**Case** $d(U, L_4^m) \leq e$: Next, let $(U^w, U^x, U^y, U^z) \in (L^m)^4$ be the codes that are close to $(U^{w*}, U^{x*}, U^{y*}, U^{z*})$ and encode the messages
(w, x, y, z). Recall that there exist no w, x, y, z that satisfy all the following constraints:

- x = P_xw,
- y = P_yw,
- z = P_zw,
- \(x \odot y + (-1)^{m\ell} \odot z = 0^{m\ell}\), and
- \(P_{\text{add}}w = 0^{m\ell}\).

We can conclude from Lemma 5.21 and Lemma 5.23 by applying a union bound on the corresponding tests that the verifier rejects except with probability

\[
\frac{2}{|F|} + \left(\frac{e + k + \ell}{n}\right)^t + \left(\frac{e + 2k}{n}\right)^t < 2 \cdot \left(\frac{1}{|F|} + \left(\frac{e + 2k}{n}\right)^t\right).
\]

The following theorem follows from the construction described above and the preceding lemmas.

**Theorem 5.26.** Fix parameters \(n, m, \ell, k, t, e\) such that \(e < \frac{n-k}{4}\). Let \(C : \mathbb{F}^{n_i} \to \mathbb{F}\) be an arithmetic circuit of size \(s\), where \(|F| \geq n\) and \(m \cdot \ell > n_i + s\). Then protocol IPCP(C, F) satisfies the following:
• Completeness: If $C(\overline{a}) = 1$ and oracle $\pi$ is generated honestly as described in the protocol, then

$$\Pr[\text{out}_2(\langle P(w), V^\pi \rangle(C)) = 1] = 1.$$ 

• Soundness: If there is no $\overline{a}$ such that $C(\overline{a}) = 1$, then for every (unbounded) prover strategy $P^*$ and every $\tilde{\pi} \in \mathbb{F}_4^{4mn}$,

$$\Pr[\text{out}_2(\langle P^*, V^{\tilde{\pi}} \rangle(C)) = 1] \leq \frac{d + 2}{|\mathbb{F}|} + \left(1 - \frac{e}{n}\right)^t + 2 \left(\frac{e + 2k}{n}\right)^t.$$ 

• Complexity: The number of field operations performed is $\text{poly}(|C|, n)$. The number of field elements communicated by $P$ to $V$ is $k + (k + \ell - 1) + (2 \cdot k - 1)$ whereas $V$ reads $t$ symbols from $\mathbb{F}_4^{4m}$.

The first term in the communication cost is the communication incurred by the test-interleaved protocol. The second term is due to the linear constraint test. The final term results from our quadratic constraint test.

### 5.3.5 IPCP for Boolean Circuits

In order to obtain the benefits in soundness from running our IPCP over a large field $\mathbb{F}$, we show how we can prove the validity of a boolean circuit $C : \{0, 1\}^{n_i} \rightarrow \{0, 1\}$ by encoding the witness in any larger field $\mathbb{F}$. First, the prover will map the boolean $0$ within the witness to the additive
identity $\epsilon_0$ in $\mathbb{F}$, and the boolean 1 to the multiplicative identity $\epsilon_1$ in $\mathbb{F}$. We enforce that this is done by introducing a linear constraint. We can enforce that each element in the witness is a 0 or 1 by introducing a quadratic constraint $\beta^2 - \beta = 0$.

Next, given that binary constraints are already enforced, we proceed to demonstrating how we incorporate the constraints based on the XOR and ADD gates. In fact, we will show that all gate constraints can be expressed as a linear relation on the witness bits. Let $x$ be a column vector consisting of the witness string. We will construct a matrix $A$ and a column vector $w$ such that if $w$ is a binary valid witness then the elements of $Aw$ will all be 0, and if $w$ is binary and is not a valid witness then at least one element of $Aw$ will be nonzero. For each XOR and AND gate in the circuit we will create a row in the matrix corresponding to the enforcement of that relation in the witness. Specifically, besides including the input bits $x$, the vector $w$ will include one additional bit for each XOR and AND gate. We explain the purpose of these extra bits next.

Given integers $b_1$ and $b_2$ consider the arithmetic constraint $b_1 + b_2 = r_0 + 2 \cdot r_1$ over the integers. In this constraint, if we enforce that all values are bits then $r_0$ is the XOR of $b_1$ and $b_2$ and $r_1$ is the AND of $b_1$ and $b_2$. In order to make sure that $b_1$ XOR $b_2$ equals $b_3$ in $w$, we require the prover to include in the witness an auxiliary bit $d$ and enforce the linear constraint $b_1 + b_2 = b_3 + 2 \cdot d$, as well as the binary constraints that $b_3$ and $d$ are bits. Analogously, to ensure $b_1$ AND $b_2$ equals $b_3$, we include an auxiliary
bit $d$ and enforce the linear constraint $b_1 + b_2 = d + 2 \cdot b_3$ and the binary constraint that $b_3$ and $d$ are bits. To conclude, we observe that if the values have been enforced to be a binary constraint then checking the arithmetic constraints over integers can be done by checking the equation modulo a sufficiently large prime ($p > 3$).

We can extend this idea to consider more complex gates such as addition modulo $2^{32}$ over 32-bit inputs and outputs. This can also be expressed as a linear constraint over the bits. Suppose $a = (a_0, \ldots, a_{31})$, $b = (b_0, \ldots, b_{31})$ and $c = (c_0, \ldots, c_{31})$ are the input and output bits, the constraint $a + b = c \mod 2^{32}$ can be expressed as

$$\sum_{i=0}^{31} 2^i \cdot a_i + \sum_{i=0}^{31} 2^i \cdot b_i = 2^{32} \cdot d + \sum_{i=0}^{31} 2^i \cdot c_i$$

where $d$ is an auxiliary input bit, and all values are enforced to be binary. However, this will require using a finite field $\mathbb{F}$ with characteristic $p > 2^{33}$.

### 5.3.6 Achieving Zero Knowledge

Note first that the verifier obtains two types of information in two different building blocks of the IPCP. First, it obtains linear combinations of codewords in a linear code $L$. Second, it probes a small number of symbols from each codeword. Since codewords are used to encode the NP witness, both types of information give the verifier partial information about said NP witness, so the IPCC we described is not zero-knowledge.
in its current form. Fortunately, ensuring zero knowledge requires introducing only small modifications to the construction and analysis. Specifically, the second type of “local” information about the codewords is made harmless by making the encoding randomized, so that probing just a few symbols in each codeword reveals no information about the encoded message. The high-level idea for making the first type of information harmless is to use an additional random codeword for blinding the linear combination of codewords revealed to the verifier. However, this needs to be done in a way that does not compromise soundness. Below we describe the modifications required for each of the IPCP ingredients.

**Zero-Knowledge Testing of Interleaved Linear Codes**

Recall that in the verification algorithm Test-Interleaved from Section 5.3.1, Ψ obtains a linear combination of the form \( w = r^T U \), where \( U \in \mathbb{F}^{m \times n} \) is a matrix whose rows should be codewords in \( L \). A natural approach for making this linear combination hide \( U \) is by allowing the prover to add to the rows of \( U \) an additional random codeword \( u' \) that is used for blinding. We next describe a simple implementation of this idea that provides a slightly inferior soundness guarantee. Apply the algorithm TestInterleaved to \( L^{m+1} \), with an extended oracle \( U' \) such that first \( m \) rows of \( U' \) contain the rows of \( U \) and the last row of \( U' \) is \( u' \). Let \( w' = r^T U + r'u' \) be the random linear combination obtained by Ψ. The test fails to be zero knowledge when \( r' = 0 \), which occurs with proba-
Alternatively, settling for a slightly-worse soundness guarantee (where \( e_{\mathbb{F}} \) is replaced by \( e_{\mathbb{F} - 1} \)), one could just let \( r' \) be a random nonexistent field element, and get perfect zero knowledge. It turns out, however, that one could fix \( r' \) to 1 and still get the same soundness guarantee about \( U \) as in Lemma 5.14, since we can apply the same decomposition argument. We describe this “affine” variant of TestInterleaved is described and analyzed in Section 5.7.3.

**Zero-Knowledge Testing of Linear Constraints over Interleaved Reed-Solomon Codes**

The verification algorithm for the linear constraints \( Ax = b \) samples a random vector \( r \), obtains \( r^T Ax \), and compares it with \( r^T b \). Looking more carefully at our actual protocol, the verifier obtains a polynomial \( q \) and checks whether \( \sum_{c \in [\ell]} q(\zeta_c) = \sum_{i \in [m], c \in [\ell]} r_{ic} b_{ic} \). While the sum itself \((r^T b)\) does not reveal any additional information beyond what is already known, the individual evaluations of \( q \), i.e. \( q(\zeta_c) \), may reveal information about the inputs. One possible idea of how to hide this information is for \( P \) to provide an additional vector \( u' \) along with \( U \) that encodes a message \((\gamma_1, \ldots, \gamma_\ell)\) such that \( \sum_{c \in [\ell]} \gamma_c = 0 \), and append to \( A \) the constraints that sum the entries in the message encoded in \( u' \).

However, as before, this will yield a suboptimal soundness guarantee. Instead, we consider the following approach, which provides the same soundness guarantee as the original nonaffine version of the test. We ap-
ply the algorithm TestLinearConstraintsIRS to $L^{m+1}$, where $L = RS_{f,n,k,\eta}$, with an extended oracle $U'$ which has the $m$ rows of $U$ as its first $m$ rows and has a last row of $u'$, where additionally $u'$ encodes a message $(\gamma_1, \ldots, \gamma_\ell)$ such that $\sum_{c \in [\ell]} \gamma_c = 0$. Let $r_{\text{blind}}$ be a polynomial of degree at most $k + \ell - 1$ corresponding to $u'$. Let $q = \sum_{i=1}^{m} r_i \cdot p_i + r_{\text{blind}}$ be the polynomial obtained by $V$. We can show that the soundness of the resulting scheme will be the same as in Lemma 5.21. We describe and analyze this “affine” variant of TestLinearConstraints in Section 5.7.4.

**Zero-Knowledge Testing of Quadratic Constraints over Interleaved Reed-Solomon Codes**

Next we modify the quadratic constraint testing procedure in the same way as we modified the linear constraint testing. Let $u'$ encode a message $0^\ell$. Let the extended oracle $U'$ contain $U^x, U^y, U^z$ as its first $m$ rows and have $u'$ as its last row. We apply the algorithm TestQuadraticConstraints to $L^{3m+1}$. Let $r_{\text{blind}}$ be a polynomial of degree at most $2k - 1$ corresponding to $u'$. Let $p_0 = \sum_{i=1}^{m} r_i \cdot p_i + r_{\text{blind}}$ be the polynomial obtained by $V$. We can show that the soundness of the resulting scheme will be the same as in Lemma 5.23. We describe and analyze this “affine” variant of TestQuadraticConstraints in Section 5.7.5.
5.3.7  The Final ZKIPCP

In this section provide a self-contained description of the final ZKIPCP protocol, combining all of the previous subprotocols. In this section, we provide our ZKIPCP for arithmetic circuits over a large field $\mathbb{F}$. At a high level, the protocol is essentially the IPCP construction from Section 5.3.4 with the exception that we replace all the tests with the generalized affine version (with repetitions).

**Protocol ZKIPCP** $(C, \mathbb{F})$

- **Input:** The prover $P$ and the verifier $V$ share as common input an arithmetic circuit $C : \mathbb{F}^{n_i} \rightarrow \mathbb{F}$ and input statement $x$. $P$ has a private input of the form $\bar{\alpha} = (\alpha_1, \ldots, \alpha_{n_i})$ such that $C(\bar{\alpha}) = 1$. Let $s$ be the number of gates in $C$.

- **Oracle $\pi$:** Let $m, \ell$ be integers such that $m \cdot \ell > n_i + s$. For $i \in [n_i]$ let $\beta_i$ be the output of the $i^{th}$ gate when evaluating $C(\bar{\alpha})$. $P$ generates an extended witness $w \in \mathbb{F}^{m\ell}$ where the first $n_i + s$ entries of $w$ are $(\alpha_1, \ldots, \alpha_{n_i}, \beta_1, \ldots, \beta_s)$. $P$ constructs vectors $(x, y, z) \in (\mathbb{F}^{m\ell})^3$ such that the $j^{th}$ entry of $x, y,$ and $z$ contains the values $\beta_a, \beta_b,$ and $\beta_c$ corresponding to the $j^{th}$ multiplication gate in $w$. $P$ and $V$ construct matrices $(P_x, P_y, P_z) \in (\mathbb{F}^{m\ell \times m\ell})^3$ such that $x = P_xw$, $y = P_yw$, and $z = P_zw$.

Finally, $P$ constructs matrix $P_{\text{add}} \in \mathbb{F}^{m\ell \times m\ell}$ such that the $j^{th}$ row of $P_{\text{add}}w$ equals $\beta_a + \beta_b - \beta_c$ where $\beta_a, \beta_b,$ and $\beta_c$ correspond to the
$j^{th}$ addition gate of the circuit in $w$. The linear constraints can be summarized as one large matrix as before:

$$A = \begin{bmatrix} I_{3m\ell \times 3m\ell} & -P \\ 0_{m\ell \times 3m\ell} & P_{\text{add}} \end{bmatrix}, P = \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix}, b = 0^{4m\ell}.$$

Let $\zeta = (\zeta_1, \ldots, \zeta_\ell)$ be a sequence of distinct elements disjoint from $(\eta_1, \ldots, \eta_n)$. Let $L = \text{RS}_{\mathbb{F}_{\ell n}, k, \eta}$. $P$ samples random codewords $(U^w, U^x, U^y, U^z) \in (L^m)^4$ subject to $w = \text{Dec}_\zeta(U^w), x = \text{Dec}_\zeta(U^x), y = \text{Dec}_\zeta(U^y), \text{ and } z = \text{Dec}_\zeta(U^z)$. Let $u^0_h, u^\text{add}_h$ be auxiliary rows sampled randomly from $L$ for every $h \in [\sigma]$ where each of $u^\text{add}_h$ encodes an independently-sampled random $\ell$ messages $(\gamma_1, \ldots, \gamma_\ell)$ subject to $\sum_{c \in [\ell]} \gamma_c = 0$ and $u^0_h$ encodes $0^\ell$. Let $U \in L^{4m}$ be the vertical juxtaposition of the matrices $U^w, U^x, U^y,$ and $U^z$. $P$ sets the oracle to $U$.

- **The Interactive Protocol:**

1. For every $h \in [\sigma]$, $V$ picks the random elements $r_h \in \mathbb{F}^{4m}, r^\text{add}_h \in \mathbb{F}^{4m\ell}$ and $r^q_h \in \mathbb{F}^{m}$ and sends them to $P$.
2. For every $h \in [\sigma]$, $P$ responds with
   - (Interleaved Reed-Solomon Testing)
   $$v_h = (r_h)^T U + u'_h \in \mathbb{F}^n,$$
– (Linear Constraints Testing) polynomial $q_{h}^{\text{add}}$ of degree at most $k + \ell - 1$ where

$$q_{h}^{\text{add}} = r_{\text{blind},h}^{\text{add}} + \sum_{i=1}^{m} r_{h,i}^{\text{add}} \cdot p_{i},$$

such that

* $p_{i}$ is the polynomial of degree less than $k$ corresponding to row $i$ of $U^{w}$,
* $r_{h,i}^{\text{add}}$ is the unique polynomial of degree less than $\ell$ such that for every $c \in [\ell]$ it holds that $r_{h,i}^{\text{add}}(\zeta_{c}) = \left((r_{h}^{\text{add}})^{T} P\right)_{i,c}$, and
* $r_{\text{blind},h}^{\text{add}}$ is the polynomial of degree less than $k + \ell - 1$ corresponding to $u_{h}^{\text{add}}$.

– (Quadratic Constraints Testing)

$$p_{0,h} = r_{\text{blind},h}^{0} + \sum_{i=1}^{m} \left[ (r_{h}^{q})_{i} \cdot (p_{i}^{x} \cdot p_{i}^{y} - p_{i}^{z}) \right]$$

where for $a \in \{x,y,z\}$, it holds that

* $p_{i}^{a}$ is the polynomial of degree less than $k$ corresponding to row $i$ of $U^{a}$,
* $r_{\text{blind},h}^{0}$ is the polynomial of degree less than $2k - 1$ corresponding to $u_{h}^{0}$.

3. $V$ picks a random index set $Q \subseteq [n]$ of size $t$. For all $j \in$
Q, V queries $U[j]$ where $U[j]$ is the vertical juxtaposition of $U^w_h[j], U^x_h[j], U^y_h[j], U^z_h[j], u^\text{add}_h[j], u'_h[j]$, and accepts if and only if the following conditions hold for every $h \in [\sigma]$:

- for every $j \in Q$,
  
  $$\sum_{i=1}^{4m} r_h[j] \cdot U_{i,j} + u'_h[j] = v_h[j],$$

- $\sum_{c \in [\ell]} q^\text{add}_h (\zeta_c) = 0$,

- for every $j \in Q$,
  
  $$u^\text{add}_h[j] + \sum_{i=1}^{4m} r^\text{add}_{h,i}(\eta_j) \cdot U_{i,j} = q^\text{add}_h(\eta_j),$$

- for every $c \in [\ell]$, $p_{0,h}(\zeta_c) = 0$

- for every $j \in Q$,

  $$u'_h[j] + \sum_{i=1}^m (r^0_h)_i \cdot \left[U^x_{i,j} \cdot U^y_{i,j} - U^z_{i,j}\right] = p_{0,h}(\eta_j).$$

The completeness of our ZKIPCP follows from the next lemma.

**Lemma 5.27.** If $P$ is honest and $(U^w, U^x, U^y, U^z) \in (L^m)^4$ encode vectors $(w, x, y, z) \in (\mathbb{F}_m^\ell)^4$ satisfying

- $x = P_x w$,
- $y = P_y w$,
- $z = P_z w$, 

• \( x \circ y + (-1)^{m\ell} \circ z = 0^{m\ell} \), and

• \( P_{\text{add}}w = 0^{m\ell} \)

then \( \mathcal{V} \) always accepts.

We argue soundness with the following lemma.

**Lemma 5.28.** Let \( e \) be a positive integer such that \( e < \frac{d}{4} \). Suppose that there exist no \( \overline{\pi} \) such that \( C(\overline{\pi}) = 1 \). Then, for any maliciously formed oracle \( U^* \) and any malicious prover strategy, the verifier accepts with probability at most \( \frac{d+2}{|F|} + (1 - \frac{e}{n})^t + 2 \left( \frac{e+2k}{n} \right)^t \).

The proofs of the preceding two lemmas follow analogously to the proofs of Lemma 5.24 and Lemma 5.25. The next lemma establishes the honest verifier zero-knowledge property.

**Lemma 5.29.** If \( k > \ell + t \), \((\mathcal{P}, \mathcal{V})\) is an (honest verifier, perfect) zero-knowledge IPCP.

**Proof:** To demonstrate zero knowledge against the honest verifier, we need to provide a simulator \( S \) that can generate a transcript given the randomness of the honest verifier \( \mathcal{V} \). For every \( h \in [\sigma] \), \( S \) first generates:

- a random polynomial \( q_{h}^{\text{add}} \) of degree less than \( k + \ell - 1 \) such that
  \[ \sum_{c \in [\ell]} q_{h}^{\text{add}}(\zeta_c) = 0. \]

- a random polynomial \( p_{0,h} \) of degree less than \( 2k - 1 \) such that for every \( c \in [\ell] \) it holds that \( p_{0,h}(\zeta_c) = 0. \)
• a random vector \( v_h \in \mathbb{F}^n \).

Next for every \( j \in Q \), \( S \) samples random elements from \( \mathbb{F} \) for \( U_h^x[j] \), \( U_h^y[j] \), \( U_h^z[j] \), and \( U_h^w[j] \). Finally, given the random challenges from \( \mathcal{V} \), for every \( j \in Q \), \( S \) sets \( u'_h[j] \), \( u_h^{\text{add}}[j] \), and \( u_0^h[j] \) as follows:

- \( u'_h[j] = \sum_{i=1}^{4m} (r_h[j] \cdot U_{i,j} - v_h[j]), \)
- \( u_h^{\text{add}}[j] = \sum_{i=1}^{4m} (r_{h,i}^{\text{add}}(\eta_j) \cdot U_{i,j} - q_h^{\text{add}}(\eta_j)), \)
- \( u_0^h[j] = \sum_{i=1}^{m} \left( (r_q^h)_i \cdot \left[ U_{i,j}^x \cdot U_{i,j}^y - U_{i,j}^z \right] - p_{0,h}(\eta_j) \right). \)

Our simulation achieves perfect zero knowledge. This follows from the fact that in an interaction with the honest prover \( P \), the distribution of \( \left( \mathcal{U}_h^x[j], \mathcal{U}_h^y[j], \mathcal{U}_h^z[j], \mathcal{U}_h^w[j] \right)_{j \in Q} \) is uniform and given that \( u'_h \), \( u_h^{\text{add}} \), \( u_h^x \), and \( u_0^h \) are uniformly chosen, the polynomials \( q_h^{\text{add}} \), and \( p_{0,h} \) and the vector \( v_h \) are uniformly distributed in their respective spaces.

The following theorem follows from the construction described above and the preceding lemmas.

**Theorem 5.30.** Fix parameters \( n, m, \ell, k, t, e \) such that \( e < \frac{n-k}{4} \). Let \( C : \mathbb{F}^{n_i} \rightarrow \mathbb{F} \) be an arithmetic circuit of size \( s \), where \( |\mathbb{F}| \geq \ell + n, m \cdot \ell > n_i + s \), and \( k > \ell + t \). Then protocol ZKIPCP(\( C, \mathbb{F} \)) satisfies the following:

- **Completeness:** If \( \bar{\alpha} \) is such that \( C(\bar{\alpha}) = 1 \) and oracle \( \pi \) is generated
honestly as described in the protocol, then

\[ \Pr[\text{out}_2(\langle P(\bar{\alpha}), V^{\bar{\pi}} \rangle(C)) = 1] = 1. \]

- **Soundness**: If there is no \( \bar{\pi} \) is such that \( C(\bar{\pi}) = 1 \), then for every (unbounded) prover strategy \( P^* \) and every \( \bar{\pi} \in F^{4mn} \),

\[ \Pr[\text{out}_2(\langle P^*, V^{\bar{\pi}} \rangle(C)) = 1] \leq \frac{d + 2}{|F|^\sigma} + \left(1 - \frac{e}{n}\right)^t + 2 \left(\frac{e + 2k}{n}\right)^t. \]

- **Zero Knowledge**: For every verifier \( V^* \), there exists a simulator \( S \) such that the distribution of the output of \( S^{V^*}(C) \) is distributed identically to the distribution \( \text{view}_2(\langle P(\bar{\alpha}), V^{s\bar{\pi}} \rangle(C)) \).

- **Complexity**: The number of field \( F \) operations performed is \( \text{poly}(|C|, n) \). The number of field elements communicated by \( P \) to \( V \) is \( \sigma \cdot n + \sigma \cdot (k + \ell - 1) + \sigma \cdot (2 \cdot k - 1) \). The number of symbols \( V \) reads symbols from \( F^{4m + 5\sigma} \) is \( t \).

## 5.4 From ZKIPCP to ZK

In this section we describe variants of known transformations from (sublinear) zero-knowledge PCP to (sublinear) zero-knowledge arguments. The latter can either be interactive using collision-resistant hash functions, or noninteractive in the random oracle model.
5.4.1 The Interactive Variant

General transformations from (noninteractive) ZKPCP to (interactive) zero-knowledge arguments that make a black-box use of collision-resistant hash functions were given in [IMS12, IW14]. Here we address the more general case of ZKIPCP, where in addition to the proof oracle there is interaction between the prover and the verifier. Using the ZKIPCP, we now describe an honest-verifier zero-knowledge protocol. The prover commits to each entry of the proof oracle using a statistically-hiding commitment scheme then compresses the commitment using a Merkle hash tree (see Section 5.1.1). Note that both steps can be realized by making black-box use of any family $\mathcal{H}$ of collision-resistant hash functions. The rest of the zero-knowledge protocol mimics the ZKIPCP, where the prover opens the committed values that correspond to the verifier’s queries. Malicious verifiers can be handled using standard techniques, as shown in [IMS12, IW14].

The communication complexity of the zero-knowledge argument includes the communication complexity of the ZKIPCP protocol and communication resulting from committing the oracle $\pi$ and decommitting to the queries $Q$. 
5.4.2 The Noninteractive Variant

By following the Fiat-Shamir transform [FS86] it is possible to directly compile our previous protocol into a noninteractive protocol using a random oracle, where the verifier’s messages are emulated by applying the random oracle on the partial transcript in each round. A formal description and analysis of this transformation is presented in [BCS16] for the interactive oracle proofs (IOP) model, which generalizes public-coin IPCP.

In slightly more detail, in this transformation the prover uses the random oracle to generate the verifier’s messages and complete the execution (computing its own messages) based on the emulated verifier’s messages, where instead of using an oracle, the prover commits to its proof and messages using Merkle hash trees. Completeness follows directly. If we start with an IOP that is zero-knowledge (ZKIPCP in our case), [BCS16] show that this transformation preserves the (statistical) zero-knowledge property. Namely, the resulting protocol can be proven to be zero knowledge in the random oracle model.

In [BCS16], the soundness of the transformed protocol is shown to essentially match the soundness of the original protocol up to an additive term that roughly depends on the product of $q^2$ and $2^{-\lambda}$ where $q$ is an upper bound on the number of queries made to the random oracle by a malicious prover and $\lambda$ is the output length of the random oracle. More precisely, [BCS16] relate the soundness of the transformed protocol to the state restoration soundness of the underlying IPCP and the colli-
sion probability of queries to the random oracle. State-restoration soundness refers to the soundness of the IOP protocol against cheating prover strategies that may rewind the verifier back to any previously seen state, where every new continuation from a state invokes the “next message” function of the verifier with fresh randomness. In [BCS16], they show that for any (IOP) the state-restoration soundness of an IOP protocol is bounded by \((\frac{T}{k(x)}) \cdot \epsilon(x)\) and the soundness of the transformed protocol is \((\frac{T}{k(x)}) \cdot \epsilon(x) + O(T^2 \cdot 2^{-\lambda})\) where \(T\) is an upper bound on the number of queries made by cheating provers to the random oracle, \(k(x)\) is the round complexity of the IOP, and \(\epsilon(x)\) is the (standard) soundness of the IOP.

Next, we tighten the analysis presented in [BCS16] for the particular ZKIPCP constructed in Section 5.3.7 and show that the soundness of the transformed protocol is \(T \cdot \epsilon(x) + O(T^2 \cdot 2^{-\lambda})\) where \(\epsilon(x)\) is the soundness of the ZKIPCP, \(T\) bounds the number of queries made by cheating prover to the random oracle and \(\lambda\) is the output length of the random oracle.

In [CCH+19], Canetti et al. show that if we have a public-coin honest-verifier interactive zero-knowledge argument with round-by-round soundness \(\epsilon\), then the Fiat-Shamir transformation yields a non-interactive zero-knowledge argument system with soundness \(\epsilon\).

We repeat (verbatim) the definition of round-by-round soundness and the lemma from [CCH+19] for completeness.

**Definition 5.31.** A 2\(r\)-round protocol \(\Pi\) has round-by-round soundness error \(\epsilon(\cdot)\) if and only if there exists a (possibly inefficient) mapping State
from the tuple \((x, \tau)\) where \(x\) is the instance and \(\tau\) a partial transcript of interaction using \(\Pi\) to \{accept, reject\} such that the following hold:

1. If \(x \not\in L\), then \(\text{State}(x, \emptyset) = \text{reject}\), where \(\emptyset\) denotes the empty transcript.

2. If \(\text{State}(x, \tau) = \text{reject}\) for a partial transcript up to \(2i\)-rounds, then for every prover message \(\alpha\), it holds that

\[
\Pr[\beta \leftarrow V(x, (\tau, \alpha)) : \text{State}(x, (\tau, \alpha, \beta)) = \text{accept}] \leq \epsilon(|x|, |\tau|).
\]

3. For any full transcript \(\tau\), if \(\text{State}(x, \tau) = \text{reject}\), then \(V(x, \tau) = 0\).

First, we compile our zero-knowledge interactive probabilistically-checkable proof to an interactive zero-knowledge argument as described above where we instantiate the collision-resistant hash function with the random oracle. Suppose the prover makes at most \(T\) queries. The probability it finds a collision is bounded by \(T^22^{-\lambda}\). Next, we analyze the round-by-round soundness of the compiled interactive zero-knowledge argument. By our preceeding analysis, \(\epsilon(|x|, |\tau|)\) where the \(\tau\) is the partial transcript at the end of the first round is \(\frac{d+2}{|F|^s}\) and at the end of the third round is \((1 - \frac{\epsilon}{n})^t + 2\left(\frac{e+2k}{n}\right)^t\). Finally, we apply the Fiat-Shamir transformation, where the prover generates the verifier’s message by applying the random oracle on the partial transcript to obtain the randomness for the verifier. To argue the soundness of the transformed protocol, we observe
that the adversary succeeds only if $\text{State}(x, (\tau, \alpha, \beta)) = \text{accept}$. Suppose that the adversary makes $T_1$ queries with $\tau$ as a partial transcript at the end of the first round and $T_2$ queries for partial transcripts at the end of the third round where $T_1 + T_2 \leq T$. Then the probability the adversary succeeds is bounded by

$$T_1 \cdot \frac{d + 2}{|F|^\sigma} + T_2 \cdot \left( (1 - \frac{e}{n})^t + 2\left(\frac{e + 2k}{n}\right)^t \right) + T^2 2^{-\lambda} \leq T \cdot \left( \frac{d + 2}{|F|^\sigma} + (1 - \frac{e}{n})^t + 2\left(\frac{e + 2k}{n}\right)^t \right) + T^2 2^{-\lambda}.$$

### 5.4.3 Sublinear Zero-Knowledge Argument

In this section, we describe how to set the parameters of our zero-knowledge argument to obtain communication that is sublinear in the circuit size. We consider first an arithmetic circuit over a large field $F$. Let $h$ be the output length of the collision-resistant hash function. Following our transformation, the communication complexity of the zero-knowledge protocol that is compiled based on our ZKIPCP is

$$\left[ k \cdot \sigma \right] + \left[ (k + \ell - 1) \cdot \sigma \right] + \left[ 2 \cdot k - 1 \right] \cdot [1g |F|] + t \cdot \left[ \log n \cdot h \right].$$
For a security parameter $\kappa$, when $|F|$ is large (i.e. $|F| > O(2^{\kappa})$) we can set $\sigma = 1$ for $2^{-\kappa}$-security. The other terms in the soundness are $\left(1 - \frac{e}{n}\right)^t$ and $\left(\frac{e+2k}{n}\right)^t$ where $e < \frac{d}{4} = \frac{n-k}{4}$ and $k = t + \ell$. We can optimize our parameters by setting $k = O(\kappa)$, $n = O(k)$ and $e = \frac{n-k}{4}$ to achieve $2^{-\kappa}$-security.

The next constraint on the parameters requires that $m \cdot \ell$ is at least as large as the witness size, which is $O(s)$ where $s$ is the number of gates in the circuit. Optimal values for $m$ and $\ell$ can be obtained by equating the dominating costs in the communication, namely, $O(\ell)$ and $O(t \cdot m)$ which implies $\ell = O\left(\sqrt{s\kappa}\right)$ and $m = O\left(\sqrt{\frac{s}{\kappa}}\right)$. Overall the communication complexity with these parameters is $O\left(\sqrt{s\kappa|\log|F|}|\right)$.

Optimizing for boolean circuits requires additional effort. In this case there is yet another degree of freedom, namely the field size. For performance reasons we wish to select the smallest finite field that fulfills our requirements. Naturally, we require that the finite field be small enough that we can compute arithmetic in that field within polynomial time. There certainly must also be at least $\ell + n$ evaluation points, but this is not the limiting factor on the field size for realistic values of the security parameter. The soundness requirement is the most interesting constraint on the field size. For a given statistical security parameter $\kappa$, we can show that for sufficiently-large circuits, the optimal field size for achieving soundness error within that bound is $|F| = O(\sqrt{s})$, and that this achieves communication complexity $O(k \cdot \sqrt{s\log s})$. For smaller circuits, the optimal value requires a more careful analysis as the low order terms are significant and
we present our results in the next section. Before this, we briefly discuss other constraints on the finite field.

Depending on implementation optimizations, there are other requirements on $F$. Instead of enforcing AND and XOR gates in the “direct” way we may instead enforce only that wire values are binary and that there is a certain linear relation between wires and the XOR of their values and the AND of their values. If $X, Y, A,$ and $B$ are binary and $X + Y = A + 2B$ then $A$ is the XOR of $X$ and $Y$ and $B$ is the AND of $X$ and $Y$. If we enforce these constraints, we will automatically enforce that that part of the circuit be evaluated correctly. If we use this optimization, then in order to prevent “overflow” we must use a finite field with characteristic strictly greater than three.

Suppose we are constructing proofs of a circuit which contains one or many 32-bit binary addition subcircuits and that only the results of the additions are used elsewhere in the circuit. In a similar way we can skip directly enforcing the correct evaluation of that part of the circuit and instead enforce a linear constraint and that the relevant wire values must be binary. If the verifier enforces the linear relation $A_0 + 2A_1 + \ldots 2^{31}A_{31} + B_0 + 2B_1 + \ldots 2^{31}B_{31} = X_0 + 2X_1 + \ldots 2^{31}X_{31} + 2^{32}X_{32}$ this will help us effectively reduce the size of the circuit, because we will not need to enforce that the intermediate results to the final sum are evaluated correctly. As long as the input to that part of the circuit is correct, and the output is correct, then all is well.
In order for this linear relation to be enforced correctly, the characteristic of the finite field must be greater than $2^{33}$. This “addition trick” may help performance in special cases, but we cannot use it effectively for all circuits on which we might want to construct zero-knowledge proofs, so it is a potential optimization in a limited domain, not an improvement to worst-case asymptotic performance. The first (smaller) addition trick can be used on any circuit, but it has only a very minor requirement on the finite field size, so this also does not affect our asymptotic analysis.

Depending on our choice of FFT algorithm we may require, for example, that there be a $2^z$-th primitive root of unity in the finite field for some convenient $z$, which may also increase the necessary field size. There are also implementations of FFT which can operate efficiently on any finite field without this requirement. While the choice of FFT algorithm can significantly affect the prover and verifier computation times, it does not affect the soundness error, the communication complexity of the interactive version of the proof, or the proof length of the noninteractive version. Therefore we are justified in ignoring these constraints on the size of the finite field in our asymptotic worst-case performance analysis.

5.4.4 Multi-Instance Amortization

If we want to prove that $C(x_i, \cdot)$ is satisfiable for $N$ public inputs $x_i$, we can simplify our ZKIPCP construction as follows. The prover $P$ first computes the combined witness $w = w_1, \ldots, w_N$ that is comprised of $N$ wit-
nesses, each is computed as in the single-instance case. Next, \( \mathcal{P} \) arranges
the witnesses in blocks of size \( \ell = N \), where block \( j \) contains the \( j^{th} \) bits
of each of the \( N \) witnesses. The public inputs \( x_i \) define public blocks. The
number of nonpublic blocks equals the size of the witness of a single in-
stance, which is \( m = |w_i| = O(s) \). \( \mathcal{P} \) then encodes the blocks of messages
into \( U \in L^m \).

Even for moderately large \( N \), the multi-instance variant provides sig-
nificant savings in both computation time and communication complex-
ity. This is because we do not need to rearrange the wire values in the
multi-instance case and we do in the single-instance case. The total com-
munication complexity is

\[
\begin{align*}
& k \cdot \sigma + (2N + t) \cdot \sigma + (2N + t - 1) \cdot \sigma + \\
& t \cdot (s + 3 \cdot \sigma) \cdot \lceil \log |\mathbb{F}| \rceil
\end{align*}
\]

\underbrace{\text{degree test}} \ + \underbrace{\text{linear test}} \ + \underbrace{\text{quadratic test}} \ + \underbrace{t \cdot \lceil \log n \rceil \cdot h}.

For sufficiently-large fields, we can set \( t = O(\kappa) \) and if \( N > O(\kappa^2) \),
then the proof length is shorter than \( sN \) bits. Note that this threshold is
independent of the circuit size.
5.5 Implementation and Results

We implemented our protocol in C++ using Shoup’s NTL library for the finite field operations. We used BOOST MPI to facilitate the message passing between prover and verifier. We chose a prime that had a sufficiently large power of two primitive root of unity and set $\eta_i$ and $\zeta_j$ values to be the roots of unity. This enabled us to perform interpolation and evaluation of sharing polynomials using FFT and inverse FFT operations. We ran our experiments on a machine with an Intel Core i7-4720HQ CPU 2.60 GHz processor with 4 cores and 8 GiB RAM. We used SHA-256 for our collision resistant hash function.

We primarily compare our work with ZKBoo/ZKB++ [GMO16,
CDG\textsuperscript{+17}, which qualitatively match our result in most aspects. The crucial advantage of our approach over ZKBoo/ZKB++ is that our communication is sublinear in the circuit size whereas ZKBoo/ZKB++ incurs communication that is proportional in the circuit size. Moreover, in the amortized setting, our approach provides significantly-better communication cost and runtimes compared to ZKBoo/ZKB++.

The primary boolean predicate we used to demonstrate our implementation was verifying a SHA-256 certificate, as it is the common benchmark used in prior works. Namely, on a common input a 256-bit string $y$ and a private 512-bit input $x$ of the prover, the prover convinces the verifier that SHA256$(x) = y$.

![Figure 5.7](image1.png) **Figure 5.7:** Amortized prover and verifier running times for verifying multiple instances of circuit with 2048 gates and SHA-256 circuit.

![Figure 5.8](image2.png) **Figure 5.8:** Amortized proof lengths for multiple instances of circuit with 2048 gates and SHA-256 circuit.
**Optimizations** A standard computation of the SHA-256 certificate involves AND gates and XOR gates and addition modulo $2^{32}$ gates. One approach is to simply implement the addition modulo $2^{32}$ gates using boolean AND and XOR gates. We follow a different approach, where we express the addition modulo $2^{32}$ gate consistency as a linear constraint over the bits of the inputs and output of the gate. Following Section 5.3.5, this can be efficiently realized if we rely on a prime field larger than $2^{33}$. However, as mentioned in the previous section, to obtain optimal communication for a given witness size requires choosing a field of a specific size. To handle this, we incorporate these addition gates by considering a word size of $\lceil \log |F| \rceil$ and performing 32-bit additions using arithmetic over the smaller word size. We express the sum of each “digit” of the addends as a linear constraint on the bits of the witness, where digits are each a number of bits equal to the word size. Following these optimizations results in a witness size of 33928 bits for the SHA-256 certificate (for $|F| \geq 2^{14}$). We ran our protocol for different circuit sizes and for each size, we ran ad-hoc optimizers to obtain optimal parameters for soundness $2^{-40}$. We remark that a tighter soundness error for the tests described in Section 5.3 which in turn is used in our ZKIPCP can be obtained by discarding the last inequality in Lemma 5.14, 5.18, and 5.23. We relied on these better bounds in our optimizer (as opposed to the cleaner bounds that appear in the lemma statements). For the case of $2^{-80}$ soundness error the communication and computation costs doubles
(as in [GMO16, CDG17]). For boolean circuits, the quadratic constraints only involve checking whether each element of the witness is binary and we can simplify the test in Section 5.3.3 by eliminating $x, y, z$ and having the prover compute $p_0 = \sum_i r_i \cdot (p_{i}^{w} \cdot p_{i}^{w} - p_{i}^{w})$.

In Figure 5.5, we compare the prover and verifier running times for verifying circuits of sizes varying from 2048 gates to 400000 gates. The computational complexity of both the prover and the verifier in the single-instance setting are proportional to $O(s \log s)$ field operations, where $s$ is the circuit size. The optimal field size can be asymptotically shown to be $O(\log s)$, resulting in an overall computational complexity of $O(s \log^2 s)$. We remark here that if we make uniformity assumptions on the circuit, then the verifier’s computational complexity becomes sublinear in the circuit size. In fact, the multi-instance setting can be seen as a uniformity assumption and here the verifier’s complexity is indeed smaller than the computational complexity.

In Figure 5.6, we provide the communication complexity of our zero-knowledge argument measured in kilobytes (KiB). We plot two instantiations of our protocol. We provide communication cost for our provable variant (labelled Ligero) and for the variant that assumes Conjecture 5.17 (labelled Ligero-Strong). We observe that Ligero-Strong yields a 20% re-

\footnote{Note that our proof length and computation times are not influenced by circuit topology and only depend on the witness size which in turn depends only on the number of gates. In the case of boolean circuits, they are independent of gate composition (assuming the circuit comprises of only XOR and AND gates).}
duction in communication cost on average. The communication cost for a SHA-256 certificate is 44KiB with Ligero and 34KiB with Ligero-Strong. We can also see in Figure 5.6 that when the circuit is larger than 3 million gates our communication cost is smaller than the circuit size.

We compare our complexity with ZKB++ [CDG+17, GMO16]). We first note that the complexity of ZKB++ only depends on the number of boolean AND gates (XOR gates are free). In our implementation we relied on prime fields and our communication cost depends on the number of AND and XOR gates. However, if we use an extension field of $\mathbb{GF}_2$, we can eliminate the cost of XOR gates the same way ZKB++ does. In this variant of our protocol, each AND gate will incur 3 bits in the witness. In Figure 5.6, in order to make a fair comparison with ZKB++ for a circuit of size $s$, we plot the communication cost incurred by the ZKB++ protocol for a circuit of size $2s/3$. The idea is that for a circuit comprising of $2s/3$ AND gates the cost of ZKB++ is compared with the communication cost of our protocol assuming a characteristic-2 FFT implementation. The threshold for which our approach incurs lesser communication than ZKB++ is roughly 3000 (AND) gates.

In Figure 5.7 and Figure 5.8 we provide our prover and verifier running times and communication for the multi-instance version of our protocol. We take a 2048 gate circuit and the SHA-256 circuit to illustrate our performance. The heaviest part of the verification involves performing FFTs over domains of sizes $N$ and $s$ where $s$ is the circuit size. Since we
considered 1 to 4096 instances, even for moderately-sized circuits the FFT over domain of size $s$ dominates the runtime of the verifier. The prover runtime, on the other hand, is about $O(N s \log s)$. We see a reduction in the amortized prover’s cost per instance with periodic jumps because we perform FFT over a larger domain, with a number of elements equal to a power of 2.

The communication complexity varies additively in $N$ and $s$. The amortized communication cost per instance decreases linearly because, similar to the verifier complexity, $s$ dominates the complexity until $N$ becomes significant compared to $s$.

## 5.6 Case $e < d/3$: Proof of Lemma 5.17

In this section, we provide the proof of our main lemma for the case when $e < \frac{d}{3}$. The proof of Claim 5.33 below is due to Ronny Roth and Gilles Zémor [RZ17].

**Lemma 5.32** (restatement of Lemma 5.17). Let $e$ be a positive integer such that $e < \frac{d}{3}$. Suppose $d(U, L^m) > e$. Then, for a random $w^*$ in the row-span of $U$, we have

$$\Pr[d(w^*, L) \leq e] \leq \frac{e + 1}{|F|}.$$

**Proof.** Suppose that $d(U^*, L^m) > e$ and $L^*$ is the span of the vectors in $U^*$. Assume for a contradiction that $d(v^*, L) \leq e$ for all $v^* \in L^*$. Suppose $v^*_0 \in L^*$ maximizes the distance from $L$. Since $d(U^*, L^m) > e$, there must
be a row $U_i^*$ such that $\Delta(U_i^*, L) \setminus \Delta(v_0^*, L) \neq \emptyset$. Let $v_0^* = u_0 + \chi_0$ and $U_i^* = u_i + \chi_i$ for $u_0, u_i \in L$ and $\chi_0, \chi_i$ of weight $\leq e$. We argue that there exists $\alpha \in F$ such that for $\hat{\vartheta} = v_0^* + \alpha U_i^*$ we have $d(\hat{\vartheta}, L) > d(v_0^*, L)$, contradicting the choice of $v_0^*$. This follows by a union bound, noting that for any $j \in \Delta(v_0^*, L) \cup \Delta(U_i^*, L)$ there is at most one choice of $\alpha$ such that $\hat{\vartheta}_j = 0$.

Now, it suffices to show that in any affine subspace of $F^n$, either all points are $e$-close to $L$ or almost all are not. This reduces to showing the following claim. We state an explicit version of the conjecture for the case of Reed-Solomon codes.

Claim 5.33. Let $L$ be an arbitrary linear code over $F$ of length $n$. Let $e$ be a positive integer such that $e < \frac{d}{3}$. Then for every $(u, v) \in (F^n)^2$, defining an affine line $\ell_{u, v} = \{u + \alpha v : \alpha \in F\}$, either (1) for every $x \in \ell_{u, v}$ we have $d(x, L) \leq e$, or (2) for at most $d$ points $x \in \ell_{u, v}$ we have $d(x, L) \leq e$.

We begin with the observation that for any two length $n$ vectors $u$ and $v$ of weight at most $e$, $\ell_{u, v}$ contains $N$ points at most distance $e$ from $L$ if and only if $\ell_{u, v + c}$ contains $N$ points of distance at most $e$ from $L$ for any codeword $c \in L$. This means it suffices to prove the claim for vectors $u$ and $v$ of weight at most $e$.

We now prove the lemma in two cases

**Case 1:** $|\text{Support}(u) \cup \text{Support}(v)| \leq e$ This means that $\ell_{u, v}$ is entirely contained in the ball $B_e(0)$ where $0$ is the all os vector which in
turn means all the vectors in the line are at most \( t \) from \( L \).

**Case 2:** \(|\text{Support}(u) \cup \text{Support}(v)| \geq e + 1\) Since \( u \) and \( v \) each have weight at most \( e \), the intersection of their supports can be of cardinality at most \( e - 1 \). For each of the coordinates in the intersection of the supports, there can be at most one vector in \( \ell_{u,v} \) such that the entry in that coordinate is 0. Therefore, there are at most \( e - 1 \) vectors in \( \ell_{u,v} \) that are contained in the ball \( B_e(0) \) where 0 is the all 0s vector.

To conclude this case, we need to demonstrate that there exists no codeword \( c \neq 0 \) such that the line \( \ell_{u,v} \) intersects with a vector inside the ball of radius \( e \) around \( c \). Assume for contradiction there exists a codeword \( c \) and vector \( w \) of weight at most \( e \) such that \( c + w \in \ell_{u,v} \). Then we have that

\[
c + w = u + \alpha v
\]

This means that \( c \) is equal to the sum of three vectors each of weight at most \( e \). Now we arrive at a contradiction because the minimum distance of \( L \) is \( d \) and \( e < \frac{d}{3} \).

### 5.7 Generalizing IPCP Tests

In this section, we provide the generalized versions of the tests in our basic IPCP. This is required for improving the soundness analysis and
achieving better concrete parameters.

### 5.7.1 Generalized Interleaved Linear Code Testing

In this section we present a generalized version of the testing algorithm that uses $\sigma$ linear combinations to amplify soundness. This algorithm is useful for obtaining better soundness over a small field $\mathbb{F}$.

**Oracle:** An alleged $L^m$-codeword $U$. Depending on the context, we may view $U$ either as a matrix in $\mathbb{F}^{m \times n}$ in which each row is an alleged $L$-codeword, or as a sequence of $n$ symbols $(U_1, \ldots, U_n)$, $U_i \in \mathbb{F}^m$.

**Parameters:**
- Probing parameter $t < n$ (number of symbols $U_i$ read by $V$).
- Repetition parameter $\sigma$ (number of random linear combinations).

**Interactive testing:**

1. $V$ picks $\sigma$ random linear combinations $(r_1, \ldots, r_\sigma) \in (\mathbb{F}^m)^\sigma$ and sends them to $P$.
2. $P$ responds with $w_h = r_h^T U \in \mathbb{F}^n$, for each $h = 1, \ldots, \sigma$.
3. $V$ queries a set $Q \subseteq [n]$ of $t$ random symbols $U_j$, $j \in Q$.
4. $V$ accepts if and only if all $w_h$ are in $L$ and are consistent with $U_Q$ and $r_h$. That is, for every $j \in Q$ and $1 \leq h \leq \sigma$, we have
   \[
   \sum_{i=1}^m (r_h)_j \cdot U_{i,j} = (w_h)_j.
   \]

**Figure 5.9:** GeneralTestInterleaved($\mathbb{F}, L[n,k,d], m, t, \sigma; U$)

**Lemma 5.34.** If $U \in L^m$ and $P$ is honest, then $V$ always accepts.
Lemma 5.35. Let $e$ be a positive integer such that $e < \frac{d}{4}$. Suppose $d(U^*, L^m) > e$. Then, for a random $w^*$ in the row-span of $U^*$, we have

$$\Pr[d(w^*, L) \leq e] \leq \frac{e + 1}{|\mathbb{F}|^\sigma}.$$

Theorem 5.36. Let $e$ be a positive integer such that $e < \frac{d}{4}$. Suppose $d(U^*, L^m) \geq e$. Then for any malicious $\mathcal{P}$ strategy, the oracle $U^*$ is rejected by $V$ except with probability at most $(1 - \frac{e}{n})^t + \frac{e + 1}{|\mathbb{F}|^\sigma}$.

We provide a formal proof of a generalization of this test in Section 5.7.3.

5.7.2 Affine Interleaved Linear Code Testing

For the purpose of obtaining a zero-knowledge IPCP, the following “affine” variant of $\text{TestInterleaved}$ is useful. Whenever $V$ requests a random linear combination of the rows of $U$, this linear combination will be masked with an additional blinding vector $u' \in \mathbb{F}^n$. The vector $u'$, which is also given as part of the proof oracle, will be picked by an honest $\mathcal{P}$ at random from $L$ and will therefore hide all information about $U$ whose rows are from $L$. The soundness of the test should hold even when $u'$ is adversarially-chosen and is not necessarily a codeword.

Completeness follows directly from the description.

Lemma 5.37. If $U \in L^m$, $u' \in L$, and $\mathcal{P}$ is honest, then $V$ always accepts.
Oracle: An alleged $L^m$ codeword $U$ and an additional auxiliary row vector $u' \in \mathbb{F}^n$.

Interactive Testing

1. $V$ picks a random linear combinations $r \in \mathbb{F}^m$ and sends $r$ to $P$.
2. $P$ responds with $w = r^T U + u' \in \mathbb{F}^n$.
3. $V$ queries a set $Q \subseteq [n]$ of $t$ random symbols $U_j$, $j \in Q$, as well as $u'_j$, $j \in Q$.
4. $V$ accepts if and only if $w \in L$ and $w$ is consistent with $U_Q$, $u'_Q$, and $r$. That is, for every $j \in Q$ we have $\sum_{i=1}^m r_j \cdot U_{ij} + u'_j = w_j$.

Figure 5.10: Affine Test Interleaved ($\mathbb{F}, L[n, k, d], m, t; U, u'$)

Our soundness analysis will rely on the following lemma.

**Lemma 5.38.** Let $e$ be a positive integer such that $e < \frac{d}{4}$. Suppose $d(U^*, L^m) > e$. Then, for arbitrary $u' \in \mathbb{F}^n$ and a random $w^*$ in the row span of $U^*$, we have $\Pr[d(w^*, L) \leq e] \leq \frac{e+1}{|\mathbb{F}|}$.

**Theorem 5.39.** Let $e$ be a positive integer such that $e < \frac{d}{4}$. Suppose $d(U^*, L^m) \geq e$. Then, for an arbitrary $u' \in \mathbb{F}^n$ and any malicious $P$ strategy, the oracle $U^*$ is accepted by $V$ with probability at most $(1 - \frac{e}{n})^t + \frac{e+1}{|\mathbb{F}|}$.

We provide a formal proof of a generalization of this test in the next section.
5.7.3 Generalized Affine Interleaved Linear Code Testing

For the purpose of obtaining a zero-knowledge IPCP, the following “affine” variant of TestInterleaved is useful. Whenever $V$ requests a random linear combination of the rows of $U$, this linear combination will be masked with an additional blinding vector $u' \in \mathbb{F}^n$. The vector $u'$, which is also given as part of the proof oracle, will be picked by an honest $P$ at random from $L$ and will therefore hide all information about $U$ whose rows are from $L$. The soundness of the test should hold even when $u'$ is adversarially chosen and is not necessarily a codeword. We generalize it further following the previous section to achieve better soundness by repetition.

**Oracle:** An alleged $L^m$ codeword $U$ and additional auxiliary row vectors $(u'_1, \ldots, u'_\sigma) \in (\mathbb{F}^n)^\sigma$.

**Interactive Testing:**

1. $V$ picks a random linear combinations $(r_1, \ldots, r_\sigma) \in (\mathbb{F}^m)^\sigma$ and sends $r$ to $P$.
2. $P$ responds with $w_h = r_h^T U + u'_h \in \mathbb{F}^n$, $h = 1, \ldots, \sigma$.
3. $V$ queries a set $Q \subseteq [n]$ of $t$ random symbols $U_j$, $j \in Q$, as well as $(u'_\sigma)_j$, $j \in Q$.
4. $V$ accepts if and only if all $w_h \in L$ and are consistent with $w_h$ and $U_Q$, $u_h$, and $r_h$. That is, for every $j \in Q$ we have $\sum_{i=1}^m (r_h)_j \cdot U_{i,j} + (u'_h)_j = w_j$.

Completeness follows directly from the description.
Lemma 5.40. If $U \in L^m$, $(u'_1, \ldots, u'_\sigma) \in L^\sigma$, and $P$ is honest, then $V$ always accepts.

Our soundness analysis will rely on the following lemma.

Lemma 5.41. Let $e$ be a positive integer such that $e < \frac{d}{4}$. Suppose $d(U^*, L^m) > e$. Then, for arbitrary $(u'_1, \ldots, u'_\sigma) \in (\mathbb{F}^n)^\sigma$ and a random $w^*$ in the row span of $U^*$, it holds that $\Pr[\forall h \in [\sigma], d(w^* + u'_h, L) \leq e] \leq \frac{e+1}{|\mathbb{F}|^\sigma}$.

Proof: The proof of Lemma 5.41 is almost identical to the proof of Lemma 5.14. The high-level reason why the same argument works is that the decomposition $w^* = \alpha v^* + x$ where $x$ is independent of $\alpha$ still holds even for $w^* + u'_h$ (since $u'_h$ is a fixed vector, and so $u'_h + x$ is independent of $\alpha$). We provide the full proof below for completeness.

Let $L^*$ be the row span of $U^*$. We consider two cases similar to our proof of Lemma 5.14.

Case 1: There exists $v^* \in L^*$ such that $d(v^*, L) > 2e$. In this case, we show that

$$\Pr_{w^* \in R L^*}[\forall h \in [\sigma], d(w^* + u'_h, L) \leq e] \leq \frac{1}{|\mathbb{F}|^\sigma}. \tag{5.2}$$

Indeed, using a basis for $L^*$ that includes $v^*$, a random $w^* \in L^*$ can be written as $\alpha v^* + x$, where $\alpha \in_R \mathbb{F}$ and $x$ is distributed independently of $\alpha$. We argue that, conditioned on any choice of $x$, there can be at most one choice of $\alpha$ such that $d(\alpha v^* + x + u'_h, L) \leq e$. We
can conclude the case from this as the probability over \((r_1, \ldots, r_h)\) that 
\[ d((r_h)^T U + u'_h, L) \leq e \] holds for every \(h\) is at most \(\frac{1}{|F|}\). This follows by observing that if 
\[ d(\alpha v^* + x_0 + u'_h, L) \leq e \] and 
\[ d(\alpha' v^* + x_0 + u'_h, L) \leq e \] for \(\alpha \neq \alpha'\), then by the triangle inequality we have 
\[ d((\alpha - \alpha')v^*, L) \leq 2e. \] By assumption \(d(v^*, L) > 2e\), which implies \(d(v^*, L) > 2e\), so we arrive at a contradiction.

**Case 2:** For every \(v^* \in L^*, d(v^*, L) \leq 2e\). We show that in this case 
\[ \Pr_{w^* \in L^*} [\forall h \in [\sigma], d(w^* + u'_h, L) \leq e] \leq \frac{e+1}{|F|}. \] Let \(U_i^*\) be the \(i^{th}\) row of \(U^*\) and let 
\[ E_i = \Delta(U_i^*, L). \] Note that since \(2e < \frac{d}{2}\), each \(U_i^*\) can be written uniquely as \(U_i^* = u_i + \chi_i\) where \(u_i \in L\) and \(\chi_i\) is nonzero exactly in its \(E_i\) entries. Let 
\[ E = \bigcup_{i=1}^m E_i. \] Since \(d(U^*, L^m) > e\), we have \(|E| > e\).

We show below that for \(j \in E_i\), except with \(\frac{1}{|F|}\) probability over a random choice of \(w^*\) from \(L^*\), either \(j \in \Delta(w^* + u'_h, L)\) or \(d(w^* + u'_h, L) > e\). First, we conclude the case and the proof of lemma assuming this holds.

We observe that this implies that with probability at most \(\frac{1}{|F|}\) over the choice of \((r_1, \ldots, r_\sigma)\), it holds that, for all \(h, j \notin \Delta((r_h)^T U + u'_h, L)\) and 
\[ d((r_h)^T U + u'_h, L) \leq e. \] Taking a union bound over the first \(e + 1\) elements of \(E\) the claim follows.

Suppose \(j \in E_i\) and fix an arbitrary \(h \in [\sigma]\). As before, we write \(w^* = \alpha U_i^* + x\) for \(\alpha \in \mathbb{F}\) and \(x\) distributed independently of \(\alpha\). Condition on any possible choice \(x_0\) of \(x\). Define a bad set

\[ B_j = \{ \alpha : j \notin \Delta(\alpha U_i^* + x_0 + u_h, L) \land d(\alpha U_i^* + x_0 + u_h, L) \leq e \}. \]
We show that $|B_j| \leq 1$. Suppose toward contradiction that there exists $(\alpha, \alpha') \in \mathbb{F}^2$ such that $\alpha \neq \alpha'$ and for $z = \alpha U_i^* + x_0 + u_h$ and $z' = \alpha' U_i^* + x_0 + u_h$ we have $d(z, L) \leq e$, $d(z', L) \leq e$, $j \notin \Delta(z, L)$, and $j \notin \Delta(z', L)$. Since $d > 4e$, for any $z^*$ in the linear span of $z$ and $z'$ we have $j \notin \Delta(z^*, L)$. Since $(\alpha - \alpha')U_i^* = z - z'$ is in this linear span, we have $j \notin \Delta(U_i^*, L)$, in contradiction to the assumption that $j \in E_i$. 

**Theorem 5.42.** Let $e$ be a positive integer such that $e < \frac{d}{4}$. Suppose $d(U^*, L^n) \geq e$. Then, for arbitrary $(u'_1, \ldots, u'_\sigma) \in \mathbb{F}^e$ and any malicious $\mathcal{P}$ strategy, the oracle $U^*$ is accepted by $\mathcal{V}$ with probability at most

$$\left(1 - \frac{e}{n}\right)^t + \frac{e + 1}{|\mathbb{F}|^e}.$$ 

### 5.7.4 Generalized Affine Linear Constraint Testing over Interleaved Reed-Solomon Codes

For the purpose of obtaining a zero-knowledge IPCP, we provide the following “affine” variant of TestLinearConstraintsIRS. Whenever $\mathcal{V}$ provides the challenge vector $r$, the linear combination $r^T A$ of the rows of $U$, will be masked with an additional blinding vector $u' \in \mathbb{F}^n$ that encodes messages that sum up to 0. The vector $u'$, which is also given as part of the proof oracle, will be picked by an honest $\mathcal{P}$ at random from $L$ subject to the condition that it encodes messages that sum up to 0 and will therefore hide all information about the individual column sums in the computa-
tion of $r^T Ax$. The soundness of the test should hold even when $u'$ is adversarially chosen and is not necessarily a codeword. We will further generalize the test to achieve better soundness. Namely, instead of relying on repetition, we improve soundness by considering the challenge space from an extension field. The test is given in Figure 5.12. We will analyze the test under the promise that the (possibly badly-formed) $U$ is close to $L^{m+1}$. Completeness follows directly as $u'$ does not affect the verification. We argue soundness next.

**Lemma 5.43.** Let $e$ be a positive integer such that $e < \frac{d}{2}$. Suppose that a (badly-formed) oracle $U^*$ that is vertically juxtaposed with an arbitrary $u'$ is $e$-close to a codeword $V \in L^{m+1}$, where $V$ contains the codewords $U \in L^m$ and $u^* \in L$ vertically juxtaposed, and $U$ encodes $x \in \mathbb{F}^{m\ell}$ such that $Ax \neq b$. Then, for any malicious $\mathcal{P}$ strategy, $U^*$ is accepted by $V$ with probability at most

$$\frac{1}{|\mathbb{F}|^{\sigma}} + \left(\frac{e + k + \ell}{n}\right)^t.$$

**Proof:** Let $p_i$ be the polynomial of degree less than $k$ corresponding to the $i^{th}$ row of $U$ and $(\gamma_1, \ldots, \gamma_c)$ the message encoded by $u^*$ via the polynomial $r_{\text{blind}}$. Let $q$ be the polynomial computed following the honest strategy in Step 3 using $p_i$s and $r_{\text{blind}}$. Since we assume $Ax \neq b$, we have $\Pr[r^T Ax = r^T b] = \frac{1}{|\mathbb{F}|} = \frac{1}{|\mathbb{F}|^{\sigma}}$. In fact, for any arbitrary $(\gamma_1, \ldots, \gamma_c)$, we can further claim that $\Pr[r^T Ax + \sum_{c \in [\ell]} \gamma_c = r^T b] = \frac{1}{|\mathbb{F}|^{\sigma}}$. Therefore it follows that except with probability $\frac{1}{|\mathbb{F}|^{\sigma}}$ over the choice of $r$ in Step 1, the
Oracle: An alleged $L^m$ codeword $U$ that should encode a message $x \in \mathbb{F}^{m \ell}$ satisfying $Ax = b$ and an additional auxiliary row vector $u' \in \hat{\mathbb{F}}^n$ that encodes the message $(\gamma_1, \ldots, \gamma_\ell)$ such that $\sum_{c \in [\ell]} \gamma_c = 0$ where $\hat{\mathbb{F}}$ is an extension field of $\mathbb{F}$ such that $|\hat{\mathbb{F}}| = |\mathbb{F}|^{\sigma}$.

Interactive testing:

1. $V$ picks a random vector $r \in \hat{\mathbb{F}}^{m \ell}$ and sends $r$ to $P$.

2. $V$ and $P$ compute

$$r^T A = (r_{11}, \ldots, r_{1\ell}, \ldots, r_{m1}, \ldots, r_{m\ell})$$

and for $i \in [m]$, let $r_i$ be the unique polynomial of degree less than $\ell$ such that $r_i(\zeta_c) = r_{ic}$ for every $c \in [\ell]$.

3. $P$ sends the $k + \ell - 1$ coefficients of the polynomial defined by $q(\cdot) = \sum_{i=1}^m r_i(\cdot) \cdot p_i(\cdot) + r_{\text{blind}}(\cdot)$, where $p_i(\cdot)$ is the polynomial of degree $< k$ corresponding to row $i$ of $U$ and $r_{\text{blind}}(\cdot)$ is the polynomial of degree $< k$ corresponding to $u'$.

4. $V$ queries a set $Q \subseteq [n]$ of $t$ random symbols $U_j$, $j \in Q$, as well as $u'_{j'}$, $j' \in Q$.

5. $V$ accepts if and only if the following conditions hold:

(a) $\sum_{c \in [\ell]} q(\zeta_c) = \sum_{i \in [m], c \in [\ell]} r_{ic} b_{ic}$, and

(b) for every $j \in Q$ we have $u'_{j'} + \sum_{i=1}^m r_i(\eta_j) \cdot U_{i,j} = q(\eta_j)$.

Figure 5.12: GeneralAffineTestLinearConstrsIRS ($\mathbb{F}$, $\text{RS}_{\mathbb{F}, n, k, \eta}$, $m$, $t$, $\zeta$, $A$, $b$, $\sigma$; $U$)

following equation fails to hold:

$$\sum_{c \in [\ell]} \left( \sum_{i \in [m]} r_i(\zeta_c) p_i(\zeta_c) + \gamma_c \right) = \sum_{i \in [m], c \in [\ell]} r_{ic} b_{ic}.$$ 

Indeed, we have $\sum_{i \in [m]} \sum_{c \in [\ell]} r_i(\zeta_c) p_i(\zeta_c) = (r^T A)x$ and further
\[ \sum_{i \in [k], c \in [\ell]} r_{ic} b_{ic} = r^T b. \] In other words, the polynomial \( q \), if provided by \( P \), would fail in Step 5a except with probability \( \frac{1}{|F|} \).

Next, we analyze the probability that a malicious \( P \) strategy is rejected conditioned on \( q \) failing as above. Let \( q' \) be the polynomial sent by the prover. If \( q' = q \), then \( V \) rejects in Step 3 with probability \( \frac{1}{|F|} \). If \( q' \neq q \) then using the fact that \( q \) and \( q' \) have degree at most \( k + \ell - 2 \), we have that the number of indices \( j \in [n] \) for which \( q(\eta_j) = q'(\eta_j) \) is at most \( k + \ell - 2 \). Let \( Q' \) be the set of indices on which they agree. Then \( V \) rejects in Step 5a whenever \( Q \) selected in Step 2 contains an index \( i \notin Q' \cup E \), where \( E = \Delta(U^*, L^m) \). This fails to happen with probability at most

\[ \frac{(e+k+\ell-2)^t}{n^t} \leq \left( \frac{e+k+\ell}{n} \right)^t. \]

The lemma now follows by a simple union bound. \( \Box \)

5.7.5 Generalized Testing Quadratic Constraints over Interleaved Reed Solomon Codes

Finally, in this section we extend our quadratic constraint test over Interleaved Reed Solomon codes via parallel repetition to improve soundness. The complete test description is provided in Figure 5.13. Next, the following lemma follows again directly from the description.

**Lemma 5.44.** If \( (U^x, U^y, U^z) \in (L^m)^3 \) encode vectors \( (x, y, z) \in (\mathbb{F}^{m\ell})^3 \) satisfying \( x \odot y + a \odot z = b \) and \( P \) is honest, then \( V \) always accepts.
Oracle Alleged $L^m$ codewords $(U^x, U^y, U^z)$ that should encode messages $(x, y, z) \in (\mathbb{F}^m)^3$ satisfying $x \odot y + a \odot z = b$.

Interactive Testing

1. Let $U^a = \text{Enc}_\xi(a)$ and $U^b = \text{Enc}_\xi(b)$.

2. $\mathcal{V}$ picks random linear combinations $(r_1, \ldots, r_\sigma) \in (\mathbb{F}^m)^\sigma$ and sends $r$ to $\mathcal{P}$.

3. $\mathcal{P}$ sends the $2k - 1$ coefficients of the $\sigma$ polynomials $(p_0^h, \ldots, p_\sigma^h)$ defined by $p_i^h = \sum_{i=1}^m (r_h)_i \cdot p_i$, where $p_i = p_i^x \cdot p_i^y + p_i^a \cdot p_i^z - p_i^b$ for $h \in [\sigma]$, and where $(p_1^x, p_1^y, p_1^z)$ are the polynomials of degree less than $k$ corresponding to row $i$ of $(U^x, U^y, U^z)$, and $(p_1^a, p_1^b)$ are the polynomials of degree less than $\ell$ corresponding to row $i$ of $(U^a, U^b)$.

4. $\mathcal{V}$ picks a random index set $Q \subseteq [n]$ of size $t$, and queries $(U^x_i, U^y_i, U^z_i)$, $j \in Q$.

5. $\mathcal{V}$ accepts if and only if the following conditions hold for every $h \in [\sigma]$:

   (a) for every $c \in [\ell]$ it holds that $p_0^h(\xi_c) = 0$, and

   (b) for every $j \in Q$, it holds that

   $$p_0^h(\eta_j) = \sum_{i=1}^m (r_h)_i \cdot \left[ U^x_{i,j} \cdot U^y_{i,j} + U^a_{i,j} \cdot U^z_{i,j} - U^b_{i,j} \right].$$

Figure 5.13: GeneralTestQuadraticConstsIRS($\mathbb{F}, L = \text{RS}_{\mathbb{F}, n, k, \eta, m, t, \xi, a, b, \sigma; U^x, U^y, U^z}$)

We argue soundness with the following lemma.

**Lemma 5.45.** Let $e$ be a positive integer such that $e < \frac{d}{2}$. Let $U^{x*}, U^{y*}, U^{z*}$ be badly-formed oracles and let $U^* \in \mathbb{F}^{3m \times n}$ be the matrix obtained by vertically juxtaposing the corresponding $m \times n$ matrices. Suppose $d(U^*, L^m) \leq e$. Let $(U^x, U^y, U^z)$, respectively, be the (unique) codewords
in $L^m$ that are closest to $(U^{x*}, U^{y*}, U^{z*})$. Suppose $(U^x, U^y, U^z)$ encode $(x, y, z)$ such that $x \odot y + a \odot z \neq b$. Then, for any malicious $\mathcal{P}$ strategy, $(U^{x*}, U^{y*}, U^{z*})$ is accepted by $V$ with probability at most

$$\frac{1}{|F|^\sigma} + \left(\frac{e + 2k}{n}\right)^t.$$

**Proof:** Let $p_0^h$ be the polynomial generated in Step 3 following the honest $\mathcal{P}$ strategy on $(U^x, U^y, U^z)$. Since $(x, y, z)$ do not satisfy the constraint $x \odot y + a \odot z = b$, the polynomial $p_0^h$ satisfies the condition in Step 5a with probability $1/|F|$ over the choice of $r_h$ in Step 2. We have $p_0^h = \sum_{i=1}^m r_{hi} \cdot p_i$ and there must exist an $i$ and $\zeta_c$ such that $p_i(\zeta_c) \neq 0$. This implies that the probability that there is no $h \in [\sigma]$ such that $p_0^h$ fails in this way is at most $1/|F|$. 

Next we analyze the probability that a malicious $\mathcal{P}$ strategy is rejected conditioned on $p_0$ failing as above. Let $p_0^{h'}$ be the polynomial sent by the prover for $h \in [\sigma]$. If $p_{00}^{h'} = p_0^h$, then $V$ rejects in Step 5a with probability $1/|F|$. If $(p_{00}^{h'})_0 \neq p_0^h$, then using the fact that $p_0^h$ and $p_0^{h'}$ have degree at most $2k - 2$, we have that the number of indices $j \in [n]$ for which $p_0^h(\eta_j) = p_0^{h'}(\eta_j)$ is at most $2k - 2$. Let $Q'$ be the set of indices on which $p_0$ and $p_0'$ agree. Then $V$ rejects in Step 5b whenever $Q$ selected in Step 4 contains an index $i \not\in Q' \cup E$, where $E = \Delta(U^*, L^{3m})$. This fails to happen with probability at most

$$\frac{(e + 2k - 2)}{t \choose n} \leq \left(\frac{e + 2k}{n}\right)^t.$$
The lemma now follows by a union bound.
Chapter 6

Conclusion

In this thesis, we investigate the limits of designing efficient cryptographic primitives from symmetric-key primitives, namely, one-way functions and collision-resistant hash-functions where the constructions rely on the underlying primitives in a black-box manner.

The two main primitives considered in this thesis are zero-knowledge proofs and universal one-way hash-functions and the main complexities investigated are round-complexity, communication complexity, and query complexity. Below, we summarize our findings and highlight open problems.

Precise zero-knowledge: We show that for all languages in NP and 
\[ r(n) \in \omega(\frac{\log n}{\log \log n}) \] there exists an \( r(n) \)-round precise zero-knowledge argument with the precision \( p(n, t) = cg(n)t + cn \) for some \( c \) with a (partial) black-box simulator. We provide a matching lower bound where we argue that for any \( c \) if a language \( L \) has a 
\[ \frac{c \log n}{\log \log n} \]-round precise zero-knowledge proof with precision \( p(n, t) = \]
$cg(n)t + n^{c-1}$, then $L \in \text{BPP}$. This resolves the round complexity of precise zero-knowledge completely. A concrete next step in this line of work is to generalize precise zero-knowledge for multiple dimensions, eg: space and time.

**Universal one-way hash functions** Our main result here provides a construction of a UOWHF from regular one-way functions of unknown regularity, where the underlying one-way function is queried only $O(n \log n)$ times. An interesting observation in this construction is that our construction yields a general approach that can be instantiated to obtain both PRGs (expanding) and UOWHFs (compressing). While PRGs require pseudoentropy UOWHFs require “inaccessible” entropy. A natural question here is to unify other constructions of PRGs and UOWHFs via a unified notion of entropy that can yield both these primitives.

**Succinct non-interactive zero-knowledge arguments** We design and implement a simple zero-knowledge proof protocol for $\text{NP}$ with communication complexity proportional to the square root of the verification circuit size. This is the first sublinear proof protocol that simultaneously avoids heavy PCP machinery and the use of public-key cryptography. The protocol can be constructed based on any collision-resistant hash function in a black-box way. The proof system can be noninteractive in the random oracle model, yielding
concretely-efficient zk-SNARKs that do not require a trusted setup or public-key cryptography. Since our work, the design of concretely efficient transparent proof systems has become a major field of research with several published works. A major bottleneck in most constructions (including ours) is memory usage: it seems to require memory proportional to the circuit size. Borrowing ideas from programming languages and compiler design should enable these proof systems to be applicable to wider real world use cases.

This work exemplifies the need to study the black-box complexity of cryptographic primitives. Our concretely efficient zero-knowledge system is a variant of the MPC-in-the-head paradigm introduced by Ishai et al. [IKOS07] where the focus was on designing a zero-knowledge system that treated the underlying NP relation in a “black-box” manner. While black-box complexity has largely been considered as a theoretical endeavor, this work shows for the first time that it can lead to very practical constructions.
Bibliography


[AS98] Sanjeev Arora and Shmuel Safra. Probabilistic checking of


[BCG+16] Eli Ben-Sasson, Alessandro Chiesa, Ariel Gabizon, Michael Riabzev, and Nicholas Spooner. Short interactive oracle


[GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended


