A Study of Efficient Secret Sharing

by

Prashant Vasudevan

B.Tech., Computer Science and Engineering, IIT Madras (2013)

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

Master of Science in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2015

Author .....
Department of Electrical Engineering and Computer Science
August 27, 2015

Certified by .....
Vinod Vaikuntanathan
Associate Professor
Thesis Supervisor

Accepted by .....
Leslie A. Kolodziejski
Professor of Electrical Engineering
Chairman, Department Committee for Graduate Students

A Study of Efficient Secret Sharing

by

Prashant Vasudevan

Submitted to the Department of Electrical Engineering and Computer Science
on August 27, 2015, in partial fulfillment of the
requirements for the degree of
Master of Science in Computer Science and Engineering

Abstract

We show a general connection between various types of statistical zero-knowledge (SZK) proof systems and (unconditionally secure) secret sharing schemes. Viewed through the SZK lens, we obtain several new results on secret-sharing:

- *Characterizations:* We obtain an almost-characterization of access structures for which there are secret-sharing schemes with an efficient sharing algorithm (but not necessarily efficient reconstruction). In particular, we show that for every language $L \in \mathsf{SZK_L}$ (the class of languages that have statistical zero knowledge proofs with log-space verifiers and simulators), a (monotonized) access structure associated with L has such a secret-sharing scheme. Conversely, we show that such secret-sharing schemes can only exist for languages in SZK.

- *Constructions:* We show new constructions of secret-sharing schemes with efficient sharing and reconstruction for access structures that are in P, but are not known to be in NC, namely Bounded-Degree Graph Isomorphism and constant-dimensional lattice problems. In particular, this gives us the first combinatorial access structure that is conjectured to be outside NC but has an efficient secret-sharing scheme. Previous such constructions (Beimel and Ishai; CCC 2001) were algebraic and number-theoretic in nature.

- *Limitations:* We show that *universally-efficient* secret-sharing schemes, where the complexity of computing the shares is a polynomial independent of the complexity of deciding the access structure, cannot exist for all (monotone languages in) P, unless there is a polynomial q such that $\mathsf{P} \subseteq \mathsf{DSPACE}(q(n))$.

Thesis Supervisor: Vinod Vaikuntanathan
Title: Associate Professor

A Study of Efficient Secret Sharing

by

Prashant Vasudevan

Submitted to the Department of Electrical Engineering and Computer Science
on August 27, 2015, in partial fulfillment of the
requirements for the degree of
Master of Science in Computer Science and Engineering

Abstract

We show a general connection between various types of statistical zero-knowledge (SZK) proof systems and (unconditionally secure) secret sharing schemes. Viewed through the SZK lens, we obtain several new results on secret-sharing:

- *Characterizations:* We obtain an almost-characterization of access structures for which there are secret-sharing schemes with an efficient sharing algorithm (but not necessarily efficient reconstruction). In particular, we show that for every language $L \in \mathsf{SZK}_{\mathbf{L}}$ (the class of languages that have statistical zero knowledge proofs with log-space verifiers and simulators), a (monotonized) access structure associated with L has such a secret-sharing scheme. Conversely, we show that such secret-sharing schemes can only exist for languages in SZK.

- *Constructions:* We show new constructions of secret-sharing schemes with efficient sharing and reconstruction for access structures that are in P, but are not known to be in NC, namely Bounded-Degree Graph Isomorphism and constant-dimensional lattice problems. In particular, this gives us the first combinatorial access structure that is conjectured to be outside NC but has an efficient secret-sharing scheme. Previous such constructions (Beimel and Ishai; CCC 2001) were algebraic and number-theoretic in nature.

- *Limitations:* We show that *universally-efficient* secret-sharing schemes, where the complexity of computing the shares is a polynomial independent of the complexity of deciding the access structure, cannot exist for all (monotone languages in) P, unless there is a polynomial q such that $\mathsf{P} \subseteq \mathsf{DSPACE}(q(n))$.

Thesis Supervisor: Vinod Vaikuntanathan
Title: Associate Professor

Acknowledgments

I am very grateful to my advisor Vinod Vaikuntanathan for guiding me through these first two years of my PhD candidacy. It has been, as I am sure it will continue to be, great working with him on various problems.

I cannot overstate the role my family, especially my parents, sister, and grandmother, have played. Their moral support has been invaluable, especially in whatever little times of listlessness and uncertainty there were.

I am grateful to all my friends, both here at Cambridge and elsewhere, without the company of whom the days would have been a little shorter and the nights filled somewhat less with merriment and mirth and music.

And finally, to Isaac Asimov, for The Gods Themselves, The End of Eternity, and a treasure of other fascinating and wondrous tales from fictional futures of human society, and to Bonny and Sujjeet for getting me the books.

Contents

Chapter 1

Introduction

Secret-sharing [33, 9], a foundational primitive in information-theoretic cryptography, enables a dealer to distribute shares of a secret to n parties such that only some predefined authorized sets of parties will be able to reconstruct the secret from their shares. Moreover, the shares of any unauthorized set of parties should reveal *no information* about the secret, even if the parties are computationally unbounded. The (monotone) collection of authorized sets is called an *access structure*.

We call a secret-sharing scheme *efficient* if both the sharing algorithm (executed by the dealer) and reconstruction algorithm (executed by the parties) run in time polynomial in n. Associating sets $S \subseteq [n]$ with their characteristic vectors $x_S \in \{0,1\}^n$, we can define a language $L_{\mathcal{A}}$ associated with an access structure $\mathcal{A}$.[1] Namely, $L_{\mathcal{A}}$ is simply the set of all x_S such that $S \in \mathcal{A}$. For an access structure $\mathcal{A}$ to have an efficient secret sharing scheme, it must be the case that the language $L_{\mathcal{A}}$ is computable in polynomial time.

A major open question in information-theoretic cryptography is:

Q1: Characterize access structures with efficient secret-sharing schemes.

Indeed, this question has been widely studied [33, 9, 7, 22, 23], culminating with the result of Karchmer and Wigderson [23] who showed efficient secret sharing schemes for

[1] More formally, we have to speak of a family of access structures $\{\mathcal{A}_n\}_{n\in\mathbb{N}}$, one for every n. We abuse notation slightly and denote $\mathcal{A}$, consisting of subsets of n parties, as the access structure.

various log-space classes.[2] We refer the reader to Beimel's survey [4] for more details. In any event, it is wide open whether all of mP, the class of languages recognized by monotone polynomial-size circuits, has efficient secret sharing schemes.

Restricting the reconstruction algorithm to be a linear function of the shares gives us a special kind of secret-sharing scheme called a *linear secret-sharing scheme*. The Karchmer-Wigderson secret sharing scheme [23] for log-space classes is a linear secret-sharing scheme. We also know that linear and even the slightly more general quasi-linear schemes [23, 5] cannot exist for access structures outside NC, the class of languages computable by boolean circuits of polylogarithmic depth. Finally, Beimel and Ishai [5] showed *non-linear* secret-sharing schemes for two specific access structures associated to algebraic problems (related to computing quadratic residuosity and co-primality) which are in P but are believed not to be in NC.

We will also study secret-sharing schemes (which we call *semi-efficient*) where the dealer is efficient, namely runs in time polynomial in n, however the reconstruction algorithm need not be efficient. Aside from their theoretical interest, such secret-sharing schemes may find use in scenarios where sharing happens in the present (and thus has to be efficient) but reconstruction happens in a future where computational resources might be cheaper. This also justifies our desire to achieve information-theoretic (unconditional) security since not only the honest parties, but also the adversary gains more computational resources with time.

Beimel and Ishai [5] show a semi-efficient secret-sharing scheme for the language of quadratic residuosity modulo a composite, which is believed not to be in P. However, quite surprisingly, a characterization of access structures with semi-efficient secret-sharing schemes also appears to be open:

Q2: Characterize access structures with semi-efficient secret-sharing schemes.

As a parenthetical remark, we note that a different interpretation of *efficiency* is sometimes used in the secret-sharing literature. Namely, a secret-sharing scheme is termed *efficient* [24, 12, 10] if the total length of the n shares is polynomial in n.

[2] We use this as a short-hand to say "secret sharing schemes for access structures $\mathcal{A}$ whose associated language $L_{\mathcal{A}}$ can be recognized in log-space".

Let us call this notion *size efficiency*. This makes no reference to the complexity of either the sharing or the reconstruction algorithms. In this work, we use the strong interpretation of *efficient*, namely where both the sharing and reconstruction algorithms run in time $\mathsf{poly}(n)$ and that of *semi-efficient* where only the sharing algorithm needs to run in time $\mathsf{poly}(n)$. We note that either of these two notions is stronger than size efficiency.

It is against this backdrop that we begin our study. Our main contribution is to develop an *interactive proof lens* to study these questions. As concrete results of this connection, we obtain an almost-characterization of access structures with semi-efficent secret-sharing schemes (almost solving $Q2$), new combinatorial access structures conjectured to lie outside NC which have efficient secret-sharing schemes (extending [5]), and limitations on an ambitious notion of universally efficient secret-sharing. We describe our results in detail below.

1.1 Our Results

(All results stated here were originally published in [36].)

Our central tool is a special type of two-message interactive proof system (that we call *Special Interactive Proofs*). Roughly speaking, the restriction on the proof system for a language L (aside from the fact that it has two messages) is that the verifier uses a special procedure to accept or reject. In particular, the verifier V on input x and a uniformly random bit b, comes up with a message m to send to the prover. The prover wins (the verifier accepts) if he can guess the bit b, given m. If $x \in L$, the prover should have a distinguishing (and therefore an accepting) strategy. However, if $x \notin L$, the verifier messages for bits 0 and 1 should be statistically indistinguishable.

Before we proceed, we must clarify what it means to have a secret sharing scheme for a language L which is not necessarily monotone. We follow the approach of Beimel and Ishai [5] and define a (monotonized) access structure on $2n$ parties $\{P_{i,0}, P_{i,1}\}_{i\in[n]}$ associated with L (more precisely, $L \cap \{0,1\}^n$): for every i, the pair of parties $\{P_{i,0}, P_{i,1}\}$ is in the access structure, as is every set of parties $\{P_{1,x_1}, P_{2,x_2}, \ldots, P_{n,x_n}\}$

for all $x \in L$. These are the minimal sets that make up the access structure $\mathcal{A}_L$. Note that the complexity of deciding whether a set $S \in \mathcal{A}_L$ is precisely the complexity of deciding the language L.

Our research in this direction was motivated by the fact that if, for some language L, $\mathcal{A}_L$ has a semi-efficient secret sharing scheme, then L has a special interactive proof: the verifier simply shares a random bit b according to the sharing algorithm and sends the prover the shares corresponding to the input, and the prover has to guess b. The honest prover runs the reconstruction algorithm, and completeness and soundness are guaranteed by correctness and privacy of the secret sharing scheme, respectively. We then investigated the circumstances under which the converse might also hold. We were able to show the following:

Theorem 1.1 (Informal)**.** *Let L be a language and let $\mathcal{A}_L$ be the associated access structure. If L has a special interactive proof with a log-space verifier, then $\mathcal{A}_L$ has a semi-efficient secret-sharing scheme. Conversely, if $\mathcal{A}_L$ has a semi-efficient secret-sharing scheme, then L has a special interactive proof.*

Our proof goes through the notion of partial garbling schemes, defined and studied in the work of Ishai and Wee [21].

Characterizing Semi-Efficient Secret-Sharing. Using Theorem 1.1, we characterize access structures that have semi-efficient secret-sharing schemes: we show that all languages in $\mathsf{SZK}_{\mathbf{L}}$, the class of languages with statistical zero knowledge proof systems [32] where the verifier and simulator run in log-space, have semi-efficient secret-sharing schemes. This follows from the observation, using a result of Sahai and Vadhan [32], that L has a special interactive proof with a log-space verifier if and only if $L \in \mathsf{SZK}_{\mathbf{L}}$. Conversely, it is easy to see that if a language L has a semi-efficient secret-sharing scheme, then $L \in \mathsf{SZK}$, the class of languages with statistical zero knowledge proof systems with polynomial-time verifier and simulator. Together, this almost characterizes languages with semi-efficient secret-sharing schemes.

The class $\mathsf{SZK}_{\mathbf{L}}$, which is contained in SZK, and hence in $\mathsf{AM} \cap \mathbf{co}\mathsf{AM}$, contains several problems of both historical and contemporary significance to cryptography,

such as *Quadratic Residosity*, *Discrete Logarithm*, and the *Approximate Closest Vector Problem*, as well as other well-studied problems like *Graph Isomorphism*. For further details, including those about complete problems and about prospects of basing cryptography on the worst-case hardness of $\mathsf{SZK}_{\mathbf{L}}$, see [13]. As a result of these containments, our characterization captures as a special case the Beimel-Ishai secret-sharing scheme for the language of quadratic residuosity modulo composites [5].

We also show a version of this theorem for efficient (as opposed to semi-efficient) secret-sharing schemes. In particular:

Theorem 1.2 (Informal)**.** *Let L be a language and let $\mathcal{A}_L$ be the associated access structure. If L has a special interactive proof with a log-space verifier and a polynomial-time prover, then $\mathcal{A}_L$ has an efficient secret-sharing scheme. Conversely, if $\mathcal{A}_L$ has an efficient secret-sharing scheme, then L has a special interactive proof with a polynomial-time prover.*

Constructions of Efficient Secret-Sharing Schemes. We show new constructions of efficient secret-sharing schemes for languages that are in P but are not known to be in NC, namely Bounded-Degree Graph Isomorphism [29, 2], and lattice Shortest and Closest Vector problems in constant dimensions [27, 19]. Our constructions arise from special interactive proofs for these languages together with an application of Theorem 1.2. In particular, our construction for Bounded-Degree Graph Isomorphism gives us *the first* efficient secret-sharing scheme for a *combinatorial* access structure conjectured to be in $\mathsf{P} \setminus \mathsf{NC}$ (The results of Beimel and Ishai were for algebraic access structures associated to quadratic residuosity modulo primes and co-primality). Moreover, our interactive proofs and secret-sharing schemes are simple, natural and easy to describe.

Limitations on Universally Efficient Secret-Sharing Schemes. Consider secret sharing schemes that are defined not for a given access structure, but uniformly for some class of access structures. The sharing algorithm in such a case gets a description of the access structure, in the form of a circuit or a Turing machine that decides membership in the access structure. Typically, the sharing algorithm runs

for as much time as the Turing machine (and therefore as much time as required to decide membership). However, there is no a-priori reason why this should be the case. Indeed, one can reasonably require that the sharing algorithm runs in some fixed polynomial time $t(n)$, even though the access structure may take arbitrary polynomial time to decide. (We allow the reconstruction algorithm to run in arbitrary polynomial time to make up for the deficiency of the sharing algorithm). As a side-effect, the size of the shares is bounded by $t(n)$, independent of the complexity of deciding the language. Can such *succinct universally efficient* secret-sharing schemes exist?

Our definition is inspired by the recent progress on (computationally secure) succinct randomized encodings [8, 28, 11, 26]. Indeed, these works show, assuming indistinguishability obfuscation [3, 16], that P has *computationally secure* succinct randomized encoding schemes. One could also reasonably ask: Can such *succinct* randomized encodings exist unconditionally for all of P? It was observed in [8] that this cannot be the case under certain complexity-theoretic assumptions about speeding up non-deterministic algorithms.

Using our interactive proof characterization, we show that unconditionally secure succinct universally efficient secret-sharing schemes (and succinct randomized encodings) cannot exist for all languages in P, unless there is a fixed polynomial q such that $\mathsf{P} \subseteq \mathsf{DSPACE}(q(n))$ (the class of languages computable by a deterministic single-tape Turing machine with $q(n)$ space). We remind the reader that $\mathsf{P} \neq \mathsf{DSPACE}(q(n))$ for any fixed q, although non-containment either way is not known.

1.2 Related Work

In this work, we insist on statistical (or unconditional) security from our secret-sharing schemes. A number of works relax this to computational security and achieve stronger positive results. Settling for computational security and assuming the existence of one-way functions, Yao [40] and [37] showed an efficient secret-sharing scheme for all monotone languages in P recognized by polynomial-sized monotone circuits. We mention that even here, we are far from a characterization as there are monotone

languages in P that cannot be recognized by polynomial-sized monotone circuits [31, 34].

Komargodski, Naor and Yogev [25] also exploit the relaxation to computational security, and show secret-sharing schemes for all of monotone NP, where the sharing algorithm is polynomial-time, and the reconstruction algorithm is polynomial-time given the NP witness. Their result relies on strong computational assumptions related to indistinguishability obfuscation [3, 16].

1.3 Organisation

Chapter 2 contains definitions of various objects and some simple lemmas about secret sharing, randomised encodings, and statistical zero knowledge that will be useful in the remainder of the presentation. The main theorem (Theorem 3.1) is stated and proved in Chapter 3. Some constructions of efficient secret sharing schemes for various access structures, as implied by our main theorem, are presented in Chapter 4. Chapter 5 talks about the implausibility of a strong notion of uniform secret sharing. Chapter 6 points out some open problems relevant to the current study that follow from our and others' work. Proofs of some lemmas that were deemed too long to include in the main text may be found in the appendices.

Chapter 2

Preliminaries and Definitions

Notation. Given a set S, we denote by 2^S the set of all subsets of S. Let $A = (a_1, \ldots, a_n)$ and $B = \{i_1, \ldots, i_m\} \subseteq [n]$; A_B is used to denote the tuple $(a_{i_1}, \ldots, a_{i_m})$.

We use languages and Boolean functions interchangeably. Given a language L, we overload L to also denote the corresponding Boolean function, namely, $L(x) = 0$ if $x \notin L$ and $L(x) = 1$ otherwise. Given a randomized algorithm A, we denote by $A(x)$ the random variable arising from running A on x, and by $A(x; r)$ the output when A is run on x with randomness r.

Given a distribution D over a finite set X and an $x \in X$, we denote by $D(x)$ the probability mass D places on x, and for a subset $S \subseteq X$, $D(S) = \sum_{x \in S} D(x)$. $x \leftarrow D$ indicates that x is a sample drawn according to the distribution D. For a set S, $x \leftarrow S$ indicates that x is drawn uniformly at random from S.

We use the notion of statistical distance (also called total variation distance or ℓ_1 distance) between distributions, defined as follows.

Definition 2.1 (Statistical Distance)**.** The *statistical distance* between two distributions D_1 and D_2 over the domain X is defined as

$$d(D_1, D_2) = \frac{1}{2} \sum_{x \in X} |D_1(x) - D_2(x)| = \max_{S \subseteq X} (D_1(S) - D_2(S))$$

Of particular interest to us is the following relationship of statistical distance to the

advantage of any unbounded procedure in distinguishing between two distributions given a uniform prior.

Fact 2.1. *Given distributions D_1, D_2 over a domain X, for functions $f : X \to \{0,1\}$, we have:*

$$\max_f Pr[f(x) = b : \ b \leftarrow \{0,1\}, x \leftarrow D_b] = \frac{1}{2} + \frac{d(D_1, D_2)}{2}$$

2.1 Complexity Classes

Before anything else, we shall briefly define the following complexity classes that are referred to occasionally in the rest of the document. To start with, P is the class of languages decidable in deterministic polynomial time.

Definition 2.2 (P)**.** P is the class of languages L for which there exists a deterministic polynomial-time Turing machine M such that for any input x, $x \in L \Leftrightarrow M(x) = 1$.

NC^k is the class of languages decidable by circuits of depth $O((\log n)^k)$. A language is in NC if it is in NC^k for some k.

Definition 2.3 (NC^k)**.** For any $k \in \mathbb{N} \cup \{0\}$, NC^k is the class of languages L for which there exists a family of boolean circuits $\{C_n\}_{n \in \mathbb{N}}$ such that:

- There is a constant c such that for all n, C_n has depth at most $c(\log n)^k$.
- For any input x of length n, $x \in L \Leftrightarrow C_n(x) = 1$

BPP is the class of languages decidable by probabilistic polynomial-time Turing machines. Note that in the below definition, the constants $\frac{2}{3}$ and $\frac{1}{3}$ may be improved to $1 - 2^{-n}$ and 2^{-n}, respectively, by repetition.

Definition 2.4 (BPP)**.** BPP is the class of languages L for which there exists a probabilistic polynomial-time Turing machine M such that for any input x:

- $x \in L \implies \Pr[M(x) = 1] \geq \frac{2}{3}$
- $x \notin L \implies \Pr[M(x) = 1] \leq \frac{1}{3}$

$\mathsf{DSPACE}(p(n))$ is the class of languages decidable by deterministic Turing machines running with space $p(n)$.

Definition 2.5 (DSPACE)**.** For any positive-valued function p, $\mathsf{DSPACE}(p(n))$ is the class of languages L for which there exists a deterministic Turing machine L such that for any input x:

- $x \in L \Leftrightarrow M(x) = 1$
- M uses at most $p(|x|)$ cells on its work tape.

And finally, SZK consists of languages that have Statistical Zero Knowledge (SZK) proofs, which are interactive proofs with some additional properties, as described below.

Definition 2.6 (SZK)**.** A language L is in SZK if there exist a tuple of Turing machines (P, V, S), where the *verifier* V and *simulator* S run in probabilistic polynomial time, satisfying the following:

- (P, V) is an interactive proof for L with negligible completeness and soundness errors.
- Let $(P, V)(x)$ denote the distribution of transcripts of the interaction between P and V on input x. For any $x \in L$ of large enough size,

$$d(S(x), (P, V)(x)) \leq negl(|x|)$$

The above is actually a definition of honest-verifier Statistical Zero Knowledge, but we know from [32] that any language with an honest-verifier SZK proof also has an SZK proof against cheating verifiers. So this follows as a definition of SZK as well. Refer [35] for more extensive definitions and explanations.

$\mathsf{SZK_L}$ is the same as SZK, but with the verifier and simulator running with only logarithmic space. In this case too the above definition is only for honest verifiers, but as this would only define a larger class, and we show positive results for this class, we will work with this definition.

2.2 Secret Sharing

Definition 2.7 (Access Structure)**.** Given a set of parties $P = \{P_1, \ldots, P_n\}$, an *access structure* $\mathcal{A}$ is a monotone collection of subsets of P. That is, if $S \in \mathcal{A}$ and $T \supseteq S$, then $T \in \mathcal{A}$.

In the context of a secret-sharing scheme, the access structure consists of all subsets of parties that are allowed to reconstruct a secret shared among them. Of course, as the access structure is monotone, it suffices to specify its minimal elements. Along the lines of [5], we associate with every language L an family of access structures $\{\mathcal{A}_{L,n}\}_{n\in\mathbb{N}}$ where $\mathcal{A}_{L,n}$ is defined for $2n$ parties. As will be evident from the definition below, the complexity of deciding whether a set $S \in \mathcal{A}_{L,n}$ is exactly the hardness of deciding the language.

Definition 2.8 (Access Structure associated with Language L)**.** For a language L, its *associated access structure* $\mathcal{A}_{L,n}$ for $2n$ parties $\mathcal{P}_n = \{P_{i,b}\}_{i\in[n],b\in\{0,1\}}$ is defined by the following minimal elements:

- $\forall i : \{P_{i,0}, P_{i,1}\} \in \mathcal{A}_{L,n}$
- $\forall x \in L \cap \{0,1\}^n : \{P_{1,x_1}, \ldots, P_{n,x_n}\} \in \mathcal{A}_{L,n}$

We use the following definition of secret sharing schemes.

Definition 2.9 (Statistical Secret Sharing)**.** An (ϵ, δ)-*Secret Sharing Scheme* for n parties $\mathcal{P} = \{P_1, \ldots, P_n\}$ and a domain of secrets D under access structure $\mathcal{A} \subseteq 2^{\mathcal{P}}$ is a pair of algorithms (S, R), where

- S is the randomized sharing algorithm that takes as input a secret $s \in D$ and outputs a sequence of shares $(s_1, s_2, \ldots, s_n)$; and
- R is the deterministic reconstruction algorithm that takes as input a subset of parties $B \subseteq [n]$ and the corresponding subset of shares $(s_i)_{i\in B}$ and outputs either a secret s or a special symbol $\perp$.

We require (S, R) to satisfy the following conditions:

1. *Correctness*: For any $B \in \mathcal{A}$ and any $s \in D$, the reconstruction algorithm R works: $\Pr[R(B, S(s)_B) = s] \geq 1 - \epsilon(n)$

2. *Privacy*: For any $B \notin \mathcal{A}$ and any $s, s' \in D$: $d(S(s)_B, S(s')_B) \leq \delta(n)$.

The scheme is said to be *semi-efficient* if S is computable in $poly(n)$ time, and it is said to be *efficient* if both S and R are computable in $poly(n)$ time.

Unless otherwise specified, the domain of secrets for all schemes we talk about in this work shall be $\{0, 1\}$, which is without loss of generality.

Remark 2.1. *When we talk about access structures associated with promise problems, we require no guarantees from a secret sharing scheme for sets corresponding to inputs that do not satisfy the promise (even though technically they are not part of the associated access structure, and so privacy would otherwise be expected to hold).*

While much of the literature on secret sharing schemes studies the size of the shares (and call schemes that produce shares of size $\mathsf{poly}(n)$ efficient), we use a stronger interpretation of efficiency. Namely, in all our exposition, the sharing algorithm S is required to run in time polynomial in n. Thus, we will not discuss the sizes of the shares produced by the schemes, which is always $\mathsf{poly}(n)$.

2.3 Partial Randomized Encodings

We use the notion of partial randomized encodings (defined as *partial garbling schemes* in [21]). They are essentially randomized encodings [20] where part of the input is allowed to be public.

Definition 2.10 (Partial Randomized Encodings)**.** An (ϵ, δ)*-partial randomized encoding (PRE)* of a (bi-variate) function $f : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ is a pair of (randomized) functions (E_f, D_f), called the encoding and decoding functions, respectively, that satisfy the following conditions for all n, n':

1. *Correctness*: $\forall (x,z) \in \{0,1\}^n \times \{0,1\}^{n'}$:

$$\Pr\left[D_f(x, E_f(x,z)) = f(x,z)\right] \geq 1 - \epsilon(n)$$

Note that the decoder gets the first half of the input, namely the public part x, in addition to the randomized encoding $E_f(x,z)$.

2. *Privacy*: $\forall x \in \{0,1\}^n$ and $\forall z_1, z_2 \in \{0,1\}^{n'}$:

$$f(x,z_1) = f(x,z_2) \implies d(E_f(x,z_1), E_f(x,z_2)) \leq \delta(n)$$

Furthermore:

- (E_f, D_f) is *local* (or *locally computable*) if E_f can be decomposed into a set of functions $\{E_f^{(i)}(x_i, z)\}_{i \in [|x|]}$, where $E_f^{(i)}$ depends only on the ith bit of x and on z.
- (E_f, D_f) is *perfect* if $\epsilon(n) = \delta(n) = 0$.
- (E_f, D_f) is said to be *semi-efficient* if E_f is computable in $poly(|x|, |z|)$ time, and it is said to be *efficient* if both E_f and D_f are computable in $poly(|x|, |z|)$ time.

We can extend the above definition to PREs of randomized functions in a natural way. Namely, to construct an (ϵ, δ)-PRE for a randomized function $A(x, z; r)$, simply construct an (ϵ, δ)-PRE $(E_{A'}, D_{A'})$ for the deterministic function $A'(x, (z, r)) = A(x, z; r)$, and let $E_A(x, z)$ be the random variable $E_{A'}(x, (z, r))$ when r is chosen uniformly at random, and have D_A be the same as $D_{A'}$. We then have the following lemma, whose proof is in Appendix A.

Lemma 2.2. *Let $A(x,z)$ be a randomized function, and (E_A, D_A) be an (ϵ, δ)-PRE of A as described above. Then, for any x and any z_1, z_2:*

$$d(A(x,z_1), A(x,z_2)) \leq \delta' \implies d(E_A(x,z_1), E_A(x,z_2)) \leq \delta(|x|) + \delta'$$

We also use the following lemma.

Lemma 2.3 ([1, 21])**.** *Every function* $f : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ *that can be computed in* $\mathsf{L/poly}$ *has efficient perfect locally computable PREs, with encoding in* NC^0 *and decoding in* NC^2*.*

Finally, we abuse notation slightly and define partial randomized encodings for languages (boolean functions) a bit differently, for somewhat technical reasons (instead of calling this object something different).

Definition 2.11 (PREs for languages)**.** An (ϵ, δ)*-partial randomised encoding (PRE)* of a language $L \subseteq \{0,1\}^*$ is a pair of (randomised) functions (E_L, D_L), called the encoding and decoding functions, respectively, that satisfy the following conditions:

1. *Correctness*: $\forall x \in L$ and $b \in \{0,1\}$: $\Pr[D_L(x, E_L(x,b)) = b] \geq 1 - \epsilon(|x|)$.

2. *Privacy*: $\forall x \notin L$, $d(E_L(x,0), E_L(x,1)) \leq \delta(|x|)$.

Semi-efficiency, efficiency and locality are defined as for general partial randomised encodings.

2.4 Special Interactive Proofs

We define a special type of interactive proof system with two messages. Roughly speaking, the restriction on the proof system (aside from the fact that it has two messages) is that the verifier uses a special procedure to accept or reject. In particular, the verifier V on input x and a uniformly random bit b, comes up with a message m to send to the prover. The prover wins if he can guess the bit b, given m.

Definition 2.12 (Special Interactive Proof)**.** An (ϵ, δ)*-Special Interactive Proof (SIP)* for a language L is a pair (P, V), where:

1. V is a PPT algorithm that takes as input an instance x and a bit b, and outputs a message m; and

2. P takes as input the instance x and the verifier message m, and outputs a bit b'.

We require (P, V) to satisfy the following conditions, when $b \leftarrow \{0,1\}$:

1. *Completeness*: $\forall x \in L$, $\Pr[P(x, V(x,b)) = b] \geq 1 - \epsilon(|x|)$.

2. *Soundness*: $\forall x \notin L$, and for any P^*, $\Pr[P^*(x, V(x,b)) = b] \leq 1/2 + \delta(|x|)$.

While the restrictions imposed on these proofs seem rather severe, they turn out to be quite general. In fact, it follows from the work of Sahai and Vadhan [32] that the set of languages with such proofs is exactly the class SZK. See Theorem 2.5.

2.5 Statistical Zero Knowledge

Recall that the class SZK is the set of languages that have statistical zero-knowledge proofs, and the class $\mathsf{SZK_L}$ is set of languages that have statistical zero-knowledge proofs where the verifier and the simulator (for a statistically close simulation) both run in log-space.

Definition 2.13 (Promise Problems SD, SD_L)**.** The promise problem (ϵ, δ)-*Statistical Difference* (SD) is defined by the following YES and NO instances:

$$SD^{YES} = \{(M_1, M_2, 1^n) : d(M_1^n, M_2^n) > 1 - \epsilon(n)\}$$
$$SD^{NO} = \{(M_1, M_2, 1^n) : d(M_1^n, M_2^n) < \delta(n)\}$$

where M_1, M_2 are deterministic Turing machines, and M_1^n, M_2^n represent the random variables corresponding to their outputs when the input is distributed uniformly at random in $\{0,1\}^n$.

If M_1 and M_2 are log-space machines, then the language is called (ϵ, δ)-*Statistical Difference for Log-space Machines, or simply* SD_L.

Theorem 2.4 ([32])**.** *For every* $\epsilon(n), \delta(n) = 2^{-n^{O(1)}}$ *such that* $\delta(n) < (1 - \epsilon(n))^2$*, the* (ϵ, δ)*-SD problem is complete for* SZK*, and the* (ϵ, δ)*-*SD_L *problem is complete for* $\mathsf{SZK_L}$.

We will use the following theorem which is a slightly stronger version of Theorem 2.4. We describe the proof (which follows from the proof of Theorem 2.4 in [32]) in Appendix B for completeness.

Theorem 2.5 ([32]). *There exist negligible functions $\epsilon(n), \delta(n) = n^{-\omega(1)}$ such that for any language $L \in \mathsf{SZK}$, L has an (ϵ, δ)-special interactive proof system (P, V). Furthermore, if $L \in \mathsf{SZK_L}$, then the verifier V can be computed in log-space.*

Proof Sketch. For the main statement, we observe that the complete problem for SZK, namely (ϵ, δ)-SD, has a simple $(\epsilon/2, \delta/2)$-special interactive proof which works as follows.

- The verifier V, on input an instance $(M_0, M_1, 1^n)$ of the SD problem chooses a uniformly random bit b, and outputs a sample from M_b^n; and
- The prover's goal is to guess the bit b.

By Fact 2.1, it follows that the best success probability of any prover in this game is $\frac{1+d(M_0^n, M_1^n)}{2}$. By the completeness of SD (Theorem 2.4), we get that SZK has (ϵ, δ)-special interactive proofs for some $\epsilon(n), \delta(n) = n^{-\omega(1)}$.

The proof for $\mathsf{SZK_L}$ works in exactly the same way, except it is now a concern that the verifier has to first run the SZK-completeness reduction to obtain an instance of the statistical distance problem SD_L, since it is not guaranteed that the reduction runs in log-space. However, we show that the Sahai-Vadhan reduction indeed does. We refer the reader to appendix B for more details. □

In fact, the connection between languages with special interactive proofs and SZK goes both ways. Namely,

Fact 2.6. *Let $(1 - 2\epsilon(n))^2 > 2\delta(n)$. If a language L has an (ϵ, δ)-SIP, then $L \in \mathsf{SZK}$.*

This is because deciding a language L that has an (ϵ, δ)-SIP (P, V) is the same as deciding whether $(V_{(x,0)}, V_{(x,1)}, 1^{|r(|x|)|}) \in (2\epsilon, 2\delta)$-$SD$, where $V_{(x,b)}(r) = V(x, b; r)$, and $(2\epsilon, 2\delta)$-SD is in SZK for ϵ and δ satisfying the given property.

Chapter 3

From Zero Knowledge to Secret Sharing and Back

In this chapter, we show tight connections between languages with special interactive proofs, partial randomized encodings (PRE), and secret sharing schemes. In particular, we show:

Theorem 3.1 (Main theorem)**.** *For any language L and parameters $\epsilon(n)$ and $\delta(n)$, the following three statements are equivalent:*

1. *There are parameters $\epsilon_1 = O(\epsilon)$ and $\delta_1 = O(\delta)$ such that L has an (ϵ_1, δ_1)-special interactive proof (P, V), where the verifier V has a semi-efficient, locally computable, (ϵ_1, δ_1)-PRE.*

2. *There are parameters $\epsilon_2 = O(\epsilon)$ and $\delta_2 = O(\delta)$ such that L has a semi-efficient, locally computable, (ϵ_2, δ_2)-PRE.*

3. *There are parameters $\epsilon_3 = O(\epsilon)$ and $\delta_3 = O(\delta)$ such that for all n, there is a semi-efficient (ϵ_3, δ_3)-secret sharing scheme under the access structure $\mathcal{A}_{L,n}$.*

We will prove Theorem 3.1 in Section 3.1, and here we state a number of interesting corollaries. The first two corollaries "almost" characterize the languages L whose associated access structure $\mathcal{A}_{L,n}$ (as defined in Definition 2.8) has a semi-efficient

secret-sharing scheme. Corollary 3.2 shows that any language in $\mathsf{SZK}_{\mathbf{L}}$ has a semi-efficient secret-sharing scheme. Corollary 3.3 shows that furthermore, if $\mathsf{P/poly}$ has semi-efficient, locally computable PREs, then any language in the entire class SZK has a semi-efficient secret-sharing scheme. Moreover, it also says that no language outside SZK has semi-efficient secret-sharing schemes, implying that our characterization is almost tight.

Corollary 3.2. *Let $\epsilon(n), \delta(n) = n^{-\omega(1)}$ be negligible functions. For any language $L \in \mathsf{SZK}_{\mathbf{L}}$, and for every n, there is a semi-efficient (ϵ, δ)-secret sharing scheme under the associated access structure $\mathcal{A}_{L,n}$.*

Proof. Theorem 2.5 asserts that for any $L \in \mathsf{SZK}_{\mathbf{L}}$, there is an (ϵ, δ)-special interactive proof (P, V) for some $\epsilon(n), \delta(n) = n^{-\omega(1)}$, where the verifier algorithm V can be computed in log-space. Therefore, by Lemma 2.3, V has an efficient (and not just semi-efficient) perfect, locally computable PRE. Applying Theorem 3.1 (in particular, that (1) $\Rightarrow$ (3)), there is a semi-efficient $(O(\epsilon), O(\delta))$-secret sharing scheme for $\mathcal{A}_{L,n}$. □

Corollary 3.3. *Let $\epsilon(n), \delta(n) = n^{-\omega(1)}$ be negligible functions.*

- *Assume that $\mathsf{P/poly}$ has semi-efficient (ϵ, δ)-locally computable PREs. Then, for any language $L \in \mathsf{SZK}$, and for every n, there is a semi-efficient (ϵ, δ)-secret sharing scheme under the associated access structure $\mathcal{A}_{L,n}$.*
- *Conversely, if $\mathcal{A}_{L,n}$ has a semi-efficient (ϵ, δ)-secret sharing scheme, then $L \in \mathsf{SZK}$.*

This follows from the same arguments as corollary 3.2, but with the absence of something like lemma 2.3 to complete the argument. In fact, one may replace $\mathsf{P/poly}$ in corollary 3.3 with any complexity class $\mathbf{C}$ that is closed under the operations involved in the reduction used in the proof of theorem B.1 (while replacing SZK with the appropriate $\mathsf{SZK}_{\mathbf{C}}$). The converse is true because of Theorem 3.1 and fact 2.6

We also have the following theorem about *efficient* secret sharing schemes, where both the sharing and reconstruction algorithms run in time polynomial in n. The dif-

ference from Theorem 3.1 is that here, we require the prover in the special interactive proof to be *efficient*, namely run in time polynomial in n. We view this theorem as an avenue to constructing efficient secret sharing schemes for languages L outside L: namely, to construct a secret-sharing scheme for $\mathcal{A}_{L,n}$, it suffices to construct special interactive proofs for L wherein the verifier algorithm can be computed in L.

The proof of Theorem 3.4 follows directly from that of Theorem 3.1.

Theorem 3.4. *For any language L and parameters $\epsilon(n)$ and $\delta(n)$, the following three statements are equivalent:*

1. *There are parameters $\epsilon_1 = O(\epsilon)$ and $\delta_1 = O(\delta)$ such that L has an (ϵ_1, δ_1)-special interactive proof (P, V), where* the prover algorithm is computable in polynomial time*, and the verifier V has an efficient, locally computable, (ϵ_1, δ_1)-PRE.*
2. *There are parameters $\epsilon_2 = O(\epsilon)$ and $\delta_2 = O(\delta)$ such that L has an efficient, locally computable, (ϵ_2, δ_2)-PRE.*
3. *There are parameters $\epsilon_3 = O(\epsilon)$ and $\delta_3 = O(\delta)$ such that for all n, there is an efficient (ϵ_3, δ_3)-secret sharing scheme under the access structure $\mathcal{A}_{L,n}$.*

3.1 Proof of the Main Theorem

We prove Theorem 3.1 by showing that (1) $\implies$ (2) $\implies$ (3) $\implies$ (1).

(1) $\implies$ (2). Let (P, V) be an (ϵ, δ)-special interactive proof for L, and let (E_V, D_V) be the hypothesized semi-efficient, locally computable (ϵ, δ)-PRE for V. The PRE for the language L works as follows:

- $E_L(x, b) = E_V(x, b)$
- $D_L(x, y) = P(x, D_V(x, y))$

We first show correctness. Let $x \in L$ and $b \in \{0, 1\}$. From the correctness of the PRE for the verifier algorithm V, we know that:

$$D_V(x, E_V(x, b)) = V(x, b)$$

with probability at least $1 - \epsilon$. Now, by the completeness of the special interactive proof, we know that:

$$P(x, V(x, b)) = b$$

with probability at least $1 - 2\epsilon$ (because this probability is at least $1 - \epsilon$ when b is chosen at random).

, and we have the following for any $b \in \{0, 1\}$:

$$\Pr_{b' \leftarrow \{0,1\}} [P(x, V(x, b')) = b'] = \frac{1}{2}\Pr [P(x, V(x, b)) = b] + \frac{1}{2}\Pr [P(x, V(x, 1 - b)) = 1 - b]$$

Putting these together, we have:

$$D_L(x, E_L(x, b)) = P\big(x, D_V(x, E_V(x, b))\big) = P\big(x, V(x, b)\big) = b$$

with probability at least $1 - 3\epsilon$.

Next, we turn to privacy. Let $x \notin L$. We will show that $E_L(x, 0)$ and $E_L(x, 1)$ are statistically close. First, note that by the δ-soundness of the special interactive proof, we know that the distributions $V(x, 0)$ and $V(x, 1)$ are $O(\delta)$-close. Now, by Lemma 2.2 and using the δ-privacy of the PRE scheme for V, this means that $E_V(x, 0)$ and $E_V(x, 1)$ are also $O(\delta)$-close. This demonstrates privacy of our PRE scheme for L.

Since E_L is the same as E_V, it is clear that if the PRE scheme (E_V, D_V) is locally computable, so is (E_L, D_L). Moreover, if (E_V, D_V) is semi-efficient, so is (E_L, D_L). Finally, if (E_V, D_V) is efficient and the prover P in the special interactive proof is computable in polynomial time, then (E_L, D_L) is also efficient.

(2) $\implies$ (3). This implication follows from the work of Ishai and Wee [21]. We provide a proof here for completeness.

Given a locally computable (ϵ, δ)-PRE (E_L, D_L) for a language L, let $\{E_L^{(i)}(x_i, b)\}_{i \in [n]}$ be the local decomposition of $E_L(x, b)$. The following is the secret sharing scheme (S, R) for the access structure $\mathcal{A}_{L,n}$:

- *Sharing*: Let $s \in \{0,1\}$ be the secret bit to be shared. $S(s)$ works as follows:

  1. For each i, pick $s_{i,0}, s_{i,1} \in \{0,1\}$ at random such that $s_{i,0} \oplus s_{i,1} = s$, and give $s_{i,b}$ to the party $P_{i,b}$.
  2. Select bits $\{s_0, \ldots, s_n\}$ at random such that $\bigoplus_{i=0}^{n} s_i = s$. For each $i \in [n]$, give s_i to both $P_{i,0}$ and $P_{i,1}$.
  3. Choose a random string r, compute $\psi_{i,b} \leftarrow E_L^{(i)}(b, s_0; r)$ for every $i \in [n]$ and $b \in \{0,1\}$, and give $\psi_{i,b}$ to party $P_{i,b}$.

- *Reconstruction*: Any authorized set $B \in \mathcal{A}_{L,n}$ reconstructs the secret as follows:

  – If B contains $P_{i,0}$ and $P_{i,1}$ for some i, the secret s can be retrieved as $s = s_{i,0} \oplus s_{i,1}$.

  – If not, then $B = \{P_{i,x_i}\}$ for some $x \in L$. This means that between them, the parties contain $E_L(x, s_0; r) = \{E_L^{(i)}(x_i, s_0; r)\}_{i \in [n]}$. Output

  $$D_L(x, E_L(x, s_0; r)) \oplus \bigoplus_{i \in [n]} s_i$$

  as the secret.

For correctness, note that there are two possible types of authorized sets B in $\mathcal{A}_{L,n}$. If the set B contains parties $P_{i,0}$ and $P_{i,1}$ for some i, they recover the secret as $s_{i,0} \oplus s_{i,1}$. If not, the authorized set contains the parties $P_{1,x_1}, \ldots, P_{n,x_n}$ for some $x = (x_1, x_2, \ldots, x_n) \in L$. By the correctness of the PRE scheme for L, we know that $D_L(x, E_L(x, s_0; r)) = s_0$ with probability at least $1 - \epsilon$. Thus, the recovered secret is

$$D_L(x, E_L(x, s_0; r)) \oplus \bigoplus_{i \in [n]} s_i = \bigoplus_{i \in \{0,1,\ldots,n\}} s_i = s$$

with probability at least $1 - \epsilon$.

For privacy, there are again two types of sets B that are not present in $\mathcal{A}_{L,n}$. If there is an i such that the set of parties B does not contain either of $P_{i,0}$ and $P_{i,1}$, then B's shares look completely random due to the absence of any information about

s_i. The other case is when $B = \{P_{i,x_i}\}$ for some $x \notin L$. In this case, $d(S(0)_B, S(1)_B)$ is exactly the distance between $E_L(x, 0)$ and $E_L(x, 1)$ due to how the s_i's are picked, which is at most δ by the privacy of the randomised encoding of L.

It is also easy to see from the definition of S and R that if (E_L, D_L) is semi-efficient, then so is (S, R); and the same if it is efficient.

(3) $\implies$ (1). Given an (ϵ, δ)-secret sharing scheme (S, R) for the access structure $\mathcal{A}_{L,n}$, we construct a special interactive proof (P, V) for L, as follows:

- The verifier V, on input x and a bit b, outputs $S(b)_{B_x}$, where $B_x = \{P_{i,x_i}\}$.
- The prover P on input x and the verifier message m, outputs $R(B_x, m)$, where $B_x = \{P_{i,x_i}\}$.

For completeness, we have that for any $x \in L$, when $b \leftarrow \{0, 1\}$,

$$\Pr\left[P(x, V(x, b)) = b\right] = \Pr\left[R(B_x, (S(b)_{B_x}) = b\right] \geq 1 - \epsilon$$

by the correctness of secret sharing scheme, as $B_x \in \mathcal{A}_{(L.n)}$.

For privacy, we have that for any $x \notin L$, when $b \leftarrow \{0, 1\}$, for any P^*,

$$\Pr\left[P^*(x, V(x, b)) = b\right] \leq \frac{1 + d(V(x, 0), V(x, 1))}{2} \leq \frac{1}{2} + \frac{\delta}{2}$$

by privacy of the secret sharing scheme, as $B_x \notin \mathcal{A}_{L,n}$.

V is a PPT algorithm if (S, R) is semi-efficient, and P is computable in polynomial time if (S, R) is efficient. Also, V is local because it can be split into the collection $\{V^{(i)}(x_i, b) = S(b)_{\{P_{i,x_i}\}}\}$, so it serves as its own semi-efficient locally computable PRE.

Chapter 4

Positive Results on Efficient Secret Sharing

In this chapter we present efficient secret sharing schemes for access structures associated with Bounded-Degree Graph Non-Isomorphism, Lattice Closest Vector in small dimensions, and Co-Primality. These are obtained by the application of Theorem 3.4 (in particular, the implication (1) $\implies$ (2) in the theorem).

Useful throughout this chapter is the fact that arithmetic over integers (and rational numbers) may be performed in NC^1 (see [39] for details).

4.1 Bounded-Degree Graph Non-Isomorphism

Notation. Given an upper triangular matrix $M \in \{0,1\}^{n \times n}$, denote by $G(M)$ the undirected graph whose adjacency matrix is $(M + M^T)$, and for a symmetric matrix M, the undirected graph whose adjacency matrix is M. The degree of a graph, $deg(G)$, is the maximum degree of any vertex in the graph. If G_1 and G_2 are isomorphic, we denote this as $G_1 \equiv G_2$.

Definition 4.1 (d-$BDGNI$)**.** *d-Bounded Degree Graph Non-Isomorphism* is the promise problem given by the following sets of YES and NO instances over pairs of upper tri-

angular matrices:

$$d\text{-}BDGNI^{YES} = \{(M_0, M_1) | G(M_0) \not\equiv G(M_1); deg(G(M_0)), deg(G(M_1)) \leq d\}$$
$$d\text{-}BDGNI^{NO} = \{(M_0, M_1) | G(M_0) \equiv G(M_1); deg(G(M_0)), deg(G(M_1)) \leq d\}$$

While Graph (Non-)Isomorphism is not known to be in P, there is a classical polynomial time algorithm known for $d\text{-}BDGNI$ due to Luks [29]. However, it appears to be a long open question whether $d\text{-}BDGNI$ is in NC (or even in **RNC**) [2].

Theorem 4.1. *For every constant d and every n, there is an efficient (perfect) secret sharing scheme for the access structure $\mathcal{A}_{d\text{-}BDGNI,n}$. The complexity of the reconstruction algorithm grows as $n^{O(d)}$, whereas sharing runs in time polynomial in n.*

Proof. We prove this by showing a special interactive proof for $d\text{-}BDGNI$ where the verifier runs in log-space (and therefore, has efficient perfect locally computable PREs) and the prover runs in polynomial time. This satisfies statement (1) in Theorem 3.4, and hence implies the existence of the required secret sharing scheme.

The SIP proof (P, V) works along the lines of the classical SZK proof for Graph Non-Isomorphism [18], as follows:

- The verifier $V((M_0, M_1), b)$, on input upper triangular matrices $M_0, M_1 \in \{0,1\}^{n \times n}$ and bit b, selects a random permutation matrix $P \in S_n$, and outputs $P(M_b + M_b^T)P^T$.

- The prover $P((M_0, M_1), M)$, checks whether $G(M) \equiv G(M_0)$. If so, it outputs 0, else 1.

Note that the operation $P(M + M^T)P^T$ is equivalent to permuting the vertices of the graph $G(M)$ by the permutation P.

Perfect completeness of this protocol follows from the fact that if $M_0 \not\equiv M_1$, then the verifier's output M will be such that $G(M)$ is isomorphic to exactly one of $G(M_0)$ and $G(M_1)$, and P can identify which by running the algorithm for $d\text{-}BDGNI$ [29].

The protocol is perfectly sound because if $M_0 \equiv M_1$, then the distribution of the verifier's output is the same whether $b = 0$ or 1, and P has probability exactly $1/2$ of guessing b correctly.

The complexity of the verifier V in the above protocol is that of selecting a random permutation and performing two matrix multiplications, both of which can be done in log-space. Hence by Lemma 2.3, V has efficient perfect locally computable PREs. The prover P is computable in polynomial time because all the prover does is run the (polynomial time) algorithm for $d\text{-}BDGNI$.

(That the running time of reconstruction algorithm of the resulting secret sharing scheme is $n^{O(d)}$ can be seen by tracing its dependence on the running time of the algorithm for $d\text{-}BDGNI$ - the one in [29] runs in time $n^{O(d)}$ - in the proof of Theorem 3.1.) □

4.2 Lattice Closest Vectors

Notation. For a full-rank (over $\mathbb{Q}$) matrix $B \in \mathbb{Z}^{d\times d}$, let $\Lambda(B)$ denote the integer lattice (of dimension d) whose basis is B, and $\mathcal{P}(B)$ denote the fundamental parallelepiped of the same lattice (the parallelepiped formed by the column vectors of B and the origin). We denote by $\mathcal{B}(y, \delta)$ the set of points in the ball of radius δ centered at the point y (note that as we work with discretised space and not with $\mathbb{R}^d$, the number of points in this set is finite).

Given full-rank matrix $B \in \mathbb{Z}^{d\times d}$, a vector $y \in \mathbb{Z}^d$, $\delta \in \mathbb{Z}^+$ and $\gamma \in [0, 1]$, the (decision version of the) gap closest vector problem in d dimensions ($GapCVP_{\gamma,d}$) asks whether the Euclidean distance of y from (any point in) $\Lambda(B)$ is at most $(\gamma\delta)$ or at least δ.

While classical algorithms due to Gauss, and Lenstra, Lenstra and Lovasz [27] show that for any d, $GapCVP_{\gamma,d}$ is in P for any γ, it is not known to be (and conjectured not to be) in NC. We are interested in the complement of this problem, as defined below.

Definition 4.2 (**coGapCVP**$_{\gamma,\mathbf{d}}$)**.** For any $d \in \mathbb{Z}^+$ and $\gamma \in [0,1]$, $coGapCVP_{\gamma,d}$ is the promise problem defined by the following YES and NO instances over triples (B, y, δ), where $B \in \mathbb{Z}^{d\times d}$ is full-rank over $\mathbb{Q}$, $y \in \mathbb{Z}^d$ and $\delta \in \mathbb{Z}^+$:

$$coGapCVP^{YES}_{\gamma,d} = \{(B, y, \delta) \mid \forall x \in \Lambda(B) : ||y - x|| > \delta\}$$
$$coGapCVP^{NO}_{\gamma,d} = \{(B, y, \delta) \mid \exists x \in \Lambda(B) : ||y - x|| \leq \gamma\delta\}$$

The following theorem asserts the existence of efficient secret sharing schemes under access structures associated with the above problem. A number of lemmas used in its proof may be found in Appendix C.

Theorem 4.2. *For every c, d, n, and any $\gamma = \left(1 - \Omega(\frac{1}{n^c})\right)$, there is an efficient $(o(1), o(1))$-secret sharing scheme under the access structure $\mathcal{A}_{coGapCVP_{\gamma,d},n}$.*

Proof. We prove this theorem by constructing a $(o(1), o(1))$-Special Interactive Proof for $coGapCVP_{\gamma,d}$ with a log-space verifier and a poly time prover. As the verifier is computable in log-space, it has efficient perfect locally computable PREs, by Lemma 2.3. The existence of such an SIP, along with Theorem 3.4, implies the efficient secret sharing schemes we need.

Our SIP is a slight modification of the protocol of Goldreich and Goldwasser [17]. Let A be the logspace program promised by Lemma C.2. The protocol is as follows:

- The verifier gets as input the instance (B, y, δ) and a bit b, and does the following:
  - It picks bits $b_1, \ldots, b_k$ such that $b_1 \oplus \cdots \oplus b_k = b$ (for k determined later in the proof).
  - For each $i \in [k]$, if $b_i = 0$, it picks $z'_i \leftarrow A(0, \delta/2, 1^n)$, else $z'_i \leftarrow A(y, \delta/2, 1^n)$.
  - For each i, it sets $z_i = z'_i \bmod \mathcal{P}(B)$.
  - It outputs $(z_1, \ldots, z_k)$.
- The prover gets as input (B, y, δ) and the verifier's output $(z_1, \ldots, z_k)$.

– For each $i \in [k]$, it checks if the distance of z_i from the lattice $\Lambda(B)$ is at most $\delta/2$.

– If so, it sets $b'_i = 0$, else $b'_i = 1$.

– It outputs $b' = b'_1 \oplus \cdots \oplus b'_k$ as its guess.

If y is δ-far from the lattice, all points in the set $\mathcal{B}(y, \delta/2)$ are more than $\delta/2$-far from the lattice. By Lemma C.2, except with probability $\frac{1}{2^n}$, this is also true of samples from $A(y, \delta/2, 1^n)$. Points from $A(0, \delta/2, 1^n)$, on the other hand, are always within $\delta/2$ from the lattice. These properties are not affected by reducing modulo $\mathcal{P}(B)$.

Hence, except with negligible probability, the prover can guess each b_i correctly by running the algorithm from [27]. By the union bound, the prover can in fact guess all b_i's, and hence b, correctly except with negligible probability, and hence the protocol is $(1 - o(1))$-complete.

If y is less than $\gamma\delta$ from the lattice, let x be the lattice point closest to y. By Lemma C.3,

$$d(\mathcal{B}(x, \delta/2), \mathcal{B}(y, \delta/2)) = 1 - c''(1 - \gamma)^d$$

for some constant c''. Using guarantees from Lemma C.2, we have:

$$\begin{aligned} d(A(x, \delta/2, 1^n), A(y, \delta/2, 1^n)) &\leq d(\mathcal{B}(x, \delta/2), \mathcal{B}(y, \delta/2)) + negl(n) \\ &\leq 1 - c'(1 - \gamma)^d \end{aligned}$$

for another constant c'.

On reducing modulo $\mathcal{P}(B)$, this distance does not increase, since the set of points in the intersection of any two sets remain in the intersection after reduction. Also, after the reduction, $A(x, \delta/2, 1^n)$ and $A(0, \delta/2, 1^n)$ are the same. Hence, for each

$i \in [k]$,

$$\begin{aligned} d(z_i|b_i = 0, z_i|b_i = 1) &= d(A(0, \delta/2, 1^n) \bmod \mathcal{P}(B), A(y, \delta/2, 1^n) \bmod \mathcal{P}(B)) \\ &\leq d(A(x, \delta/2, 1^n), A(y, \delta/2, 1^n)) \\ &\leq 1 - c'(1-\gamma)^d \end{aligned}$$

Considering what the verifier does - choosing $b_1, \ldots, b_k$ that XOR to b and selecting z_i's appropriately - Lemma C.4 tells us that:

$$\begin{aligned} d((z_1, \ldots, z_k)|b = 0, (z_1, \ldots, z_k)|b = 1) &= d(z_i|b_i = 0, z_i|b_i = 1)^k \\ &\leq (1 - c'(1-\gamma)^d)^k \end{aligned}$$

As long as $\gamma = \left(1 - \Omega(\frac{1}{n^c})\right)$ for some c, we can choose k as some $poly(n)$ to make this quantity negligible, meaning that the distributions of verifier messages when $b = 0$ and $b = 1$ are negligibly close. This means the prover cannot guess b with non-negligible probability, giving us the required $o(1)$-soundness.

The verifier here runs the logspace program from Lemma C.2 and the reduction modulo $\mathcal{P}(B)$ on its output, which can also be done in logspace by Lemma C.5. As a constant number of compositions of logspace programs still gives a logspace program, the verifier can be computed in logspace. The prover simply runs the algorithm from [27] several times, and is hence computable in polynomial time.

□

4.3 Co-primality

Efficient secret sharing schemes for non-co-primality and semi-efficient ones for quadratic non-residuosity were shown by Beimel and Ishai [5] as an illustration of the power of non-linear secret sharing schemes over linear ones. We note that these follow as implications of our Theorem 3.1 given the existence of SZK proofs for these languages with logspace verifiers (which are indeed known to exist).

We demonstrate here, as an example, the case of non-co-primality, which is in P, but again, as noted in [5], not known to be in NC.

Definition 4.3 (Non-co-primality)**.** The language *Non-co-primality (NCoP)* consists of pairs of positive integers that are not co-prime, represented as strings, that is,

$$NCoP = \{(u, v) \mid u, v \in \mathbb{Z}^+, gcd(u, v) > 1\}$$

Theorem 4.3 asserts the existence of statistically correct, statistically private efficient secret sharing schemes under the access structure associated with $NCoP$.

Theorem 4.3. *For every n, there is an efficient $(o(1), o(1))$-secret sharing scheme under the access structure $\mathcal{A}_{NCoP,n}$.*

Proof. Again, we prove this by demonstrating a $(o(1), o(1))$-SIP for Non-co-primality where the prover is efficient and the verifier has efficient perfect locally computable PREs. This implies what we need, by Theorem 3.4.

We denote by $|u|$ the length of the representation of u as a boolean string. Below, we assume $|u| \geq |v|$. The SIP proof (P, V) is roughly as follows, for some $m = \Theta(|u|)$:

- The verifier V takes as input (u, v) and a bit b.
  - If $b = 1$, it outputs m random multiples of u modulo v; that is, it picks m random numbers $\{r_i\}_{i\in[m]} \leftarrow \{0, 1\}^{|u|}$ and outputs $\{(r_i u)(mod\ v)\}$.
  - If $b = 0$, it outputs m random numbers in $[v]$.
- The prover P takes as input (u, v) and the verifiers message, which is a set of m numbers $\{a_i\}_{i\in[m]}$. If $gcd(\{a_i\}) = 1$, the prover outptus 0, else 1.

The above SIP is complete because if $gcd(u, v) > 1$, then if $b = 1$, all multiples of u modulo v will be divisible by $gcd(u, v)$, and the prover will always output 1, and if $b = 0$, with high probability the gcd of m random numbers in $[v]$ will be 1 and the prover will output 0. It is sound because when $gcd(u, v) = 1$, the distribution of multiples of u (drawn from a large enough range) modulo v is negligibly close to uniform, and the cases $b = 0$ and $b = 1$ are indistinguishable.

The verifier V is computable in L, as all it does is multiply n-bit numbers, and so has efficient perfect locally computable PREs, by Lemma 2.3. The prover is efficient, as all it has to do is compute the *gcd* of some numbers. □

Chapter 5

Negative Results on Universally Efficient Secret Sharing

In this chapter, we show that a natural strengthening of efficient secret-sharing, that we call *universally efficient secret-sharing*, cannot exist for all of P, if for every polynomial t, $\mathsf{P} \not\subseteq \mathsf{DSPACE}(t(n))$.

Notation. Below, by L we denote both a language in a class $\mathcal{C}$, and its standard representation as a member of this class, say, for example, as a Turing machine that decides the language in case $\mathcal{C} = \mathsf{P}$. For a function f that takes two arguments (as $f(x, y)$), by $f(x, \cdot)$, we denote f curried with x, that is, the function $g(y) = f(x, y)$; this extends naturally to the case where f takes more than two arguments.

Definition 5.1 (Universal Secret Sharing)**.** An (ϵ, δ)-*Universally Efficient Secret Sharing Scheme (USS)*, or simply a universal secret sharing scheme, for a class of languages $\mathcal{C}$ over a domain D is a pair of (randomized) algorithms (S, R) such that for any $L \in \mathcal{C}$ and any n, $(S(L, 1^n, \cdot), R(L, 1^n, \cdot, \cdot))$ is an (ϵ, δ)-secret sharing scheme under the access structure $A_{L,n}$ over the domain D.

For any polynomial t, a universal secret sharing scheme is said to be t-*semi-efficient* if for any $L \in \mathcal{C}$, $S(L, 1^n, \cdot)$ is computable in time $t(n)$. The scheme is said to be t-*efficient* if both $S(L, 1^n, \cdot)$ and $R(L, 1^n, \cdot, \cdot)$ are computable in time $t(n)$.

Theorem 5.1. *Let, for all n, $1 - \epsilon(n) > \delta(n)$. If a class of languages $\mathcal{C}$ has t-semi-efficient (ϵ, δ)-universal secret sharing (USS) schemes, then there exists t' such that $t'(n) = O(t(n))$ and $\mathcal{C} \subseteq \mathsf{DSPACE}(t'(n))$.*

Proof Sketch. Suppose (S, R) is a t-semi-efficient (ϵ, δ) USS scheme for the class $\mathcal{C}$. Theorem 5.1 follows from applying lemma 5.2 to each language $L \in \mathcal{C}$, using the fact that by definition, $(S(L, 1^n, \cdot), R(L, 1^n, \cdot, \cdot))$ is an (ϵ, δ)-secret sharing scheme for $A_{L,n}$ where the sharing algorithm runs in time $t(n)$. □

In particular, Theorem 5.1 implies that if P had a t-semi-efficient USS scheme, then it would be contained in $\mathsf{DSPACE}(t(n))$ for some polynomial $t(n)$.

Lemma 5.2. *Let, for all n, $1 - 3\epsilon(n) > 3\delta(n)$. If, for some language L, there is an (ϵ, δ)-secret sharing scheme (S, R) for $A_{L,n}$ for all n, where S runs in time $t(n)$, then $L \in \mathsf{DSPACE}(t'(n))$, where $t'(n) = O(t(n))$.*

The proof below is adapted from that of a more general statement from [15].

Proof. We start by using Theorem 3.1 to recognize the existence of an (ϵ', δ')-SIP (P, V) for L where V runs in time $t(n)$, where $\epsilon' = 3\epsilon$ and $\delta' = 3\delta$ (the constant 3 comes out of the proof of Theorem 3.1), and we have $1 - \epsilon'(n) > \delta'(n)$.

In order to decide whether $x \in L$, it is sufficient to determine whether any P' can guess b given $V(x, b)$ with probability $\geq (1 - \epsilon'(|x|))$ or only $\leq (1/2 + \delta'(|x|)/2)$. This is equivalent to whether $d(V(x, 0), V(x, 1))$ is $\geq (1 - \epsilon(|x|))$ or $\leq \delta(|x|)$. But $d(V(x, 0), V(x, 1))$ itself can be computed in space $O(t(|x|))$ as follows.

First, for any v of length at most $t(|x|)$, $\Pr_r[V(x, b; r) = v]$ can be computed by iterating over the possible values of r – note that $|r| \leq t(|x|)$– and simulating V to see if it outputs v, and counting the number of r's for which it does. This requires only $O(t(|x|))$ space because V can be simulated in this much space, and the count of r's is at most $2^{t(|x|)}$.

So for each v, we can also compute

$$p(v) := |\Pr_r[V(x, 0; r) = v] - \Pr_r[V(x, 1; r) = v]|$$

in $O(t(|x|))$ space. What we need is the sum $\left(\sum_{v:|v|\leq t(|x|)} p(v)\right)$. To compute this, we simply iterate over all the v's, storing at the end of each iteration only the sum $\left(\sum_{v':v'\leq v} p(v)\right)$. As each $p(v) \geq 2^{-t(|x|)}$, and the cumulative sum is at most 1, this adds at most $O(t(|x|))$ space to what is needed for each iteration. Hence, the entire computation of $d(V(x,0), V(x,1))$ can be done in space $t'(|x|) = O(t(|x|))$, and hence $L \in \mathsf{DSPACE}(t'(n))$. □

Chapter 6

Conclusion and Future Directions

In this work, we have shown close relationships between computationally efficient secret sharing and zero knowledge proofs. We showed how zero knowledge proofs with certain properties for a language may be used to obtain efficient secret sharing schemes for access structures related to that language, and noted that efficient secret sharing schemes in turn yield zero knowledge proofs. This was a step toward understanding the class of access structures that have such secret sharing schemes. There is a lot of ground to cover in this regard, and below we mention a few of the problems in this direction that demand to be studied, particularly in the wake of our work.

Notation. Let mSS denote the class of (monotone) languages corresponding to access structures that have semi-efficient secret sharing schemes, and SS denote the class of languages whose monotonized access structures have semi-efficient secret sharing schemes. Let $\mathsf{SS}^{\mathsf{eff}}$ and $\mathsf{mSS}^{\mathsf{eff}}$ denote the corresponding classes for efficient secret sharing schemes (that is, those that also have efficient reconstruction). Let $\mathsf{SZK}_{\mathsf{L}}^{\mathsf{eff}}$ denote the class of languages that have SZK proofs with logspace verifier and polynomial-time prover. Let mL denote the class of languages in L that can be computed by polynomial-sized monotone branching programs.

The following containments follow from our work (as described in the previous sections) and [23]:

$$\mathsf{SZK}_{\mathbf{L}} \subseteq \mathsf{SS} \subseteq \mathsf{SZK}$$

$$\mathsf{SZK}_{\mathsf{L}}^{\mathsf{eff}} \subseteq \mathsf{SS}^{\mathsf{eff}} \subseteq \mathsf{BPP}$$

$$\mathsf{mL} \subseteq \mathsf{mSS} \subseteq \mathsf{SZK}$$

$$\mathsf{mL} \subseteq \mathsf{mSS}^{\mathsf{eff}} \subseteq \mathsf{BPP}$$

A number of questions related to these classes remain open, including:

1. Can one non-trivially characterize mSS and $\mathsf{mSS}^{\mathsf{eff}}$?

   - The containment of mL in mSS and $\mathsf{mSS}^{\mathsf{eff}}$ was obtained with just linear secret sharing schemes in mind. Could considering non-linear secret sharing schemes help us find larger classes of monotone languages than mL whose access structures have (semi-)efficient secret sharing schemes?
   - Is there a better upper bound on either? At least, can one demonstrate monotone languages in SZK (resp. BPP) that do not have semi-efficient (resp. efficient) secret sharing schemes?

2. Can one better characterize SS and $\mathsf{SS}^{\mathsf{eff}}$?

   - Do these classes have natural complete problems?
   - Are they closed under operations like composition and complementation?
   - Is $\mathsf{SS} = \mathsf{SZK}$? Is $\mathsf{SS}^{\mathsf{eff}} = \mathsf{BPP}$?

A closely related topic to study would be that of the classes $\mathsf{SZK}_{\mathbf{L}}$ and $\mathsf{SZK}_{\mathsf{L}}^{\mathsf{eff}}$ (which, as mentioned earlier, are contained in SS and $\mathsf{SS}^{\mathsf{eff}}$ respectively). $\mathsf{SZK}_{\mathbf{L}}$ is notable for containing a number of problems whose hardness much of cryptography depends on, from quadratic residuosity to several lattice problems. This class has been studied in the past and is known to have complete problems similar to the statistical complete problems for SZK [14].

Regarding $\mathsf{SZK}_{\mathsf{L}}^{\mathsf{eff}}$, while there has been research into SZK proofs with efficient provers, this has been in slightly different settings - [6] showed that any language in SZK has an SZK proof with a polynomial-time prover that is given oracle access to an NP-hard problem, and [30] showed that given any language L in $\mathsf{SZK} \cap \mathsf{NP}$, there is an SZK proof for L where the prover runs in polynomial-time given the NP witness for the input. $\mathsf{SZK}_{\mathsf{L}}^{\mathsf{eff}}$, though, is about provers that are efficient without such aids, and not much is known in this case.

The following are a few questions to start an investigation into $\mathsf{SZK}_{\mathsf{L}}^{\mathsf{eff}}$ with:

1. Does $\mathsf{SZK}_{\mathsf{L}}^{\mathsf{eff}}$ have natural complete problems?
2. Does it matter whether the randomness used by the verifier is public or private?
3. Is it closed under complement?
4. How does it relate to BPP?

Bibliography

[1] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography in nc^0. *SIAM J. Comput.*, 36(4):845–888, 2006.

[2] Vikraman Arvind and Jacobo Torán. Isomorphism testing: Perspective and open problems. *Bulletin of the EATCS*, 86:66–84, 2005.

[3] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. *J. ACM*, 59(2):6, 2012.

[4] Amos Beimel. Secret-sharing schemes: A survey. In YeowMeng Chee, Zhenbo Guo, San Ling, Fengjing Shao, Yuansheng Tang, Huaxiong Wang, and Chaoping Xing, editors, *Coding and Cryptology*, volume 6639 of *Lecture Notes in Computer Science*, pages 11–46. Springer Berlin Heidelberg, 2011.

[5] Amos Beimel and Yuval Ishai. On the power of nonlinear secret-sharing. *IACR Cryptology ePrint Archive*, 2001:30, 2001.

[6] Mihir Bellare and Erez Petrank. Making zero-knowledge provers efficient. In *Proceedings of the Twenty-fourth Annual ACM Symposium on Theory of Computing*, STOC '92, pages 711–722, New York, NY, USA, 1992. ACM.

[7] Josh Cohen Benaloh and Jerry Leichter. Generalized secret sharing and monotone functions. In Shafi Goldwasser, editor, *Advances in Cryptology - CRYPTO '88*, volume 403 of *Lecture Notes in Computer Science*, pages 27–35. Springer, 1988.

[8] Nir Bitansky, Sanjam Garg, and Sidharth Telang. Succinct randomized encodings and their applications. *IACR Cryptology ePrint Archive*, 2014:771, 2014.

[9] George Blakley. Safeguarding cryptographic keys. *Proceedings of the National Computer Conference*, 48:313–âĂŞ317, 1979.

[10] Carlo Blundo, Alfredo De Santis, Roberto De Simone, and Ugo Vaccaro. Tight bounds on the information rate of secret sharing schemes. *Des. Codes Cryptography*, 11(2):107–122, 1997.

[11] Ran Canetti, Justin Holmgren, Abhishek Jain, and Vinod Vaikuntanathan. Indistinguishability obfuscation of iterated circuits and RAM programs. *IACR Cryptology ePrint Archive*, 2014:769, 2014.

[12] László Csirmaz. The size of a share must be large. *J. Cryptology*, 10(4):223–231, 1997.

[13] Zeev Dvir, Dan Gutfreund, Guy N. Rothblum, and Salil Vadhan. On approximating the entropy of polynomial mappings. In *In Proceedings of the 2nd Innovations in Computer Science Conference*, pages 460–475, 2011.

[14] Zeev Dvir, Dan Gutfreund, Guy N Rothblum, and Salil P Vadhan. On approximating the entropy of polynomial mappings. In *ICS*, pages 460–475, 2011.

[15] Lance Fortnow and Carsten Lund. Interactive proof systems and alternating time-space complexity. *Theor. Comput. Sci.*, 113(1):55–73, 1993.

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

[17] Oded Goldreich and Shafi Goldwasser. On the limits of non-approximability of lattice problems. In Jeffrey Scott Vitter, editor, *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998*, pages 1–9. ACM, 1998.

[18] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity and a methodology of cryptographic protocol design (extended abstract). In *27th Annual Symposium on Foundations of Computer Science, Toronto, Canada, 27-29 October 1986*, pages 174–187, 1986.

[19] Raymond Greenlaw, H. James Hoover, and Walter L. Ruzzo. *Limits to Parallel Computation: P-completeness Theory.* Oxford University Press, Inc., New York, NY, USA, 1995.

[20] Yuval Ishai and Eyal Kushilevitz. Randomizing polynomials: A new representation with applications to round-efficient secure computation. In *FOCS*, pages 294–304, 2000.

[21] Yuval Ishai and Hoeteck Wee. Partial garbling schemes and their applications. In *ICALP 2014*, pages 650–662, 2014.

[22] M. Ito, A. Saio, and Takao Nishizeki. Multiple assignment scheme for sharing secret. *J. Cryptology*, 6(1):15–20, 1993.

[23] Mauricio Karchmer and Avi Wigderson. On span programs. In *Proceedings of the Eigth Annual Structure in Complexity Theory Conference, San Diego, CA, USA, May 18-21, 1993*, pages 102–111. IEEE Computer Society, 1993.

[24] Ehud D. Karnin, J. W. Greene, and Martin E. Hellman. On secret sharing systems. *IEEE Transactions on Information Theory*, 29(1):35–41, 1983.

[25] Ilan Komargodski, Moni Naor, and Eylon Yogev. Secret-sharing for NP. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology - ASIACRYPT 2014*, volume 8874 of *Lecture Notes in Computer Science*, pages 254–273. Springer, 2014.

[26] Venkata Koppula, Allison Bishop Lewko, and Brent Waters. Indistinguishability obfuscation for turing machines with unbounded memory. *IACR Cryptology ePrint Archive*, 2014:925, 2014.

[27] A.K. Lenstra, Jr. Lenstra, H.W., and L. LovÃąsz. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534, 1982.

[28] Huijia Lin and Rafael Pass. Succinct garbling schemes and applications. *IACR Cryptology ePrint Archive*, 2014:766, 2014.

[29] Eugene M. Luks. Isomorphism of graphs of bounded valence can be tested in polynomial time. In *FOCS*, pages 42–49, 1980.

[30] Minh-Huyen Nguyen and Salil Vadhan. Zero knowledge with efficient provers. In *Proceedings of the Thirty-eighth Annual ACM Symposium on Theory of Computing*, STOC '06, pages 287–295, New York, NY, USA, 2006. ACM.

[31] Alexander Razborov. Lower bounds on the monotone complexity of some boolean functions, 1985.

[32] Amit Sahai and Salil P. Vadhan. A complete problem for statistical zero knowledge. *Electronic Colloquium on Computational Complexity (ECCC)*, 7(84), 2000.

[33] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

[34] Éva Tardos. The gap between monotone and non-monotone circuit complexity is exponential. *Combinatorica*, 8(1):141–142, 1988.

[35] Salil Vadhan. *A Study of Statistical Zero-Knowledge Proofs*. PhD thesis, Massachusetts Institute of Technology, 1999.

[36] Vinod Vaikuntanathan and Prashant Nalini Vasudevan. From statistical zero knowledge to secret sharing. *IACR Cryptology ePrint Archive*, 2015:281, 2015.

[37] V. Vinod, Arvind Narayanan, K. Srinathan, C. Pandu Rangan, and Kwangjo Kim. On the power of computational secret sharing. In *Progress in Cryptology - INDOCRYPT 2003*, pages 162–176, 2003.

[38] Arnold Walfisz. Über gitterpunkte in mehrdimensionalen kugeln iii. *Acta Arithmetica*, 6(2):193–215, 1960.

[39] Ingo Wegener. *The Complexity of Boolean Functions*. John Wiley & Sons, Inc., New York, NY, USA, 1987.

[40] Andrew Yao. Unpublished manuscript, 1989. Presented at Oberwolfach and DIMACS Workshops.

Appendix A

Proof of Lemma 2.2

In this section, we restate and prove lemma 2.2. This essential lemma extends the privacy properties of PREs to the case of PREs of randomised functions - while the original definition of PREs (for deterministic functions) states that if for some x, $f(x, z_1) = f(x, z_2)$, then $E_f(x, z_1)$ and $E_f(x, z_2)$ are statistically close, lemma 2.2 states that even for a randomised function g, if $g(x, z_1)$ and $g(x, z_2)$ are statistically close, then so are $E_g(x, z_1)$ and $E_g(x, z_2)$.

Note that PREs for randomised functions are defined as described in section 2: To construct an (ϵ, δ)-PRE for a randomized function $A(x, z; r)$, simply construct an (ϵ, δ)-PRE $(E_{A'}, D_{A'})$ for the deterministic function $A'(x, (z, r)) = A(x, z; r)$, and let $E_A(x, z)$ be the random variable $E_{A'}(x, (z, r))$ when r is chosen uniformly at random, and have D_A be the same as $D_{A'}$.

Lemma A.1. *Let $A(x, z)$ be a randomized function, and (E_A, D_A) be an (ϵ, δ)-PRE of A as described above. Then, for any x and any z_1, z_2:*

$$d(A(x, z_1), A(x, z_2)) \leq \delta' \implies d(E_A(x, z_1), E_A(x, z_2)) \leq \delta(|x|) + \delta'$$

Proof. As above, consider the deterministic function $A'(x, (z, r)) = A(x, z; r)$. By definition, $d(E_A(x, z_1), E_A(x, z_2)) = d(E_{A'}(x, (z_1, r)), E_{A'}(x, (z_2, r)))$, which is given

by:

$$\sum_{\hat{v}} |\Pr\left[E_{A'}(x,(z_1,r)) = \hat{v}\right] - \Pr\left[E_{A'}(x,(z_2,r)) = \hat{v}\right]|$$

where r is distributed uniformly over its domain. We wish to prove that this expression is small. From the privacy of PREs, we have promises on the behaviour of $E_{A'}$ on inputs for which A' has the same output value. Towards exploiting this, we expand the above expression, conditioning on possible values of A' to get:

$$\sum_{\hat{v}} \Bigg| \sum_{v} \Pr\left[A'(x,(z_1,r)) = v\right] \Pr\left[E_{A'}(x,(z_1,r)) = \hat{v} \Big| A'(x,(z_1,r)) = v\right] - \sum_{v} \Pr\left[A'(x,(z_2,r)) = v\right] \Pr\left[E_{A'}(x,(z_2,r)) = \hat{v} \Big| A'(x,(z_2,r)) = v\right] \Bigg|$$

For the same reason - so that we may compare $E_{A'}$ on points where A' has the same output value - we add and subtract $(\sum_v \Pr\left[A'(x,(z_1,r)) = v\right])$ to the factor in the second term above and use the triangle inequality to say that what we have is at most:

$$\begin{aligned} \sum_{v} \Pr\left[A'(x,(z_1,r)) = v\right] \Big(\sum_{\hat{v}} \Big| & \Pr\left[E_{A'}(x,(z_1,r)) = \hat{v} \Big| A'(x,(z_1,r)) = v\right] \\ & - \Pr\left[E_{A'}(x,(z_2,r)) = \hat{v} \Big| A'(x,(z_2,r)) = v\right] \Big| \Big) \\ + \sum_{v} \sum_{\hat{v}} & \Pr\left[E_A(x,(z_2,r)) = \hat{v} \Big| A(x,(z_2,r)) = v\right] \cdot \\ & |\Pr\left[A(x,(z_1,r)) = v\right] - \Pr\left[A(x,(z_2,r)) = v\right]| \end{aligned}$$

The first summand above is a convex combination of several terms, each of which is at most $\delta(|x|)$ by the privacy guarantee of $E_{A'}$ (as each of these terms is some convex combination of the distance between $E_{A'}$ on input values for which A' produces the

same output). The second summand is simply equal to $d(A'(x, z_1), A'(x, z_2)) = \delta'$. Hence the whole thing is at most $(\delta(|x|) + \delta')$, which is what we wanted to prove.

□

Appendix B

A Refined Completeness Theorem for $\mathsf{SZK_L}$

In this section, we complete the proof sketch of Theorem 2.5. In order to do so, we shall first demonstrate Lemma B.1.

Lemma B.1 ([32]). *There exist negligible functions $\epsilon(n), \delta(n) = n^{-\omega(1)}$ such that every language L in $\mathsf{SZK_L}$ reduces to (ϵ, δ)-SD_L. Furthermore, there is a logspace program D_L such that, if an instance x is mapped to the instance (C_0, C_1) by the above reduction, $D_L(b, x, r) = C_b(r)$.*

Given D_L from Lemma B.1 for a language $L \in \mathsf{SZK_L}$, we can prove Theorem 2.5 by constructing a special interactive proof (P, V) for L as follows:

- $V(x, b; r) = D_L(b, x, r)$
- $P(x, m)$ outputs 0 if $\Pr[D_L(0, x, r) = m] > \Pr[D(1, x, r) = m]$, and 1 otherwise.

Note that the above is an $(\epsilon/2, \delta/2)$-SIP proof for L where the verifier can be computed in logspace.

We shall now sketch a proof of Lemma B.1, for which we shall need the following amplification lemma for statistical distance of distributions.

Lemma B.2 (Polarisation Lemma, [32]). *Let $\alpha, \beta \in [0, 1]$ be constants such that $\alpha^2 > \beta$. Given two logspace machines $X_0, X_1 : \{0,1\}^n \to \{0,1\}^m$, there are logspace*

machines $Y_0, Y_1 : \{0,1\}^{n'} \to \{0,1\}^{m'}$ *(where* n'*,* m' *grow polynomially with* n*,* m*) that use* X_0, X_1 *only as blackboxes such that:*

$$d(X_0, X_1) \geq \alpha \implies d(Y_0, Y_1) \geq 1 - 2^{-n'}$$
$$d(X_0, X_1) \leq \beta \implies d(Y_0, Y_1) \leq 2^{-n'}$$

Both the above lemmas are not stated in precisely this manner in either [32] or [35], but these extensions follow easily from the proofs of statements that are indeed made in these works.

Proof Sketch. of Lemma B.1 (The lemma follows directly from the proof of completeness of SD for SZK presented in [35], noticing that the reduction from any $L \in SZK$ to SD, outlined below, leads to logspace machines if one starts with an $L \in SZKL$, as L has a logspace simulator.)

Suppose L has an SZK proof (P, V) in which, on inputs of length n, the total communication is $t(n)$ over $v(n)$ messages, V uses $r(n)$ bits of randomness, and there is a logspace simulator S that achieves deviation $\mu(n) \leq 1/(Ct(n)^2)$, for some constant C to be determined. Let S_i denote the distribution of the output of S (on a given input) truncated to the first i rounds. We assume, without loss of generality, that the prover speaks first, messages alternate, and that the last message of the verifier consists of all its randomness. We shall describe now distributions that witness the reduction of L to SD_L. Proofs and further details may be found in [35], chapter 3.

Define the following distributions:

$X : S_2 \otimes S_4 \otimes \cdots \otimes S_{2v}$

$Y_1 : S_1 \otimes S_3 \otimes \cdots \otimes S_{2v-1} \otimes U_{r-7}$

$Y_2 :$ Run S $8ln(tv+2)$ times, and if the transcript is rejecting in a majority of these, output U_{tv+2}, else output nothing.

$Y : Y_1 \otimes Y_2$

We may arrange, again without loss of generality, for a given input length n of L, for both X and Y to use at most m' bits of randomness and have output length n'. Let $q = 9km'^2$ for some constant k to be determined later.

Let $X'' = \otimes^q X$ and $Y'' = \otimes^q Y$, and m'' and n'' be the (upper bound on) number of bits of randomness used and output length of X'' and Y''. Let $H = H_{m''+n'',m''}$ be a family of 2-universal hash functions from $\{0,1\}^{m''+n''} \to \{0,1\}^{m''}$. Define now the following distributions:

A : Choose $r \leftarrow \{0,1\}^{m''}, h \leftarrow H, y \leftarrow Y$, let $x = X''(r)$. Output $(x, h, h(r, y))$.

B : Choose $x \leftarrow X'', h \leftarrow H, z \leftarrow \{0,1\}^{m''}$. Output (x, h, z).

As proven in [35], if $x \in L$, then $d(A, B) \geq 1 - O(2^{-k})$, and if $x \notin L$, $d(A, B) \leq 2^{-\Omega(k)}$. Note that all steps involved so far, including evaluating the hash function, may be done in logspace, meaning that there is a randomised logspace program that on input x can sample A (or B).

This lets us apply Lemma B.2 to (A, B) to get distributions (A', B') which are still sampleable in logspace given x (as they are logspace programs that only use the samplers for A and B as blackboxes), and are such that if $x \in L$, $d(A', B') \geq 1 - 2^{-r}$ and if $x \notin L$, $d(A', B') \leq 2^{-r}$, where r (a polynomial in $|x|$ and $\Omega(|x|)$) is the amount of randomness used by the sampler for A' (or B'). This gives us the reduction to SD_L.

We now define D_L to simply emulate the above steps. On input (b, x, r), where $|r|$ is a function of $|x|$ resulting from above operations (D_L is undefined on input lengths that do not obey this relation between $|x|$ and $|r|$), if $b = 0$, D_L runs the logspace sampler for A' with input x and randomness r, and simliarly the sampler for B' if $b = 1$. Note that D_L is still in logspace, and that if $x \in L$, $d(D_L(0, x, r), D_L(1, x, r)) \geq 1 - 2^{-|x|}$, and if $x \notin L$, $d(D_L(0, x, r), D_L(1, x, r)) \leq 2^{-|x|}$. □

Appendix C

Lemmas for Section 4.2

In this section we prove several facts that are used in the proof of theorem 4.2 concerning the Gap Closest Vector Problem, which were stated in appendix C.

An issue that needs to be addressed at the onset of a discussion of a problem such as $GapCVP$ is that of precision, that is, how many bits will be used to represent numbers and vectors. Arguments and algorithms that are guided by intuition from the behaviour of real quantities tend to work only when the set of numbers representable is dense enough. But one is also constrained by being unable to use more than a polynomial number of bits of precision for efficiency.

Below, as in section 4.2, for any d-dimensional vector y and $r \in \mathbb{Q}^+$, we denote by $\mathcal{B}(y, r)$ the set of points in the ball of radius r centered at y *that are representable* using whatever scheme it is that we use to represent vectors. In general, this will be as a tuple of rational numbers, but in turn only those rational numbers that are representable given the number of bits of precision we use. By l bits of precision, we mean that the number of bits used to represent the fractional part of rational number is l.

Another concern is that a number of intuitive propositions that one takes for granted in 2 or 3 dimensions break completely in higher dimensions. The fact that we are only interested in low-dimensional spaces alleviates such concerns significantly, and we are able to make use of the following results, some of which would not hold in higher dimensions.

The following lemma is implied by, among others, [38], and states that the number of integer points in a d-dimensional sphere is more or less what one would expect it to be, namely, of the order of the volume of the sphere.

Lemma C.1. *For any d, the number of integer points in a d-dimensional sphere of radius r centered at the origin is $\Theta(r^d)$.*

This implies that if we use l bits of precision, then for any d, there are constants c_1 and c_2 such that for any point $y \in \mathbb{Z}^d$ and $r \in \mathbb{Q}$:

$$c_1(r2^l)^d \leq |\mathcal{B}(y,r)| \leq c_2(r2^l)^d$$

We shall next look at the task of sampling a point uniformly at random from such a ball $\mathcal{B}(y,r)$, and make use of the above bounds to do so easily by rejection sampling. Hereon we shall use l, as above, to denote the number of bits of precision.

Lemma C.2 (Sampling from balls)**.** *For $y \in \mathbb{Z}^d$ and $r \in \mathbb{Q}$, let $B(y,r)$ be the uniform distribution over the points in $\mathcal{B}(y,r)$. There is a (randomised) logspace program A such that, for any k, $d(A(y,r,1^k), B(y,r)) \leq \frac{1}{2^k}$.*

Proof. Let s be the integer such that $2^{s-1} \leq 2r \leq 2^s$. A repeatedly samples random points in the d-dimensional hypercube of side 2^s centered at the origin until it finds a point that is at a distance of at most r from the origin, or until it fails ck times for some constant c to be determined. If it finds such a point, it shifts the point by y and outputs it, else it outputs y.

To see that this may be done with only logspace, note that selecting a random point from the hypercube of side 2^s simply involves picking $s + l$ random bits, and checking that the chosen point, say $(x_1, \ldots, x_d)$, is within distance r is the same as checking that $\sum_{i=1}^{d} x_i^2 \leq r^2$, which can be done in NC^1 (because squaring, addition of a constant number of integers and comparison can be), and hence in L.

Conditioned on A finding a point in the ball, its output is distributed the same as B. Hence, $d(A(y,r,1^k), B(y,r))$ is at most the probability that A fails to do so,

which is:

$$\left(1 - \frac{\textit{no. points in ball}}{\textit{no. points in hypercube}}\right)^{ck} = \left(1 - \frac{|\mathcal{B}(y,r)|}{2^{d(s+l)}}\right)^{ck}$$
$$\leq \left(1 - \frac{c_1 r^d 2^{ld}}{(4r)^d 2^{ld}}\right)^{ck}$$
$$= \left(1 - \frac{c_1}{4^d}\right)^{ck}$$

where c_1 is the constant from the bound on $|\mathcal{B}(y,r)|$ above and the term inside the brackets at the end is a constant, and so c can be chosen so that the whole thing is at most $\frac{1}{2^k}$, giving us what we need.

□

The following lemma states, in a sense, that balls whose centers are close (relative to their radii) have significant overlap. What shall be salient to us here is that if the fractional distance between the centers is noticeably bounded away from 1 (that is, $(1-\gamma) = \Omega\left(\frac{1}{poly(n)}\right)$), then so is the distance between uniform distributions over these balls.

Lemma C.3. *There is a constant c such that for any $\gamma \in [0,1)$, $r \in \mathbb{Q}^+$ and d-dimensional vector y, if $||y|| \leq \gamma r$, then $d(\mathcal{B}(0,r/2), \mathcal{B}(y,r/2)) = 1 - c(1-\gamma)^d$.*

Proof. Let $B_0 = \mathcal{B}(0,r/2)$ and $B_1 = \mathcal{B}(y,r/2)$ for notational convenience.

As we are concerned with uniform distributions over these sets, we have $d(B_0,B_1) = \frac{|B_0 \backslash B_1|}{|B_0|}$. Considering B_0 and B_1 as balls in $\mathbb{R}^d$, it is easy to see that it is possible to embed a ball of radius $(1-\gamma)\frac{r}{2}$ in $B_0 \cap B_1$ (its center is at $y/2$). Let B' be the set of points situated in the space of this ball. This implies that $|B_0 \backslash B_1| = |B_0| - |B_0 \cap B_1| \leq |B_0| - |B'|$.

Using the bounds on sizes of balls obtained earlier, we have:

$$\begin{aligned} d(B_0, B_1) &= \frac{|B_0 \setminus B_1|}{|B_0|} \\ &\leq 1 - \frac{|B'|}{|B_0|} \\ &\leq 1 - \frac{c_1(1-\gamma)^d(r/2)^d 2^{ld}}{c_2(r/2)^d 2^{ld}} \\ &= 1 - \frac{c_1}{c_2}(1-\gamma)^d \end{aligned}$$

□

We will also be using the following lemma from [32] (which was, in fact, originally used there to prove lemma B.2).

Lemma C.4 (Yet another XOR lemma)**.** *Given distributions X_0, X_1 over the same domain and $k \in \mathbb{Z}^+$, define distributions Y_0, Y_1 by the following sampling procedure for Y_b:*

- *Select bits $b_1, \ldots, b_k$ such that $b_1 \oplus \cdots \oplus b_k = b$.*
- *For each $i \in [k]$, sample $c_i \leftarrow X_{b_i}$.*
- *Output $(c_1, \ldots, c_k)$.*

Then, $d(Y_0, Y_1) = d(X_0, X_1)^k$.

Another procedure that we shall need to perform is that of reducing a vector modulo the fundamental parallelepiped $\mathcal{P}(B)$ of a lattice $\Lambda(B)$. This too can be performed in logarithmic space, as evidenced by the following lemma.

Lemma C.5. *Given a full-rank (over $\mathbb{Q}$) matrix $B \in \mathbb{Z}^{d \times d}$ and vector $y \in \mathbb{Q}^d$, $y \bmod \mathcal{P}(B)$ can be computed in logspace.*

Proof. As B is full rank, there is a vector $x \in \mathbb{Q}^d$ such that $Bx = y$. x may be written as $x_1 + x_2$ for some $x_1 \in \mathbb{Z}^d$ and $x_2 \in [0, 1)^d$. What we wish to compute is $y \bmod \mathcal{P}(B) = Bx_2$. We shall do so by first computing x as $B^{-1}y$, taking the fractional parts of each of its coordinates to get x_2 and then computing Bx_2.

As B is of constant dimension d, its inverse can be computed in NC^1 by the standard method of computing determinants of its minors. $B^{-1}y$ and Bx_2 can also be computed in NC^1, as addition and multiplication of any constant number of integers (or rationals) can be done in NC^1. x_2 can also be obtained from x in NC^1, as it simply involves one addition or subtraction per co-ordinate of x. Thus, $y \bmod \mathcal{P}(B)$ can be computed in NC^1 given B and y, and hence in logspace. □