UNIVERSITY OF CALIFORNIA
Santa Barbara

Zero-Knowledge and Obfuscation

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Mathematics

by

Eric Martin Demer

Committee in Charge:

Professor Daryl Cooper, Co-Chair

Professor Ananth Prabhanjan, Co-Chair

Professor Paul Atzberger

June 2020

The Dissertation of
Eric Martin Demer is approved:

Professor Paul Atzberger

Professor Daryl Cooper, Committee Co-Chair

Professor Ananth Prabhanjan, Committee Co-Chair

June 2020

Zero-Knowledge and Obfuscation

by

Eric Martin Demer

I dedicate this to my mother, Linda Demer, and my advisor, Daryl Cooper. Their effort made it so my procrastination, though excessive, was not enough to prevent me from finishing this thesis.

Acknowledgements

I thank Daryl Cooper for very extensive discussions regarding the layout of this thesis and editing this thesis, as well as fixing the references and arranging the component documents into a form suitable for being a thesis. I thank Prabhanjan Ananth for minor assistance with the obfuscation result. Lastly, my discussions with Stefano Tessaro provided part of the foundation for the obfuscation result.

Curriculum Vitæ

Eric Martin Demer

Education University of California, Santa Barbara

PhD in Mathematics expected June 2020

MA in Mathematics 2013-2015

BS in Mathematics 2009-2013

Skills

Algorithm Design

Mathematical Cryptography

Proficiency with Python programming language

Proficiency with C++ programming language

Team juggling

Teaching at UCSB

Grader September 2014 to December 2014,

Teaching Assistant September 2014 to 2020

Publication

Wong M N, Nguyen T P, Chen T H, Hsu J J, Zeng X, Saw A, Demer E M, Zhao X, Tintut Y, and Demer L L.

Preferred mitotic orientation in pattern formation by vascular mesenchymal cells. *Am J Physiol Heart Circ Physiol.* 303:H1411-1417, 2012.

http://ajpheart.physiology.org/content/303/12/H1411.long

Statistical Consultant for the publication

Preferred mitotic orientation in pattern formation by vascular mesenchymal cells

Abstract

Zero-Knowledge and Obfuscation

Eric Martin Demer

Zero-Knowledge Protocols and Witness Encryption are usually defined for NP relations. I show that they can be extended to handle promiseMA relations. For witness encryption, this simply results in polynomially-longer instances and witnesses. For zero-knowledge protocols, this may require one additional round of interaction, since the prover needs to receive an allegedly random string from the verifier, but the structure of the protocol does not otherwise change, and soundness holds even if the prover chooses its instance after seeing that allegedly random string.

It is known that efficient virtual black-box obfuscation of functions is impossible, but there are candidate constructions of indistinguishability obfuscation, and one can trivially build witness encryption from indistinguishability obfuscation. I show that if witness encryption exists, then there is no virtual black-box obfuscation of *distributions* against auxiliary input.

Contents

List of Figures

Chapter 1

Summary of Results

NP relations roughly correspond to types of puzzles, such that whenever there *is* a solution, there is a *not-too-long* solution that is efficiently-and-deterministically verifiable. The best example of this is probably Sudoku. For that example, the instances are partially-filled grids, and the witnesses are valid ways of filling in the empty squares.

For NP relations, Zero-Knowledge Protocols allow one to show that one knows a solution to an instance, without revealing anything else, and *Witness Encryption* allows one to encrypt plaintext messages m using an instance x, producing a ciphertext c, such that the following hold.

- If there is no witness for x, then the encryption hides everything about m other than the length of m.

- Given c and a witness for x, one can efficiently find m.

I show how to extend zero-knowledge protocols and witness encryption from NP relations to what are called *promiseMA relations*. These relations allow randomness to be used in determining whether or not something is a witness, rather than requiring that the verifier be deterministic. The simplest example of this is probably minesweeper, for which given a fast algorithm that plays the game, one can efficiently estimate that algorithm's win probability by having it play a large number of independent games, but there is no obvious efficient way of *deterministically* estimating that probability. After all, how would one deterministically choose where to put the mines?

Very roughly, *obfuscation* means changing a computer program so that the resulting code does the same thing as the original code, in whatever the relevant sense may be, but is otherwise unintelligible. It is known that when the thing the code does is compute a function, there is no close-to-ideal way to obfuscate code. I show that, if witness encryption exists for all NP relations, then there is no close-to-ideal way to obfuscate code when the thing the code does is sample from a distribution.

Chapter 2

Zero–knowledge protocols

In the following discussions about zero–knowledge protocols, I use the names Peggy and Victor because Peggy is the Prover and Victor is the Verifier.

Suppose Peggy and Victor both know a pair of networks which differ only in the labeling of their nodes, and that Peggy knows a bijective correspondence between the labels that converts the networks to each other. These sorts of networks are called *graphs*, and such correspondences between the labels are called isomorphisms. Is there some way for Peggy to prove to Victor that she knows an isomorphism between the two graphs *without* revealing the isomorphism?

The answer is *yes*, and the simplest way of doing this is described in the next section and is also the simplest example of a zero–knowledge proof.

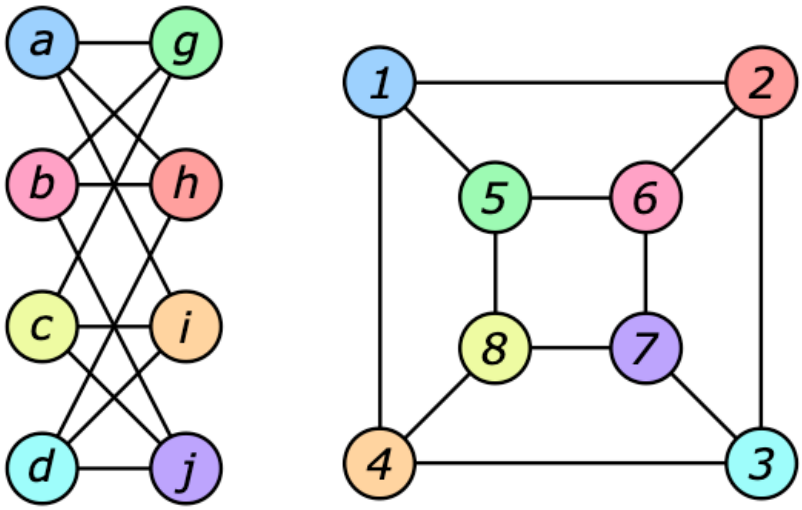


Figure 2.1: This is an example of two graphs that are isomorphic.

2.1 Giving nothing away

This section describes proving and verifying the existence of an isomorphism without conveying extra knowledge.

The Basic Protocol for Graph Isomorphism

0. An ordered pair of graphs (G_0, G_1) is common input to Peggy and Victor, and an isomorphism w between those graphs is an input to just Peggy.

1. Peggy chooses a bit b and lets L be the set of labels for G_b.

2. Peggy lets n be the number of elements of L.

3. Peggy computes a random bijection $f : L \to \{0, 1, 2, \cdots, n-1\}$.

4. Peggy relabels G_b by replacing each label x with the new label $f(x)$.

5. Peggy sends the new graph, G_{new}, to Victor.

6. Victor chooses a bit c uniformly at random.

7. Victor sends c to Peggy.

8. Peggy sends to Victor a function f_{sent} defined as follows.

If $c = b$, then $f_{\text{sent}} = f$. If $c \neq b$ then $f_{\text{sent}} = f \circ w$.

9. If f_{sent} is an isomorphism from G_c to G_{new}, then Victor accepts,

else Victor rejects.

As an example, suppose Peggy picks the right of the two networks in figure 2.1. In this case, $L = \{1, 2, 3, 4, 5, 6, 7, 8\}$ and $n = 8$.

In this way, Victor becomes half-convinced that Peggy knows an isomorphism between the graphs, but without learning anything else. The reason why Victor is only half-convinced is that even when there is no isomorphism between the graphs, Peggy has a simple strategy which has probability half of getting Victor to

accept. By repeating the protocol multiple times, Victor can become increasingly convinced that the graphs are isomorphic.

There are four conditions for a protocol to be a zero–knowledge protocol. In the context of graph isomorphism, the term *witnesses* used below refers to isomorphisms between the graphs.

Efficiency: The verifier can efficiently do its part of the protocol and, given a witness, the prover can efficiently do its part of the protocol.

Correctness or *Completeness*: If the prover knows a witness and follows the protocol, and the verifier also follows the protocol, then the verifier will accept.

Zero–Knowledge: Even a possibly malicious verifier can't learn more than that the prover knows a witness.

Soundness: The *basic* notion of soundness is, if there does *not* exist a witness, then the verifier will have negligible probability of accepting. One frequently also wants a stronger property called *knowledge-soundness*, which I go into in section (2.1.3).

In theoretical cryptography and theoretical computer science, something being *efficient* by default means its runtime is bounded by a polynomial in the length of the input, and that is the meaning used in this exposition.

With respect to the present example basic protocol, because it is simple enough, efficiency clearly holds, and completeness will clearly hold when $c = b$.

Whether completeness holds when $c \neq b$ is slightly more complex, and it is addressed as follows. If $c \neq b$, then

- w is an isomorphism from G_c (the graph Victor chose) to G_b (the graph Peggy chose).
- f is an isomorphism from G_b to G_{new}.

So f_{sent} is an isomorphism from G_c to G_{new}, which means Victor accepts. Thus completeness holds.

In most cases, one can't have both perfect zero–knowledge and perfect soundness. Instead, these are usually measured with a *security parameter*, which is usually denoted k. For example, the verifier might have a 2^{-k} chance of learning the witness, and the prover might have a 2^{-k} chance of convincing the verifier of a false statement.

Because the basic protocol does not satisfy soundness (Peggy has a 50% chance of convincing Victor even when the graphs are not isomorphic) and can be simulated essentially perfectly, I do not have it use a security parameter.

An additional wrinkle here is that, in computational complexity theory, efficiency is generally measured in terms of the *length* of the input, whereas for a positive integer L, the number $2^L - 1$ is only L bits long. For that reason, one gives unary rather than binary for k. I use overline to denote the unary repre-

sentation of a natural number, so $\overline{k}$ is the string of k ones. For example, $\overline{6}$ is the string 111111.

2.1.1 The Zero–Knowledge Condition

Although the zero–knowledge property was described earlier as *the verifier learns nothing beyond that the prover knows a witness*, its formal definitions involve a *simulator* for the distribution of the verifier's views of its interaction with the prover. Roughly, the zero–knowledge property is that there is an efficient simulator such that interactions with the real prover are indistinguishable from outputs of the simulator. To be more specific, about that, I need to give significantly more explanation.

The *instance* is what the prover is trying to show something about. For graph isomorphism, the *instance* is the ordered pair of graphs (G_0, G_1). *View*s are usually defined as including more than my definition does, but the additional information does not affect the definition of zero–knowledge protocols. I define a possibly malicious verifier $\mathcal{A}$'s *view* of an interaction as consisting of $\mathcal{A}$'s randomness and the sequence of messages sent by the prover. In particular, graph isomorphism, Victor's *view* is the ordered triple $(c, G_b, f_{\text{sent}})$.

The zero–knowledge property requires that the simulator satisfy efficiency and indistinguishability.

To describe these requirements, I first note that the zero–knowledge property can be either *computational* or *statistical*. I begin with computational zero-knowledge, since it is easier to go from that to statistical zero-knowledge than the other way around.

Strict efficiency is that the simulator is efficient in the worst-case. *Expected efficiency* is that it is infeasible to find an input on which the simulator's average runtime is too long.

In the following, **g&d** stands for *generate and distinguish*. For a prover $\mathcal{P}$ and a simulator $\mathcal{S}$, *computational indistinguishability* means there is no feasible team of adversaries $(\mathcal{A}_{\text{g\&d}}, \mathcal{A}_{\text{ver}})$ such that Prob $(\mathcal{A}_{\text{g\&d}}$ outputs 1) differs non–negligibly between the following two experiments.

Real	**Simulated**
$\mathcal{A}_{\text{g\&d}}$ receives $\overline{k}$.	
$\mathcal{A}_{\text{g\&d}}$ chooses an instance x and a witness w for x and an auxiliary input z.	
Let $\mathcal{A}_{\text{ver}}\left(\overline{k}, x, z\right)$ interact with $\mathcal{P}\left(\overline{k}, x, y\right)$.	Run $\mathcal{S}\left(\overline{k}, x, z, \mathcal{A}_{\text{ver}}\right)$.
Send $\mathcal{A}_{\text{ver}}$'s view of that interaction to $\mathcal{A}_{\text{g\&d}}$.	Send the output of that to $\mathcal{A}_{\text{g\&d}}$.
$\mathcal{A}_{\text{g\&d}}$ outputs a bit.	

The definition of *statistical indistinguishability* instead requires that there are no such adversaries, feasible or otherwise. Similarly, for statistical zero-knowledge, the definition of expected efficiency is that there are no inputs on which the simulator's average runtime is too long. The definition of strict efficiency is the same for statistical zero-knowledge as it is for computational zero-knowledge.

Importantly, the simulator does *not* receive a witness. Thus, given an efficient simulator satisfying a suitable level of indistinguishability, if Victor wanted to use a run of the protocol as input to some other algorithm, then Victor could just as well instead use the simulator's output as the input to the other algorithm.

This section describes the properties that a simulator must satisfy to be a zero–knowledge simulator. In the next section, I build such a simulator for the basic protocol.

2.1.2 Simulating the Basic Protocol

In order to build such a simulator for the basic protocol, I start with a pre-simulator.

a Pre-simulator for the Basic Protocol

0. An ordered pair of graphs (G_0, G_1) is common input to the pre-simulator and Victor,

1. The pre-simulator chooses a bit b uniformly at random, and lets L be the set of labels for G_b.

2. - 7. These use the pre-simulator instead of Peggy, but are otherwise the same as 2-7 from the basic protocol.

8. If $c = b$, then the pre-simulator sends its function f to Victor. If $c \neq b$, then the pre-simulator fails.

Efficiency clearly holds, so consider the extent to which the pre-simulator satisfies the zero–knowledge property.

Fix the common input (G_0, G_1), assume that these graphs are isomorphic, and let H be a graph that Peggy or the pre-simulator might send to Victor in step 5. Since steps 2 and 3 result in the new labels corresponding to old labels that are different from each other but otherwise uniformly random, the probability of Victor receiving H in step 7 depends on neither of the following.

- whether Victor is interacting with Peggy or the pre-simulator
- which graph Peggy or the pre-simulator chose in step 1.

In particular, for each graph H that Victor might receive from the pre-simulator, Victor is equally likely to receive H from the pre-simulator when $b = 0$ as Victor

is to receive H from the pre-simulator when $b = 1$. Thus, regardless of which graph H Victor receives, and which bit c Victor chooses, the pre-simulator has probability exactly $1/2$ of failing. On the other hand, in each of the following cases,

- Victor interacts with Peggy and $b = 0$
- Victor interacts with Peggy and $b = 1$
- Victor interacts with the pre-simulator and $c = b$

the function, f_{sent} that Victor receives from Peggy is equally likely to be any of the isomorphisms from G_b to G_{new}.

Thus, Victor's views of his interactions with Peggy are indistinguishable from his views of his interactions with the pre-simulator *conditioned on the pre-simulator not failing.*

a Simulator for the Basic Protocol

Receive as input $\overline{k}$ and the ordered pair of graphs (G_0, G_1) and the auxiliary input z and Victor's algorithm.

Using independent randomness each time, run interactions of Victor's algorithm with the pre-simulator until getting a run in which the pre-simulator does not fail.

Output the view from Victor's algorithm of its interaction with the pre-simulator in which the pre-simulator did not fail.

Since the probability of the pre-simulator failing is $1/2$, the average number of runs is 2, so the simulator is efficient on average. Furthermore, it follows that the distribution of the simulator's outputs is indistinguishable from the distribution of Victor's views of his interactions with Peggy.

2.1.3 Soundness

The basic notion of soundness is, if there does *not* exist a witness, then the verifier will have negligible probability of accepting. However, like zero–knowledge, this soundness can also be *computational* or *statistical*. When soundness is statistical, protocols as described here are called interactive proofs, and when soundness is only assumed to be computational, protocols as described here are called interactive arguments. For a set Π and a verifier $\mathcal{V}$, a protocol for showing $x \in \Pi$ is

computationally sound if and only if there does not exist a feasible adversary $\mathcal{A}$ with non-negligible probability of succeeding in the following.

Soundness Experiment

$\mathcal{A}$ receives $\overline{k}$

$\mathcal{A}$ chooses an instance x

If $x \in \Pi$ then $\mathcal{A}$ fails.

(in which case the rest of this does not happen)

Have $\mathcal{A}$ interact with $\mathcal{V}\left(\overline{k}, x\right)$.

If $\mathcal{V}$ accepts, then $\mathcal{A}$ succeeds, else $\mathcal{A}$ fails.

$\mathcal{V}\left(\overline{k}, x\right)$ denotes the verifier with inputs $\overline{k}$ and x. In other words, it is not feasible to get the verifier to accept something that is not in Π.

The definition of *statistical soundness* instead requires that there are no such adversaries, feasible or otherwise. However, quite often, one wants the verifier to be convinced not merely that there *exists* a witness, but that the prover *knows* a witness: i.e., one wants *knowledge-soundness*. To define this, the set Π is not enough. One instead needs to pick a *witness relation*, which is a relation R such that w is a witness for x if and only if xRw. Although, for most natural sets Π with at least one such R, there is an obvious such R, I am not aware of any results to the effect that one can generally pick such an R which is canonical in any sense.

One can define *strict* knowledge-soundness or *expected* knowledge-soundness. Similarly to the way that the definition zero-knowledge involves a simulator, the definition of strict knowledge-soundess involves a witness-extended emulator and the definition of expected knowledge-soundness involves a knowledge extractor. The main condition that a witness-extended emulator must satisfy is fairly complicated, and the explanation of why that condition is used is far more complicated, so I do not give that definition in this exposition. I instead just give the definition of expected knowledge-soundness. As mentioned, this definition involves a *knowledge-extractor*, henceforth called an *extractor*.

Expected knowledge-soundness requires that the extractor satisfy expected efficiency and expected knowledge-extraction.

As mentioned before, the knowledge-soundness property can be either *computational* or *statistical*. I begin with computational knowledge-soundness, since it is easier to go from that to statistical knowledge-soundness than the other way around.

Expected efficiency is the condition that there is a polynomial *poly* such that it is infeasible to find a positive integer $\overline{T}$ and other inputs to the extractor such that the extractor's average runtime on $\overline{T}$ and those other inputs is greater than $T \cdot \text{poly}(\ell)$, where ℓ is the combined length of those other inputs. For a verifier $\mathcal{V}$ and an extractor $\mathcal{E}$, *expected knowledge-extraction* means no feasible teams of ad-

versaries $(\mathcal{A}_{\text{gen}}, \mathcal{A}_{\text{prv}})$ has non-negligible probability of succeeding in the following experiment.

Expected knowledge-extraction Experiment

$\mathcal{A}_{\text{gen}}$ receives $\overline{k}$

$\mathcal{A}_{\text{gen}}$ chooses an instance x and an auxiliary input z and a positive integer $\overline{T}$ in unary.

If the probability of $\mathcal{A}_{\text{prv}}\left(\overline{k}, x, z\right)$ convincing $\mathcal{V}(\overline{k}, x)$ is at most $1/T$, then the team of adversaries fails. In this case we stop here.

Run $\mathcal{E}\left(\overline{k}, x, z, \mathcal{A}_{\text{prv}}, \overline{T}\right)$. If it outputs a witness for x, then the team of adversaries fails, else the team of adversaries succeeds.

The definition of *statistical knowledge-soundness* still restricts to values of T such that $\overline{T}$ can feasibly be produced, but otherwise requires that there are no such adversaries. i.e., for *statistical knowledge-soundness*, the probability of convincing the verifier still can't be too small, but otherwise one does not restrict to feasible adversaries.

We say that a protocol is *an argument of knowledge* if the knowledge extractor satisfies computational knowledge-soundness. We say that a protocol is a *proof of knowledge* if the knowledge extractor satisfies statistical knowledge-soundness.

2.1.4 Graph Isomorphism

We describe a sound protocol for Graph Isomorphism. To achieve soundness, one uses sequential composition.

Statistical Zero-Knowledge Proof-of-Knowledge for Graph Isomorphism

0. $\overline{k}$ and (G_0, G_1) are common inputs to the prover and the verifier, and an isomorphism w between those graphs is an input to just the prover.

1. Run the basic protocol for graph isomorphism k times. If Victor accepts every time in the basic protocol, then this verifier accepts, otherwise it rejects.

Since the security parameter k is given in unary and the basic protocol satisfies efficiency, this protocol also satisfies efficiency. Since the basic protocol satisfies completeness, this protocol also satisfies completeness.

The Simulator

0. Receive as input $\overline{k}$ and (G_0, G_1) and an auxiliary input z and $\mathcal{A}_{\text{ver}}$.

1. Let z_0 be the state in which $\mathcal{A}_{\text{ver}}$ starts.

2. Produce a sequence of states $z_1, z_2, \cdots, z_{k-1}$ as follows. Simulate the basic protocol for graph isomorphism, using as verifier the algorithm that starts in state z_i and ends when the basic protocol ends, and otherwise works as $\mathcal{A}_{\text{ver}}\left(\overline{k}, (G_0, G_1), z\right)$.

Let z_{i+1} be the state that $\mathcal{A}_{\text{ver}}\left(\overline{k}, (G_0, G_1), z\right)$ reaches from z_i by following that simulator's output.

3. Output the concatenation of the randomness for $\mathcal{A}_{\text{ver}}$ from each of those simulations, followed by the sequence of simulated messages from the prover for those simulations.

To show indistinguishability, one considers the distribution of Simulated and Real views, denoted distrib_0 and distrib_k, as well as $k-1$ hybrids $\text{distrib}_1, \cdots, \text{distrib}_{k-1}$. The hybrid, distrib_n, first performs $k-n$ simulated iterations followed by n iterations that are interactions with the prover.

For each experiment, let p_n be $\text{Prob}\left(\mathcal{A}_{\text{g\&d}} \text{ outputs } 1\right)$ in distrib_n. For positive integers k, if $|p_0 - p_k| \geq \delta$, then the average of $p_0 - p_1, p_1 - p_2, \cdots, p_{k-1} - p_k$ has absolute value at least δ/k. So one could distinguish between simulated and real views of the basic protocol by at least δ/k by having the distinguisher for

the basic protocol choose $n \in \{0, 1, \cdots, k-1\}$ uniformly at random and run the distinguisher for the full protocol on input that is from either distrib_n or distrib_{n+1}.

Incidentally, this application was the original reason for the introduction of auxiliary input: A malicious verifier for the full protocol might make its behavior in each execution of the basic protocol depend on its views of the earlier executions of the basic protocol, so in this case, its views of the earlier executions of the basic protocol are included in the auxiliary input.

Knowledge Extractor

0. Receive as input $\overline{k}$ and (G_0, G_1) and an auxiliary input

   z and a malicious prover $\mathcal{A}_{\text{prv}}$ and an integer $\overline{T}$ in unary.

1. If $T > 2^{k-1}$ then the extractor fails.

2. Set $\mathcal{B} = \mathcal{A}_{\text{prv}}(\overline{k}, (G_0, G_1), z)$.

3. Repeat the following up to $T \cdot k$ times:

   Start $\mathcal{B}$ as if it was interacting with a verifier.

   Repeat the following up to k times:

   Create a graph G_{new} by running $\mathcal{B}$.

   Choose a bit c uniformly at random, and send $1 - c$ to $\mathcal{B}$.

   Create a function f_{1-c} by running $\mathcal{B}$.

   Rewind $\mathcal{B}$ to just before $1 - c$ was sent to it.

   Send c to $\mathcal{B}$, then create a function f_c by running $\mathcal{B}$.

   If $f_c : G_c \to G_{\text{new}}$ is *not* an isomorphism,

   then go to the next iteration of step 3, else

   if $f_c^{-1} \circ f_{1-c} : G_{1-c} \to G_c$ is an isomorphism,

   then output it and halt.

4. If step 3 did not find an isomorphism, then the extractor fails.

By inspection, one can see that expected efficiency holds. For example, the extractor runs $\mathcal{B}$ at most $T \cdot k \cdot k \cdot 3$ times. Because k is large, producing a string whose length exceeds 2^{k-1} is infeasible, so assume $T \leq 2^{k-1}$.

Aside from brief excursions to see what the response of the adversary to the other bit would have been, and the extractor possibly halting with a witness, each iteration of step 3 is equivalent to the adversary interacting with an honest verifier.

Let p be the probability of the malicious prover convincing an honest verifier. If $p < 1/T$ then the team of adversaries fails in the expected knowledge-extraction experiment, so assume $1/T \leq p$.

Each time a bit c is chosen in the inner loop, that choice is uniformly at random, and $\mathcal{B}$ is in the same state when it receives $1 - c$ as when it receives c. Thus the probability of $\mathcal{B}$ outputting an isomorphism from G_d to G_{new} when $d = c$ but not when $d = 1 - c$ is at most $1/2$. Thus, for each of the at most $k \cdot T$ iterations of step 3, the probability of the extractor finding an isomorphism is at least $p - 2^{-k}$. Since $T \leq 2^{k-1}$ and $1/T \leq p$ it follows that

$$2 \cdot 2^{-k} = 2^{-(k-1)} \leq 1/T \leq p \quad \therefore \quad p - 2^{-k} \geq (1/2) \cdot p \geq (1/2) \cdot (1/T)$$

Thus, the probability that the extractor does not find an isomorphism is at most $(1-(1/2)\cdot(1/T))^{T\cdot k}$. Using $\ln(1+x) < x$ gives

$$\begin{aligned}
&\ln\left((1-(1/2)\cdot(1/T))^{T\cdot k}\right) \\
&= \ln(1-(1/2)\cdot(1/T))\cdot T\cdot k \\
&\leq -(1/2)\cdot(1/T)\cdot T\cdot k \\
&= -(1/2)\cdot k
\end{aligned}$$

so $(1-(1/2)\cdot(1/T))^{T\cdot k} \leq e^{-(1/2)\cdot k}$. Since $e^{-(1/2)\cdot k}$ is negligible, knowledge-soundness holds.

2.2 Zero-Knowledge Protocols

The protocol for graph isomorphism can easily be extended to other isomorphism problems for which one can do something corresponding to randomly relabeling the vertices. However, that is still fairly narrow. Based on certain computational hardness assumptions, which I will describe later, the notions of computational zero-knowledge and computational soundness seem to allow one to build zero-knowledge protocols for *every* witness relation R such that

- R is decidable in time polynomial in the length of the input

- there is a polynomial poly such that for all x and w, if xRw then length$(w) \leq$ poly(length(x)).

The use of *polynomial* in these conditions is due to the theoretical definition of efficient as polynomial-time. Such relations are called *NP relations*, because by definition, the complexity class NP is the set of decision problems Π such that there exists such a witness relation R for Π.

As an example, consider the *Hamiltonian Path Problem*. Given a directed graph G, does G have a *Hamiltonian path*, i.e., a path in G that visits each vertex exactly once?

For this problem, I use the obvious witness relation. The witnesses for a graph G are the Hamiltonian paths in G. This problem is *NP-complete* [22], which roughly means that it is in NP and for each other problem Π in NP, given an instance x for Π, one can efficiently compute a graph G such that G has a Hamiltonian path if and only if the answer to x is yes.

Furthermore, by examining the proofs, one sees that for each NP relation R, one can do this in a way which respects knowledge of witnesses. For any instance x for R and corresponding graph G, given a witness for x one can efficiently find a Hamiltonian path in G, and given a Hamiltonian path in G one can efficiently find a witness for x.

Thus, a zero-knowledge protocol for the Hamiltonian Path Problem, with the obvious witness relation, yields a zero-knowledge protocol for every NP relation.

2.2.1 Hamiltonian Paths

Temporarily, assume that the prover can provide the verifier with ideal *locked boxes* each containing a bit, such that for each box:

- with the help of the prover, the verifier can learn which bit the box contains
- without the help of the prover, the verifier can't distinguish boxes containing 0 from boxes containing 1

Once I describe how to use this in basic protocol for the Hamiltonian Path problem, I will then describe how to replace them with a digital version based on computational hardness assumptions.

Basic Protocol with Boxes

0. A graph G is common input to the prover and verifier, and a Hamiltonian path called *witpath* in G is an input to just the prover.

1. The prover lets n be the number of vertices.

2. The prover chooses a random bijection

$randpath : \{0, 1, \cdots, n-1\} \to \{0, 1, \cdots, n-1\}$.

3. For each $u \neq v \in \{0, 1, \cdots, n-1\}$, the prover creates a locked box labeled (u, v). For each $m \in \{0, 1, \cdots, n-2\}$, the prover puts 1 into the box $(\text{randpath}(m), \text{randpath}(m+1))$, and puts 0 into the remaining boxes.

4. The prover sends those locked boxes to the verifier.

5. The verifier chooses a bit c uniformly at random and sends it to the prover.

6. **If** $c = 1$ then the prover sends *randpath* to the verifier, and helps the verifier open all the boxes. If they all open to the bits described in step 3, then the verifier accepts, else the verifier rejects.

If $c = 0$ then the prover lets V be the set of vertices of G and

$f : V \to \{0, 1 \cdots, n-1\}$ is $f(vertex\ m\ of\ \text{witpath}) = \text{randpath}(m)$.

The prover sends f to the verifier, and helps the verifier open those boxes $(f(u), f(v))$ for which G does *not* have an edge from u to v. If they all open to 0, then the verifier accepts, else the verifier rejects.

Efficiency clearly holds, and completeness will clearly hold when $c = 1$. Since witpath is a path in G, it only goes along G's edges. Thus, for every ordered pair of distinct vertices (u, v), if G does *not* have an edge from u to v, then witpath does not go directly from u to v. In this case $f(u), f(v)$ will not be consecutive outputs of randpath, so the (u, v) box will have a 0. Thus completeness will also hold when $c = 0$.

The zero-knowledge property is far easier with ideal locked boxes than with the digital implementation described later. If $\mathcal{A}_{\text{ver}}$ chooses $c = 1$, then the simulator chooses randpath as in step 2 and for each box, tells $\mathcal{A}_{\text{ver}}$ that the box opened to the bit as described in step 3. If $\mathcal{A}_{\text{ver}}$ chooses $c = 0$, then the simulator chooses a random bijection f from the vertices of G to $\{0, 1, 2, \cdots, n-1\}$, and for each of the boxes that are to be opened, tells $\mathcal{A}_{\text{ver}}$ that the box opened to 0. Indistinguishability holds when witpath does not depend on randpath, because in that case randpath being uniformly random means that for an honest prover, f will be uniformly random.

This protocol does not yet satisfy the definition of soundness that one is ultimately after, but it does come close in the same way as graph isomorphism. Given a valid response for $c = 1$ *and* a valid response for $c = 0$, one can easily find a Hamiltonian path in G. That is because f must send non-edges to pairs

that randpath does *not* follow, so $f^{-1} \circ \text{randpath}$ must be a path that follows G's edges.

2.3 Commitment Schemes

The weakest digital version of such locked boxes, is what is called a *commitment scheme*. In general, such a protocol consists of a *commitment* phase followed by an *reveal* phase, between two parties, a *committer* and a *receiver*. The security parameter in unary is common input to the parties, and a message m is an input to just the committer, and at the end, the receiver outputs either a message $\hat{m}$ or rejects.

In order to replace the boxes in the above protocol, a commitment scheme just needs to handle single-bit messages m, so the definitions given here will be specific to bit commitment. The properties that such a scheme must satisfy are the following.

Efficiency: The parties can efficiently do their parts of the protocol.

Correctness or *Completeness*: If the parties both follow the protocol, then the receiver's output will be the same as the committer's input message.

Hiding: Even a possibly malicious receiver can't learn about m before the reveal phase.

Binding: After the commitment phase, even a malicious committer won't both be able to reveal a 0 and be able to reveal a 1.

As with the zero-knowledge property, hiding can be either computational or statistical. *Statistical hiding* means that for all adversaries $\mathcal{A}$, $|p_0 - p_1|$ is negligible, where p_m is the probability that $\mathcal{A}(\overline{k})$ outputs 1 between the commitment phase and the reveal phase when interacting with a committer whose input message is m. The definition of *computational hiding* instead only requires this for feasible adversaries.

For a receiver $\mathcal{R}$, *computational binding* means that, for all feasible adversaries $\mathcal{A}$, the probability of $\mathcal{A}$ succeeding in the following experiment is negligible.

Binding Experiment

$\mathcal{A}$ and $\mathcal{R}$ receive $\overline{k}$ as common input.

$\mathcal{A}$ interacts with $\mathcal{R}$ in a commitment phase.

$b \in \{0, 1\}$ is chosen, depending on $\mathcal{A}$ and $\mathcal{A}$'s state and $\mathcal{R}$'s state, so as to minimize $\mathcal{A}$'s probability of succeeding in the following, with ties broken towards 0.

$\mathcal{A}$ receives b.

$\mathcal{A}$ interacts with $\mathcal{R}$ in a reveal phase.

If $\mathcal{R}$ outputs b, then $\mathcal{A}$ succeeds, else $\mathcal{A}$ fails.

For most natural bit commitment schemes, the definition of binding can be simplified. Specifically, suppose that reveal is simple, in the following sense.

- During the reveal phase, $\mathcal{R}$ is deterministic.
- For each of $\mathcal{R}$'s actions (sending a message or giving an output) during that phase, $\mathcal{R}$ only uses the sequence of messages exchanged prior to that action.

In that case, the committer can easily figure out exactly what such an $\mathcal{R}$ would do in the reveal phase. Thus the reveal phase is, without loss of generality, non-interactive. For the reveal phase, the committer sends $\mathcal{R}$ all at once the sequence of messages that it would have sent in the interactive reveal phase. For the same reason, given the transcript of the commitment phase, and an attempted reveal by the committer, anyone can determine what the the output of receiver would be. For such bit commitment schemes, the binding property as defined above is equivalent to it being the case that for all feasible adversaries $\mathcal{A}$, the probability of $\mathcal{A}$ succeeding in the following experiment is negligible.

Binding Experiment for Simple Reveal

$\mathcal{A}$ and $\mathcal{R}$ receive $\overline{k}$ as common input.

$\mathcal{A}$ interacts with $\mathcal{R}$ in a commitment phase.

$\mathcal{A}$ outputs x_0 and x_1

If the receiver outputs 0 when the committer sends x_0, and outputs 1 when x_1 is sent, then $\mathcal{A}$ succeeds, else $\mathcal{A}$ fails.

The definition of *statistical binding* requires that the probability of succeeding be negligible for *all* adversaries $\mathcal{A}$, rather than just for all *feasible* adversaries $\mathcal{A}$. The equivalence of the two definitions for protocols with simple reveal also holds for statistical binding. Given a bit commitment scheme, one can replace the locked boxes with commitments to the bits that would've been inside those boxes.

2.4 Hamiltonian Path Without Boxes

Basic Protocol for Hamiltonian Path

Instead of putting bits in boxes, the prover commits to bits.

Instead of helping the verifier open boxes, the prover reveals commitments.

Otherwise, this is the same as the Basic Protocol with Boxes.

Efficiency of this protocol follows easily from efficiency of the protocol with boxes and efficiency of the bit commitment scheme. The same applies to completeness.

If G has fewer than 2 vertices, then it has at most one Hamiltonian path, and it is trivial to find that path when there is one. Thus, the simulator can just find that path, halting with output equal to the empty view if there is no such path, and then use the path it found to honestly prove that it knows such a path. Accordingly, assume that the graph G always has at least 2 vertices.

Here, establishing the zero-knowledge property is substantially trickier. For the protocol with boxes, a simulator could just go through the first six steps,

and wait until it receives the bit c to decide what bits the simulator should say the opened boxes contained. Although there are more sophisticated notions of commitment which would allow that to be done via a simulated commitment phase, the standard notion of commitment does not support that. Instead, one starts with a pre-simulator, similarly to how to handle the basic protocol for the graph isomorphism problem.

a Pre-simulator for Hamiltonian Path

0. $\overline{k}$ and a graph G are common input to the pre-simulator and the verifier.

1. The pre-simulator lets n be the number of vertices.

2. The pre-simulator chooses a random bijection *randpath* of $\{0, \cdots, n-1\}$.

3. The pre-simulator chooses a bit b uniformly at random.

4. Using security parameter k, the pre-simulator commits to a bit for each pair from step 3 of the basic protocol. If $b = 1$ then those bits are as in that step. If $b = 0$ then those bits are all 0.

5. The pre-simulator receives a bit c from the verifier.

6. If $c = b$ then the pre-simulator reveals committed bits as in step 6 of the basic protocol.

   If $c \neq b$ then the pre-simulator fails.

Efficiency of the pre-simulator clearly follows from efficiency of the commitment scheme. As was done for graph isomorphism, one would like to show two things relevant to the zero-knowledge property.

- The pre-simulator has a noticeable probability of not failing.
- The verifier's views of its interactions with an honest prover are indistinguishable from the verifier's views of its interactions with the pre-simulator *conditioned on the pre-simulator not failing*.

The difficulty with these is that the hiding property of the commitment scheme was defined for a *single* commitment. To deal with this, one shows those properties via a hybrid argument, by changing one commitment at a time. We call there hybrids *pseudo–pre-simulator*, abbreviated to *PPS*.

Pseudo–pre-simulator for Hamiltonian Path

0. $\overline{k}$ and a graph G are common input to PPS and the verifier. PPS also receives as input a Hamiltonian path witpath in G and a non-negative integer j.

1. PPS lets n be the number of vertices.

2. PPS chooses a random bijection *randpath* of $\{0, \cdots, n-1\}$.

3. PPS chooses a bit b uniformly at random.

4. If $b = 1$ then the prover sets $L = n - 1$, else the prover sets $L = \min(n-1, j)$.

5. Using security parameter k, for each ordered pair (u, v) of distinct elements of $\{0, \cdots, n-1\}$, PPS commits to a bit for that pair. For each $m \in \{0, \cdots, L-1\}$, the bit is 1 for the pair $(\text{randpath}(m), \text{randpath}(m+1))$, and the bit is 0 for the remaining pairs.

6. and 7. PPS does the same in these steps as the pre-simulator does in steps 5 and 6 respectively.

By efficiency of the commitment scheme, PPS is efficient. A subscript on PPS indicates the value of j for PPS.

Observe that the verifier's view of an interaction with PPS_0 is identical to the verifier's view of an interaction with the pre-simulator. On the other hand, the verifier's view of interactions with PPS_{n-1} differ from the verifier's view of

interactions with an honest prover only in that, independently of the verifier's view up to the point at which the prover would reveal committed bits, PPS_{n-1} has probability 1/2 of failing instead of revealing those bits.

Going from j to $j+1$ can affect the bit committed for $(\mathrm{randpath}(j), \mathrm{randpath}(j+1))$, but does not affect the bits committed to by the other commitments. This lets one use the commitment scheme's hiding property to show that PPS_0 is indistinguishable from PPS_{n-1}. To do so, suppose $\mathcal{D}_{\mathrm{zk}}$ is a distinguisher for those two cases and let $\mathcal{D}_{\mathrm{com}}$ be the distinguisher for the commitment scheme that works as follows.

- Choose $j \in \{0, 1, \cdots, n-2\}$ uniformly at random.
- Have the verifier interact with PPS_j, except that if PPS_j chooses $b = 0$, then for the pair $(\mathrm{randpath}(j), \mathrm{randpath}(j+1))$, use the commitment to an unknown bit instead of necessarily committing to 0.
- Run $\mathcal{D}_{\mathrm{zk}}$ on the verifier's view of that interaction, and give the same output as $\mathcal{D}_{\mathrm{zk}}$.

Claim: the amount by which $\mathcal{D}_{\mathrm{zk}}$ distinguishes its two cases is $n-1$ times the amount by which $\mathcal{D}_{\mathrm{com}}$ distinguishes commitments to 0 from commitments to 1.

Proof of Claim: For each non-negative integer j, let p_j be the probability of $\mathcal{D}_{\mathrm{zk}}$ outputting 1 when its input is a view of an interaction with PPS_j. For each

bit b and $j \in \{0, 1, \cdots, n-2\}$, when the unknown bit is b and the distinguisher chooses that j, the probability of $\mathcal{D}_{\text{com}}$ outputting 1 will be p_{j+b}. Thus, the amount by which $\mathcal{D}_{\text{com}}$ distinguishes commitments to 0 from commitments to 1 is the absolute value of the average of $p_0 - p_1, p_1 - p_2, \cdots, p_{n-2} - p_{n-1}$. That average is $|p_0 - p_{n-1}|/(n-1)$, and $|p_0 - p_{n-1}|$ is the amount by which $\mathcal{D}_{\text{zk}}$ distinguishes its two cases. □

Thus, by the commitment scheme's hiding property, the verifier's views of its interactions with PPS using $j = 0$ are indistinguishable from the verifier's views of its interactions with PPS using $j = n - 1$.

Now, recall the behavior of PPS_0 and PPS_{n-1}. The verifier's view of an interaction with PPS is identical to the verifier's view of an interaction with the pre-simulator. The verifier's view of interactions with PPS_{n-1} differ from the verifier's view of interactions with an honest prover only in that, independently of the verifier's view up to the point at which the prover would reveal committed bits, PPS has probability 1/2 of failing instead of revealing those bits.

Thus probability of the pre-simulator failing differs at most negligibly from 1/2, so for all sufficiently large k, that probability is greater than 1/3. In particular, the pre-simulator and PPS_{n-1} both have noticeable probability of not failing. Since views of interactions with them are indistinguishable, this means views of interactions with the *conditioned on them not failing* are also indistin-

guishable. Furthermore, conditioned on PPS_{n-1} not failing, the verifier's view of interactions with PPS_{n-1} are *identical* to the verifier's views of interactions with the honest prover. Therefore the verifiers interactions with the honest prover are indistinguishable from the verifier's view of interactions with the pre-simulator conditioned on the pre-simulator not failing.

Lastly, one wants the property which will be used to show that sequential composition of the basic protocol satisfies soundness.

Let $\mathcal{A}_{\mathrm{prv}}$ be a possibly-malicious prover. Interact with it as an honest verifier until just before the verifier would choose a bit c in step 4. At that point, make a copy of $\mathcal{A}_{\mathrm{prv}}$, and send $c = 1$ to one and $c = 0$ to the other. If the verifier accepts for both copies, then $f^{-1} \circ \mathrm{randpath}$ is probably a Hamiltonian path in the graph. That is formalized as the following claim.

Let f be the function from the $c = 0$ branch and randpath the path from the $c = 1$ branch. The *special-soundness error* is the probability that the verifier accepts in both branches, and $f^{-1} \circ \mathrm{randpath}$ is *not* a Hamiltonian path in G

Claim 0: The special-soundness error is negligible.

Proof of claim 0: Just before the honest verifier chooses a bit c in step 4, for each of the pairs (u, v), let ${}_ub_v$ be a bit that minimizes the probability that an honest verifier will accept a reveal of the (u, v) commitment as being to the bit $1 - {}_ub_v$, with ties broken towards 0.

It is important that one does *not* need to assume that anyone *knows* these bits ${}_ub_v$. Although there are more sophisticated notions of commitment which would enforce such knowledge, the standard notion of commitment does not necessarily do so. Instead, the bits $1 - {}_ub_v$ correspond to the bits b from the binding experiment.

For the purpose of proving that, I say that $\mathcal{A}_{\text{prv}}$ *breaks multi-binding* if and only if there is a pair (u, v) such that the verifier accepts a reveal of the (u, v) commitment as being to $1 - {}_ub_v$ is negligible.

Claim 1: The probability that $\mathcal{A}_{\text{prv}}$ breaks multi-binding is negligible.

Proof of Claim 1:

Consider the the adversary $\mathcal{A}_{\text{com}}$ which works as follows.

- $\mathcal{A}_{\text{com}}$ chooses one of the pairs (u, v) uniformly at random.
- Have $\mathcal{A}_{\text{prv}}$ interact with an honest verifier, except that the (u, v) commitment is part of the binding experiment instead of with the verifier.

subclaim: The probability that $\mathcal{A}_{\text{prv}}$ breaks multi-binding is negligible is at most $(n \cdot (n-1))/2$ times the probability of $\mathcal{A}_{\text{com}}$ succeeding in the binding experiment.

Proof of subclaim: The commitments and reveals are independent of which pair (u, v) was chosen by $\mathcal{A}_{\text{com}}$, so the bits ${}_ub_v$ are also independent of which pair

(u, v). Similarly, the bit b for the binding experiment equals $1 - {}_u b_v$. Since the reveals are independent of which pair (u, v) was chosen by $\mathcal{A}_{\text{com}}$, the probability that $\mathcal{A}_{\text{com}}$ succeeds in the binding experiment is at least the probability that $\mathcal{A}_{\text{prv}}$ breaks multi-binding divided by the number of pairs (u, v). That number is n choose 2, which equals $(n \cdot (n-1))/2$.□

Thus, by the binding property of the commitment scheme, Claim 1 holds.□

Now, *other than* when $\mathcal{A}_{\text{prv}}$ breaks multi-binding, it can only reveal the commitments to the corresponding bits ${}_u b_v$, and those were *determined* before the verifier chose its bit c, even if they are not *known*. In such cases, the reasoning from the boxes protocol works here too. Given a valid response for $c = 1$ *and* a valid response for $c = 0$, one can easily find a Hamiltonian path in G. That is because f must send non-edges to pairs that randpath does *not* follow, so $f^{-1} \circ \text{randpath}$ must be a path that follows G's edges. Thus, $f^{-1} \circ \text{randpath}$ is a Hamiltonian path in G.

Therefore the special-soundness error is at most the probability that $\mathcal{A}_{\text{prv}}$ breaks multi-binding, which is negligible by Claim 1. This proves Claim 0. □

Finally, using the pre-simulator and Claim 0, one can build a computational zero-knowledge argument of knowledge for Hamiltonian Path by sequential composition of the basic protocol for Hamiltonian Path, just as was done for Graph Isomorphism. If the commitment scheme is statistically binding, then this gives a

computational zero-knowledge proof of knowledge, and if the commitment scheme is statistically hiding, then this gives a statistical zero-knowledge argument of knowledge.

2.5 Commitment Schemes

2.5.1 Naor Commitment

The simplest commitment scheme uses a *pseudorandom generator*, and is due to [25]. For bit commitment, it uses an efficiently-computable function PRG : $\{0,1\}^k \to \{0,1\}^{3 \cdot k}$ such that the outputs of PRG on uniformly random inputs are computationally indistinguishable from uniformly random elements of $\{0,1\}^{3 \cdot k}$. In other words, one needs that for all feasible functions $\mathcal{D} : \{0,1\}^{3 \cdot k} \to \{0,1\}$, when $x \in \{0,1\}^k$ and $y \in \{0,1\}^{3 \cdot k}$ are chosen uniformly at random

$$|\text{Prob}(\mathcal{D}(f(x)) = 1) - \text{Prob}(\mathcal{D}(y) = 1)|$$

is negligible.

For such a pseudorandom generator PRG, the bit commitment scheme works as follows, where $\oplus$ denotes bitwise exlusive-or.

- The receiver chooses $r \in \{0,1\}^{3 \cdot k}$ uniformly at random, and sends r to the committer.

- The committer choses $x \in \{0,1\}^k$ uniformly at random.
- If $b = 0$ then the committer sends $\mathrm{PRG}(x)$ to the receiver.

  If $b = 1$ then the committer sends $\mathrm{PRG}(x) \oplus r$ to the receiver.
- To reveal the committed bit, the committer sends b and x to the receiver.
- Let y by the value sent by the committer in the commit phase.

  If $b = 0$ and $\mathrm{PRG}(x) = y$ then the receiver outputs 0.

  If $b = 1$ and $\mathrm{PRG}(x) \oplus r = y$ then the receiver outputs 1.

  If neither of those are the case then the receiver rejects.

Efficiency and correctness both clearly hold. This scheme has simple reveal, so one can use the simpler definition of binding. Suppose

$$\mathrm{PRG}(x_0) = y = \mathrm{PRG}(x_1) \oplus r$$

In that case, $r = \mathrm{PRG}(x_1) \oplus \mathrm{PRG}(x_1) \oplus r = \mathrm{PRG}(x_1) \oplus \mathrm{PRG}(x_0)$. However, there are only 2^k possibilities for x_0 and for x_1, whereas r gets chosen uniformly at random from a set with $2^{3 \cdot k}$ elements. Thus, at most a 2^{-k} fraction of the possibilities for r are such that there is a way to break binding. Thus this scheme is statistically binding.

To see that this scheme satisfies computational hiding, note that for $r \in \{0,1\}^{3 \cdot k}$ chosen by the adversary, for $x \in \{0,1\}^k$ and $y \in \{0,1\}^{3 \cdot k}$ chosen uni-

formly at random, the distribution of $\mathrm{PRG}(x)$ is computationally indistinguishable from the distribution of y, which is identical to the distribution of $y \oplus r$, which is computationally indistinguishable from the distribution of $\mathrm{PRG}(x) \oplus r$. To see the last of those three relations, note that if it would help, the distinguisher $\mathcal{D}$ for PRG could simply start by applying $\oplus r$ to its input.

2.5.2 Commitment from Collision-Resistance

The next-simplest bit-commitment scheme uses a *collision-resistant hash family*. For bit commitment, suppose the receiver can efficiently sample from a distribution of boolean circuits computing functions from $\{0,1\}^{2 \cdot k}$ to $\{0,1\}^k$ such that, given the circuit H, it is infeasible to find strings x_0 and x_1 in $\{0,1\}^{2 \cdot k}$ such that $H(x_0) = H(x_1)$ and $x_0 \neq x_1$. For such a family of circuits, the bit commitment scheme works as follows, where $\oplus$ denotes exclusive-or and $\odot$ denotes inner product modulo 2.

- The receiver samples H from the distribution of circuits, and sends H to the committer.
- The committer chooses r and x uniformly at random from $\{0,1\}^{2 \cdot k}$, and sends r and $H(x)$ and $(r \odot x) \oplus b$ to the receiver.
- To reveal the committed bit, the committer sends x to the receiver.

- Let r, h, a be the values sent by the committer in the commit phase. If $H(x) = h$, then the receiver outputs $(r \odot x) \oplus a$, else the receiver rejects.

Efficiency clearly holds. Since $(r \odot x) \oplus (r \odot x) \oplus b = b$, completeness also holds. As was the case for the scheme from a pseudorandom generator, this scheme has simple reveal. For the simpler definition of binding, the adversary succeeding against this commitment scheme requires that the adversary find a collision for the circuit H chosen by the receiver. Thus this scheme is computationally binding.

Statistical Hiding: For all elements x_0 and x_1 of $\{0,1\}^{2 \cdot k}$, if $x_0 \neq x_1$ then exactly half of the elements $r \in \{0,1\}^{2 \cdot k}$ satisfy $r \odot x_0 = r \odot x_1$. One can see this by considering a position i as which x_0 differs from x_1, fixing the bits of r at all other positions, and then observing that exactly one of the two possibilities for the remaining bit of r is such that $r \odot x_0 = r \odot x_1$. Thus, the family of functions $x \mapsto r \odot x$ from $\{0,1\}^{2 \cdot k}$ parameterized by $r \in \{0,1\}^{2 \cdot k}$ is a 2-universal hash family, as defined in the appendix, which allows the use of the leftover hash lemma, as stated and proved in the appendix.

Fix the function H chosen by the verifier, and then partition $\{0,1\}^{2 \cdot k}$ according to the outputs of H. Since there only 2^k outputs, fewer than $2^{(5/3) \cdot k}$ inputs are in parts of size less than $2^{(2/3) \cdot k}$. Call those parts the *small* parts and the remaining parts the *large* parts, and let W be the union of the large parts. Since there are

$2^{2 \cdot k}$ possible inputs, that means a uniformly random input has probability less than $2^{-k/3}$ of being in a small part. In all other cases, the input is in $\mathcal{W}$.

For each non-empty subset S of $\{0,1\}^{2 \cdot k}$, let precom_S denote the distribution of $(r, H(x), r \odot x)$ conditioned on $x \in S$, and let U_S denote the distribution of $(r, H(x), b)$ for $b \in \{0,1\}$ chosen independently and uniformly at random, still conditioned on $x \in S$. By the leftover hash lemma, for each large piece $\mathcal{X}$, there is no way to distinguish $\text{precom}_{\mathcal{X}}$ from $U_{\mathcal{S}}$ by at least $2^{-k/3}$. That is because the values $H(x)$ will all be the same, so if it would help then a distinguisher could just start by putting that value in. The distribution $\text{precom}_{\mathcal{W}}$ is a mixture the distributions $\text{precom}_{\mathcal{X}}$ for large parts $\mathcal{X}$, and similarly for $U_{\mathcal{W}}$ and the distributions $U_{\mathcal{X}}$. Thus, by weighting the triangle inequality, there is no way to distinguish $\text{precom}_{\mathcal{W}}$ from $U_{\mathcal{W}}$ by at least $2^{-k/3}$.

Set $\text{precom} = \text{precom}_{\{0,1\}^{2 \cdot k}}$ and $U = U_{\{0,1\}^{2 \cdot k}}$. The distribution precom is a mixture of $\text{precom}_{\mathcal{W}}$ with another distribution where the weight of the latter is less than $2^{-k/3}$, so there is no way to distinguish precom from $\text{precom}_{\mathcal{W}}$ by at least $2^{-k/3}$. The same applies to U and $U_{\mathcal{W}}$. Thus, by the triangle inequality, there is no way to distinguish precom from U by at least $3 \cdot 2^{-k/3}$.

Now, consider commitments to 0, commitments to 1, and what I'll call pseudo-commitments, which are obtained by using U instead of precom. The bit being committed or pseudo-committed to only affects the last entry of the triple, and

for the triples from U, the last entry is a bit chosen uniformly and independently of the other two entries. Thus, for the pseudo-commitments, the bit the bit being pseudo-committed to does not affect the distribution of outputs.

Furthermore, for each fixed bit being committed or pseudo-committed to, a distinguisher could start by xor-ing the last entry of the triple with the fixed bit, so there is no way to distinguish commitments from pseudo-commitments by at least $3 \cdot 2^{-k/3}$. Thus, by the triangle inequality, there is no way to distinguish commitments to 0 from commitments to 1 by at least $6 \cdot 2^{-k/3}$. Therefore, this commitment scheme is statistically hiding.

2.5.3 Binding vs Hiding

In general, statistical hiding is better. This is because statistically binding commitments can be stored for a long time and perhaps eventually forced open, either due to continually increasing computational resources or due to someone finding a fast attack against the computational assumption. On the other hand, with statistical hiding, breaking the computational assumption in either of those two senses, a fast attack or way more computational resources than planned for, only matters if it happens while the receiver is still ready to accept a reveal of the commitment. With statistical hiding, the property which can remain relevant for a longer period of time holds unconditionally.

However, statistical hiding seems to require either lots of interaction [20] or stronger computational assumptions [18]. The assumption of a pseudo-random generator is a minimal assumption for bit commitment [5], even when both hiding and binding are just computational [19], and gives statistically binding commitments with barely any interaction needed.

2.6 Beyond NP

The construction of zero-knowledge protocols based on reduction to an NP problem requires that the size of the resulting instance correspond to an upper-bound on worst-case amount of time needed to check original the witness relation. For that reason, essentially everything about the protocol grows with that upper bound, in addition to growing with the security parameter.

[2] gives a way of avoiding this. Given a collision-resistant hash family, it shows how to construct arguments for which the amount of communication needed and the verifier's runtime both only depend on the security parameter and the length of the *instance*, whereas the prover is still efficient when additionally given the witness. [9] shows how to extend those arguments to show *quasi-knowledge*, in a sense defined there.

However, these all involve an efficient *deterministic* algorithm for the witness relation. What if one only has a randomized algorithm for it?

For example, suppose that Peggy and Victor like to play m-by-n minesweeper with i mines, and Peggy claims that she knows a small and fast algorithm which does very well at the game with those parameters. Given a claimed such algorithm, one can easily estimate how well it does by simply running it in independent games with those parameters. However, that involves using randomness to choose where to put the mines in those runs, so there is not obviously a suitable NP relation. Is there nonetheless some way for Peggy to convince Victor that she knows a such an algorithm, without revealing the algorithm to Victor?

Chapter 4 of my thesis resolves this issue by showing how to go from zero-knowledge protocols for NP relations to zero-knowledge protocols for relations R such that evaluating xRw uses randomness.

2.7 Appendix

Let $|\cdot|_1$ denote the ℓ_1 norm. In order to show the relevance of the leftover hash lemma, I begin by showing that the ℓ_1-norm bounds the amount by which finite distributions can be distinguished.

Statistical Distance Claim: For all finite sets S, for all probability distributions $X : S \to [0,1]$ and $Y : S \to [0,1]$, for all possibly-randomized functions $\mathcal{D} : S \to \{0,1\}$, we have

$$|\text{Prob}_{x \leftarrow X}(\mathcal{D}(x) = 1) - \text{Prob}_{y \leftarrow Y}(\mathcal{D}(y) = 1)| \leq \frac{1}{2} \cdot |X - Y|_1$$

Definition: For non-empty finite sets A and B, a *2-universal hash family* from A to B consists of a non-empty finite set I and a function $\mathcal{H} : I \times A \to B$ such that for all elements x and y of A, if $x \neq y$ then $\text{Prob}_{i \leftarrow I}(\mathcal{H}(i,x) = \mathcal{H}(i,y)) \leq \frac{1}{|B|}$.

Leftover Hash Lemma: For all non-empty finite sets A and B, for all 2-universal hash families $\mathcal{H} : I \times A \to B$, for all distributions $\mathcal{X}$ on A,

$$\frac{1}{2} \cdot |\mathbf{p} - \mathbf{u}|_1 < \frac{1}{2} \cdot \sqrt{|B| \cdot \underset{x \leftarrow \mathcal{X}, y \leftarrow \mathcal{X}}{\text{Prob}}(x = y)}$$

where $\mathbf{p} : I \times B \to [0,1]$ is the distribution of pairs $(i, \mathcal{H}(i,x))$ for i chosen uniformly from I and x chosen from $\mathcal{X}$, and $\mathbf{u} : I \times B \to [0,1]$ is the uniform distribution.

The point of the leftover hash lemma is, if $\mathcal{X}$ has significantly more collision-entropy that the uniform distribution on B, then applying a random member of a

2-universal hash family to a sample from $\mathcal{X}$ produces an element of B that is close to uniformly random, even given *which* member of the hash family was applied.

Proof of statistical distance claim. For each $s \in S$, let $d_s = \text{Prob}(\mathcal{D}(s) = 1)$. One has

$$\text{Prob}_{x \leftarrow X}(\mathcal{D}(x) = 1) \ = \ \sum_{s \in S}(X(s) \cdot d_s) = \sum_{s \in S}(d_s \cdot X(s))$$

and similarly $\text{Prob}_{y \leftarrow Y}(\mathcal{D}(y) = 1) \ = \ \sum_{s \in S}(d_s \cdot Y(s))$, so

$$\text{Prob}_{x \leftarrow X}(\mathcal{D}(x) = 1) - \text{Prob}_{y \leftarrow Y}(\mathcal{D}(y) = 1) \ = \ \sum_{s \in S}(d_s \cdot (X(s) - Y(s)))$$

Since each d_s is a probability, one has $0 \leq d_s \leq 1$ for all $s \in S$. Fix X and Y. The values of s such that $X(s) = Y(s)$ are irrelevant, and otherwise one extremizes $\sum_{s \in S}(d_s \cdot (X(s) - Y(s)))$ as follows.

- One minimizes $\sum_{s \in S}(d_s \cdot (X(s) - Y(s)))$ by setting

  $d_s = 0$ when $X(s) > Y(s)$ and $d_s = 1$ when $X(x) < Y(s)$.

- One maximizes $\sum_{s \in S}(d_s \cdot (X(s) - Y(s)))$ by setting

  $d_s = 1$ when $X(s) > Y(s)$ and $d_s = 0$ when $X(x) < Y(s)$.

Thus, when $S_X = \{s \in S : X(s) > Y(s)\}$ and $S_Y = \{s \in S : X(s) < Y(s)\}$, one has the following bounds.

$$\sum_{s \in S_Y}(X(s) - Y(s)) \leq \sum_{s \in S}(d_s \cdot (X(s) - Y(s))) \leq \sum_{s \in S_X}(X(s) - Y(s))$$

Now

$$\begin{aligned}
&\sum_{s\in S_Y}(X(s)-Y(s))+\sum_{s\in S_X}(X(s)-Y(s))\\
&=\sum_{s\in S_Y\cup S_X}(X(s)-Y(s))\\
&=\sum_{s\in S}(X(s)-Y(s))\\
&=\sum_{s\in S}X(s)-\sum_{s\in S}Y(s)\\
&=1-1=0
\end{aligned}$$

so $\sum_{s\in S_Y}(X(s)-Y(s))=-\sum_{s\in S_X}(X(s)-Y(s))$. Furthermore, by the choice of S_Y,

$$-\sum_{s\in S_Y}(X(s)-Y(s))=\sum_{s\in S_Y}-(X(s)-Y(s))=\sum_{s\in S_Y}|X(s)-Y(s)|$$

and $\sum_{s\in S_X}(X(s)-Y(s))=\sum_{s\in S_X}|X(s)-Y(s)|$, so

$$\begin{aligned}
2\cdot\sum_{s\in S_X}(X(s)-Y(s))&=-\sum_{s\in S_Y}(X(s)-Y(s))+\sum_{s\in S_X}(X(s)-Y(s))\\
&=\sum_{s\in S_Y}|X(s)-Y(s)|+\sum_{s\in S_X}|X(s)-Y(s)|\\
&=\sum_{x\in S_Y\cup S_X}|X(s)-Y(s)|\\
&=\sum_{x\in S}|X(s)-Y(s)|\\
&=|X-Y|_1
\end{aligned}$$

Thus $\sum_{s \in S_X} (X(s) - Y(s)) = \frac{1}{2} \cdot |X - Y|_1$, and

$$-\frac{1}{2} \cdot |X - Y|_1 \leq \sum_{s \in S} (d_s \cdot (X(s) - Y(s))) \leq \frac{1}{2} \cdot |X - Y|_1$$

Therefore

$$\begin{aligned} &|\text{Prob}_{x \leftarrow X}(\mathcal{D}(x) = 1) - \text{Prob}_{y \leftarrow Y}(\mathcal{D}(y) = 1)| \\ &= \left| \sum_{s \in S} d_s \cdot (X(s) - Y(s)) \right| \leq \frac{1}{2} \cdot |X - Y|_1 \end{aligned}$$

□

Leftover Hash Lemma, restated: If A and B are non-empty finite sets, and $\mathcal{H} : I \times A \to B$ is 2-universal hash family, and $\mathcal{X}$ is a distribution on A, then

$$\frac{1}{2} \cdot |\mathbf{p} - \mathbf{u}|_1 < \frac{1}{2} \cdot \sqrt{|B| \cdot \underset{x \leftarrow \mathcal{X}, y \leftarrow \mathcal{X}}{\text{Prob}}(x = y)}$$

where $\mathbf{p} : I \times B \to [0, 1]$ is the distribution of pairs $(i, \mathcal{H}(i, x))$ for i chosen uniformly, and x chosen from $\mathcal{X}$, and $\mathbf{u} : I \times B \to [0, 1]$ is the uniform distribution.

The following proof closely tracks the proof from [LHL] of a slightly weaker version of the statement.

Proof of Leftover Hash Lemma. Regard $\mathbf{p}$ and $\mathbf{u}$ as vectors in the inner product space $\mathbb{R}^{H \times B}$ with the usual inner product, let $|| \cdot ||_2$ denote the Euclidean norm, and set $\text{col}(\mathcal{X}) = \underset{x \leftarrow \mathcal{X}, y \leftarrow \mathcal{X}}{\text{Prob}}(x = y)$. By the Cauchy-Schwarz inequality,

$$|\mathbf{p} - \mathbf{u}|_1 \leq \sqrt{|I| \cdot |B|} \cdot ||\mathbf{p} - \mathbf{u}||_2$$

Since $\mathbf{p}$ and $\mathbf{u}$ are probability vectors, the sum of the entries in $\mathbf{p} - \mathbf{u}$ is zero, so $\mathbf{p} - \mathbf{u}$ is orthogonal to $\mathbf{u}$. Since $\mathbf{p} = (\mathbf{p} - \mathbf{u}) + \mathbf{u}$, by the Pythagorean theorem, one has $||\mathbf{p} - \mathbf{u}||_2 = \sqrt{||\mathbf{p}||_2^2 - ||\mathbf{u}||_2^2}$.

Now, $||\mathbf{u}||_2^2 = |I| \cdot |B| \cdot \frac{1}{(|I| \cdot |B|)^2} = \frac{1}{|I| \cdot |B|}$, so

$$\begin{aligned} |\mathbf{p} - \mathbf{u}|_1 &\leq \sqrt{|I| \cdot |B|} \cdot ||\mathbf{p} - \mathbf{u}||_2 \\ &= \sqrt{|I| \cdot |B|} \cdot \sqrt{||\mathbf{p}||_2^2 - ||\mathbf{u}||_2^2} \\ &= \sqrt{|I| \cdot |B|} \cdot \sqrt{||\mathbf{p}||_2^2 - \frac{1}{|I| \cdot |B|}} \end{aligned}$$

Most of what's left is bounding $||\mathbf{p}||_2^2$. Now $||\mathbf{p}||_2^2 = \sum_{(i,b) \in I \times B} (\mathbf{p}(i,b))^2$ is the probability that two independent samples from $\mathbf{p}$ are equal. Thus,

$$\begin{aligned} ||\mathbf{p}||_2^2 &= \operatorname*{Prob}_{\substack{i \leftarrow I, x \leftarrow \mathcal{X} \\ j \leftarrow I, y \leftarrow \mathcal{X}}} \Big[(i, \mathcal{H}(i,x)) = (j, \mathcal{H}(j,y)) \Big] \\ &= \operatorname*{Prob}_{\substack{i \leftarrow I, x \leftarrow \mathcal{X} \\ j \leftarrow I, y \leftarrow \mathcal{X}}} \Big[i = j \ \wedge \ \mathcal{H}(i,x) = \mathcal{H}(j,y) \Big] \end{aligned}$$

Split this event into when $x = y$ and when $x \neq y$. One gets the following bound for the $x = y$ part of the event.

$$\begin{aligned}
&\operatorname*{Prob}_{\substack{i \leftarrow I, x \leftarrow \mathcal{X} \\ j \leftarrow I, y \leftarrow \mathcal{X}}} \Big[i = j \ \wedge \ \mathcal{H}(i,x) = \mathcal{H}(j,y) \ \wedge \ x = y \Big] \\
&= \operatorname*{Prob}_{\substack{i \leftarrow I, x \leftarrow \mathcal{X} \\ j \leftarrow I, y \leftarrow \mathcal{X}}} \Big[i = j \ \wedge \ x = y \Big] \\
&= \operatorname*{Prob}_{i \leftarrow I, j \leftarrow I} (i = j) \cdot \operatorname*{Prob}_{x \leftarrow \mathcal{X}, y \leftarrow \mathcal{X}} (x = y) \\
&= \frac{1}{|I|} \cdot \mathrm{col}(X)
\end{aligned}$$

Bounding the $x \neq y$ part of the event is more complicated.

$$\begin{aligned}
&\operatorname*{Prob}_{\substack{i \leftarrow I, x \leftarrow \mathcal{X} \\ j \leftarrow I, y \leftarrow \mathcal{X}}} \Big[i = j \ \wedge \ \mathcal{H}(i,x) = \mathcal{H}(j,y) \ \wedge \ x \neq y \Big] \\
&= \frac{1}{|I|} \cdot \operatorname*{Prob}_{\substack{x \leftarrow \mathcal{X}, y \leftarrow \mathcal{X} \\ i \leftarrow I}} \Big[\mathcal{H}(i,x) = \mathcal{H}(i,y) \ \wedge \ x \neq y \Big]
\end{aligned}$$

By the 2-universality of $\mathcal{H}$, for each fixed x and y, if $x \neq y$ then

$$\mathrm{Prob}_{i \leftarrow I}[\mathcal{H}(i,x) = \mathcal{H}(i,y)] \ \leq \ \frac{1}{|B|}$$

Since $\mathcal{X}$ is a distribution on a finite set, that means

$$\operatorname*{Prob}_{\substack{x \leftarrow \mathcal{X}, y \leftarrow \mathcal{X} \\ i \leftarrow I}} \Big[\mathcal{H}(i,x) = \mathcal{H}(i,y) \ \wedge \ x \neq y \Big] \ < \ \frac{1}{|B|}$$

Thus one gets the following bound for the $x \neq y$ part of the event.

$$\begin{aligned}
&\operatorname*{Prob}_{\substack{i \leftarrow I, x \leftarrow \mathcal{X} \\ j \leftarrow I, y \leftarrow \mathcal{X}}} \Big[i = j \ \wedge \ \mathcal{H}(i,x) = \mathcal{H}(j,y) \ \wedge \ x \neq y \Big] \\
&= \frac{1}{|I|} \ \cdot \operatorname*{Prob}_{\substack{x \leftarrow \mathcal{X}, y \leftarrow \mathcal{X} \\ i \leftarrow I}} \Big[\mathcal{H}(i,x) = \mathcal{H}(i,y) \ \wedge \ x \neq y \Big] \\
&< \frac{1}{|I|} \cdot \frac{1}{|B|} = \frac{1}{|I| \cdot |B|}
\end{aligned}$$

Combining the two bounds gives

$$||\mathbf{p}||_2^2 = \operatorname*{Prob}_{\substack{i \leftarrow I, x \leftarrow \mathcal{X} \\ j \leftarrow I, y \leftarrow \mathcal{X}}} \Big[i = j \ \wedge \ \mathcal{H}(i,x) = \mathcal{H}(j,y) \Big] \ < \frac{\mathrm{col}(X)}{|I|} + \frac{1}{|I| \cdot |B|}$$

Finally, plugging that in to the bound on $|\mathbf{p} - \mathbf{u}|_1$ gives

$$\begin{aligned}
|\mathbf{p} - \mathbf{u}|_1 < \ & \sqrt{|I| \cdot |B|} \cdot \sqrt{\frac{\mathrm{col}(X)}{|I|} + \frac{1}{|I| \cdot |B|} - \frac{1}{|I| \cdot |B|}} \\
&= \sqrt{|I| \cdot |B|} \cdot \sqrt{\frac{\mathrm{col}(X)}{|I|}} \\
&= \sqrt{|B| \cdot \mathrm{col}(X)}
\end{aligned}$$

so $\frac{1}{2} \cdot |\mathbf{p} - \mathbf{u}|_1 < \frac{1}{2} \cdot \sqrt{|B| \cdot \mathrm{col}(X)} = \frac{1}{2} \ \cdot \ \sqrt{|B| \ \cdot \operatorname*{Prob}_{x \leftarrow \mathcal{X}, y \leftarrow \mathcal{X}} (x = y)}$. $\square$

Chapter 3

Obfuscation

In 2010, a group of leaders in the field of cryptography collaborated on the paper *On the (Im)possibility of Obfuscating Programs* [10], in which they stated "However, there still remain several important problems in cryptography about which theory has had little or nothing to say. One such problem is that of *program obfuscation*." (emphasis in original). Theory has since had lots to say on that problem [13], [1], [6], [12] , but so far nothing on one of the problems posed in [10], which is whether or not one can obfuscate *sampling algorithms*. Roughly speaking, *program obfuscation* is changing a program's source code to make the new source code *unintelligible*, without affecting the program's *functionality*. Ideally, one would want the new source code to reveal nothing more than what can be learned

from just querying the functionality: i.e., the new source code should be a *virtual black box* which provides just input-output access to the functionality.

In the context of obfuscation, when applied to general programs, the default *functionality* of a program P, is the function F given by $F(x) = (P(x), t)$ where t is a coarse approximation to the amount of time it takes for P to calculate $P(x)$.

One might wonder why the amount of time is involved in the functionality. There are two reasons for this. The simpler reason is that one would want obfuscations to be relatively efficient. In particular, one would want the obfuscated program to not take far longer to run than the original program. The more complicated reason is, even when the outputs are actually simple, being sure that the outputs are simple can be hard.

To illustrate the more complicated reason, suppose one has sets of programs S_0 and S_1 such that the programs in $S_0 \cup S_1$ run for a long time and the programs in S_0 compute the zero function and the programs in S_1 compute a function g such that all programs which compute g take a long time. To make this consideration more precise, note the following observations.

1. Obviously, there are extremely-efficient programs that compute the zero function.

2. With respect to the functionality F_0 given by $F_0(x) = P(x)$, fulfilling the *virtual black box* goal would require that obfuscations of programs in S_0 be

hard to distinguish from obfuscations of other programs that compute the zero function.

3. In order for the obfuscator to be relatively efficient, obfuscations of programs from (1) would have to be efficient.

4. By combining observations 2 and 3, obfuscations of programs in S_0 would have to be efficient.

5. By the assumption on S_1, obfuscations of programs in S_1 can't be efficient.

6. By running obfuscations of the input program for a moderate amount of time, one can reliably distinguish programs in S_0 from programs in S_1.

7. Observation 6 applies even when H is a set for which membership can be decided by a short program and on input x, the programs $P_y \in S_0 \cup S_1$ are "If $y \in H$ then output $g(x)$ else output 0.".

8. Observation 7 applies even then L is a positive integer and H is a set of binary strings of length L which maximizes size of circuits needed to reliably distinguish H from its complement.

9. By brute force over the possibilities for H, there is a short program that decides membership in such an H.

Thus, with respect to the functionality F_0 given by $F_0(x) = P(x)$, the existence of an efficient *virtual black box* obfuscator would imply that every set can of binary strings be reliably distinguished from its complement in a moderately-efficient way.

Obfuscation of general programs is not the main focus of cryptographic research on obfuscation. Research has focused primarily on obfuscation of circuits, since circuits are more restrictive. But a not very long bound on the runtime lets one convert a program into a circuit and most cryptographic uses of obfuscation know the security parameter before running the obfuscator and the runtime is bounded by a function of the *security parameter* that does not grow very quickly, and which controls the security level.

One example of using obfuscation for circuits, is witness encryption for NP relations, a concept I go into in section 3.4 . As far as I am aware, prior to my thesis, witness encryption has only been defined for NP relations. My thesis extends the definition of witness encryption to the class called promiseMA, which is the generalization of NP that allows randomized verifiers. In section 4.5, I show that given witness encryption for NP, one also gets witness encryption for promiseMA. Furthermore, I use this result to show that if witness encryption for NP exists, then the auxiliary input setting does not allow virtual black box obfuscation of sampling algorithms.

In the context of obfuscation, when applied to circuits, the default *functionality* of a circuit C is the function computed by C. Here, there is to include an amount of time t as part of the functionality, since circuits can always be evaluated efficiently given the input and the circuit.

The idea that obfuscation might be possible comes from the following. *Practical experience*, in which code tends to naturally become hard to understand if precautions are not taken against this. *Theorems and Conjectures* such as the Rice-Shapiro Theorem [29] and the Strong Exponential Time Hypothesis [7]. Combined these would appear to make looking at the code useless, so as written in [10] *the only useful thing that one can do with a program or circuit is to run it (on inputs of one's choice).* This is explained in section 2.

[10] proves that nonetheless, efficient *virtual black box* obfuscation is not possible, for programs or for circuits. [10] also asks the question of whether *virtual black box* obfuscation is possible for *sampling algorithms*, and that is the focus of my thesis. Sampling algorithms can similarly be divided into programs and circuits, although their functionalities are different from the default.

Let F be the default functionality of the program P. As a sampling algorithm, the *functionality* of P is the function $\hat{F}$ given by $\hat{F}(x)$ is the distribution of $F(r)$ for r chosen uniformly at random from inputs with the same length as x.

As a sampling algorithm, the *functionality* of a circuit C is the distribution of outputs of C on uniformly random inputs.

In the setting of [10], when defining what it would mean for a candidate obfuscator to be a virtual black-box obfuscator, the distinguisher is given *just* the obfuscated code. I do not achieve a result for when the distinguisher is given just the obfuscated code, but I do achieve a result for the auxiliary-input setting. This corresponds to the distinguisher having context information, rather than just being given the obfuscation in a vacuum.

The result is that there is a pair $\mathcal{C}_0, \mathcal{C}_1$ of families of sampling circuits, and an algorithm auxgen that takes as input the original code and an upper bound on the lengths of obfuscations of the code, such that (A) and (B) are true.

(A) There is no feasible algorithm $\mathcal{D}$ that distinguishes between $b = 0$ and $b = 1$ in the following.

- $\mathcal{D}$ chooses a feasible length L
- C is chosen from $\mathcal{C}_b$
- set $z = \text{auxgen}(C, L)$
- $\mathcal{D}$ receives z
- $\mathcal{D}$ can can request samples from $\hat{F}(C)$. Each time it does, $\mathcal{D}$ receives one sample from $\hat{F}(C)$.

(B) There is an efficient algorithm $\mathcal{D}$ that distinguishes between $b = 0$ and $b = 1$ in

- $\mathcal{D}$ chooses a feasible length L
- C is chosen from $\mathcal{C}_b$
- set $z = \text{auxgen}(C, L)$
- let C' be an obfuscation of C with length at most L
- $\mathcal{D}$ receives z
- $\mathcal{D}$ receives C'.

I show that if there are weak obfuscators with respect to the default functionality then there are no *virtual black box* obfuscators against auxiliary input for sampling algorithms.

3.1 Overview

The two theoretical reasons for thinking that obfuscation might be possible are the Rice-Shapiro theorem and the Strong Exponential Time Hypothesis. In section 3.2, I give the Rice-Shapiro theorem, and the computability background for it. In section 3.2.1, I give the Strong Exponential Time Hypothesis, and the logical background for it.

3.2 Computability and Complexity Background

A subset S of $\mathbb{N}$ is *computably enumerable* if and only if there is a Turing machine M such that for all $n \in \mathbb{N}$, M halts on input n if and only if $n \in S$. A *partial function* is like a function, but can have inputs for which it is undefined. (This includes ordinary functions, which do not have any inputs for which they are undefined.) Let $\mathrm{PF}(\mathbb{N})$ be the set of partial functions from $\mathbb{N}$ to $\mathbb{N}$. For any $f \in \mathrm{PF}(\mathbb{N})$, a Turing machine M computes f if and only if for all $n \in \mathbb{N}$,

- if $f(n)$ is defined then, on input n, M halts and its output is $f(n)$
- if $f(n)$ is undefined then, on input n, M does not halt.

The reason for this concept is that, similarly to the undecidability of the halting problem, there are computable $f \in \mathrm{PF}(\mathbb{N})$ such that f cannot be extended to a computable total function from $\mathbb{N}$ to $\mathbb{N}$, so there is no clear way to go from an arbitrary Turing machine to a total function from $\mathbb{N}$ to $\mathbb{N}$. On the other hand, it should be clear that each Turing machine computes exactly one $f \in \mathrm{PF}(\mathbb{N})$.

Let $\mathrm{CPF}(\mathbb{N})$ (for "computable partial functions") be the set of $f \in \mathrm{PF}(\mathbb{N})$ such that there exists a Turing machine that computes f. For f and g in $\mathrm{CPF}(\mathbb{N})$, one says that f *extends* g if and only if for all $n \in \mathbb{N}$, if $g(n)$ is defined then so is $f(n)$ and they are equal.

Fix a computable enumeration $M_0, M_1, M_2, M_3, ...$ of the set of all Turing machines. For example, one could go first by length and then lexicographically. Let $J : 2^{\mathrm{CPF}(\mathbb{N})} \to 2^{\mathbb{N}}$ be given by

$$J(S) = \{i \in \mathbb{N} : M_i\text{'s partial function is in } S\}$$

Theorem 3.2.1. *[Rice-Shapiro] For all subsets S of* $\mathrm{CPF}(\mathbb{N})$*, if $J(S)$ is computably enumerable then for all $f \in \mathrm{CPF}(\mathbb{N})$,*

$$f \in S \iff (\exists g \in S)\big(f \text{ extends } g \text{ and } |\{n \in \mathbb{N} : g(n) \text{ exists}\}| < \infty\big)$$

I now describe a relevant consequence of this theorem. For pairs

$$X = \{x_0, x_1, ..., x_{L-1}\}, \qquad Y = (y_0, y_1, ..., y_{L-1})$$

such that all variables on the right-hand side are natural numbers and $x_0 < x_1 < ... < x_{L-1}$, one can easily construct a Turing machine, $T(g)$, which computes the partial function g given by $g(x_i) = y_i$ for $0 \leq i < L$ and $g(n)$ is undefined for all other inputs. Any reasonable way of choosing such a Turing machine will work. By the Rice-Shapiro theorem, for all subsets S of $\mathrm{CPF}(\mathbb{N})$, if $J(S)$ is computably enumerable then the following holds.

One can computably enumerate $J(S)$ by halting on the indices i from the triples X, Y, i such that

- the pair $\{x_0, x_1, ..., x_{L-1}\}$, $(y_0, y_1, ..., y_{L-1})$ is as described above

- the index of $T(g)$ is in S
- M_i halts on all elements of X
- M_i produces and output of y_k for input x_k.

In particular, one can verify a computable partial function's membership in S by just *running* an algorithm for it, rather than using anything else about the algorithm's code.

As an interesting aside, this consequence also has a topological interpretation: When $\{\text{undefined}\} \cup \mathbb{N}$ is topologized by the basis

$$\{\{\text{undefined}\} \cup \mathbb{N}\} \cup \{\{n\} : n \in \mathbb{N}\}$$

the computably enumerable subsets of CPF($\mathbb{N}$) correspond to the computably enumerable unions of basic open subsets of $(\{\text{undefined}\} \cup \mathbb{N})^{\mathbb{N}}$.

3.2.1 The Strong Exponential Time Hypothesis

The other theoretical reason for thinking that obfuscation might be possible is the Strong Exponential Time Hypothesis (SETH). SETH concerns the complexity of the sequence of problems *k-SAT* as k goes to infinity, and those problems involve *k-CNF formulas*. Accordingly, I start by giving definitions leading up to a pair of definitions for what a *k-CNF formula* is, then define the sequence of problems, k-SAT, then describe what the Strong Exponential Time Hypothesis is.

k-CNF and k-SAT

A *propositional variable* is a variable whose universe is $\{\text{True}, \text{False}\}$. This is also known as a Boolean variable. A *literal* is a propositional variable or the negation of a propositional variable. For example, if x is a propositional variable, then "x" and "$\neg x$" are each literals. The *sign* of a literal indicates whether the literal is a propositional variable or the negation of a propositional variable: For a propositional variable x, x is a *positive* literal and $\neg x$ is a *negative* literal.

A *clause* is a disjunction of one or more literals. The *length* of a clause is the number of literals in the clause. For example, if w, x, y, z are propositional variables, then $w \vee \neg x \vee y \vee \neg z$ is a clause whose length is 4.

CNF is an acronym for *conjunctive normal form*. A *CNF formula* is a conjunction of clauses, for example $(a \vee b \vee c) \wedge (d \vee e) \wedge f$.

Definition A *k-CNF formula* is a CNF formula whose clauses each have length at most k.

If one instead requires the length to always equal k, then some results remains true, but with altered proofs. I call this the *strict* definition.

For example, if $C_0, C_1, C_2, C_3, C_4, C_5$ are clauses whose lengths are each 4, then $C_0 \wedge C_1 \wedge C_2 \wedge C_3 \wedge C_4 \wedge C_5$ is a 4-CNF-formula with six clauses under both definitions.

I use n to denote the number of variables in a CNF formula. In this exposition, I only consider values of k that are at most logarithmic in n, since for such k, unless one gets much more precise than I will need to get, it barely matters whether one:

(a) uses the strict definition for k-CNF formula

(b) is allowed to repeat propositional variables within a clause

(c) is allowed to repeat clauses within a CNF formula.

The unimportance of (a) is shown by introducing k new variables and the $2^k - 1$ clauses which rule out all assignments to those variables other than all of them being false, and putting some of those variables into the clauses that had fewer than k variables.

For (b), the literals could have the same sign or different signs. In the case of repeated literals with *different* signs, such as $x \vee y \vee \neg x$, the clause can just be deleted: At least one of the literals is always true, so the clause is always true.

Repeated literals with the *same* sign can trivially be replaced with just a single occurrence of the literal, such as going from $x \vee y \vee x$ to $x \vee y$.

Similarly, for (c), repetitions of clauses within a CNF formula can be replaced with just a single occurrence of the clause, such as going from $(w \vee x) \wedge (y \vee z) \wedge (w \vee x)$ to $(w \vee x) \wedge (y \vee z)$.

Repetitions as in (b) or (c) can efficiently be eliminated, so they only matter when the time needed to do so is at least a noticeable fraction of the time needed to handle the formula that results from eliminating those repetitions.

For the values of k that I consider, the runtimes of the algorithms that I will refer to and describe are approximately $2^{c \cdot n}$ for positive constants c, so unless one gets much more precise than I will need to get, the time needed to eliminate those repetitions can only matter if there is a positive integer r such that infinitely often, the total length of the formula is much more than $2^{n/r}$. Under those circumstances, the formula would necessarily be almost entirely repetition, since when neither (b) nor (c) are allowed, the formulas all have length much less than $2^{\left((\log(n))^3\right)}$.

Accordingly, in this exposition, for convenience I say that neither (b) nor (c) are allowed.

Following [11], for a k-CNF formula Φ, I define freq(Φ) as the maximum number of times a propositional variable occurs in Φ, and I define sol(Φ) to be the the set of satisfying *assignments* for Φ: i.e., the set of ways of assigning True or False to each variable so that Φ comes out as True.

k-SAT is the problem of, given as input a k-CNF-formula Φ, determining whether or not sol(Φ) has at least one assignment.

3.2.2 k-SAT and NP

We now discuss k-SAT and its Relation to the Class NP of Decision Problems: i.e., the Qualitative Hardness of k-SAT. It turns out that for $k \geq 3$, the problem k-SAT, as defined in the preceding section, is one of the hardest of the problems in the class of decision problems called NP.

Decision problems are problems such that for each input, the output is either *yes* or *no*. *NP* is the set of decision problems such that whenever the answer is *yes*, there is a polynomial-length proof of that which is verifiable in polynomial time. In other words, *NP* is the set of decision problems such that there is a polynomial p and a deterministic algorithm V such that the following hold.

- V takes as input the decision problem's input (the *instance*) and an alleged-proof, and outputs *accept* or *reject*.
- On input (x, y), the running time of V is at most $p(\text{length}(x) + \text{length}(y))$.
- For every *yes* instance x, there is at least one alleged-proof y such that $\text{length}(y) \leq p(\text{length}(x))$ and $V(x, y)$ accepts.
- For every *no* instance x and every alleged-proof y, $V(x, y)$ rejects.

In some cases, it may be possible to reduce a decision problem L_0 to a decision problem L_1. In general, a reduction from a problem L_0 to a problem L_1 is a way

of using a solver for L_1 to solve L_0. In order to give the relationship of k-SAT to NP, I introduce the most common type of reduction, although this section's conclusion applies even to far more restrictive notions of reduction.

A *polynomial-time many-one reduction* from a decision problem L_0 to a decision problem L_1 is a function f such that

- The inputs for f are the same as the inputs for L_0.
- f is computable in time bounded by a polynomial in the length of its input.
- For every instance x of L_0, then $f(x)$ is an instance of L_1 with the same answer.

The existence of such a function f limits how much harder L_0 can be than L_1, since it lets one solve an instance x of L_0 by solving the instance $f(x)$ of L_1.

For any given notion of reduction, one has the following:

- A decision problem L_0 is *reducible* to a decision problem L_1 if and only if there is a *reduction* from L_0 to L_1.
- A decision problem L is *NP-hard* if and only if *every* problem in NP is reducible to L.
- A problem is *NP-complete* if and only if the problem is in NP and is NP-hard.

Recall that k-SAT is the problem of, given as input a k-CNF-formula Φ, determining whether or not $\text{sol}(\Phi)$ has at least one assignment. Accordingly, for each k, the problem k-SAT is in NP because $V(\Phi, y)$ can simply check whether or not y is an assignment in $\text{sol}(\Phi)$. Furthermore, it is known that, for every integer $k \geq 3$, the problem k-SAT is NP-hard.

The NP-hardness of 3-SAT was first shown in [3]. For $k_0 < k_1$, the problem k_1-SAT cannot be more than a tiny bit easier than k_0-SAT, so k-SAT is also NP-hard for $k > 3$. Thus, by definition, given that k-SAT is in NP and it is also NP-hard for $k \geq 3$, therefore the problem k-SAT is NP-complete for $k \geq 3$.

Roughly speaking, this means k-SAT for $k \geq 3$ are among the hardest problems in NP.

3.2.3 Strong Exponential Time Hypothesis

The Strong Exponential Time Hypothesis concerns the complexity of solving k-SAT, and I am going to explain how one arrives at it.

One can trivially solve k-SAT instances in an amount of time that scales roughly as 2^n, by simply trying each of the 2^n possible assignments to the variables. One might wonder: How much faster than that can one solve k-SAT? Well, somewhat signficant improvements *are* known. There are known algorithms that

(a) solve 3-SAT in time between $2^{0.3862 \cdot n}$ [21] and $2^{0.4118 \cdot n}$ [23]

(b) solve 4-SAT in time between $2^{0.554 \cdot n}$ [27] and $2^{0.585 \cdot n}$ [24]

(c) for $k > 4$, solve k-SAT in time $2^{(1-(\mu/k)) \cdot n}$, where μ depends on k and whether the algorithm can be randomized or must be deterministic.

For part (c), with currently known algorithms [23] , [27], each such μ will be between 4/3 and 2. The values of μ are far less important than the observation that the values of $1 - (\mu/k)$ approach 1 as k goes to infinity. One might further wonder: Can one solve k-SAT in time $2^{c \cdot n}$ for a *universal* constant $c < 1$? SETH is essentially the hypothesis that the answer to that question is no.

(SETH) The *Strong Exponential Time Hypothesis* states that solving k-SAT in time $2^{c_k \cdot n}$ requires that c_k go to 1 as k goes to infinity.

Other than [30], I am not aware of any results regarding whether or not the hypothesis depends on the type of algorithm or depends on whether one requires *for infinitely many n* or *for all but finitely many n*. Accordingly, I am not aware of any proof of equivalence between the different ways in which one could formalize SETH.

However, SETH *is* known to be robust in four ways:

1. If SETH is true, then there is a sequence of constants $d_3, d_4, d_5, ...$ such that SETH remains true when restricting to k-SAT instances Φ such that $\text{freq}(\Phi) \leq d_k$.

2. It does not matter whether one asserts

   $\liminf_{k\to\infty} c_k = 1$ or $\sup(\{c_3, c_4, c_5, ...\}) = 1$.

3. One can require that, on input Φ, the solver *find* an assignment

   in $\mathrm{sol}(\Phi)$ whenever that set has at least one assignment.

4. For randomized algorithms, if SETH is true then it remains true when restricting to k-SAT instances Φ such that $\mathrm{sol}(\Phi)$ has at most one assignment.

Since each clause has at least one variable, a k-SAT instance Φ such that $\mathrm{freq}(\Phi) \leq d_k$ has at most $d_k \cdot n$ clauses. Thus, it follows from (1) that, if SETH is true then it remains true when one restricts the number of clauses so that for each k, the number of clauses is at most linear in n.

The justification of those four claims follows.

Proof of claim (1). For each integer $k \geq 3$ and Δ, let c_k be such that solving k-SAT uses time $2^{c_k \cdot n}$ and let $c_{k,\Delta}$ be such that solving k-SAT on instances Φ with $\mathrm{freq}(\Phi) \leq \Delta$ uses time $2^{c_{k,\Delta} \cdot n}$. Increasing Δ only allows more instances, so $c_{k,\Delta}$ is non-decreasing in Δ, which means $\lim_{\Delta\to\infty} c_{k,\Delta}$ exists.

Claim (Weird): For all integers $k \geq 3$, one has that $c_k \leq \lim_{\Delta\to\infty} c_{k,\Delta}$.

The proof of that claim involves the following definition and lemma.

For an algorithm A that produces a finite sequence of outputs, we say A has *delay* t if and only if for all inputs to A,

- A eventually halts
- A takes at most time t between each pair of consecutive *events*, where the *events* are A starting, and A halting, and each time A gives an output

The reference for the following lemma
does not use the strict definition of k-CNF.

Lemma 3.2.1 (Sparsification Lemma (ref [cwcd]))**.** *There is a deterministic algorithm A and a function f so that $\forall k, \epsilon > 0$ and $\Phi \in k$-CNF then $A(k, \epsilon, \Phi)$ outputs $\Phi_1, ..., \Phi_s \in k$-CNF s.t.*

1. $s \leq 2^{\epsilon \cdot n}$
2. $\text{sol}(\Phi) = \cup_i \text{sol}(\Phi_i)$
3. $\forall i \ \text{freq}(\Phi_i) \leq f(k, \epsilon)$
4. *each Φ_i is output with* $\text{poly}(n)$ *delay, although the degree of the polynomial may depend on k, ϵ*
5. $f(k, \epsilon) \in \text{poly}(\frac{1}{\epsilon})$*, but the degree of the polynomial may depend on k*

The outputs that A produces do not qualify as k-CNF formulas under the strict definition. My proof handles going back to the strict definition.

Proof of Claim: Let A and f be as in the lemma. Fix an integer $k \geq 3$ and a real number $0 < \epsilon < 1$, let $d = f(k, \epsilon)$ as in part 3 of the lemma, let n be an integer such that

$$\frac{2 \cdot k}{\epsilon} \leq n \tag{3.1}$$

and let Φ be an input k-CNF formula that has n variables. Run $A(k, \epsilon, \Phi)$, and for each of its outputs Φ_i do the following.

A clause if *short* if it has *length* $< k$. We *expand* the short clauses to have length k as follows. Split the short clauses in Φ_i into $\lfloor \frac{\epsilon \cdot n}{k} \rfloor$ sets of approximately equal size, called *groups*. Call the variables in Φ_i the *old* variables. For each group G, introduce k *new* variables and the $2^k - 1$ clauses which rule out all assignments to the new variables other than all of the new variables being false, and add one or more of the new variables to the clauses in G so that the resulting clauses are not short.

Let Φ' be the resulting formula, and note that Φ' is a k-CNF formula the strict definition. The formula Φ' has k new variables for each of the $\lfloor \frac{\epsilon \cdot n}{k} \rfloor$ groups, so since $\lfloor k \cdot \frac{\epsilon \cdot n}{k} \rfloor \leq k \cdot \frac{\epsilon \cdot n}{k} = \epsilon \cdot n$ and there are n old variables, Φ_i has at most $(1+\epsilon) \cdot n$ variables.

Since $\text{freq}(\Phi_i) \leq d$ by the choice of d, and each clause has at least one variable, Φ_i has at most $d \cdot n$ clauses, so in particular Φ_i has at most $d \cdot n$ short clauses.

Since those were split into $\lfloor \frac{\epsilon \cdot n}{k} \rfloor$ groups of approximately equal size, each group has at most $\lceil (d \cdot n) \big/ \lfloor \frac{\epsilon \cdot n}{k} \rfloor \rceil$ clauses.

Using (3.1) gives that $\frac{\epsilon \cdot n}{k} \geq (\epsilon \cdot \frac{2 \cdot k}{\epsilon})/k = 2$, then

$$\left\lfloor \frac{\epsilon \cdot n}{k} \right\rfloor > \frac{\epsilon \cdot n}{k} - 1 \geq \frac{\epsilon \cdot n}{2 \cdot k}$$

so

$$\left\lceil (d \cdot n) \Big/ \left\lfloor \frac{\epsilon \cdot n}{k} \right\rfloor \right\rceil \leq \left\lceil (d \cdot n) \Big/ \left(\frac{\epsilon \cdot n}{2 \cdot k} \right) \right\rceil = \left\lceil \frac{2 \cdot k \cdot d}{\epsilon} \right\rceil < \frac{2 \cdot k \cdot d}{\epsilon} + 1$$

Thus each group has fewer than $\frac{2 \cdot k \cdot d}{\epsilon} + 1$ clauses. In Φ', for each group G, each of the k new variables introduced for G occurs in each of the $2^k - 1$ expanded clauses, and can occur in any of the at most $\frac{2 \cdot k \cdot d}{\epsilon}$ clauses obtained from the clauses in G by adding one or more of the new variables, but can't occur anywhere else.

Thus, in Φ', each new variable occurs fewer than $\frac{2 \cdot k \cdot d}{\epsilon} + 2^k$ times. Also, each old variable occurs in Φ' exactly as many times as it occurs in Φ, so by the choice of d, each old variable occurs in Φ' at most d times. Since $k \geq 3$ and $\epsilon < 1$, one has $d < \frac{2 \cdot k \cdot d}{\epsilon} + 2^k$, so that means

$$\text{freq}(\Phi') < \frac{2 \cdot k \cdot d}{\epsilon} + 2^k$$

By the construction of Φ', the set $\text{sol}(\Phi')$ is exactly the set of extensions of assignments in $\text{sol}(\Phi_i)$ formed by setting each of the new variables to False. Thus, $\text{sol}(\Phi')$ has the same number of assignments as $\text{sol}(\Phi_i)$, and given an element of

either one can trivially find an element of the other. By part 2 of the sparsification lemma, every element of $\text{sol}(\Phi_i)$ is an element $\text{sol}(\Phi)$.

Furthermore, again by part 2 of the sparsification lemma 3.2.1, if $\text{sol}(\Phi)$ has at least one assignment then at least one of the outputs Φ_i of the algorithm A is such that $\text{sol}(\Phi_i)$ has at least one assignment. Thus, one can determine whether or not $\text{sol}(\Phi)$ has at least one assignment via the following procedure:

- Run the algorithm A on input (k, ϵ, Φ).
- For that algorithm's outputs Φ_i:
  - Construct Φ' from Φ_i as above.
  - If $\text{sol}(\Phi')$ has at least one assignment then:
    - Output *yes* .
    - Given an assignment in $\text{sol}(\Phi')$, the restriction of the assignment to the old variables in Φ' is an assignment in $\text{sol}(\Phi)$.
    - Halt.
- If none of those sets had at least one assignment then output *no* .

Each Φ' from this procedure is a k-CNF formula under the strict definition, and is such that $\text{freq}(\Phi') < \frac{2 \cdot k \cdot d}{\epsilon} + 2^k$ and Φ' has at most $(1 + \epsilon) \cdot n$ variables. By

part 4 of the sparsification lemma 3.2.1, generating the formulas Φ_i takes time at most $\text{poly}(n) \cdot 2^{\epsilon \cdot n}$ although the degree of the polynomial may depend on k and ϵ. Nonetheless, as n goes to infinity, that will be much less than $2^{2 \cdot \epsilon \cdot n}$.

Temporarily fix i. Since $2^k \leq n$, going from Φ_i to the corresponding Φ' takes at most $\text{poly}(n)$ time. Set $\Delta = \lceil \frac{2 \cdot k \cdot d}{\epsilon} + 2^k \rceil$, so that $\text{freq}(\Phi') < \Delta$. The formula Φ' also has at most $(1+\epsilon) \cdot n$ variables, so by the choice of $c_{k,\Delta}$, determining whether or not $\text{sol}(\Phi')$ has at least one assignment uses time at most $2^{c_{k,\Delta} \cdot (1+\epsilon) \cdot n}$. Thus the i-th iteration of the for loop takes time at most $\text{poly}(n) + 2^{c_{k,\Delta} \cdot (1+\epsilon) \cdot n}$.

Now unfix i. Generating the formulas Φ_i takes time much less than $2^{2 \cdot \epsilon \cdot n}$ as n goes to infinity, and each of the at most $2^{\epsilon \cdot n}$ iterations of the for loop takes time at most $\text{poly}(n) + 2^{c_{k,\Delta} \cdot (1+\epsilon) \cdot n}$, and the final step would take at most constant time, so the procedure takes time at most

$$2^{2 \cdot \epsilon \cdot n} + \left(2^{\epsilon \cdot n} \cdot \left(\text{poly}(n) + 2^{c_{k,\Delta} \cdot (1+\epsilon) \cdot n}\right)\right) + \text{constant}$$

The value $c_{k,\Delta}$ was chosen to be such that solving a specific computational problem uses time $2^{c_{k,\Delta} \cdot n}$, so $0 \leq c_{k,\Delta}$. Now, assume that n is sufficiently large. If

$c_{k,\Delta} = 0$ then

$$
\begin{aligned}
& 2^{2\cdot\epsilon\cdot n} + \left(2^{\epsilon\cdot n} \cdot \left(\text{poly}(n) + 2^{c_{k,\Delta}\cdot(1+\epsilon)\cdot n}\right)\right) + \text{constant} \\
&= 2^{2\cdot\epsilon\cdot n} + \left(2^{\epsilon\cdot n} \cdot \left(\text{poly}(n) + 2^{0\cdot(1+\epsilon)\cdot n}\right)\right) + \text{constant} \\
&= 2^{2\cdot\epsilon\cdot n} + \left(2^{\epsilon\cdot n} \cdot (\text{poly}(n) + 1)\right) + \text{constant} \\
&< 2^{2\cdot\epsilon\cdot n} + \left(2^{\epsilon\cdot n} \cdot 2^{\epsilon\cdot n}\right) + 2^{2\cdot\epsilon} \\
&= 3 \cdot 2^{2\cdot\epsilon\cdot n} \\
&< 2^{3\cdot\epsilon\cdot n} \\
&\leq 2^{(3\cdot\epsilon + c_{k,\Delta}\cdot(1+\epsilon))\cdot n}
\end{aligned}
$$

and if $c_{k,\Delta} > 0$ then

$$
\begin{aligned}
& 2^{2\cdot\epsilon\cdot n} + \left(2^{\epsilon\cdot n} \cdot \left(\text{poly}(n) + 2^{c_{k,\Delta}\cdot(1+\epsilon)\cdot n}\right)\right) + \text{constant} \\
&< 2^{2\cdot\epsilon\cdot n} + \left(2^{\epsilon\cdot n} \cdot 2 \cdot 2^{c_{k,\Delta}\cdot(1+\epsilon)\cdot n}\right) + \text{constant} \\
&< \left(2 \cdot 2^{2\cdot\epsilon\cdot n}\right) + \left(2^{2\cdot\epsilon\cdot n} \cdot 2 \cdot 2^{c_{k,\Delta}\cdot(1+\epsilon)\cdot n}\right) \\
&< 2 \cdot 2^{2\cdot\epsilon\cdot n} \cdot 2 \cdot 2^{c_{k,\Delta}\cdot(1+\epsilon)\cdot n} \\
&= 2^{2+2\cdot\epsilon\cdot n + c_{k,\Delta}\cdot(1+\epsilon)\cdot n}
\end{aligned}
$$

bounding the exponent

$$
\begin{aligned}
& 2 + 2 \cdot \epsilon \cdot n + c_{k,\Delta} \cdot (1+\epsilon) \cdot n \\
&= (3 \cdot \epsilon + c_{k,\Delta} \cdot (1+\epsilon)) \cdot n
\end{aligned}
$$

Combining the preceding two results gives

$$\begin{aligned}
&2^{2\cdot\epsilon\cdot n} + \left(2^{\epsilon\cdot n}\cdot\left(\text{poly}(n) + 2^{c_{k,\Delta}\cdot(1+\epsilon)\cdot n}\right)\right) + \text{constant} \\
&< 2^{2+(2\cdot\epsilon\cdot n)+(c_{k,\Delta}\cdot(1+\epsilon)\cdot n)} \\
&< 2^{((3\cdot\epsilon)+(c_{k,\Delta}\cdot(1+\epsilon)))\cdot n}
\end{aligned}$$

In each case,

$$2^{2\cdot\epsilon\cdot n} + \left(2^{\epsilon\cdot n}\cdot\left(\text{poly}(n) + 2^{c_{k,\Delta}\cdot(1+\epsilon)\cdot n}\right)\right) + \text{constant} < 2^{((3\cdot\epsilon)+(c_{k,\Delta}\cdot(1+\epsilon)))\cdot n}$$

Thus, the procedure I gave determines whether or not $\text{sol}(\Phi)$ has at least one assignment in time less than $2^{((3\cdot\epsilon)+(c_{k,\Delta}\cdot(1+\epsilon)))\cdot n}$. This holds for all sufficiently large n and all n-variable k-CNF formulas Φ, so $c_k \leq ((3\cdot\epsilon)+(c_{k,\Delta}\cdot(1+\epsilon))$. In turn, this proof gives such a Δ for every real number $0 < \epsilon < 1$, so $c_k \leq \limsup_{\Delta\to\infty} c_{k,\Delta}$, which proves claim Weird. □

The rest of the proof of (1) is now easy. For each integer $k \geq 3$, let d_k be an integer such that $c_k \leq c_{k,d_k} + (1/k)$. Thus $\liminf_{k\to\infty} c_k = \liminf_{k\to\infty} c_{k,d_k}$ and $\sup(\{c_3, c_4, c_5, ...\}) = \sup(\{c_{3,d_3}, c_{4,d_4}, c_{5,d_5}, ...\})$, so if SETH is true then it remains true when restricting to k-SAT instances Φ such that $\text{freq}(\Phi) \leq d_k$. This completes the proof of (1). □

Proof of (2). By the definition of k-CNF formula which only requires that the the clauses have length *at most* k, one gets that for all integers k_0 and k_1, if $k_0 \leq k_1$ then all k_0-CNF formulas are also k_1-CNF formulas, so k_0-SAT is no harder than

k_1-SAT. Thus $c_3, c_4, c_5, ...$ is non-decreasing, so $\liminf_{k\to\infty} c_k = \sup(\{c_3, c_4, c_5, ...\})$.

□

Proof of (3). Let Φ_0 be the input instance. Determine whether or not Φ_0 has a satisfying assignment. If it does not, then there is nothing else to do here, so assume it does. The formula Φ_0 can't have a pair of length-1 clauses with the same variable that differ in the literal's sign, since there would be no way to satisfy both clauses in such a pair. Set $\Phi = \Phi_0$. For each variable x such that neither x nor $\neg x$ is a length-1 clause in the current formula ϕ:

Determine whether or not $\Phi \wedge x$ is satisfiable. If it is, then replace Φ with $\Phi \wedge x$, else replace Φ with $\Phi \wedge \neg x$.

For each Φ obtained throughout this loop, $\text{sol}(\Phi) \subseteq \text{sol}(\Phi_0)$. After finishing the loop, Φ is such that

- for each variable x, Φ has exactly one 1-clause involving x
- the signs of the literals in those 1-clauses

  indicate a satisfying assignment for Φ_0.

This only involves determining satisfiability $n + 1$ times and a trivial amount of additional work, so it takes only $\text{poly}(n) \cdot 2^{c_k \cdot n}$ time, which will be much less than $2^{(c_k+(1/k))\cdot n}$ time when n is sufficiently large. Clearly, adding $1/k$ to each c_k does not affect whether $c_3, c_4, c_5, ...$ goes to 1.□

Lastly, the proof of (4) is given by Theorem 1 of [8].

3.3 Uses for a Good Obfuscator

3.3.1 Relations Between Primitives

Below, I discuss the following cryptographic primitives, listed roughly in increasing value. Obfuscation might allow one to advance from less useful primitives to more useful primitives.

1. symmetric encryption
2. symmetric encryption with three specific properties (to be specified later)
3. pseudorandom function families
4. public-key encryption with one specific property (to be specified later)
5. fully homomorphic encryption.

I omit the adjective *secure*, because when discussing implications involving the existence of these and other cryptographic primitives, security is part of the definitions of these things. I will give the specific properties for (3) and (4) when I go into more detail on the meaning of (2) and (4).

It is known that if (1) exists, then so do (2) and (3) [5],[16],[28]. I do not go over those proofs since they are beyond the scope of this work. On the other hand, neither the existence of (4) nor the existence of (5) is known to follow from the existence of (1), and there is a known obstacle to any proof that the existence of either does follow from (1) [4]. As two examples of how a good-enough obfuscator could be used, I will show how such an obfuscator could be used to go from (2) and (3) to (4), and then from (2) and (4) to (5).

3.3.2 Symmetric and Public-Key Encryption

To describe these two types of encryption, I use the characters Alice, Bob, and Eve, as is traditional in cryptographic literature. These do not necessarily refer to humans; they can instead be computers. Eve is the eavesdropper. She is listening in on the communication channel between Alice and Bob.

In cryptography, Kerckchoff's Principle states that a cryptosystem should be secure even if Eve knows everything about the system, except for a single secret parameter, the key. Although the character "Eve" was not used in this context until much later, this was otherwise stated in an essay published by Auguste Kerckchoffs in 1883, and is accepted by all modern cryptosystems.

In symmetric encryption, keys used for encryption and decryption are either the same or trivially related to each other. This is the paradigm that has been used for most of human history. For symmetric encryption,

- Alice and Bob already have *either* a shared key *or else* one key each.
- Eve knows which of those is the case and knows the lengths of the key(s), but knows nothing else about the key(s).
- The goal of Alice and Bob is to use their key(s) to get information, such as a text message, from Bob to Alice in a way which does not let Eve learn anything useful about that information beyond the length of that information, such as the number of symbols.

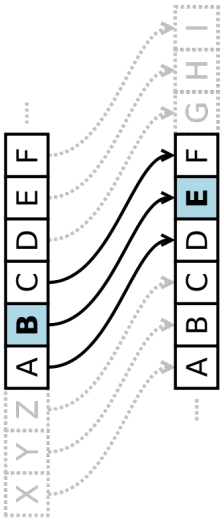
For this, the *plaintext* is the original message, and the *ciphertext* is what Alice sends to Bob over the communication channel that Eve is listening in on. Eve will see the ciphertext; Alice and Bob want Eve to not learn more about the plaintext than the plaintext's length.

By the pigeon-hole principle, most possible plaintexts require sending a number of symbols that is not much less than the number of symbols in the message. For this, the pigeons are the possible plaintexts and the holes are the possible ciphertexts. In particular, hiding significant differences in plaintext length, such

as the difference between “Hi Alice” and “Hello Alice, Charlie, David, and Fred. How are you?”, would require corresponding increases in ciphertext length.

Unless there is a limit to plaintext length, no matter how much additional communication is used to hide differences in plaintext length, there will always be possible plaintexts with a greater difference between their lengths so that ciphertext length tends to reveal the difference between those possible plaintexts. This is why encryption simply makes no attempt at hiding the length of the plaintext.

One extremely simple example, which is far from actually being secure but nonetheless might convey the idea, is the Caesar cipher. That cipher is almost-always and most-conveniently described as having the keys be the same and letting encryption involve adding the key and decryption involve subtracting the key.



This is how encryption with the Caesar cipher works when the encryption key is 3.

It could alternatively be described as having the keys be negatives of each other modulo 26 (the number of letters in the alphabet) and letting both encryption and decryption involve adding the relevant key.

The development of fast computers has massively increased the amount of resources available for Alice and Bob and Eve. Although it might seem as though those changes should cancel out, an important point is that the *number of possible keys* of a given length grows much faster than that length - exponential versus linear. Thus, it is possible for encryption to be secure even if Eve can find the keys or key much faster than by just trying each possible key: *much faster than brute force* does not necessary mean feasible. This allows for systems which seem to achieve a more ambitious goal than symmetric encryption.

The more ambitious goal is for Alice and Bob is to use one public key and one secret key to get information from Bob to Alice in a way which does not let Eve learn anything useful about that information beyond the length of that information. That is, Alice generates a key-pair on her own - an encryption key and a decryption key - keeps the decryption key for herself (private key), and publishes the encryption key (public key), which we assume results in both Bob and Eve learning the encryption key. This concept is called *public key encryption*. One reason why public key encryption is useful is it allows other parties besides Bob to send her encrypted messages. For example, it allows for confidential commu-

nication between parties who have had no prior contact, such as customers and online companies.

The first methods that seem to achieve public key encryption were found in the 1970s. Those methods were based on number theory, specifically modular arithmetic in the integers, although those are no longer the only methods that seem to achieve public key encryption. However, it is known that public key encryption is qualitatively [4] harder than symmetric encryption, and the the existence of public key encryption is not known to be implied by the existence of symmetric encryption.

3.3.3 Properties for Symmetric Encryption

As mentioned in 3.1, it *is* known that if symmetric encryption exists, then so does symmetric encryption with the following three properties:

- the encryption and decryption keys are the same
- encryption and decryption are non-interactive: i.e., Bob uses the key and an original message and randomness to compute a ciphertext, then Alice uses the key and ciphertext to recover the plaintext
- security holds even when Eve gets to see encryptions of plaintexts of her choice with randomness of her choice.

3.3.4 Circuit Obfuscation and Encryption

A good-enough circuit obfuscator would enable one to easily go from symmetric encryption with those properties to public-key encryption. One would:

1. use the same decryption key privkey as used for the symmetric encryption that one starts with, and set $L = \text{length(privkey)}$

2. find an upper bound B on the number of random bits that the symmetric encryption might use to encrypt a message whose length is at most L with a key whose length equals L

3. use as the encryption key an obfuscation of a circuit C where C works as follows: The circuit C takes as input an ordered pair (m, r) of strings such that

   - $\text{length}(m) \leq L$
   - $\text{length}(r) = B$

   and outputs the encryption of m with key privkey using r as randomness.

The last of the three properties from the previous section for the symmetric encryption is that access to the *functionality* of such a circuit does not let Eve break the security of the symmetric encryption. Accordingly, if the obfuscator does what one might hope it does - roughly, stop the adversary from learning

more than the adversary could learn from just the functionality - then the public-key encryption system constructed in this way satisfies the basic notion of security for public-key encryption.

In order to let a *circuit* obfuscator suffice, rather than requiring a *program* obfuscator, the public-key encryption constructed in this way only *directly* works for messages that are not very long. However, this issue of only working for short messages issue applies to essentially every construction of public key encryption that I am aware of. Furthermore, this issue has a generic solution called *hybrid encryption*, which combines public key encryption with symmetric encryption. Specifically, it uses the public key encryption to encrypt a key for symmetric encryption, and use symmetric encryption with that key on the actual message. Fortunately, symmetric encryption is not limited to messages which are not very long.

3.3.5 Pseudorandom Function Families

Roughly speaking, *Pseudorandom function families* (PRFs) are functions $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ such that $\mathcal{K}$ is non-empty and for prfkey chosen at random from $\mathcal{K}$, the function $f : \mathcal{X} \to \mathcal{Y}$ given by $f(x) = F(\text{prfkey}, x)$ looks like a random function $\mathcal{X} \to \mathcal{Y}$ to someone who does not know prfkey. Namely, for prfkey

chosen uniformly at random from $\mathcal{K}$, Eve should not be able to tell the difference between receiving the following two things:

(1) the outputs $F(\text{prfkey}, x)$ for inputs $x \in \mathcal{X}$ of her choice

versus

(2) The outputs of the following procedure applied to inputs of her choice, where Q is initially empty:

- receive $x \in \mathcal{X}$ as input
- if x is the left entry of an ordered pair in Q, then output that pair's right entry.
- Otherwise, choose y uniformly at random from $\mathcal{Y}$, add (x, y) to Q, and output y.

In other words, (2) outputs random elements of $\mathcal{Y}$ *except that* it outputs the same element when given the same input. It is known that if symmetric encryption exists, then so do pseudorandom function families.

3.3.6 Fully Homomorphic Encryption

Given that symmetric encryption implies pseudorandom function families, a good-enough obfuscator could be used again to make the public-key encryption be fully homomorphic. For the following, note that *ciphertext* means the result

of encrypting. In order to be *fully homomorphic*, an encryption scheme should come with an efficient algorithm that, given encryptions of bits x and y, computes encryptions of $\neg x$ and $x \wedge y$.

Since all boolean operations can be expressed in terms of $\neg$ and $\wedge$, such an algorithm suffices to be able to compute encryptions of the outputs of arbitrary circuits given encryptions of their inputs. Given a pseudorandom function family f and a public key encryption scheme and a good-enough obfuscator, the fully homomorphic encryption scheme works as follows:

1. Every plaintext is a bit.

2. The private key privkey is the same as in the starting public key encryption.

3. Find an upper bound B_0 on the number of random bits that the original decryption algorithm might use and an upper bound B_1 on the number of random bits that the original encryption algorithm might use.

4. Let $\hat{\mathrm{CT}}$ be a set that includes every possible ciphertext for the original encryption scheme, and set $\mathcal{X} = \mathrm{CT} \times \mathrm{CT} \times \{0,1\}$.

5. Let $F : \mathcal{K} \times \mathcal{X} \to \{0,1\}^{2 \cdot B_0 + B_1}$ be a pseudorandom function family.

6. Choose a key prfkey for F, and let f be given by $f(x) = F(\mathrm{prfkey}, x)$. The public key is an ordered pair $(\mathrm{pub}, \mathrm{obf})$, where pub is the public key for

the starting public key encryption, and obf is an obfuscation of the circuit that takes as input ciphertexts x and y for 1-bit plaintexts and a bit b, and outputs the result of encrypting

$$\begin{cases} (x\text{'s plaintext}) \wedge (y\text{'s plaintext}) & \text{if } b = 1 \\ \neg(x\text{'s plaintext}) & \text{if } b = 0 \end{cases}$$

with privkey with $f((x, y, b))$ as randomness.

Since F is a pseudorandom function family, Eve should not be able to tell the difference between receiving

(1) the outputs of one of those circuits on inputs of her choice

versus

(2) The outputs of the following procedure on inputs of her choice, where the set Q is empty before Eve provides inputs.

- Receive a triple (x, y, b) as input from Eve.
- If that triple is the left entry of an ordered pair in Q, then output the right entry of that ordered pair.
- Otherwise, let z be the result of encrypting

$$\begin{cases} (x\text{'s plaintext}) \wedge (y\text{'s plaintext}) & \text{if } b = 1 \\ \neg(x\text{'s plaintext}) & \text{if } b = 0 \end{cases}$$

with privkey, add $((x, y, b), z)$ to Q, and output z.

Thus, a good-enough obfuscator can be used to go from symmetric encryption to fully homomorphic encryption.

3.3.7 Watermarks

Another use of a sufficiently-good obfsucator would be to create watermarks for software. This means producing many different versions of the same software, with only tiny changes in functionality. Then, if an adversary gets a few of these versions and makes a version with approximately the same functionality, and the original producer subsequently receives that version, then the original producer can identify one of the versions it produced as having been used by the adversary. Even when the adversary gets more than one version, the original producer cannot necessarily expect to identify more than one version as received by the adversary, since the adversary could just ignore all but one of its versions.

3.4 Possibility and Impossibility

When attempted in practice, program obfuscation is generally done heuristically, with neither a theoretical basis nor a formalized security goal. I believe its first theoretical treatment was [10], which proved that there is no efficient *virtual black box* obfuscator. As a result, that paper proposes two alternative notions,

one of which is called *indistinguishability obfuscation*, and the other of which is obfuscation for sampling algorithms instead of for functions. For virtual black box obfuscation, one wanted the new source code to reveal nothing more than what can be learned from just querying the functionality: i.e., the new source code should be a *virtual black box* which provides just input-output access to the functionality.

For indistinguishability obfuscation of circuits, one instead wants that for all circuits C_0 and C_1, if C_0 and C_1 have the same size and compute the same function then obfuscations of C_0 are computationally indistinguishable from obfuscations of C_1. There are candidate constructions for such obfusctors [1], and such obfuscators suffice for a huge number of cryptographic tasks [14]. One of those tasks is witness encryption, which works as follows.

Witness encryption schemes consist of an encryption algorithm and a decryption algorithm. However, unlike public key encryption, which also has a key generation algorithm, a witness encryption scheme is traditionally for an NP verifier V. The encryption algorithm takes an input

- the security parameter (the parameter which controls the security level)
- the message m
- an instance x for the verifier V.

It produces a ciphertext. The decryption algorithm takes as input

- the ciphertext
- a possible witness y.

We require that if $V(x, y)$ accepts, then the decryption algorithm outputs the message m. The security property is, if x is a *no* instance, then encryptions of different messages with the same length are computationally indistinguishable.

Indistinguishability obfuscation of circuits easily suffices to build a witness encryption scheme for an NP relation V. Let p be a polynomial such that the following hold.

- V takes as input the decision problem's input (the *instance*) and an alleged-proof, and outputs *accept* or *reject*.
- On input (x, y), the running time of V is at most $p(\text{length}(x) + \text{length}(y))$.
- For every *yes* instance x, there is at least one alleged-proof y such that $\text{length}(y) \leq p(\text{length}(x))$ and $V(x, y)$ accepts.
- For every *no* instance x, and every alleged-proof y then $V(x, y)$ rejects.

Given indistinguishability obfuscation for circuits, on instance x and message m, the encryption algorithm sets $L_x = \text{length}(x)$ and $L_m = \text{length}(m)$, and outputs an indistinguishability obfuscation of the circuit that works as follows.

- Take as input a string y of length at most $p(L_x)$.
- Run $V(x, y)$ for up to $p(L_x + \text{length}(y))$ time.
- If V accepts within that time, then output 1 followed by m, else output the string of $L_m + 1$ zeros.

The decryption algorithm simply runs the ciphertext on the possible witness y. If x is a *no* instance, then the *functions computed by* the circuits which are to be obfuscated do not depend on the message m, so security for the witness encryption scheme follows from security for the indistinguishability obfuscation.

Chapter 4

Joint Selection

In this chapter I discuss of Joint Selection Random Samples for Zero-Knowledge Protocols and Other Cryptographic Schemes and Protocols.

In cryptography, NP relations typically appear as the subject of two types of protocols: *Arguments* and *Witness Encryption*. In turn, there is a whole gamut of properties that cryptographic arguments may have. These include

succinct	non − interactive
witness − indistinguishable	strongly − witness − indistinguishable
zero − knowledge	proof
of knowledge	

The properties *witness-indistinguishable* and *zero-knowledge* themselves each come in a hierarchy of at least 3 different flavors: *computational, statistical*, and

perfect. The concept of witness encryption was introduced far more recently. Its candidate constructions are based generally on indistinguishability obfuscation, or specifically on *graded encoding systems*.

We give a transformation which shows that, at the cost of at most 1 additional message, most such protocols can also be made to work for relations with randomized verifiers. In particular, we do *not* use an extra round for witness encryption: Otherwise, the result would not be a witness encryption scheme.

4.1 On promiseMA

Suppose Alice claims to have a circuit that does surprisingly well at Minesweeper with specific parameters: for example, m-by-n boards with j mines. Can Alice prove that without revealing her circuit?

The natural approach here is zero-knowledge protocols. However, such protocols *for NP relations* do not directly suffice: NP verifiers must be deterministic, whereas even if the input circuit is deterministic, the only obvious efficient way of estimating its probability of winning the minesweeper game is using randomness to sample mine placements and running the input circuit on the sampled mine placements. Thus, to handle this situation, one presumably needs to allow randomized verifiers.

The generalization of NP to randomized verifiers for languages is called *MA*, which is short for *Merlin-Arthur*. It requires that for each string in the language, there is a polynomial-length proof (provided by *Merlin*) such that the verifier (*Arthur*) has probability at least 2/3 of accepting, whereas for each string that's *not* in the language, no matter what alleged-proof Merlin provides, Arthur has probability at most 1/3 of accepting.

One might ask, why 1/3 and 2/3? The answer to that has two parts. One part is, one needs a gap. For example, there's no obvious feasible way of distinguishing a verifier has probability $1/2 - 2^{-\text{input\_length}}$ of accepting from a verifier has probability $1/2 + 2^{-\text{input\_length}}$ of accepting.

The other part is, $1/3, 2/3$ is the simplest pair of distinct fractions between 0 and 1. Which such pair is used does not affect MA, for the same reason as it does not affect BPP. As long as the thresholds differ noticeably, one can use Chebyshev's inequality to get that the acceptance fraction over a large-enough but still easily doable number of independent trials is likely to differ from the verifier's true acceptance probability by less than half the difference between the thresholds, and then use a Chernoff bound to get that a majority vote of acceptance fractions generated independently in that way has at most an exponentially small probability of being wrong.

A significant issue is, NP is syntactic, whereas MA is not obviously syntactic. Every NP language has a verifier that is obviously a verifier for a NP language, since one can hardcode a polynomial and have the verifier reject the alleged-witness is too long or the original verifier would take too long. One does *not* obviously have anything similar for MA. How can one alter a randomized verifier so that it's clear that there are no instances for which the new verifier's maximum acceptance probability (over the space of alleged-witnesses) is between the two thresholds, without changing the language defined by the verifier?

This issue leads to the notion of *promise problem*s. We note that our definitions very closely follow [26]. By a *string*, I mean finite-length binary string. With that meaning, $\{0,1\}^*$ is the set of all strings.

Definition 4.1.1. *Promise problems* are pairs (Π_Y, Π_N) such that Π_Y and Π_N are each sets of strings and $\Pi_Y \cap \Pi_N = \{\}$.

The idea behind that name is, the solver is promised that the input is in $\Pi_Y \cup \Pi_N$.

Definition 4.1.2. A total probabilistic algorithm M *solves a promise problem* (Π_Y, Π_N) *with thresholds* $0 < p_{low} < p_{high} < 1$ if and only if for all $x \in \{0,1\}^*$

both the following hold:

$$x \in \Pi_Y \quad \Rightarrow \quad \text{Prob}(M(x) \text{ accepts}) > p_{high}$$

$$x \in \Pi_N \quad \Rightarrow \quad \text{Prob}(M(x) \text{ accepts}) < p_{low}$$

It is important that, the acceptance probability for inputs which do *not* satisfy the promise (i.e., x such that $x \notin \Pi_Y \cup \Pi_N$) can be anything. Promise-relations, and algorithms for them, are defined similarly.

Definition 4.1.3. *Promise relations* are pairs (R_Y, R_N) such that R_Y and R_N are each sets of ordered pairs of strings and $R_Y \cap R_N = \{\}$.

Definition 4.1.4. A total probabilistic algorithm M *solves a promise relation* (R_Y, R_N) *with thresholds* $0 < p_{low} < p_{high} < 1$ if and only if for all $(x, y) \in \{0,1\}^* \times \{0,1\}^*$ both the following hold:

$$(x, y) \in R_Y \quad \Rightarrow \quad \text{Prob}(M(x, y) \text{ accepts}) > p_{high}$$

$$(x, y) \in R_N \quad \Rightarrow \quad \text{Prob}(M(x, y) \text{ accepts}) < p_{low}$$

For the same reason as BPP, although whether-or-not *a specific algorithm* solves a promise problem depends on the thresholds, whether-or-not there is a polynomial-time algorithm that solves the promise-problem does *not* depend on the thresholds.

Now, the definition of promiseMA relations comes with the same unimportant ambiguity as the definition of NP relations. Does one force the promise-relation

to output zero when the alleged witness is too long, or instead have a separate polynomial length bound to get a corresponding promiseMA problem? To simplify presentation, I use the former approach:

Definition 4.1.5. A *promiseMA relation* is a promise-relation (R_Y, R_N) such that (R_Y, R_N) is solvable in polynomial time and there is a polynomial p such that for all $x \in \{0,1\}^*$ and $y \in \{0,1\}^*$, if $p(\text{length}(x)) < \text{length}(y)$ then $(x, y) \in R_N$.

Definition 4.1.6. Given a promiseMA relation (R_Y, R_N), the *induced promise problem* in promiseMA is the promise problem (Π_Y, Π_N) given by

$x \in \Pi_Y \iff (\exists y)((x, y) \in R_N)$

$x \in \Pi_N \iff (\forall y)((x, y) \in R_N)$.

My approach to going from handling NP relations to handling promiseMA relations consists of giving something which is close-enough to a reduction from promiseMA promise-relations to NP relations.

In the following, k is the security parameter in unary. Roughly, for a promiseMA promise-relation (R_Y, R_N), we want an NP relation R_{NP} and polynomials proverLen and verifierLen and efficient deterministic algorithms

- convert_instance_
- forward_witness_transfer_

- backward_witness_transfer_

Such that

- convert_instance_ takes as input k and 3 more strings
- forward_witness_transfer_ and backward_witness_transfer_

  each take as input k and 4 more strings
- length(forward_witness_transfer_(k, x, α, β, y)) may depend on length(y), but does not otherwise depend on y
- for all strings x and z, and all but an exponentially-small fraction of the elements β in $\{0,1\}^{\text{verifierLen(length}(x))}$, and all strings $\alpha \in \{0,1\}^{\text{proverLen(length}(x))}$

  if (convert_instance_$(k, x, \alpha, \beta, z)) \in R_{NP}$

  then $(x, \text{backward\_witness\_transfer\_}(k, x, \alpha, \beta, z)) \notin R_N$.
- for all elements (x, w) of R_Y, for all $\beta \in \{0,1\}^{\text{verifierLen(length}(x))}$, a majority of the elements $\alpha \in \{0,1\}^{\text{proverLen(length}(x))}$ are such that

  $$(\text{convert\_instance\_}(k, x, \alpha, \beta), \text{forward\_witness\_transfer\_}(k, x, \alpha, \beta, w))$$

  is in R_{NP}.

The *roughly* is because proverLen and verifierLen just need to be efficiently-enough computable and *bounded* by a polynomial rather than actually being polynomials.

The *for all strings* x condition is soundness: Most likely, any witness for the NP relation can be used to get something that is close to being a witness for the promiseMA promise-relation.

The *for all elements* (x, w) *of* R_Y condition provides completeness and part of security: The prover can try as many candidate αs as the prover wants.

The length condition provides the rest of security: The length of the R_{NP} witness does not reveal any more than the length of the (R_Y, R_N) witness.

Note that achieving *perfect* completeness for ZK protocols in *strict* polynomial time requires promiseBPP = promiseZPP, so it would be difficult to prove that such completeness can be achieved. For a promiseBPP statement, one can simulate an interactive proof of the statement and an interactive proof of the statement's negation, with the alleged-witness being empty. If either prover times out or either verifier rejects, then one can safely answer in the other direction.

4.2 Interactive Protocols

This section tracks [26] quite closely. An interactive protocol consists of two possibly-randomized algorithms that compute the *next-message function* of the (honest) parties in the protocol. Specifically, $A(x, a, \gamma)$ denotes the next message sent by party A when x is the list of public inputs, a is the list of A's private inputs, and γ is the list of messages exchanged so far. There are two special

messages, *accept* and *reject*, which immediately halt the interaction. We say that party A (resp. party B) is *probabilistic polynomial-time (PPT)* if the runtime of its next-message function is bounded above by a polynomial in the number of bits used to write out A's (resp. B's) sequence of inputs.

For an interactive protocol $\langle A, B\rangle$, we write $\langle A(a), B(b)\rangle(x)$ to denote the random process obtained by having A and B interact where x is the list of common inputs, a is A's list of private inputs, b is B's list of private inputs, and the coin tosses of A and B are independent.

We say $\langle A, B\rangle$ is *polynomially bounded* if and only if there is a polynomial in the number of bits used to write out the public input such that for all lists a, b of private inputs for A, B respectively, the total length of all messages exchanged in $\langle A(a), B(b)\rangle(x)$ is at most that polynomial with probability 1. Moreover, if B^* is any interactive algorithm, then A will immediately halt and reject in $\langle A(a), B^*(b)\rangle(x)$ if the total length of the messages ever exceeds that polynomial, and similarly for B interacting with any A^*.

Let transcript$(\langle A(a), B(b)\rangle(x)$ denote $(\gamma_0, \gamma_1, ..., \gamma_t)$, where the γs are the messages exchanged (including a final *accept*/*reject*). We write $\text{view}_A(\langle A(a), B(b)\rangle(x))$ to the transcript concatenated with A's randomness, that is $(\gamma_0, \gamma_1, ..., \gamma_t, r)$ where r is A's randomness. We define $\text{view}_B(\langle A(a), B(b)\rangle(x))$ similarly, using B's randomness rather that A's randomness.

The number of *rounds* in an interactive protocol is the total number of messages exchanged between A and B, not including a final *accept/reject*. We call the protocol $\langle A, B\rangle$ *public coin* if each message sent by B is simply the output of B's coin-tosses (independent of its history) except for a final *accept/reject* which B computes as a deterministic function of the transcript.

So as to simplify references to other algorithms in the definitions related to interactive proofs and interactive arguments, I start with the following definitions and notational notes.

Definition 4.2.1. $(\{0,1\}^*)^*$ is the set of finite-length sequences of strings.

Σ is the set $\{0,1\}^* \cup \{\text{accept}, \text{reject}\}$, and Σ^* is the set of finite sequences of elements of Σ.

Definition 4.2.2. For all non-negative integers n, unary(n) is the string of n ones.

Notation: k is the security parameter.

Definition 4.2.3. Given sequences of sets $X_0, X_1, \cdots, X_{m-1}$ and $Y_0, Y_1, \cdots, Y_{n-1}$,

$$A : X_0 \times X_1 \times ... \times X_{m-1} \stackrel{\text{krand}}{\rightarrow} Y_0 \times Y_1 \times ... \times Y_{n-1}$$

denotes that A is a possibly-randomized algorithm with $m+1$ inputs (when not including its own randomness) and n outputs, where its first m inputs are from

the corresponding Xs, its m-th input is unary(k), and each output is in the corresponding Y. When I refer to the inputs of such an algorithm, by default, the randomness is *not* included.

Notation: When I specify the randomness used, that randomness will be between a semi-colon and the closing parenthesis. The absence of a semi-colon in my functional notation for algorithms indicates that the randomness is independent of everything else.

For example, if A generates exactly one
random bit and outputs exactly that bit, then
$\text{Prob}\left[A(\text{unary}(k);0)=1\right]$, $\text{Prob}\left[A(\text{unary}(k))=1\right]$, $\text{Prob}\left[A(\text{unary}(k);1)=1\right]$
are $0\,, 1/2\,, 1$ respectively.

Suppression of Security Parameter: I will omit the unary(k) s from my expressions. For example, when $m=1$, I would write $A(x;r)$ or $A(x)$, rather than $A(x,\text{unary}(k);r)$ or $A(x,\text{unary}(k))$.

Definition 4.2.4. We write $\epsilon : \{0,1,2,3,...\} \to [0,1]$ for *negligible function.*

The definition of negligible function is that $f : \mathbb{N} \to [0,1]$ is negligible if and only if for all natural numbers c, for all sufficiently large n, $f(n) < n^{-c}$. However, similarly to most other results in cryptography, the results in this thesis apply to any definition of *negligible* that satisfies sufficient closure properties. For

example, one might use $2^{-(\log(n))^c}$ instead of n^{-c}, or instead require that there *exists* a positive integer c such that for all sufficiently large n, $f(n) \leq 2^{-\left(n^{1/c}\right)}$.

Definition 4.2.5. For interactive algorithms P and V, we say that $\langle P, V \rangle$ is an *efficient-prover interactive proof system* for a promise-relation (R_Y, R_N) if and only if there is a negligible function ϵ such that the following conditions all hold.

- Efficiency: $\langle P, V \rangle$ is polynomially bounded and P and V are both efficient
- Completeness: For all strings x and w, if $(x, w) \in R_Y$ then with probability at least $1 - \epsilon(k)$, V accepts in $\langle P(w), V \rangle(x)$.
- Statistical Soundness: For every algorithm P^* that can interact with V while taking no private inputs, if x is in the no set of the promise problem induced by (R_Y, R_N) then V accepts in $\langle P^*, V \rangle(x)$ with probability at most $\epsilon(k)$.

Definition 4.2.6. I define *efficient-prover interactive argument system*s as above, but with the *statistical soundness* bullet replaced with the following:

Computational Soundness: For every one-private-input feasible algorithm P^* that can interact with V, it is infeasible to find strings x and z such that x is in the no set of the promise problem induced by (R_Y, R_N) and V accepts in $\langle P^*(z), V \rangle(x)$ with probability greater than $\epsilon(k)$.

I note that if one wants a *semi-uniform* [15] notion, then one could interpret the *infeasible* in the above condition as non-uniform with respect to x and uniform with respect to z, and interpret all other *feasible* s as referring to uniform algorithms.

In any case, an efficient-prover interactive proof system is automatically an efficient-prover interactive argument system, since any $P^*(z)$ for violating computational soundness can be turned into an algorithm Q^* for violating statistical soundness, by just hard-coding z.

The following definitions use $\overset{\text{krand}}{\rightarrow}$ from Definition 4.2.3 .

Definition 4.2.7. If $\langle P, V\rangle$ is an efficient-prover interactive argument system for a promise-relation (R_Y, R_N) and circs is the set of circuits that can interact with P while taking no private inputs, then an algorithm $S : \text{circs} \times \{0,1\}^* \overset{\text{krand}}{\rightarrow} \Sigma^*$ is a *statistical zero-knowledge simulator* for $\langle P, V\rangle$ if and only if the following conditions both hold.

- S is efficient
- Statistical Zero Knowledge: There is a negligible function ϵ such that for every circuit $V^* \in$ circs, for all elements (x,w) of R_Y, the statistical distance from the distribution of $\text{view}_{V^*}(\langle P(w), V^*\rangle(x))$ to the distribution sampled from by $S(V^*, x)$ is at most $\epsilon(k)$.

Definition 4.2.8. If $\langle P, V\rangle$ is an efficient-prover interactive argument system for a promise-relation (R_Y, R_N) and algs is the set of algorithms that can interact with P while taking exactly one private input, then an algorithm

$$S : \text{algs} \times \{0,1\}^* \times \{0,1\}^* \overset{\text{krand}}{\rightarrow} \Sigma^*$$

is a *computational zero-knowledge simulator* for $\langle P, V\rangle$ if and only if the following conditions both hold.

- Relative Efficiency: S is efficient when its input algorithm is an efficient algorithm, and is feasible when its input algorithm is a feasible algorithm.
- Computational Zero Knowledge: For all feasible algorithms $V^* \in$ algs, for all feasible algorithms

  $$D : \{0,1\}^* \times \{0,1\}^* \times \Sigma^* \overset{\text{krand}}{\rightarrow} \{0,1\}$$

  there is a negligible function ϵ such that it is infeasible to find strings x,w,z such that $(x,w) \in R_Y$ and

  $$|f(\text{view}_{V^*}(\langle P(w), V^*(z)\rangle(x)) - f(S(V^*, z, x)| > \epsilon(k)$$

  where $f(t) = \text{Prob}\left[D\left(z, x, t\right) = 1\right]$

Similarly to the case for computational soundness, one can get a *semi-uniform* [15] notion by interpreting the *infeasible* in the above condition as non-uniform

with respect to x and w and uniform with respect to z, and interpret all other *feasible* s as referring to uniform algorithms.

Definition 4.2.9. If $\langle P, V \rangle$ is an interactive argument for a promise-relation (R_Y, R_N) and circs is the set of circuits that can interact with V while taking no private inputs, then an algorithm

$$E : \text{circs} \times \{0,1\}^* \stackrel{\text{krand}}{\rightarrow} \Sigma^* \times \{0,1\}^*$$

is a *statistical witness-extended emulator* for $\langle P, V \rangle$ if and only if there is a negligible function ϵ such that the following conditions all hold.

- E is efficient
- Statistical Emulation: For every circuit $P^* \in$ circs, for all strings x, the statistical distance from the distribution of $\text{view}_{P^*}(\langle P^*, V(x) \rangle)$ to the distribution of the left output of $E\,(P^*, x)$ is at most $\epsilon(k)$.
- Statistical Witness-Extension: For every circuit $P^* \in$ circs, for all strings x, $\text{Prob}_{(u,y) \leftarrow E(P^*,x)}\,[(x,y) \in R_N \text{ and } u \text{ is a view in which } V \text{ accepts}] \;\leq\; \epsilon(k)$.

Definition 4.2.10. If $\langle P, V \rangle$ is an interactive argument for a promise-relation (R_Y, R_N) and algs is the set of algorithms that can interact with V while taking exactly one private input, then an algorithm $E : \text{algs} \times \{0,1\}^* \times \{0,1\}^* \rightarrow \Sigma^* \times \{0,1\}^*$ is a *computational witness-extended emulator* for $\langle P, V \rangle$ if and only if the following conditions all hold.

- Relative Efficiency: E is efficient when its input algorithm is an efficient algorithm, and is feasible when its input algorithm is a feasible algorithm.

- Computational Emulation:

  For all feasible algorithms $P^* \in \text{algs}$, for all feasible algorithms $D : \{0,1\}^* \times \{0,1\}^* \times \Sigma^* \stackrel{\text{krand}}{\to} \{0,1\}$, there is a negligible function ϵ such that it is infeasible to find strings x and z such that

  $\left|\text{Prob}\left[D\left(z, x, \text{view}_{P^*}(\langle P^*(z), V\rangle(x)\right) = 1\right] - \text{Prob}_{(u,y)\leftarrow E(P^*,z,x)}\left[D(z,x,u) = 1\right]\right|$

  is greater than $\epsilon(k)$.

- Computational Witness-Extension: For all feasible algorithms $P^* \in \text{algs}$, there is a negligible function ϵ such that it is infeasible to find strings x and z such that

  $\text{Prob}_{(u,y)\leftarrow E(P^*,z,x)}\left[V \text{ accepts in } u \text{ and } (x,y) \in R_N\right] \; > \; \epsilon(k)$.

As before, one can get a *semi-uniform* [15] notion by interpreting the *infeasible* s as non-uniform with respect to x and uniform with respect to z, and all other *feasible* s as uniform. That completes the complicated definitions. With them out of the way, I now give definitions which simply put those concepts together.

Definition 4.2.11. For all promise-relations (R_Y, R_N), the statistical zero knowledge proof (resp. argument) systems for (R_Y, R_N) are the pairs $S, \langle P, V\rangle$ such

that $\langle P, V \rangle$ is an efficient-prover interactive proof (resp. argument) system for (R_Y, R_N) and S is a statistical zero-knowledge simulator for $\langle P, V \rangle$.

Definition 4.2.12. Computational zero-knowledge proof (resp. argument) systems for (R_Y, R_N) are defined using computational zero-knowledge simulators, but otherwise as in the definition of statistical zero-knowledge proof (resp. argument) systems.

Definition 4.2.13. For all promise-relations (R_Y, R_N), the proof (resp. argument) of knowledge systems for (R_Y, R_N) are the pairs $\langle P, V \rangle, E$ such that $\langle P, V \rangle$ is an efficient-prover interactive proof (resp. argument) system for (R_Y, R_N) and E is a statistical (resp. computational) witness-extended emulator for $\langle P, V \rangle$.

Definition 4.2.14. For all promise-relations (R_Y, R_N), the statistical zero knowledge proof (resp. argument) of knowledge systems for (R_Y, R_N) are the triples $S, \langle P, V \rangle, E$ such that $S, \langle P, V \rangle$ is a statistical zero-knowledge proof (resp. argument) system and $\langle P, V \rangle, E$ is a proof (resp. argument) of knowledge system.

Definition 4.2.15. Computational zero-knowledge proof (resp. argument) of knowledge systems for (R_Y, R_N) are defined using computational zero-knowledge simulators, but otherwise as in the definition of statistical zero-knowledge proof (resp. argument) of knowledge systems.

There are implications between those properties and flavors. However, we do not go further into those implications. Instead, we recall that, as long as the prover is required to be *efficient*, an NP *language* is not sufficient for such protocols, since the completeness condition involves the prover being given a witness, and what strings are witnesses depends on more than just the language.

Witness Encryption is a far more recent development. The idea is that it is like public-key encryption, but instead of coming with a specific key generation algorithm, they let the encryptor use any instance (of, for example, an NP relation) as a *public key* , and the corresponding *private keys* are the witnesses for that instance. Witness Encryption for NP relations can easily be constructed from Indistinguishability Obfuscation.

4.3 Informal discussion of promiseMA

Suppose we have fixed an instance x of the problem and a string y that Alice claims to be a witness for x. Let n be an upper bound on the amount of randomness which the promiseMA verifier might use on (x, y), let $S = \{0,1\}^n$, and let A be the subset of strings in S which cause the promiseMA verifier to accept (x, w). The promiseMA relation (R_Y, R_N) has the properties: $(x, y) \in R_Y$ when *most* elements of S are in A, and $(x, y) \in R_N$ when *most* elements of S are *not* in A.

The goal is to distinguish these cases. However, NP verifiers cannot themselves flip coins, so any randomness for that purpose must be provided as part of their input. Additionally, the two parties providing that input do not trust each other, so the procedure must work (with high probability) even when only one of its two alleged-randomness strings is chosen honestly. That yields the informal problem we describe next.

Let S be a non-empty finite set, and let A be a subset of S. Suppose Alice claims that *most* elements of S are in A, and Bob claims that *most* elements of S are not in A. Since S is non-empty and finite, at least one of them is lying.

Suppose Charlie would like to identify one of them as a liar, but S is too large to brute-force, and Charlie cannot make his own random choices. However, we *do* let Charlie take two strings of alleged-randomness; one from Alice and one from Bob. Charlie wants to use these strings so that he outputs either *Alice lied* or *Bob lied* with the following requirements.

If most elements of S are in A then his probability of outputting *Alice lied* is extremely small, and if most elements of S are *not* in A then his probability of outputting *Bob lied* is extremely small.

We would prefer that the alleged-randomness strings Charlie receives be fairly short, and that Charlie only needs to check a fairly small number of elements

of S. We will now give a general description, and then a formal definition, for randomness-efficient samplers.

For that purpose, we *do* let Charlie use randomness, generated on his own rather than from Alice/Bob. In this case, by the Chernoff bounds, Charlie can estimate $|A|/|S|$ with decent accuracy and high probability by just drawing independent samples from S. However, that requires a fairly large amount of randomness. We would like Charlie to make do with much less, without decreasing his accuracy too much. That is the basic idea behind randomness efficient-samplers.

4.4 Joint Sampling

We write $\mathbb{Z}_k$ for the set $\{0, 1, \cdots, k-1\}$ of integers i such that $0 \leq i < k$, and $S = \{0,1\}^n$ is all strings of length n. A collection of strings of length n is then a subset of S, and can be regarded as a function $S \to \{0,1\}$.

Healy's main theorem yields something which is stronger than what we need in two ways. It concerns functions

$$S \to [0,1]$$

instead of subsets of S. One might think of this as assigning a *degree to which* a given string is in the collection. Also, Healy's theorem allows functions that are

different for the different output-indices. This motivates the following, which is based on Definition 2 of [Healy].

Definition 4.4.1. A function

$$\Gamma = (\Gamma_0, \cdots, \Gamma_{k-1}) : \{0,1\}^m \to S^k$$

is a *strong* (γ, ϵ)*-averaging sampler* if the following holds. Given functions f_0, f_1, ..., $f_{k-1} : S \to [0,1]$, define $F : S^k \to [0,1]$ by

$$F(x_0, x_1, x_2, \cdots, x_{k-2}, x_{k-1}) = \mathrm{E}_i[f_i(x_i)] = \frac{1}{k}\sum_{i=0}^{k-1} f_i(x_i)$$

Then

$$\Pr\left[|F \circ \Gamma - \mu| > \epsilon\right] \leq \gamma$$

where $\mu = \mathrm{E}_{x \in S^k}[F(x)]$ is the mean of F. We call m the *seed-length* of the sampler, and k is the *sample complexity* of the sampler.

To show the definition's similarity to what we described, observe that

$$|F(\Gamma(s)) - \mu| \leq \epsilon$$

is equivalent to

$$|\mathrm{E}_i[f_i(\Gamma_i(s))] - \mathrm{E}_i[\mathrm{E}_s[f_i(s)]]| \leq \epsilon\ .$$

Moreover, if all the $f_i = f$ this simplifies to

$$|\mathrm{E}_i[f(\Gamma_i(s))] - \mathrm{E}_s[f(s)]| \leq \epsilon$$

Next we give similar definitions for the related objects we will be constructing.

Definition 4.4.2. The *set of strings Alice chooses from* is $\mathcal{A} = \{0,1\}^{m_A}$, and the *set of strings Bob chooses from* is $\mathcal{B} = \{0,1\}^{m_B}$.

Definition 4.4.3. A *weak* (γ, ϵ)*-joint sampler* is a function

$$J = (J_0, \cdots, J_{k-1}) : \mathcal{A} \times \mathcal{B} \to S^k$$

such that for all functions $f : S \to \{0,1\}$, when $F : S^k \to [0,1]$ is defined by

$$F(x_0, x_1, x_2, \cdots, x_{k-2}, x_{k-1}) = \mathrm{E}_i[f(x_i)] = \frac{1}{k}\sum f(x_i)$$

and μ is the mean of f, we have the following two conditions:

- *Bob's security*

  $$\mathrm{Prob}_{\beta \in \mathcal{B}}\left(\exists \alpha \in \mathcal{A} \ \ |(F \circ J)(\alpha, \beta) - \mu| > \epsilon\right) < \gamma$$

  This is a formalization of *It is unlikely that Bob's choice of string,* β*, allows Alice to chose a string,* α*, where the mean on the sampler's outputs differs from the true mean by more than* ϵ*.*

- *Alice's security*

  $$\forall\ \beta \in \mathcal{B} \quad \mathrm{E}(F \circ J) = \mu$$

  where the expectation is over the choice of $\alpha \in \mathcal{A}$ and $i \in \mathbb{Z}_k$ This is a formalization of *Whatever Bob does, the expected value using the sampler, over all Alice's choices of string and index, equals the true mean.*

We call k the *sample complexity* of the joint sampler.

In what follows, $\oplus$ can be any binary operation operation on S whose multiplication table is a Latin square: i.e., each column and each row has each symbol exactly once. In particular, $\oplus$ could be any group multiplication, for example bitwise exclusive-or.

Theorem 4.4.4. *Given $\gamma, \epsilon > 0$ and a strong $(\gamma/(2^{m_A}), \epsilon)$-averaging sampler Γ, the function*

$$J : S \times \mathcal{B} \to S^k$$

given by $J_i(\alpha, \beta) = \alpha \oplus \Gamma_i(\beta)$ is a weak (γ, ϵ)-joint sampler.

Proof: Suppose μ is the mean of a function $f : \mathcal{A} \to \{0, 1\}$. Fix an element $\alpha \in S$, and let $\hat{f} : \mathcal{A} \to [0, 1]$ be given by $\hat{f}(r) = f(\alpha \oplus r)$. By the assumption on $\oplus$, $r \mapsto \alpha \oplus r$ is a bijection on S. It follows that

$$\mathrm{E}_r(\hat{f}(r)) = \mathrm{E}_r(f(\alpha \oplus r)) = \mathrm{E}_r(f(r)) = \mu$$

This holds for all $\alpha \in \mathcal{A}$, so the probability, over the choice of β, of *any particular* $\alpha \in \mathcal{A}$ being such that

$$|(F \circ J)(\alpha, \beta) - \mu| > \epsilon \cdot k$$

is at most $\gamma/(2^{m_A})$ by (4.4.1). Therefore, by a union bound, the probability that there *exists* an element $\alpha \in \mathcal{A}$ for which that inequality holds is at most $|\mathcal{A}| \cdot \gamma/(2^{m_A}) = \gamma$. Thus Bob's security holds.

Now, fix an element $\beta \in \mathcal{B}$, and an integer $0 \leq j < k$. By the assumption on $\oplus$, $r \mapsto r \oplus \Gamma(\beta)_j$ is a bijection, so

$$\mathrm{E}_r\left(f\left(r \oplus \Gamma(r_b)_j\right)\right) = \mathrm{E}_r(f(r)) = \mu$$

Since this holds for all j, the equality will still hold when the expectation is also over the choice of j. Thus Alice's security also holds. Therefore J is a weak (γ, ϵ)-joint sampler, as claimed.□

4.5 Weak Joint Samplers, NP, and promiseMA

The efficient construction of weak joint samplers with good-enough parameters already suffices for modifying *most* schemes for handling NP relations into schemes for handling promiseMA promise-relations. For 2-party protocols designed for NP relations, if the following five conditions all apply to each time such a relation is used then the protocol can be converted to handle promiseMA relations.

- exactly one party will supply a possible witness
- before that party does so, it will know the NP relation and the instance
- if that party is honest, then the possible witness will be a valid witness

- (even if all strings are valid witnesses,) that party as an adversary can never gain by making the protocol continue as if the supplied possible witness was not a valid witness

- if the other party somehow knew that the party referred to in the previous four bullet points knows a valid witness, then there would be no need for the latter party to actually supply its witness

Converted protocols created in the manner described in this paper have three downsides:

1) For each promiseMA relation that does not have perfect completeness, each party that supplies a possible witness will usually need to be promiseZP: To continue with the protocol, those parties will need to find a string of a specific length satisfying a condition that they can efficiently-and-deterministically check, where *if* their supplied witness is a valid witness then more than 1/4 of the strings with the specified length satisfy the condition.

2) For each party, before that party supplies a possible witness for a promiseMA relation, that party must know a sufficiently long string which the other party trusts was random conditioned on the promiseMA relation, where the threshold for *sufficiently long* is a linear function of the security parameter plus an upper bound on the length of the instance plus an upper bound on the amount of randomness that the promiseMA verifier might use.

3) For each party, if that party supplies any possible witness, then the *other* party's security will be at most statistical (rather than perfect). The conversion can also be applied to *schemes* which can tolerate those downsides.

Suppose (R_Y, R_N) is a promiseMA relation. Let p be a polynomial upper-bound on its witness lengths. As mentioned just after Definition 1.6, and described in detail below, the conversion consists of applying something like a reduction from the promiseMA relation to an NP relation. Suppose $r\_ver\_(x, y, r)$ is an algorithm which solves (R_Y, R_N) with thresholds $1/3, 2/3$ using at most $\ell(n)$ bits of randomness where n is an upper bound on length(x). Suppose we are also given two polynomially-bounded functions $q_a, q_b : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ which, when measured by their inputs (instead of the length of those inputs), are polynomially-bounded and computable in polynomial time. Finally suppose we have an efficiently-computable doubly-indexed sequence

$$J_{k,n} : \mathcal{A}_n \times \mathcal{B}_n \to (S_n)^{k(n)} \qquad n \in \mathbb{N}$$

of weak $(2^{-(p(n)+n+k+2)}, 1/7)$-joint samplers where $\mathcal{A}_n = \{0,1\}^{q_a(n)}$ and $\mathcal{B}_n = \{0,1\}^{q_b(n)}$ and $S_n = \{0,1\}^{\ell(n)}$ and $k(n)$ is the *sample complexity* of J_n.

Lemma 4.5.1. *The relation R_{NP} defined as follows is an NP relation. Suppose x, y, α, β are strings. Then $(k, x, \beta)R_{NP}(y, \alpha)$ if and only if the following two conditions both hold.*

1) $\text{length}(x) = n, \quad \text{length}(\beta) = q_b(n), \quad \text{length}(\alpha) = q_a(n)$

and $\text{length}(y) \leq p(n)$.

2) $\left|\{i : \text{r\_ver\_}(x, y, ((J_{k,n}(\alpha, \beta))_i) = \textit{accept}\}\right| > k(n)/2$

i.e. A majority of the coordinates z of $J_{k,n}(\alpha, \beta)$ are such that $\text{r\_ver\_}(x, y, z)$ accepts

Proof. Since lengths are non-negative and q_a is polynomially-bounded, R_{NP}'s witness-lengths are bounded by a polynomial in the length of its instances and the functions $J_{k,n}$ are efficiently-computable, R_{NP} is also efficiently computable. □

Now, the thing which is close-enough to a reduction from (R_Y, R_N) to R_{NP}, is given by

- $\text{proverLen}(k, n) = q_a(n)$
- $\text{verifierLen}(k, n) = q_b(n)$
- $\text{convert\_instance\_}(k, x, \alpha, \beta) = (k, x, \beta)$
- $\text{forward\_witness\_transfer\_}(k, x, \alpha, \beta, y) = (y, \alpha)$
- $\text{backward\_witness\_transfer\_}(k, x, \alpha, \beta, z) = \text{left\_entry\_}(z)$

Let *Alice* be the party that will supply a possible witness, and let *Bob* be the other party. For example, for interactive arguments, Alice is the prover and Bob

is the verifier, whereas for witness encryption, Bob is the encrypter and Alice is the decrypter.

Lemma 4.5.2 (Alice has many suitable strings)**.** *Suppose x, y, α and β are strings with $\ell(x) = n,\ \ell(y) = p(n),\ \ell(\alpha) = q_a(n),\ \ell(\beta) = q_b(n)$. Suppose $(x, y) \in R_Y$, then*

$$|\{\alpha \in \mathcal{A} : (k, x, \beta) R_{NP}(y, \alpha)\}| > |\mathcal{A}|/4$$

Proof. By the definition of being a verifier for (R_Y, R_N)

$$\Pr_r(\text{rand\_verif\_}(x, y, r) \text{ accepts}) \geq 2/3.$$

Define

$$W = |\{i : 0 \leq i < k, \text{rand\_verif\_}(x, y, (J_{k,n}(\alpha, \beta))_i) \text{ accepts}\}|/k$$

By (4.4.3) Alice's security for $J_{k,n}$, it follows that

$$\mathrm{E}_\alpha \left[W\right] = \Pr_r(\text{rand\_verif\_}(x, y, r))$$

Since $W \leq 1$, and $E_\alpha(W) \geq 2/3$, then $Pr_\alpha(W > 1/2) > 1/4$. Recall that $(k, x, \beta) R_{NP}(y, \alpha)$ is the condition $W > 1/2$, so the result follows. □

Lemma 4.5.3 (Bob's security)**.**

$$\text{Prob}_\beta \left[\exists x, y, \alpha \ \ ((x, y) \in R_N \ \textit{and}\ (k, x, \beta) R_{NP}(y, \alpha))\right] < \frac{1}{2^k}$$

Here x is a string of length at most n, y is a string of length at most $p(n)$, and $\alpha \in \mathcal{A}$.

In other words, it is highly likely that witnesses for R_{NP} are either witnesses for the promiseMA relation, or strings that together with x make the promise-relation's promise fail.

Proof. For x and y as above, if $(x, y) \in R_N$ then by Bob's security for the joint sampler $J_{k,n}$, the probability (over the choice of β) of there existing an α such that $(k, x, \beta)R_{NP}(y, \alpha)$ is true is at most $2^{-(p(n)+n+k+2)}$ because $J_{k,n}$ is a weak $(2^{-(p(n)+n+k+2)}, 1/7)$-joint sampler. Observe that $2^{-(p(n)+n+k+2)} = \frac{1}{2^{n+1}} \cdot \frac{1}{2^{p(n)+1}} \cdot \frac{1}{2^k}$. Furthermore, there are fewer than 2^{n+1} strings of length at most n and there are fewer than $2^{p(n)+1}$ strings of length at most $p(n)$, so the result follows from the union bound. □

Thus, we indeed have something resembling a reduction, as described just after Definition 1.6 . Using this, one converts protocols that satisfy the conditions described at the start of this section and are for NP relations, to handling promiseMA promise-relations, as follows. Bob receives an upper-bound n on the length of the promiseMA instance, chooses $\beta \in \mathcal{B}_n$ uniformly at random, and sends β to Alice. Both parties use (k, x, β) as the instance for R_{NP}. Once Alice receives β and the possible witness y, Alice tries random elements $\alpha \in \mathcal{A}_n$ until

Alice finds a value of α that makes $(k, x, \beta)R_{NP}(y, \alpha)$ true. When Alice finds such an α, Alice uses (y, α) as the witness for the NP relation.

We now show that this conversion statistically preserves security, and preserves completeness when Alice is promiseZP, and preserves efficiency.

To start, note that one of the assumptions was that Alice can never gain by making the protocol continue as if the supplied possible witness was not a valid witness. Accordingly, for Bob's security, we only need the other direction. That direction follows directly from Lemma 5.3 and the choice of backward_witness_transfer_. We now show efficiency and completeness and Alice's security.

Since R_{NP} is an NP relation, the only thing which might not be efficient is Alice finding a string $\alpha \in \mathcal{A}_n$ such that

convert_instance_$(k, x, \alpha, \beta)R_{NP}$ forward_witness_transfer_(k, x, α, β, w).

However, that is equivalent to $(k, x, \beta)R_{NP}(y, \alpha)$, so if $(x, y) \in R_Y$ then by Lemma 5.2 more than 1/4 of the strings in $\mathcal{A}_n$ satisfy that condition, in which case a promiseZP Alice finds such an α with probability 1, and trying a large number of independently chosen candidates for α finds a suitable α with probability exponentially close to 1. Thus efficiency holds.

The original protocol or scheme was assumed able to handle NP relations, and both parties are using it with the NP relation R_{NP} and instance convert_instance_(k, x, α, β). Given a witness y for the promiseMA relation and

an α as above, forward_witness_transfer_(k, x, α, β, w) will be a witness for R_{NP}. Thus completeness holds to the extent claimed.

Given any witness y for the promiseMA relation, regardless of *which* witness y is, Alice will find a witness for R_{NP} with either with probability 1 or with probability exponentially close to 1. Furthermore, by the choice of forward_witness_transfer_, although the length of that witness may depend on length(y), the length of the witness does not otherwise depend on y, so a simulator or reduction can just use a string of length(y) zeros to find the length of the witness for R_{NP}. Thus Alice's security also holds.

4.6 Constructing strong averaging Samplers

This section is devoted to proving the following theorem which is based very closely on [Healy]'s Theorem 1. The motivation is that in order to facilitate simpler instantiations which use expander *multi*graphs rather than restricting to expander *graphs*, we want a theorem whose statement covers multigraphs too. We decided to generalize it further, to possibly time-dependent Markov processes, because the proof is not significantly harder.

Theorem 4.6.1. *Given an integer $N > 0$ set $S = \{0, 1, ..., N-1\}$ and $\mathbf{1} = (1, ..., 1)$. Suppose $0 < \lambda < 1$ and $\mathcal{M}$ is a possibly time-dependent Markov process*

on S such that for all transition matrices M of $\mathcal{M}$, one has $M\mathbf{1} = \mathbf{1}$, and for all $\mathbf{v} \in \mathbb{R}^N$, if $\mathbf{1}^T\mathbf{v} = 0$ then $||M\mathbf{v}|| \leq \lambda \cdot ||\mathbf{v}||$.

Given $0 < \epsilon < 1$ and an integer $k > 0$, and, for $0 \leq i \leq k-1$, functions $f_i : S \to [0,1]$, and given x_0 define $x_j = \mathcal{M}(x_{j-1})$ for $1 \leq j \leq k-1$. If x_0 is chosen uniformly from S then

$$\Pr\left[\ |\mathrm{E}_i(f_i(x_i)) - \mathrm{E}_i(\mathrm{E}_x\left[f_i(x)\right])|\ \geq \epsilon\right] \leq 2 \cdot \exp\left(-\frac{\epsilon^2 \cdot (1-\lambda) \cdot k}{4}\right)$$

Proof of Theorem (4.6.1). For $i \in S$, set $\mu_i = \mathrm{E}_x\left[f_i(x)\right]$. Let

$$C = \mathrm{E}_i(f_i(x_i)) - \mathrm{E}_i(\mu_i)$$

Then

$$\Pr\left[|C| \geq \epsilon\right] = \Pr\left[C \geq \epsilon\right] + \Pr\left[C \leq -\epsilon\right]$$

We will show

$$\Pr\left[C \geq \epsilon\right] \leq \exp\left(-\frac{\epsilon^2 \cdot (1-\lambda) \cdot k}{4}\right)$$

By replacing each f_i by $(1 - f_i)$, and applying the same argument, it follows that $\Pr\left[C \leq -\epsilon\right]$ has the same upper bound, and the conclusion then follows.

Introduce a parameter r with

$$0 < r < 1/2 \quad \textit{and} \quad \exp(r) \leq 1/\sqrt{\lambda} \tag{4.1}$$

then

$$\begin{aligned}
& \mathrm{E}_i(f_i(x_i)) - \mathrm{E}_i(\mu_i) \;\geq\; \epsilon \\
\iff \quad & \mathrm{E}_i(f_i(x_i)) \;\geq\; \epsilon + \mathrm{E}_i(\mu_i) \\
\iff \quad & rk \cdot \mathrm{E}_i(f_i(x_i)) \;\geq\; rk \cdot (\epsilon + \mathrm{E}_i(\mu_i)) \\
\iff \quad & \exp\left(r \cdot \left(\sum_{0 \leq i < k} f_i(x_i)\right)\right) \;\geq\; \exp\left(r \cdot \left(k \cdot \epsilon + \sum_{0 \leq i < k} \mu_i\right)\right) \\
\iff \quad & \prod_{0 \leq i < k} \exp\left(r \cdot f_i(x_i)\right) \;\geq\; \exp\left(r \cdot \left(k \cdot \epsilon + \sum_{0 \leq i < k} \mu_i\right)\right)
\end{aligned}$$

Since $\exp(t) > 0$, we apply Markov's inequality to the last of the above inequalities to get

$$\begin{aligned}
& \Pr\left[\mathrm{E}_i(f_i(x_i)) - \mathrm{E}_i(\mu_i) \geq \epsilon\right] \qquad (4.2) \\
\leq\; & \mathrm{E}\left[\prod_{0 \leq i < k} \exp\left(r \cdot f_i(x_i)\right)\right] \Bigg/ \exp\left(r \cdot \left(k \cdot \epsilon + \sum_{0 \leq i < k} \mu_i\right)\right)
\end{aligned}$$

Accordingly, we now work on getting an upper bound on the expectation

$$\mathrm{E}\left[\prod_{0 \leq i < k} \exp\left(r \cdot f_i(x_i)\right)\right]$$

Let $M_1, M_2, \cdots, M_{k-1}$ be the first $k-1$ transition matrices of $\mathcal{M}$,

and let $E_0, \cdots, E_{k-1}$ be the $N \times N$ diagonal matrices such that

$$(E_i)_{j,j} = \exp(r \cdot f_i(j)) \qquad (4.3)$$

Then

$$\mathrm{E}\left[\prod_{0 \leq i < k} \exp\left(r \cdot f_i(x_i)\right)\right] = \mathbf{1}^T E_{k-1} M_{k-1} E_{k-2} M_{k-2} \cdots E_1 M_1 E_0 \left(\frac{1}{N} \cdot \mathbf{1}\right)$$

because the right hand side is a sum of terms. Each term corresponds to a sequence of possible values for $x_0, x_1, \cdots, x_{k-1}$, and is the probability of that sequence times $\prod_{0 \le i < k} \exp\big(r \cdot f_i(x_i)\big)$.

Let M_0 be the $N \times N$ matrix whose entries are all $1/N$. Then

$$M_0\left(\frac{1}{N} \cdot \mathbf{1}\right) = \frac{1}{N} \cdot (M_0 \mathbf{1}) = \frac{1}{N} \cdot \mathbf{1}$$

so

$$\begin{aligned} &\mathrm{E}\left[\prod_{0 \le i < k} \exp\big(r \cdot f_i(x_i)\big)\right] \\ &= \mathbf{1}^T E_{k-1} M_{k-1} E_{k-2} M_{k-2} \cdots E_1 M_1 E_0 M_0 \left(\tfrac{1}{N} \cdot \mathbf{1}\right) \end{aligned} \tag{4.4}$$

We now study the sequence of vectors $\mathbf{z}_0, \mathbf{z}_1, \cdots, \mathbf{z}_k$ given inductively by

$$\mathbf{z}_0 = \frac{1}{N}\mathbf{1} \qquad \textit{and} \qquad \mathbf{z}_{i+1} = M_i E_i \mathbf{z}_i \tag{4.5}$$

If $\mathbf{v} \in \mathbb{R}^N$ and $\mathbf{1}^T v = 0$ then

$$||M_0 \mathbf{v}|| = \left|\left|\frac{1}{N} \cdot \mathbf{1}\mathbf{1}^T \mathbf{v}\right|\right| = 0 = 0 \cdot ||\mathbf{v}|| \le \lambda \cdot ||\mathbf{v}||$$

In other words, M_0 satisfies the assumptions on the other Ms, so will be handled the same way. Given $v \in \mathbb{R}^N$ let $v^{\shortparallel}$ be the component of v in the direction of $\mathbf{1} \in \mathbb{R}^N$ then $v^{\perp} = v - v^{\shortparallel}$ is the component orthogonal to $\mathbf{1}$.

We now prove the following lemma related to the $\mathbf{z}_i$:

Lemma 4.6.2. *Suppose $f : S \to [0,1]$, and M is an N-by-N stochastic matrix such that,*

$$\mathbf{v} \in \mathbb{R}^N \quad \textit{and} \quad \mathbf{1}^T \mathbf{v} = 0 \qquad \Rightarrow \qquad ||M\mathbf{v}|| \le \lambda \cdot |||\mathbf{v}||$$

Let $\mu = \mathrm{E}_x[f(x)]$*, and let* E *be the* $N \times N$ *diagonal matrix* $(E)_{j,j} = \exp(r \cdot f(j))$*. If* $0 < r < \frac{\log(1/\lambda)}{2}$ *then for* $\mathbf{z} \in \mathbb{R}^{\mathbf{N}}$

$$
\begin{aligned}
&(i) \quad \left\| (EM\mathbf{z}^{\shortparallel})^{\shortparallel} \right\| \leq (1 + (\exp(r) - 1) \cdot \mu) \cdot ||\mathbf{z}^{\shortparallel}|| \\
&(ii) \quad \left\| (EM\mathbf{z}^{\shortparallel})^{\perp} \right\| \leq \frac{\exp(r) - 1}{2} \cdot ||\mathbf{z}^{\shortparallel}|| \\
&(iii) \quad \left\| \left(EM\mathbf{z}^{\perp}\right)^{\shortparallel} \right\| \leq \frac{\exp(r) - 1}{2} \cdot \lambda \cdot \left|\left|\mathbf{z}^{\perp}\right|\right| \\
&(iv) \quad \left\| \left(EM\mathbf{z}^{\perp}\right)^{\perp} \right\| \leq \sqrt{\lambda} \cdot \left|\left|\mathbf{z}^{\perp}\right|\right|
\end{aligned}
$$

Proof. Given f, M, r and E as above and $\mathbf{z} \in \mathbb{R}^N$. Since $\mathbf{z}^{\shortparallel} = c.\mathbf{1}$ for some scaler c, and $M\mathbf{1} = \mathbf{1}$, it follows that $M\mathbf{z}^{\shortparallel} = \mathbf{z}^{\shortparallel}$. Then

$$(EM\mathbf{1})^{\shortparallel} = (E\mathbf{1})^{\shortparallel} = \frac{1}{N} \cdot \left(\mathbf{1}^T E\mathbf{1}\right)\mathbf{1} = \mathrm{E}_x[\exp(r \cdot f(x))]\mathbf{1}$$

Using linearity

$$(EM\mathbf{z}^{\shortparallel})^{\shortparallel} = \mathrm{E}_x[\exp(r \cdot f(x))]\mathbf{z}^{\shortparallel} \tag{4.6}$$

Since $\exp(r \cdot x)$ is a convex function of x, on the interval $0 \leq y \leq 1$ we get

$$\exp(r \cdot y) \leq 1 + ((\exp(r) - 1) \cdot y) \tag{4.7}$$

$$
\begin{aligned}
&\left\|(EM\mathbf{z}^{\shortparallel})^{\shortparallel}\right\| \\
&= \|\mathrm{E}_x[\exp(r \cdot f(x))]\mathbf{z}^{\shortparallel}\| && \textit{using } 4.6 \\
&= |\mathrm{E}_x[\exp(r \cdot f(x))]| \cdot ||\mathbf{z}^{\shortparallel}|| \\
&= \mathrm{E}_x[\exp(r \cdot f(x))] \cdot ||\mathbf{z}^{\shortparallel}|| \\
&\leq \mathrm{E}_x[1 + ((\exp(r) - 1) \cdot f(x)] \cdot ||\mathbf{z}^{\shortparallel}|| && \textit{using } 4.7 \\
&= (1 + ((\exp(r) - 1) \cdot \mathrm{E}_x[f(x)])) \cdot ||\mathbf{z}^{\shortparallel}|| \\
&= (1 + ((\exp(r) - 1) \cdot \mu)) \cdot ||\mathbf{z}^{\shortparallel}||
\end{aligned}
$$

This proves (i). For (ii), let I be the N-by-N identity matrix. Both maps $\mathbf{v} \mapsto \mathbf{v}^{\shortparallel}$ and $\mathbf{v} \mapsto \mathbf{v}^{\perp}$ are linear and do not increase norm, and $(\mathbf{v}^{\shortparallel})^{\perp} \equiv 0$. So given α

$$
\begin{aligned}
&(EM\mathbf{z}^{\shortparallel})^{\perp} \\
&= (E\mathbf{z}^{\shortparallel})^{\perp} \\
&= (E\mathbf{z}^{\shortparallel})^{\perp} - (\alpha I\mathbf{z}^{\shortparallel})^{\perp} + (\alpha\mathbf{z}^{\shortparallel})^{\perp} \\
&= ((E\mathbf{z}^{\shortparallel}) - (\alpha I\mathbf{z}^{\shortparallel}))^{\perp} \\
&= ((E - \alpha I)\,\mathbf{z}^{\shortparallel})^{\perp}
\end{aligned}
$$

We apply this with $\alpha = \frac{\exp(r)+1}{2}$. From (4.3) we see that the matrix E is diagonal with entries are between 1 and $\exp(r)$. Thus $E - \alpha I$ is diagonal with entries bounded in absolute value by $\frac{\exp(r)-1}{2}$. Then

$$
\left\|(EM\mathbf{z}^{\shortparallel})^{\perp}\right\| = \left\|((E - \alpha I)\,\mathbf{z}^{\shortparallel})^{\perp}\right\| \leq \|(E - \alpha I)\,\mathbf{z}^{\shortparallel}\| \leq \frac{\exp(r) - 1}{2} \cdot \|\mathbf{z}^{\shortparallel}\|
$$

This proves (ii). For (iii), since M is stochastic, $\left(M\mathbf{z}^{\perp}\right)^{"} = 0$, and

$$\begin{aligned}
&\left(EM\mathbf{z}^{\perp}\right)^{"} \\
&= \left(EM\mathbf{z}^{\perp}\right)^{"} - \left(\alpha IM\mathbf{z}^{\perp}\right)^{"} + \left(\alpha M\mathbf{z}^{\perp}\right)^{"} \\
&= \left(\left(EM\mathbf{z}^{\perp}\right) - \left(\alpha IM\mathbf{z}^{\perp}\right)\right)^{"} \\
&= \left((E-\alpha I)M\mathbf{z}^{\perp}\right)^{"}
\end{aligned}$$

Substituting for α as above gives

$$\begin{aligned}
&\left|\left|\left(EM\mathbf{z}^{\perp}\right)^{"}\right|\right| \\
&= \left|\left|\left((E-\alpha I)M\mathbf{z}^{\perp}\right)^{"}\right|\right| \\
&\leq \left|\left|(E-\alpha I)M\mathbf{z}^{\perp}\right|\right| \\
&\leq \frac{\exp(r)-1}{2} \cdot \left|\left|M\mathbf{z}^{\perp}\right|\right| \\
&\leq \frac{\exp(r)-1}{2} \cdot \lambda \cdot \left|\left|\mathbf{z}^{\perp}\right|\right|
\end{aligned}$$

This proves (iii). For (iv)

$$\begin{aligned}
&\left|\left|\left(EM\mathbf{z}^{\perp}\right)^{\perp}\right|\right| \\
&\leq \left|\left|EM\mathbf{z}^{\perp}\right|\right| \\
&\leq \exp(r) \cdot \left|\left|M\mathbf{z}^{\perp}\right|\right| \\
&\leq \exp(r) \cdot \lambda \cdot \left|\left|\mathbf{z}^{\perp}\right|\right| \\
&\leq \sqrt{\lambda} \cdot \left|\left|\mathbf{z}^{\perp}\right|\right|
\end{aligned}$$

The last step uses $\exp(r) < 1/\sqrt{\lambda}$, completing the proof of (iv).$\square$

Next we use the above lemma to show that each $\mathbf{z}_i^{\perp}$ remains short relative to those $\mathbf{z}_j^{\shortparallel}$ with $j < i$.

Lemma 4.6.3. *If* $1 \leq i \leq k$ *then* $\left|\left|\mathbf{z}_i^{\perp}\right|\right| \leq \left(\frac{\exp(r)-1}{1-\lambda}\right) \cdot \max_{0 \leq j < i} \left|\left|\mathbf{z}_j^{\shortparallel}\right|\right|$.

Proof. Using the definition (4.5)

$$\begin{aligned}
\left|\left|\mathbf{z}_{i+1}^{\perp}\right|\right| &= \left|\left|(E_i M_i \mathbf{z}_i)^{\perp}\right|\right| \\
&= \left|\left|\left(E_i M_i \left(\mathbf{z}_i^{\shortparallel} + \mathbf{z}_i^{\perp}\right)\right)^{\perp}\right|\right| \qquad \because \ \mathbf{z}_i = \mathbf{z}_i^{\shortparallel} + \mathbf{z}_i^{\perp} \\
&\leq \left|\left|(E_i M_i \mathbf{z}_i^{\shortparallel})^{\perp}\right|\right| + \left|\left|\left(E_i M_i \mathbf{z}_i^{\perp}\right)^{\perp}\right|\right|
\end{aligned}$$

Thus, by parts (ii) and (iv) of lemma (4.6.2),

$$\left|\left|\mathbf{z}_{i+1}^{\perp}\right|\right| \leq \frac{\exp(r)-1}{2} \cdot ||\mathbf{z}_i^{\shortparallel}|| + \sqrt{\lambda}\left|\left|\mathbf{z}_i^{\perp}\right|\right|$$

By definition (4.5) $\left|\left|\mathbf{z}_0^{\perp}\right|\right| = \left|\left|((1/N) \cdot \mathbf{1})^{\perp}\right|\right| = 0$, so recursively applying the bound above yields

$$\begin{aligned}
&\left|\left|\mathbf{z}_{i+1}^{\perp}\right|\right| \\
&\leq \frac{\exp(r)-1}{2} \cdot \left(\sum_{j=0}^{i} \left(\left(\sqrt{\lambda}\right)^{j+1} \cdot \left|\left|\mathbf{z}_{i-j}^{\shortparallel}\right|\right|\right)\right) \\
&\leq \frac{\exp(r)-1}{2} \cdot \left(\sum_{j=0}^{i} \left(\sqrt{\lambda}\right)^{j+1}\right) \cdot \max_{0 \leq j \leq i} \left|\left|\mathbf{z}_j^{\shortparallel}\right|\right|
\end{aligned}$$

Since $0 < \lambda < 1$,

$$\sum_{j=0}^{i} \left(\sqrt{\lambda}\right)^{j+1} \leq \frac{1}{1-\sqrt{\lambda}}$$

so we get

$$\left|\left|\mathbf{z}_{i+1}^{\perp}\right|\right| \leq \frac{\exp(r)-1}{2} \cdot \frac{1}{1-\sqrt{\lambda}} \cdot \max_{0\leq j\leq i} \left|\left|\mathbf{z}_{j}^{\shortparallel}\right|\right|$$

Finally, again using $0 < \lambda < 1$, we have

$$\frac{1}{1-\sqrt{\lambda}} = \frac{1+\sqrt{\lambda}}{1-\lambda} \leq \frac{2}{1-\lambda}$$

so

$$\left|\left|\mathbf{z}_{i+1}^{\perp}\right|\right| \leq \frac{\exp(r)-1}{1-\lambda} \cdot \max_{0\leq j\leq i} \left|\left|\mathbf{z}_{j}^{\shortparallel}\right|\right|$$

and replacing i with $i-1$ yields the claimed result. □

We now use this lemma to bound the values of $||\mathbf{z}_i^{\shortparallel}||$.

Lemma 4.6.4. *If* $1 \leq i \leq k$ *then*

$$||\mathbf{z}_i^{\shortparallel}|| \leq \exp\left((\exp(r)-1)\cdot\mu_i + \frac{\lambda\cdot(\exp(r)-1)^2}{2\cdot(1-\lambda)}\right) \cdot \max_{0\leq j<i} \left|\left|\mathbf{z}_{j}^{\shortparallel}\right|\right|$$

Proof. Using the definition (4.5) again

$$\left|\left|\mathbf{z}_{i+1}^{\shortparallel}\right|\right| = \left|\left|(E_i M_i \mathbf{z}_i)^{\shortparallel}\right|\right|$$

and using $\mathbf{z}_i = \mathbf{z}_i^{\shortparallel} + \mathbf{z}_i^{\perp}$ gives

$$\left|\left|(E_i M_i \mathbf{z}_i)^{\shortparallel}\right|\right| \leq \left|\left|(E_i M_i \mathbf{z}_i^{\shortparallel})^{\shortparallel}\right|\right| + \left|\left|\left(E_i M_i \mathbf{z}_i^{\perp}\right)^{\shortparallel}\right|\right|$$

So by parts (i) and (iii) of lemma (4.6.2),

$$\begin{aligned}
&\left|\left|\mathbf{z}_{i+1}^{\shortparallel}\right|\right| \\
&\leq \left|\left|\left(E_i M_i \mathbf{z}_i^{\shortparallel}\right)^{\shortparallel}\right|\right| + \left|\left|\left(E_i M_i \mathbf{z}_i^{\perp}\right)^{\shortparallel}\right|\right| \\
&\leq (1 + ((\exp(r) - 1) \cdot \mu_{i+1})) \cdot ||\mathbf{z}_i^{\shortparallel}|| + \frac{\exp(r) - 1}{2} \cdot \lambda \cdot \left|\left|\mathbf{z}_i^{\perp}\right|\right|
\end{aligned}$$

Using (4.6.3) gives

$$(1 + ((\exp(r) - 1) \cdot \mu_{i+1})) \cdot \|\mathbf{z}_i^{\shortparallel}\| + \frac{\exp(r) - 1}{2} \cdot \lambda \cdot \frac{\exp(r) - 1}{1 - \lambda} \cdot \max_{0 \leq j < i} \|\mathbf{z}_j^{\shortparallel}\|$$

Simplifying gives

$$\begin{aligned}
&(1 + (\exp(r) - 1) \cdot \mu_{i+1}) \cdot \|\mathbf{z}_i^{\shortparallel}\| + \frac{\lambda \cdot (\exp(r) - 1)^2}{2 \cdot (1 - \lambda)} \cdot \max_{0 \leq j < i} \|\mathbf{z}_j^{\shortparallel}\| \\
&\leq \left(1 + (\exp(r) - 1) \cdot \mu_{i+1} + \frac{\lambda \cdot (\exp(r) - 1)^2}{2 \cdot (1 - \lambda)}\right) \cdot \max_{0 \leq j \leq i} \|\mathbf{z}_j^{\shortparallel}\|
\end{aligned}$$

Using the fact that $1 + x \leq \exp(x)$ we then conclude that

$$\left|\left|\mathbf{z}_{i+1}^{\shortparallel}\right|\right| \leq \exp\left((\exp(r) - 1) \cdot \mu_{i+1} + \frac{\lambda \cdot (\exp(r) - 1)^2}{2 \cdot (1 - \lambda)}\right) \cdot \max_{0 \leq j \leq i} \left|\left|\mathbf{z}_j^{\shortparallel}\right|\right|$$

As before, replacing i with $i - 1$ yields the desired result. $\square$

To complete the proof of Theorem (4.6.1)

$$||\mathbf{z}_0^{\shortparallel}|| = \left|\left|\frac{1}{N} \cdot \mathbf{1}\right|\right| = \frac{1}{\sqrt{N}}$$

Applying (4.6.4) recursively, for $0 \leq j \leq k$,

$$\left|\left|\mathbf{z}_j^{\shortparallel}\right|\right| \leq \frac{1}{\sqrt{N}} \cdot \prod_{i=1}^{j} \exp\left((\exp(r) - 1) \cdot \mu_i + \frac{\lambda \cdot (\exp(r) - 1)^2}{2 \cdot (1 - \lambda)}\right) \tag{4.8}$$

By (4.4) and (4.5)

$$\begin{aligned}
&\mathrm{E}\left[\prod_{0\le i<k}\exp\left(r\cdot f_i(x_i)\right)\right]\\
&=\mathbf{1}^T E_{k-1}M_{k-1}E_{k-2}M_{k-2}\cdots E_1M_1E_0M_0\left(\frac{1}{N}\cdot\mathbf{1}\right)\\
&=\mathbf{1}^T\mathbf{z}_k\\
&=\mathbf{1}^T\mathbf{z}_k^{\shortparallel}
\end{aligned}$$

By Cauchy-Schwarz this is bounded above by $\sqrt{N}\cdot||\mathbf{z}_k^{\shortparallel}||$. Then using (4.8)

$$\begin{aligned}
&\sqrt{N}\cdot||\mathbf{z}_k^{\shortparallel}||\\
&\le\prod_{i=1}^{k}\exp\left((\exp(r)-1)\cdot\mu_i+\frac{\lambda\cdot(\exp(r)-1)^2}{2\cdot(1-\lambda)}\right)\\
&=\exp\left((\exp(r)-1)\cdot\mu+\left(\frac{\lambda\cdot(\exp(r)-1)^2}{2\cdot(1-\lambda)}\ \cdot k\right)\right)
\end{aligned}$$

where $\mu=\sum_{i=0}^{k-1}\mu_i\le k$ since $\mu_i\le 1$. Since $\exp(1/2)\le 5/3$, and $\exp(r)$ is convex for $0\le r\le 1/2$, it follows that in this range

$$\exp(r)-1\le 4r/3$$

Taylor's formula with remainder says

$$f''(r)=f(0)+rf'(0)+(r^2/2)\xi$$

where $\xi=f''(t)$ for some $0\le t\le r$. Applying this with $f(r)=\exp(r)-1$, and using $\xi\le 5/3$ gives

$$\exp(r)-1\le r+r^2$$

Using this gives

$$\begin{aligned}
\sqrt{N} \cdot ||\mathbf{z}_k^{||}|| &\leq \exp\left(\left(r + r^2\right) \cdot \mu + \left(\frac{\lambda \cdot ((4/3) \cdot r)^2}{2 \cdot (1-\lambda)} \cdot k\right)\right) \\
&\leq \exp\left(r \cdot \mu + r^2 \cdot k + \left(r^2 \cdot \frac{\lambda}{1-\lambda} \cdot k\right)\right) \\
&= \exp\left(r \cdot \mu + r^2 \cdot \left(\frac{1-\lambda}{1-\lambda} + \frac{\lambda}{1-\lambda}\right) \cdot k\right) \\
&= \exp\left(r \cdot \mu + r^2 \cdot \frac{1}{1-\lambda} \cdot k\right) \\
&= \exp\left(r \cdot \mu + \frac{r^2 \cdot k}{1-\lambda}\right)
\end{aligned}$$

So we have shown

$$\mathrm{E}\left[\prod_{0 \leq i < k} \exp\left(r \cdot f_i(x_i)\right)\right] \leq \exp\left(r \cdot \mu + \frac{r^2 \cdot k}{1-\lambda}\right) \tag{4.9}$$

Thus by (4.2) and (4.9) we have

$$\begin{aligned}
&\Pr\left[\left(\sum_{0 \leq i < k} f_i(x_i)\right) - \sum_{0 \leq i < k} \mu_i \ \geq k \cdot \epsilon\right] \\
&\leq \exp\left(r \cdot \mu + \frac{r^2 \cdot k}{1-\lambda}\right) \Bigg/ \exp\left(r \cdot \left(k \cdot \epsilon + \sum_{0 \leq i < k} \mu_i\right)\right) \\
&= \exp\left(r \cdot \mu + \frac{r^2 \cdot k}{1-\lambda}\right) \Bigg/ \exp(r \cdot (k \cdot \epsilon + \mu)) \\
&= \exp\left(r \cdot \mu + \frac{r^2 \cdot k}{1-\lambda} - r \cdot (k \cdot \epsilon + \mu)\right) \\
&= \exp\left(\left(\frac{r^2}{1-\lambda} - r \cdot \epsilon\right) \cdot k\right)
\end{aligned}$$

Set $r = (1-\lambda)\cdot(\epsilon/2)$. Since $0 \leq \epsilon, \lambda \leq 1$ it follows that $r \leq 1/2$ and $\exp(r) < 1/\sqrt{\lambda}$ so r satisfies (4.1). Substituting this value for r in the above gives

$$\begin{aligned}\frac{r^2}{1-\lambda} - r\cdot\epsilon &= \epsilon^2\left((1-\lambda)/4 - \frac{1-\lambda}{2}\right)\\ &\leq -\epsilon^2(1-\lambda)/4\end{aligned}$$

As mentioned earlier, this theorem's conclusion now follows by replacing each f_i with $1-f_i$, applying the same argument, and then applying a union bound. □

Before we state the relevant corollaries, we recall that finite real *symmetric* matrices are all othogonally diagonalizable over the reals.

Corollary 4.6.5. *Given an integer $N > 0$ set $S = \{0, 1, ..., N-1\}$ and $\mathbf{1} = \mathbf{1}_N$. Suppose λ is a real number such that $0 < \lambda < 1$ and $\mathcal{M}$ is a possibly time-dependent Markov process on S such that for all transition matrices M of $\mathcal{M}$, one has M is symetric, and $M\mathbf{1} = \mathbf{1}$, and for all eigenvectors $\mathbf{v}$ of M, if $\mathbf{1}^T\mathbf{v} = 0$ then the absolute value of the corresponding eigenvalue is at most λ.*

The rest of the statement is as in 4.6.1.

Proof. Note that the only change from the theorem is the assumptions on the transition matrices M. We show that the assumptions in the corollary imply the assumptions in the theorem. Let M be a transition matrix for $\mathcal{M}$. Let $e_1 = \mathbf{1}/||\mathbf{1}||, e_2, ..., e_N$ be an orthonormal basis of eigenvectors for M, and let

$\lambda_1, ..., \lambda_N$ be the corresponding eigenvalues so that $Me_i = \lambda_i e_i$. Let $\mathbf{v} \in \mathbb{R}^N$ be such that $\mathbf{1}^T\mathbf{v} = 0$, and let $c_1, ..., c_N$ be the scalars such that $\mathbf{v} = c_1 \cdot e_1 + ... + c_N \cdot e_N$. Since $e_1^T\mathbf{v} = 0/|\mathbf{1}| = 0$ and the basis is orthogonal, $c_1 = 0$. Thus $\mathbf{v} = 0 \cdot e_1 + c_2 \cdot e_2 + ... + c_N \cdot e_N = c_2 \cdot e_2 + ... + c_N \cdot e_N$, so

$$\begin{aligned}
||M\mathbf{v}|| &= ||M\left(c_2 \cdot e_2 + ... + c_N \cdot e_N\right)|| \\
&= ||c_2 \cdot Me_2 + ... + c_N \cdot Me_N|| \\
&= ||c_2 \cdot \lambda_2 \cdot e_2 + ... + c_N \cdot \lambda_N \cdot e_N|| \\
&= \sqrt{\left(c_2 \cdot \lambda_2\right)^2 + ... + \left(c_N \cdot \lambda_N\right)^2} \\
&\leq \lambda\sqrt{c_2^2 + ... + c_N^2} \\
&= \lambda \cdot ||\mathbf{v}||
\end{aligned}$$

□

Definition 4.6.6. A *frum-graph* is an undirected multigraph with finitely many vertices and edges, that is regular and has more than one vertex and at least one edge.

Those are the only graphs we will be concerned with.

Definition 4.6.7. Suppose G is a frum-graph. Let $\mathbf{1}$ be the vector in $\mathbb{R}^{\text{vertices}(G)}$ whose entries are all 1. Let M be the transition matrix for a Morkov process that is a random walk on G. Then we define $\lambda_2(G)$ to be the maximum absolute value of an eigenvalue of M for an eigenvector that is not a multiple of $\mathbf{1}$.

Corollary 4.6.8. *Given an integer $N > 1$ set $S = \{0, 1, ..., N-1\}$. Suppose $0 < \lambda < 1$ and G_j is a sequence of frum-graphs with vertex set S, and $\lambda_2(G_j) \leq \lambda$ for all j.*

Given $0 < \epsilon < 1$ and functions $f_i : S \to [0, 1]$ for $0 \leq i \leq k-1$. Choose x_0 uniformly from S then choose $x_1, x_2, ..., x_{k-1}$ so that x_{j+1} is a random neighbor of x_j in G_j. Then

$$\Pr\left[\ |\mathrm{E}_i(f_i(x_i)) - \mathrm{E}_i(\mathrm{E}_x\left[f_i(x)\right])|\ \geq \epsilon\right] \leq 2 \cdot \exp\left(-\frac{\epsilon^2 \cdot (1-\lambda) \cdot k}{4}\right)$$

Proof. For such N and λ and $\mathcal{G}$, set $\mathbf{1} = \mathbf{1}_N$, let $\mathbf{0}$ be the zero vector in $\mathbb{R}^N$, and let $\mathcal{M}$ be the possibly-time-dependent Markov process given by for each time j and state x, the next state is a random neighbor of x in G_j.

Suppose M is a transition matrix of $\mathcal{M}$. Since the multi-graphs are undirected, M is symmetric. Suppose $\mathbf{1}^T\mathbf{v} = 0$ and $Mv = \lambda v$ then $|\lambda| \leq \lambda_2(G_j)$ because $\mathbf{0}$ is the only multiple of $\mathbf{1}$ that is orthogonal to $\mathbf{1}$. Thus $\mathcal{M}$ satisfies the hypotheses in 4.6.5. By construction, the choice of $x_1, x_2, ..., x_{k-1}$ is equivalent to choosing them according to $\mathcal{M}$. Thus the conclusion follows from 4.6.5. □

Definition 4.6.9. Given an integer $n > 1$ the *8-regular frum-graph* $G(n)$ with vertices $(\mathbb{Z}/n\mathbb{Z})^2$ is defined as follows. The vertex (x, y), is connected by an edge to each of

$$(x, y \pm 2x), (x, y \pm (2x+1)), (x \pm 2\ y, y), (x \pm (2y+1), y)$$

Observe that these neighboring vertices might have repetition, in which case there is more than one edge.

Theorem 4.6.10. $\lambda_2(G(n)) \leq \frac{5\cdot\sqrt{2}}{8}$.

Proof. Theorem 3.1.1 of [31] applies to give the result. See also 1.2.7 and 1.3.1 in the same paper. □

Let $G = G(n)$ be the 8-regular frum graph. For $u, v \in G$ let $E(u, v)$ be the number of edges with endpoints u and v. In order to perform a random walk on G the edges coming out of a vertex in G are labelled by elements of $L = \{0, 1\}^3$ using a *labelling function*

$$\text{nbhr} : L \times \text{vertices}(G) \to \text{vertices}(G)$$

such that

$$|\{s \in L : \text{nbhr}(s, u) = v\}| = E(u, v)$$

Define

$$\Gamma = (\Gamma_0, \cdots, \Gamma_{k-1}) : \{0, 1\}^{(2n)} \times \prod_{i=1}^{k-1} L_i \to \left(\{0, 1\}^{2n}\right)^k$$

where $L = L_i$ as follows. Then

$$\Gamma_0 : \{0, 1\}^{(2n)} \times \prod_{i=1}^{k-1} L_i \to \{0, 1\}^{(2n)}$$

is projection onto the first factor. Moreover $\Gamma_0, \Gamma_1, \cdots, \Gamma_{k-1}$ is a random walk on G starting at the output of Γ_0, and Γ_{i+1} takes one step in the direction given by L_{i+1}, thus $\Gamma_{i+1} = \text{nbhr}(L_{i+1}, \Gamma_i)$

Corollary 4.6.11. Γ *defined above is a strong* (γ, ϵ)*-averaging sampler where*

$$\gamma = 2\exp\left(-\epsilon^2\left(1 - \frac{5\sqrt{2}}{8}\right)k/4\right)$$

Proof. Consider the constant sequence of frum graphs $G_0, G_1, ..., G_{k-2}$ that are all equal to G. For a random input, Γ's output is chosen as described in Corollary 5.8 with this sequence of frum-graphs. Then, by Theorem 5.9, the conclusion follows by setting $\lambda = \frac{5\cdot\sqrt{2}}{8}$ in Corollary 5.8 . $\square$

Chapter 5

(Im)plausibility

The paper *On the (Im)possibility of Obfuscating Programs* [3] considers formal definitions of what program obfuscation might mean, and defines what it means for an algorithm to be a *virtual black box obfuscator*. It nicely, though informally, summarizes that definition as

Definition 5.0.1. $\mathcal{O}$ is an obfuscator if and only if it is an efficient, probabilistic *compiler* that takes as input a program (or circuit) P and produces a new program $\mathcal{O}(P)$ satisfying the following two conditions:

- *functionality*: $\mathcal{O}(P)$ computes the same function as P
- *virtual black box property*: Anything that can be efficiently computed from $\mathcal{O}(P)$ can be efficiently computed given oracle access to P.

The paper then proves the following pair of results.

Theorem 5.0.1. *(3.10 [3]) If one-way functions exist, then there exists an un-obfuscatable circuit ensemble.*

Lemma 5.0.2. *(3.8 [3]) If efficient circuit obfuscators exist, then one-way functions exist.*

This rules out such obfuscation even if it's supposed to be just for circuits. *Circuits* can be thought of as programs with fixed-length inputs whose runtime does not depend on their input. [3] also discusses the analogous problems for *sampling obfuscators*. For those, the corresponding informal summary of their definition is

Definition 5.0.3. $\mathcal{O}$ is an *efficient sampling obfuscator* if it is an (efficient, probabilistic) compiler that takes as input a program (or circuit) P and produces a new program $\mathcal{O}(P)$ satisfying the following two conditions:

- *functionality*: $\mathcal{O}(P)$ samples from the same distribution as P when both are run on uniformly random inputs.
- *virtual black box property* Anything that can be efficiently computed from $\mathcal{O}(P)$ can be efficiently computed given sampling access to P - i.e., the ability to obtain, upon request, independent and uniform random samples from the distribution defined by P.

In [3], Proposition 6.7 rules out a quite strong notion of sampling obfuscation for circuits assuming the existence of one-way functions, and Proposition 6.4 shows that if a fairly weak notion of sampling obfuscation is possible for circuits then promiseSZK $\not\subseteq$ promiseBPP .

In this thesis, I show that witness encryption - a special case of obfuscation for functions - implies an analogue *for sampling algorithms* of Theorem 3.10 in [3], in the auxiliary input setting.

This is a negative result so, as in [3], to make it stronger, it applies even to the obfuscation of just *circuits* rather than more general programs. In fact, this negative result also applies to *approximate* sampling obfuscators - i.e., when $\mathcal{O}(C)$ samples from a distribution that is *approximately* $\langle\langle C \rangle\rangle$. Below I present an informal summary of my result.

Theorem 5.0.2. *(Informal) If one-way functions exist and there is a witness encryption scheme for NP relations, then there is an efficiently-sampleable ensemble of sampling circuits and a predicate* Pred *on such circuits and an auxiliary-input generator such that*

(1) the auxiliary-input generator takes as input the sampling circuit and a positive integer L

(2) the auxiliary-input generator runs in time $\mathrm{poly}(\mathrm{security\_parameter}, L)$

(3) given such auxiliary input and sampling access *to the circuit, it is infeasible to guess* Pred *with probability non-negligibly better than* 1/2

(4) against adversaries that know the auxiliary-input, such sampling circuits cannot be obfuscated into circuits of size at most L, even if the obfuscator also knows the auxiliary-input and only needs to hide Pred *and only needs to* approximately *preserve the distribution and can be inefficient*

5.1 Notation and Definitions:

Definition 5.1.1. If X and Y are sets then $\mathbb{F}(X, Y)$ is the set of all functions $f : X \to Y$.

The security parameter is K, and $\overline{k}$ is the string of k ones. If C is a circuit with m input bits then $\langle\langle C \rangle\rangle$ is the distribution on $\{0,1\}^*$ obtained by evaluating C on m uniform and independent random bits, and $\langle\langle C \rangle\rangle$ being an oracle means the oracle takes no input and returns a sample from $\langle\langle C \rangle\rangle$ that is independent of everything before the query. In particular, when queried more than once, its responses are independent of each other.

Definition 5.1.2. A circuit C_1 *is an ϵ-sampler for another circuit* C_0 if the statistical distance between $\langle\langle C_1 \rangle\rangle$ and $\langle\langle C_0 \rangle\rangle$ is at most ϵ.

When we refer to an algorithm, by default, the algorithm may be randomized.

Definition 5.1.3. Given $\epsilon : \{0, 1, 2, 3, ...\} \to [0, 1]$, an *$\epsilon$-approximate sampling obfuscator* is an algorithm $\mathcal{O}$ such that the following holds for all feasible-size circuits C, and all k

(1) (arity and number of outputs) $\mathcal{O}$ takes exactly two inputs and gives exactly one output

(2) (polynomial slowdown) There is a 2-variable polynomial p such that with certainty, $\mathcal{O}(\overline{k}, C)$ is a circuit whose size is at most $p(\overline{k}, \text{size}(C))$

(3) (approximate functionality) Except with probability at most $\epsilon(k)$, $\mathcal{O}(\overline{k}, C)$ is an $\epsilon(k)$-sampler for C

(4) (security) For all feasible adversaries $\mathcal{A}$, there is an efficient simulator $\mathcal{S}$ such that

$$\left|\text{Prob}\left[\mathcal{A}(\overline{k}, \text{Obf}(\overline{k}, C)) = 1\right] - \text{Prob}\left[\mathcal{S}^{\langle\langle C\rangle\rangle}(\overline{k}, \text{size}(C)) = 1\right]\right| < \epsilon(k)$$

Definition 5.1.4. Approximate sampling obfuscators *against auxiliary input* are defined as above, but with condition (4) replaced by

(4′)(security against auxiliary input) For all feasible adversaries $\mathcal{A}$, there is an efficient simulator $\mathcal{S}$ such that it is infeasible to find a string z such that

$$\left|\text{Prob}\left[\mathcal{A}(\overline{k}, \text{Obf}(\overline{k}, C), z) = 1\right] - \text{Prob}\left[\mathcal{S}^{\langle\langle C\rangle\rangle}(\overline{k}, \text{size}(C), z) = 1\right]\right| < \epsilon(k)$$

In what follows, $h : \{0, 1, 2, 3, ...\} \to \{3, 4, 5, 6, ...\}$ can be any function such that $\lim_{k\to\infty} h(k) = \infty$ and $\overline{h(k)}$ is efficiently computable given $\overline{k}$.

Theorem 5.1.5. *(Formal) If pseudorandom generators exist, and witness encryption for NP exists, then* $1/(h(k))$*-approximate sampling obfuscators against auxiliary input do not exist.*

To prove this theorem, I show that under its assumptions, there exist ensembles that are unobfuscatable in a very strong sense.

Theorem 5.1.10. *With the assumptions of Theorem 5, there is is a predicate* pred *on circuits and a sequence* $\langle \mathcal{G}_0, \mathcal{G}_1, \mathcal{G}_2, ... \rangle$ *of distributions on circuits and an efficient algorithm* auxG *taking as input positive integers* $\overline{k}, \overline{\ell}$ *in unary and a circuit* C *such that the following hold:*

- *(efficient computability) Given* $\overline{k}$ *as input,* $\mathcal{G}_k$ *can be efficiently sampled from.*
- *(white-box learnability) For all circuits* C *from* $\bigcup_{i=0}^{\infty} \text{support}(\mathcal{G}_i)$*, for all polynomially–bounded* ℓ*, for all sufficiently large* k*,* $\text{pred}(C)$ *is easy to compute given a possible output of* $\text{auxG}(C, \overline{\ell})$ *and any circuit* C' *of bit-length at most* ℓ *that is a* $1/(h(k))$*-sampler for* C*.*
- *(black-box unlearnability) For all feasible adversaries* $\mathcal{A}$ *and 1-variable integer-coefficient polynomials* p *such that* $1 \leq p(k)$ *for all non-negative integers* k*,*

$\max_{1 \leq \ell \leq p(k)} \left| \mathrm{Prob}_{C \leftarrow \mathcal{G}_k, z \leftarrow \mathrm{auxG}(C, \overline{\ell})} \left[\mathcal{A}^{\langle\langle C \rangle\rangle} \left(\overline{k}, z \right) = \mathrm{pred}(C) \right] - \frac{1}{2} \right|$

is negligible.

Proof. I will show the white-box learnability property for $1/(18 \cdot h(k))$-samplers. The stated results will then follow by replacing the function h with the function $k \mapsto \max(3, \lfloor (h(k))/54 \rfloor)$ and, for k such that $h(k) < 162$, letting pred be trivially learnable. For example, one way of letting pred be trivial for a given k is having $\mathcal{G}_k$ choose $b \in \{0, 1\}$ uniformly at random and output the sampler that always outputs b, and pred of such sampling circuits be the bit b.

Having established that change in the statistical distance, I will now assume that $k \geq 24$ and $h(k) \leq \frac{k}{2 \cdot \lfloor \log_2(k) \rfloor}$. Smaller values of k can be handled in the same way as described above, and if h was too large, then it could be replaced with $\min \left(h(k), \frac{k}{2 \cdot \lfloor \log_2(k) \rfloor} \right)$, since making $h(k)$ smaller makes $1/(18 \cdot h(k))$ larger, which in turn can only make it harder to satisfy white-box learnability.

Note that [17] gives a way to convert pseudorandom generators into pseudorandom function families, and the other part of my thesis gives a way to convert witness encryption schemes for NP into witness encryption schemes for promiseMA.

Given h and a pseudorandom function family PRFF with input length equal to output length and a witness encryption scheme for promiseMA, I will exhibit a predicate and sequence and algorithm as described in the result of taking the theorem statement and replacing $h(k)$ with $18 \cdot h(k)$.

Let $\{0,1\}^{\leq k/2}$,$\{0,1\}^{\leq k}$ be the sets of binary strings whose lengths are less than or equal to $k/2$,k respectively, and let

$$\mathrm{LC} : \mathbb{F}(\{0,1\}^k, \{0,1\}^k) \to \mathbb{F}(\{0,1\}^{\leq k/2}, \{0,1\}^{\leq k})$$

be the *length conversion* function defined by $(\mathrm{LC}(F))(x)$ is the first $2 \cdot \mathrm{length}(x)$ bits of $F(x||1||\mathbf{0_m})$, where $||$ denotes string concatenation and $\mathbf{0_m}$ is the string of $m = k - (\mathrm{len}(x) + 1)$ zeros.

Given a circuit $F : \{0,1\}^k \to \{0,1\}^k$ and a bit b, we generate a sampling circuit $C_b(F)$ in $\mathcal{G}_k$ as follows. Later, F will be chosen according to the PRFF.

The circuit $C_b(F)$ has b and $\mathrm{LC}(F)$ encoded in an efficient and efficiently-recoverable way that does not affect the outputs. This is analogous to putting a comment in computer code, but needs to be done for circuits. For example, one could repeatedly apply $x \wedge x$ or $x \vee x$ to the first input, according to whether a 0 or 1 is to be represented. The circuit samples an integer L in $[1, h(k) - 1]$ such that each integer has probability greater than $\frac{1}{h(k)-2}$ of being chosen (i.e., close to uniformly). If $k/2 < L \cdot \lfloor \log_2(k) \rfloor$ then the circuit outputs the empty string. Otherwise, the circuit then chooses an element $r \in \{0,1\}^{L \cdot \lfloor \log_2(k) \rfloor}$ uniformly at random and outputs the ordered pair $(r, (\mathrm{LC}(F))(r))$.

The predicate pred is given by the following: If an ordered pair $(b, \mathrm{LC}(F))$ is encoded in the input circuit in the efficiently-recoverable way used for the previous paragraph, then output b, else output 0.

The distribution $\mathcal{G}_k$ is sampled from as follows. Choose F according to the PRFF with security parameter k, choose a bit b uniformly at random and output $C_b(F)$. The algorithm auxG consists of at most four steps.

- Step 0: Using the recovery procedure for how $(b, \mathrm{LC}(F))$ would be encoded into $C_b(F)$, attempt to recover a bit b and a circuit

$$\hat{F} : \{0,1\}^{\leq k/2} \to \{0,1\}^{\leq k}$$

  from the input circuit C. If that failed or $2^{(h(k)-1)\cdot\lfloor\log_2(k)\rfloor} \leq \ell$, then output the empty string and halt.

- Step 1: Let L be the least integer in $[1, h(k)-1]$ such that $\ell < 2^{L\cdot\lfloor\log_2(k)\rfloor}$, and set $\mathrm{rLen} = L \cdot \lfloor\log_2(k)\rfloor$.

- Step 2: For each string $r \in \{0,1\}^{\mathrm{rLen}}$, set $\mathrm{truevalue}_r = \hat{F}(r)$. Let V_0 be the circuit that works as follows.

  (1) receive as input a circuit $\mathcal{O}(C)$ of bit-length at most ℓ

  (2) Choose $5400 \cdot 8^{\mathrm{rLen}}$ inputs to $\mathcal{O}(C)$ independently and uniformly at random.

  (3) Evaluate $\mathcal{O}(C)$ at those inputs.

(4) For each string $r \in \{0,1\}^{\mathrm{rLen}}$, let $\mathrm{guess}_r \in \{0,1\}^{2\cdot\mathrm{rLen}}$ maximize the number of times that step (4) produced (r, guess_r), with ties broken lexicographically.

(5) If at least half of the strings $r \in \{0,1\}^{\mathrm{rLen}}$ are such that $\mathrm{guess}_r = \mathrm{truevalue}_r$, then accept, else reject.

- Step 3: Using the witness encryption scheme, output a witness encryption of b under the promiseMA relation whose YES set is

$$\{\langle V, y\rangle : V(y) \text{ accepts with probability at least } 2/3\}$$

and whose NO set is

$$\{\langle V, y\rangle : V(y) \text{ accepts with probability at most } 1/3\},$$

where the instance is V_0.

We now check Efficient Computability. In the definition of $C_b(F)$, rejection sampling with $2 \cdot \lceil \log_2(h(k)) \rceil + 2$ attempts has probability at most $\frac{1}{2\cdot(h(k))^2}$ of not producing an integer in $[1, h(k)-1]$, so outputting 1 in that case gives an efficient way of sampling L which is close-enough to uniform. Also, we gave an example of a suitable encoding of bits into circuits. Thus, pred can be made efficient and given $\overline{k}$ as input, $\mathcal{G}_k$ can be efficiently sampled from. For all values of L and rLen

that step 1 of auxG can choose $2^{\mathrm{rLen}} \leq \ell \cdot k$ because

$$\begin{aligned} \mathrm{rLen} &= L \cdot \lfloor \log_2(k) \rfloor \\ &= (L-1) \cdot \lfloor \log_2(k) \rfloor + \lfloor \log_2(k) \rfloor \end{aligned}$$

Recall that L is minimal subject to $\ell < 2^{L \cdot \lfloor \log_2(k) \rfloor}$ so $2^{(L-1) \cdot \lfloor \log_2(k) \rfloor} \leq \ell$. Combining with $2^{\lfloor \log_2(k) \rfloor} \leq k$ gives the result. In particular, $5400 \cdot 8^{\mathrm{rLen}} \leq 5400 \cdot (\ell \cdot k)^3 = 5400 \cdot \ell^3 \cdot k^3$, so given $\overline{\ell}$ as input, step 2 of auxG can build V_0 efficiently. Thus auxG is also efficient.

Next we check White-Box Learnability. For this, the learning algorithm is:

If the obfuscated circuit C' is too large to be a witness for the auxiliary information z, then output $\perp$, else output the result of decrypting that auxiliary information with C' as alleged-witness.

Fix a circuit C from $\bigcup_{i=0}^{\infty} \mathrm{support}(\mathcal{G}_i)$ and a positive integer ℓ and a possible output z of $\mathrm{auxG}(C, \overline{\ell})$ and a circuit C' of bit-length at most ℓ that is a $1/(h(k))$-approximate sampler for C.

The goal here is showing that V_0 accepts C' with probability strictly greater than $2/3$, since that means decrypting the auxiliary information with C' will yield the value of the predicate.

Let $\mathrm{LC}(F)$ be as in the construction of C, let L be as in step 1 of auxG, and let p_L be the probability of the circuit C choosing that value of L from

$[1, h(k) - 1]$. By the construction of the sampling circuits, $\frac{1}{h(k)-2} < p_L$, so $\frac{1}{3} \leq \frac{h(k)}{h(k)-2} = h(k) \cdot \frac{1}{h(k)-2} < p_L < h(k) \cdot p_L$

Let lowprobofgood be the set of all strings $r \in \{0,1\}^{\mathrm{rLen}}$ such that C' has probability at most $2 \cdot p_L/(3 \cdot 2^{\mathrm{rLen}})$ of outputting $(r, (\mathrm{LC}(F))(r))$, and let highprobofbad be the set of strings $r \in \{0,1\}^{\mathrm{rLen}}$ such that C' has probability at least $p_L/(3 \cdot 2^{\mathrm{rLen}})$ of outputting a pair (r, y) with $y \neq (\mathrm{LC}(F))(r)$. For each $r \in \{0,1\}^{\mathrm{rLen}}$, the circuit C has probability $p_L/(3 \cdot 2^{\mathrm{rLen}})$ of outputting $(r, (\mathrm{LC}(F))(r))$ and probability zero of outputting a pair (r, y) with $y \neq (\mathrm{LC}(F))(r)$. Thus, each element of lowprobofgood $\cup$ highprobofbad corresponds to a term with value at least $p_L/(3 \cdot 2^{\mathrm{rLen}})$ in the expression for the statistical distance between $\langle\langle C \rangle\rangle$ and $\langle\langle C' \rangle\rangle$, with B denoting the cardinality of that union, one has the following.

$$(1/2) \cdot B \cdot p_L/(3 \cdot 2^{\mathrm{rLen}}) \leq 1/(54 \cdot h(k))$$

Using this gives

$$(B \cdot 3)/(2^{\mathrm{rLen}}) \leq (B \cdot 9 \cdot h(k) \cdot p_L)/(2^{\mathrm{rLen}}) \leq 1$$

Hence $B \leq (2^{\mathrm{rLen}})/3$.

Thus, lowprobofgood $\cup$ highprobofbad has at most $(2^{\mathrm{rLen}})/3$ elements. Let good be the complement of lowprobofgood $\cup$ highprobofbad, set $q = p_L/(3 \cdot 2^{\mathrm{rLen}})$, let N be a positive integer which is at least $200/(q^3)$, and temporarily fix $r \in$ good. A random output of C' has probability greater than $2 \cdot q$ of being $(r, (\mathrm{LC}(F))(r))$,

and probability less than q of being an ordered pair (r, y) with $y \neq (\mathrm{LC}(F))(r)$. Call the former the *good* pairs and the latter the *bad* pairs. Consider the standard deviations of fraction of of good pairs and the fraction of bad pairs when taking N independent samples from C'. Those are each at most $\sqrt{2 \cdot q}/\sqrt{N}$.

$$\sqrt{2 \cdot q}/\sqrt{N} = \sqrt{(2 \cdot q)/N} \leq \sqrt{(2 \cdot q)/(200/(q^3))} = \sqrt{(q^2)/100} = q/10$$

so by Chebyshev's inequality and a union bound, the probability of the number of good pairs being at most the number of bad pairs is at most $2 \cdot 1/(5^2)$, which is strictly less than $1/12$.

Now, unfix r. At least a $2/3$ fraction of the strings $r \in \{0,1\}^{\mathrm{rLen}}$ are in Good, and the value $1/2$ is $3/12$ away from the value $2/3$, so when taking N independent samples from C', the probability that a majority of the strings $r \in \{0,1\}^{\mathrm{rLen}}$ have more good pairs than bad pairs is strictly greater than $2/3$. Thus, all that's left is checking that the number of samples V_0 takes from C' is at least $200/(q^3)$.

$$\begin{aligned}
\frac{200}{q^3} &= \frac{200}{\left(p_L/(3 \cdot 2^{\mathrm{rLen}})\right)^3} \\
&= \frac{5400 \cdot 8^{\mathrm{rLen}}}{p_L^3} \\
&\leq \frac{5400 \cdot 8^{\mathrm{rLen}}}{1^3} \\
&= 5400 \cdot 8^{\mathrm{rLen}}
\end{aligned}$$

Therefore white-box learnability holds.

Lastly we check Black-Box Unlearnability. Suppose that $\mathcal{A}$ is an adversary against black-box unlearnability and p is a 1-variable polynomial. As preparation towards defining such adversaries in the normal sense, we give algorithms $\mathcal{B}_{\text{prf}}$ and $\mathcal{B}_{\text{witenc}}$ that take a unary-encoded positive integer ℓ as an additional input and are otherwise adversaries against the PRFF and the witness encryption scheme respectively.

Both algorithms $\mathcal{B}_{\text{prf}}$ and $\mathcal{B}_{\text{witenc}}$, when simulating $\mathcal{A}\left(\overline{k}, z\right)$, respond to the oracle queries from $\mathcal{A}\left(\overline{k}, z\right)$ in a similar way to $C_b(\hat{F})$, but using $\text{LC}(\mathcal{O})$ rather than LC of a function given by a circuit. In the definition of black-box unlearnability, those queries would be responded to by $\langle\langle C \rangle\rangle$.

The algorithm $\mathcal{B}_{\text{prf}}$ works as follows. Let $\mathcal{O}$ be the oracle of $\mathcal{B}_{\text{prf}}$. Then $\mathcal{O}$ is either a member of the PRFF or a truly random function. Let $\hat{F} = \text{LC}(\mathcal{O})$, and choose a bit b uniformly at random. If

$$2^{(h(k)-1)\cdot\lfloor\log_2(k)\rfloor} \leq \ell$$

then $\mathcal{B}_{\text{prf}}$ outputs zero and halts. Otherwise, $\mathcal{B}_{\text{prf}}$ runs steps 1 and 2 and 3 of auxG, which produces an output of z. Next, $\mathcal{B}_{\text{prf}}$ simulates and gives the same output as $\mathcal{A}\left(\overline{k}, z\right)$.

The algorithm $\mathcal{B}_{\text{witenc}}$ works as follows. Simulate a random function $\mathcal{O} : \{0,1\}^k \to \{0,1\}^k$. Let $\hat{F} = \text{LC}(\mathcal{O})$, and choose a bit b uniformly at random. If there is an integer L as in step 1 of auxG, then run step 1, otherwise set

rLen = 0. After that, $\mathcal{B}_{\text{witenc}}$ runs step 2 of auxG, chooses a bit b' independently and uniformly at random, and submits to its challenger

- as instance the circuit V_0
- as left plaintext the bit b
- as right plaintext the bit b'.

The challenger produces a response of z then $\mathcal{B}_{\text{witenc}}$ simulates $\mathcal{A}\left(\overline{k}, z\right)$. At the end of that simulation, if the output of $\mathcal{A}\left(\overline{k}, z\right)$ is b then $\mathcal{B}_{\text{witenc}}$ outputs 1 else $\mathcal{B}_{\text{witenc}}$ outputs 0.

Next we define some experiments, each of which returns a value of true or false.

$\text{Expt}_{\text{learn},\ell}$: Choose $C \leftarrow \mathcal{G}_k$ and $z \leftarrow \text{auxG}(C, \overline{\ell})$ and then run $\mathcal{A}^{\langle\langle C\rangle\rangle}\left(\overline{k}, z\right)$. The outcome is True if and only if the output of $\mathcal{A}$ is pred(C).

$\text{Expt}^{\text{pseudo}}_{\text{prf},\ell}$: Choose $\mathcal{O} \leftarrow$ PRFF, and run $\mathcal{B}^{\mathcal{O}}_{\text{prf}}(\overline{k}, \overline{\ell})$. The outcome is True if and only if the output of $\mathcal{B}_{\text{prf}}$ is 1.

$\text{Expt}^{\text{truerand}}_{\text{prf},\ell}$: Choose $\mathcal{O} \leftarrow \mathbb{F}(\{0,1\}^k, \{0,1\}^k)$, and run $\mathcal{B}^{\mathcal{O}}_{\text{prf}}(\overline{k}, \overline{\ell})$. The outcome is True if and only if the output of $\mathcal{B}_{\text{prf}}$ is 1.

For the next two experiments the parameter *side* can be *left* or *right*.

$\text{Expt}^{\text{side}}_{\text{witenc},\ell}$: Run $\mathcal{B}_{\text{witenc}}(\overline{k}, \overline{\ell})$ with the challenger encrypting the side plaintext. The outcome is True if and only if the output of $\mathcal{B}_{\text{witenc}}$ is 1.

$\text{Expt}^{\text{side}}_{\text{unbounded},\ell}$: Run $\mathcal{B}_{\text{witenc}}(\overline{k}, \overline{\ell})$ with the challenger encrypting the side plaintext. The outcome is True if and only if the instance that $\mathcal{B}_{\text{witenc}}$ submitted to the challenger was a NO instance and the output of $\mathcal{B}_{\text{witenc}}$ is 1.

The definition of black-box unlearnability requires $\ell \leq p(k)$. Since h is $\omega(1)$, we have $p(k) < 2^{(h(k)-1)\cdot\lfloor\log_2(k)\rfloor}$ for all sufficiently large k. Thus, in the rest of this proof, we assume $\ell < 2^{(h(k)-1)\cdot\lfloor\log_2(k)\rfloor}$.

For such ℓ, if F is the oracle of $\mathcal{B}_{\text{prf}}$ then for each value of b, the algorithm $\mathcal{B}_{\text{prf}}$ chooses z from the same distribution as $\text{auxG}(C_b(F))$, the distribution of the responses by $\mathcal{B}_{\text{prf}}$ to the oracle queries of $\mathcal{A}$ is $\langle\langle C_b(F)\rangle\rangle$, and $\text{pred}(C_b(F)) = b$.

Thus, $\text{Prob}\left[\text{Expt}_{\text{learn}}\right] = \text{Prob}\left[\text{Expt}^{\text{pseudo}}_{\text{prf}}\right]$. Furthermore, $\text{Prob}\left[\text{Expt}^{\text{truerand}}_{\text{prf}}\right] = \text{Prob}\left[\text{Expt}^{\text{left}}_{\text{witenc}}\right]$. On the other hand

$$\text{Prob}\left[\text{Expt}^{\text{right}}_{\text{witenc}}\right] = 1/2$$

since for this experiment, none of $\mathcal{O}, \overline{k}, \overline{\ell}$ depends on b. Thus, by the triangle inequality,

$$\begin{aligned}
&\left|\text{Prob}\left[\text{Expt}_{\text{learn}}\right] - \tfrac{1}{2}\right| \\
&\leq \left|\text{Prob}\left[\text{Expt}^{\text{pseudo}}_{\text{prf}}\right] - \text{Prob}\left[\text{Expt}^{\text{truerand}}_{\text{prf}}\right]\right| \\
&+ \left|\text{Prob}\left[\text{Expt}^{\text{left}}_{\text{witenc}}\right] - \text{Prob}\left[\text{Expt}^{\text{right}}_{\text{witenc}}\right]\right|
\end{aligned} \tag{5.1}$$

The term

$$\left|\text{Prob}\left[\text{Expt}^{\text{pseudo}}_{\text{prf}}\right] - \text{Prob}\left[\text{Expt}^{\text{truerand}}_{\text{prf}}\right]\right|$$

is the analogue, for algorithms that take $\overline{\ell}$ as an additional input, of the advantage of $\mathcal{B}_{\text{prf}}$ against the PRFF. However,

$$\left|\text{Prob}\left[\text{Expt}^{\text{left}}_{\text{witenc}}\right] - \text{Prob}\left[\text{Expt}^{\text{right}}_{\text{witenc}}\right]\right|$$

is *not* quite the analogue, for algorithms that take $\overline{\ell}$ as an additional input, of the advantage of $\mathcal{B}_{\text{witenc}}$ against the witness encryption scheme.

The issue specific to witenc is that we have not yet handled the possibility of the instance not being in the NO set of the promiseMA relation. The other issue is that, in actual attacks on the PRFF and the witness encryption scheme, there will *not* be an input $\overline{\ell}$. We will handle the witenc-specific issue by by giving an upper bound on the probability of the instance not being in the NO set of the promiseMA relation, and will handle the other issue after that.

To start, note that the rLen used by $\mathcal{B}_{\text{witenc}}$ depends *only* on k and $h(k)$ and ℓ. If the adversary used randomness to choose ℓ, then rLen might depend on that randomness because rLen can depend on ℓ, but rLen otherwise does not depend on any randomness. Towards that end, we define a function as follows.

If $r_{\text{ver}} \in \{0,1\}^{\ell \cdot 5400 \cdot 8^{\text{rLen}}}$ and C is a circuit of bit-length at most ℓ then

$$\text{guessed\_func}\_(r_{\text{ver}}, C) : \{0,1\}^{\text{rLen}} \to \{0,1\}^{2 \cdot \text{rLen}}$$

is the function given by

$$\text{guessed\_func}\_(r_{\text{ver}}, C)(r) = \text{guess}_r$$

where guess_r would be produced by running (1),(2),(3),(4), and (5) from step 2 of auxG, with $\mathcal{O}(C) = C$ and randomness equal to r_{ver}.

If the promiseMA instance that $\mathcal{B}_{\text{witenc}}$ submits to the challenger is *not* a NO instance, then there exists

(*) a circuit C of bit-length at most ℓ such that more than 1/3 of the strings $r_{\text{ver}} \in \{0,1\}^{\ell \cdot 5400 \cdot 8^{\text{rLen}}}$ are such that guessed_func_(r_{ver}, C) agrees with LC applied to the function being simulated by $\mathcal{B}_{\text{witenc}}$ on at least half of the inputs from $\{0,1\}^{\text{rLen}}$.

There are $2^{\left(2 \cdot \text{rLen} \cdot 2^{\text{rLen}}\right)}$ functions from $\{0,1\}^{\text{rLen}}$ to $\{0,1\}^{2 \cdot \text{rLen}}$, and each of those functions is equally likely to be the restriction to $\{0,1\}^{\text{rLen}}$ of LC of the function being simulated by $\mathcal{B}_{\text{witenc}}$.

On the other hand:

• There are less than $2^{\ell+1}$ circuits of bit-length at most ℓ

• For each of the 2^{rLen} elements of $\{0,1\}^{\text{rLen}}$, there are 2 ways to choose an element of $\{=, \neq\}$.

• For each integer in $\left[0, \left(2^{\text{rLen}}\right)/2\right]$, there are at most $2^{2 \cdot \text{rLen} \cdot \left(2^{\text{rLen}}\right)/2}$ sequences of elements of $\{0,1\}^{\text{rLen}}$ such that the length of the sequence is the integer.

There are $\beta = 2^{\left(2 \cdot \text{rLen} \cdot 2^{\text{rLen}}\right)}$ equally likely possibilities for LC applied to the function being simulated by $\mathcal{B}_{\text{witenc}}$. For any fixed $r_{\text{ver}} \in \{0,1\}^{\ell \cdot 5400 \cdot 8^{\text{rLen}}}$, the

number of these β possibilities such that there is a circuit as in (*) is less than

$$\alpha = 2^{(\ell+1)+\left(2^{\mathrm{rLen}}\right)+\left(2\cdot \mathrm{rLen}\cdot\left(2^{\mathrm{rLen}}\right)/2\right)}$$

Using that $\ell + 1 \leq 2^{\mathrm{rLen}}$ easily gives

$$\ell + 1 + 2^{\mathrm{rLen}} + \left(2 \cdot \mathrm{rLen} \cdot \left(2^{\mathrm{rLen}}\right)/2\right) \leq (2 + \mathrm{rLen}) \cdot 2^{\mathrm{rLen}}$$

Since $\mathrm{rLen} = L \cdot \lfloor \log_2(k) \rfloor$ and L is a positive integer and $k \geq 16$, one has $\mathrm{rLen} \geq 4$, so

$$(2 + \mathrm{rLen}) \cdot 2^{\mathrm{rLen}} \leq (3/2) \cdot \mathrm{rLen} \cdot 2^{\mathrm{rLen}}$$

Observe that $\mathrm{rLen} \geq 4$ and $\mathrm{rLen} = L \cdot \lfloor \log_2(k) \rfloor \geq \lfloor \log_2(k) \rfloor$ so $2^{\mathrm{rLen}} \geq k/2$ so

$$\log_2 \beta - \log_2 \alpha \geq k$$

It follows that for any fixed $r_{\mathrm{ver}} \in \{0,1\}^{\ell \cdot 5400 \cdot 8^{\mathrm{rLen}}}$, the probability, over the choice by $\mathcal{B}_{\mathrm{witenc}}$ of the function it simulates, that there exists a circuit as in (*), is at most $1/(2^k)$. By averaging, this $1/(2^k)$ bound continues to hold when $r_{\mathrm{ver}} \in \{0,1\}^{\ell \cdot 5400 \cdot 8^{\mathrm{rLen}}}$ is chosen at random rather than fixed beforehand. Thus, by averaging again, it follows that the probability is at most

$$3/(2^k) \tag{5.2}$$

that, over the choice by $\mathcal{B}_{\mathrm{witenc}}$ of the function it simulates, that for at least $1/3$ of the strings $r_{\mathrm{ver}} \in \{0,1\}^{\ell \cdot 5400 \cdot 8^{\mathrm{rLen}}}$ there is a circuit as in (*).

Therefore, the probability of the promiseMA instance that $\mathcal{B}_{\text{witenc}}$ submits to the challenger *not* being a NO instance, is at most $3/(2^k)$. That will be the upper bound on p_{invalid} for the witenc case, which we define as part of showing how to remove $\overline{\ell}$ from the inputs.

By (5.1), if $\left|\text{Prob}\left[\text{Expt}_{\text{learn}}\right] - \frac{1}{2}\right|$ is non-negligible, then either

$$\left|\text{Prob}\left[\text{Expt}_{\text{prf}}^{\text{pseudo}}\right] - \text{Prob}\left[\text{Expt}_{\text{prf}}^{\text{truerand}}\right]\right|$$

is non-negligible or

$$\left|\text{Prob}\left[\text{Expt}_{\text{witenc}}^{\text{left}}\right] - \text{Prob}\left[\text{Expt}_{\text{witenc}}^{\text{right}}\right]\right|$$

is non-negligible, or both. We handle those two cases together, by considering interaction between a two-input adversary $\mathcal{B}$ that outputs True or False and one of an ordered pair of feasible environments $\text{Env}_{\text{left}}, \text{Env}_{\text{right}}$, where there is also a predicate Valid on such interactions, but Valid does not even need to be computable.

We define four algorithms $\mathcal{B}\left(\overline{k}, \overline{\ell}\right)$ and Env_{left} and $\text{Env}_{\text{right}}$ and Valid as follows.

Case prf To show that the absolute difference of the prf probabilities is negligible:

Env_{left}: Choose $\mathcal{O} \leftarrow$ PRFF and act as $\mathcal{O}$.

$\text{Env}_{\text{right}}$: Choose $\mathcal{O} \leftarrow \mathbb{F}(\{0,1\}^k, \{0,1\}^k)$ and act as $\mathcal{O}$.

$\mathcal{B}$ sends queries to the Env, rather submitting queries to an oracle, and otherwise acts as $\mathcal{B}_{\text{prf}}$.

Valid always outputs True.

Case witenc To show that the absolute difference of the witenc probabilities is negligible:

$\text{Env}_{\text{left}}, \text{Env}_{\text{right}}$ *are* the challengers for $\mathcal{B}_{\text{witenc}}$ that encrypt the left,right plaintext respectively.

$\mathcal{B} = \mathcal{B}_{\text{witenc}}$

Valid is true if and only if the instance submitted to the Env was a NO instance.

For both cases, set

$$p_{\text{invalid}} = \max_{1 \leq \ell \leq p(k)} \sum_{\text{side} \in \{\text{left,right}\}} 1 - \text{Prob}[\text{Valid}]$$

and for each choice of side from $\{\text{left}, \text{right}\}$, let $\mathcal{B}\left(\overline{k}, \overline{\ell}\right) \leftrightarrow \text{Env}_{\text{side}}$ denote the experiment $\mathcal{B}\left(\overline{k}, \overline{\ell}\right)$ interacts with Env_{side} if $\mathcal{B}\left(\overline{k}, \overline{\ell}\right)$ outputs 1 then the outcome is True else the outcome is False.

Let $\mathcal{B}\left(\overline{k}, \overline{\ell}\right) \overset{v}{\leftrightarrow} \text{Env}_{\text{side}}$ denote $\mathcal{B}\left(\overline{k}, \overline{\ell}\right) \leftrightarrow \text{Env}_{\text{side}} \wedge \text{Valid}$. We will define $\hat{B}\left(\overline{k}\right)$ later. Once we do, let $\mathcal{B}\left(\overline{k}\right) \leftrightarrow \text{Env}_{\text{left}}$ and $\mathcal{B}\left(\overline{k}\right) \overset{v}{\leftrightarrow} \text{Env}_{\text{right}}$ denote the analogous experiments for $\hat{B}\left(\overline{k}\right)$ rather than $\mathcal{B}\left(\overline{k}, \overline{\ell}\right)$.

For case prf, since Valid is always true. Let

$$\begin{aligned}\beta &= \max_{1\le\ell\le p(k)} \left|\text{Prob}\left[\text{Expt}^{\text{pseudo}}_{\text{prf}}\right] - \text{Prob}\left[\text{Expt}^{\text{truerand}}_{\text{prf}}\right]\right| \\ &= \max_{1\le\ell\le p(k)} \left|\text{Prob}\left[\mathcal{B}\left(\overline{k},\overline{\ell}\right) \leftrightarrow \text{Env}_{\text{left}}\right] - \text{Prob}\left[\mathcal{B}\left(\overline{k},\overline{\ell}\right) \leftrightarrow \text{Env}_{\text{right}}\right]\right| \\ &= \max_{1\le\ell\le p(k)} \left|\text{Prob}\left[\mathcal{B}\left(\overline{k},\overline{\ell}\right) \overset{v}{\leftrightarrow} \text{Env}_{\text{left}}\right] - \text{Prob}\left[\mathcal{B}\left(\overline{k},\overline{\ell}\right) \overset{v}{\leftrightarrow} \text{Env}_{\text{right}}\right]\right|\end{aligned}$$

For case witenc. Let

$$\begin{aligned}\gamma &= \max_{1\le\ell\le p(k)} \left|\text{Prob}\left[\text{Expt}^{\text{left}}_{\text{witenc}}\right] - \text{Prob}\left[\text{Expt}^{\text{right}}_{\text{witenc}}\right]\right| \\ &= \max_{1\le\ell\le p(k)} \left|\text{Prob}\left[\mathcal{B}\left(\overline{k},\overline{\ell}\right) \leftrightarrow \text{Env}_{\text{left}}\right] - \text{Prob}\left[\mathcal{B}\left(\overline{k},\overline{\ell}\right) \leftrightarrow \text{Env}_{\text{right}}\right]\right|\end{aligned}$$

By the upper bound (5.2), for case witenc $p_{\text{invalid}} \le \frac{6}{2^k}$. Thus $p_{\text{invalid}} \le \frac{6}{2^k}$ holds in each case. In each case, let j be an arbitrary positive integer, and suppose k is such that

$$1/(k^j) < \max_{1\le\ell\le p(k)} \left|\text{Prob}\left[\mathcal{B}\left(\overline{k},\overline{\ell}\right) \overset{v}{\leftrightarrow} \text{Env}_{\text{left}}\right] - \text{Prob}\left[\mathcal{B}\left(\overline{k},\overline{\ell}\right) \overset{v}{\leftrightarrow} \text{Env}_{\text{right}}\right]\right| \tag{5.3}$$

i.e., that there *exists* an integer $\ell \in [1, p(k)]$ such that $\mathcal{B}\left(\overline{k},\overline{\ell}\right)$ validly distinguishes Env_{left} from $\text{Env}_{\text{right}}$. Let $\hat{\mathcal{B}}$ be the adversary that works as follows, with $N = 250 \cdot p(k) \cdot k^{3 \cdot j}$. For each integer $\ell \in [1, p(k)]$, simulate Env_{left} for N runs of $\mathcal{B}\left(\overline{k},\overline{\ell}\right) \leftrightarrow \text{Env}_{\text{left}}$ and let $q^{\text{left}}_{\ell} = y/N$, where y is the number of times $\mathcal{B}\left(\overline{k},\overline{\ell}\right) \leftrightarrow \text{Env}_{\text{left}}$ is true. Similarly define q^{right}_{ℓ} for $\mathcal{B}\left(\overline{k},\overline{\ell}\right) \leftrightarrow \text{Env}_{\text{right}}$.

Let $\ell = \hat{\ell}$ maximize $\left|q_\ell^{\text{left}} - q_\ell^{\text{right}}\right|$. If $q_\ell^{\text{left}} = q_\ell^{\text{right}}$ then output a random bit and halt. Otherwise, forward messages between $\mathcal{B}\left(\overline{k}, \overline{\hat{\ell}}\right)$ and the actual Env, and let b be the output by $\mathcal{B}\left(\overline{k}, \overline{\hat{\ell}}\right)$.

If $b = 0 \iff q_{\hat{\ell}}^{\text{left}} < q_{\hat{\ell}}^{\text{right}}$ then output 1 else output 0.

Let $\text{Expt}_{\text{wrong\_guess}}$ be the following experiment: Choose side $\in \{\text{left}, \text{right}\}$ uniformly at random, run $\hat{\mathcal{B}}\left(\overline{k}\right) \leftrightarrow \text{Env}_{\text{side}}$, and let b be the outcome of that experiment. If $b = 0 \iff \text{side} = \text{left}$ then the outcome is True, else the outcome is False.

In what follows ℓ is an integer in the interval $[1, p(k)]$ and

$p_\ell^{\text{left}} = \text{Prob}\left[\mathcal{B}\left(\overline{k}, \overline{\ell}\right) \leftrightarrow \text{Env}_{\text{left}}\right]$ and $p_\ell^{\text{right}} = \text{Prob}\left[\mathcal{B}\left(\overline{k}, \overline{\ell}\right) \leftrightarrow \text{Env}_{\text{right}}\right]$.

Consider the following three events for runs of $\text{Expt}_{\text{wrong\_guess}}$

- $\text{BadEstimate}_{\text{left}}$: for some ℓ

$$\frac{1}{10 \cdot k^j} < \left|q_\ell^{\text{left}} - p_\ell^{\text{pseudo}}\right|$$

- $\text{BadEstimate}_{\text{right}}$: for some ℓ

$$\frac{1}{10 \cdot k^j} < \left|q_\ell^{\text{right}} - p_\ell^{\text{truerand}}\right|$$

- GoodL:

$$\frac{6}{10 \cdot k^j} < \left|p_{\hat{\ell}}^{\text{left}} - p_{\hat{\ell}}^{\text{right}}\right|$$

and $p_{\hat{\ell}}^{\text{left}} - p_{\hat{\ell}}^{\text{right}}$ has the same sign as $q_{\hat{\ell}}^{\text{pseudo}} - q_{\hat{\ell}}^{\text{truerand}}$.

We show that at least one of these three events must occur. Suppose neither $\text{BadEstimate}_{\text{left}}$ nor $\text{BadEstimate}_{\text{right}}$ occur. Then, since β is the maximum of $\left|p_\ell^{\text{left}} - p_\ell^{\text{right}}\right|$, by (5.3) it follows that for values ℓ that maximize $\left|p_\ell^{\text{left}} - p_\ell^{\text{right}}\right|$ one has

$$\frac{10}{10 \cdot k^j} < \left|p_\ell^{\text{left}} - p_\ell^{\text{right}}\right|$$

hence

$$\frac{8}{10 \cdot k^j} < \left|q_\ell^{\text{left}} - q_\ell^{\text{right}}\right| \leq \left|q_{\hat{\ell}}^{\text{left}} - q_{\hat{\ell}}^{\text{right}}\right|$$

so

$$\frac{6}{10 \cdot k^j} < \left|p_{\hat{\ell}}^{\text{left}} - p_{\hat{\ell}}^{\text{right}}\right|$$

Moreover $x = q_{\hat{\ell}}^{\text{left}} - q_{\hat{\ell}}^{\text{right}}$ has the same sign as $y = p_{\hat{\ell}}^{\text{left}} - p_{\hat{\ell}}^{\text{right}}$, since because $\text{BadEstimate}_{\text{pseudo}}$ and $\text{BadEstimate}_{\text{truerand}}$ are both false, one has

$$|x - y| \leq \frac{2}{10 \cdot k^j}$$

Fix an integer $\ell \in [1, p(k)]$, and consider the random variable q_ℓ^{left}. That random variable is the average of $N = 250 \cdot p(k) \cdot k^{3 \cdot j}$ Bernoulli random variables. Since Bernoulli random variables have standard deviation at most $1/2$, the standard deviation of q_ℓ^{left} is at most $\frac{1/2}{\sqrt{N}}$. Thus $\text{Prob}\left[\left|q_\ell^{\text{left}} - p_\ell^{\text{left}}\right| > \frac{1}{10 \cdot k^j}\right] \leq \left(\frac{1/(10 \cdot k^j)}{(1/2)/(\sqrt{N})}\right)^{-2} = \frac{1}{10 \cdot p(k) \cdot k^j}$.

By a union bound, summing over ℓ, this means

$$\text{Prob}\left[\text{BadEstimate}_{\text{left}}\right] \leq p(k) \cdot \frac{1}{10 \cdot p(k) \cdot k^j} = \frac{1}{10 \cdot k^j}$$

Similarly,

$$\text{Prob}\left[\text{BadEstimate}_{\text{left}}\right] \leq \frac{1}{10 \cdot k^j}$$

The sum of those two bounds is less than 1, so $\text{Prob}\left[\text{GoodL}\right] > 0$. Since at least one of those three events must occur,

$$\begin{aligned}
&\text{Prob}\left(\text{Expt}_{\text{wrong_guess}}\right) \\
&\leq \text{Prob}\left[\text{BadEstimate}_{\text{left}}\right] \\
&+ \text{Prob}\left[\text{BadEstimate}_{\text{right}}\right] \\
&+ \text{Prob}\left[\text{GoodL}\right] \cdot \text{Prob}\left[\text{Expt}_{\text{wrong_guess}} \,\middle|\, \text{GoodL}\right] \\
&\leq \frac{2}{10 \cdot k^j} + \text{Prob}\left[\text{Expt}_{\text{wrong_guess}} \,\middle|\, \text{GoodL}\right]
\end{aligned} \tag{5.4}$$

Similarly,

$$\text{Prob}\left[\text{BadEstimate}_{\text{truerand}}\right] \leq \frac{1}{10 \cdot k^j}$$

Since k and j are positive integers, $2/(10 \cdot k^j) < 1$, so

$$\text{Prob}\left[\text{GoodL}\right] > 0$$

Since one of the three events $\text{BadEstimate}_{\text{pseudo}}$ or $\text{BadEstimate}_{\text{truerand}}$ or GoodL must occur,

$$
\begin{aligned}
&\text{Prob}\left(\text{Expt}_{\text{wrong\_guess}}\right) \\
&\leq \text{Prob}\left[\text{BadEstimate}_{\text{pseudo}}\right] \\
&+ \text{Prob}\left[\text{BadEstimate}_{\text{truerand}}\right] \\
&+ \text{Prob}\left[\text{GoodL}\right] \cdot \text{Prob}\left[\text{Expt}_{\text{wrong\_guess}} \,\middle|\, \text{GoodL}\right] \\
&\leq \frac{2}{10 \cdot k^j} + \text{Prob}\left[\text{Expt}_{\text{wrong\_guess}} \,\middle|\, \text{GoodL}\right]
\end{aligned} \tag{5.5}
$$

For each value ℓ_0 that $\hat{\ell}$ could take in a run of $\text{Expt}_{\text{wrong\_guess}}$ which satisfies GoodL,

$$
\begin{aligned}
&\text{Prob}\left[\text{Expt}_{\text{wrong\_guess}} \,\middle|\, \text{GoodL} \wedge \hat{\ell} = \ell_0\right] \\
&= \frac{1}{2} \cdot \left(\text{Prob}\left[\text{Expt}_{\text{wrong\_guess}} \,\middle|\, \begin{array}{c}\text{GoodL} \wedge \\ \hat{\ell} = \ell_0 \wedge x = \text{left}\end{array}\right]\right. \\
&\qquad \left. + \text{Prob}\left[\text{Expt}_{\text{wrong\_guess}} \,\middle|\, \begin{array}{c}\text{GoodL} \wedge \\ \hat{\ell} = \ell_0 \wedge x = \text{right}\end{array}\right] \right) \\
&= \frac{1}{2} \cdot \left(\min\left(p_{\ell_0}^{\text{left}}, p_{\ell_0}^{\text{right}}\right) + 1 - \max\left(p_{\ell_0}^{\text{left}}, p_{\ell_0}^{\text{right}}\right)\right) \\
&- \frac{1}{2} \cdot \left(1 + \min\left(p_{\ell_0}^{\text{left}}, p_{\ell_0}^{\text{right}}\right) - \max\left(p_{\ell_0}^{\text{left}}, p_{\ell_0}^{\text{right}}\right)\right) \\
&= \frac{1}{2} \cdot \left(1 - \left| p_{\ell_0}^{\text{left}} - p_{\ell_0}^{\text{right}} \right| \right) \\
&\leq \frac{1}{2} \cdot \left(1 - \frac{6}{10 \cdot k^j} \right) = \frac{1}{2} - \frac{3}{10 \cdot k^j}
\end{aligned}
$$

That holds for all such ℓ_0, so

$$
\text{Prob}\left[\text{Expt}_{\text{wrong\_guess}} \,\middle|\, \text{GoodL}\right] \leq \frac{1}{2} - \frac{3}{10 \cdot k^j}.
$$

Substituting this bound in inequality (5.5) gives

$$\text{Prob}\left[\text{Expt}_{\text{wrong\_guess}}\right] \leq \frac{1}{2} - \frac{1}{10 \cdot k^j} \tag{5.6}$$

and by the definition of $\text{Expt}_{\text{wrong\_guess}}$,

$$\begin{aligned}
&2 \cdot \text{Prob}\left[\text{Expt}_{\text{wrong_guess}}\right] \qquad (5.7)\\
&= \text{Prob}\left[\neg\left(\hat{\mathcal{B}}\left(\overline{k}\right) \leftrightarrow \text{Env}_{\text{left}}\right)\right] + \text{Prob}\left[\hat{\mathcal{B}}\left(\overline{k}\right) \leftrightarrow \text{Env}_{\text{right}}\right]\\
&= \left(1 - \text{Prob}\left[\hat{\mathcal{B}}\left(\overline{k}\right) \leftrightarrow \text{Env}_{\text{left}}\right]\right) + \text{Prob}\left[\hat{\mathcal{B}}\left(\overline{k}\right) \leftrightarrow \text{Env}_{\text{right}}\right]\\
&= 1 - \left(\text{Prob}\left[\hat{\mathcal{B}}\left(\overline{k}\right) \leftrightarrow \text{Env}_{\text{left}}\right] - \text{Prob}\left[\hat{\mathcal{B}}\left(\overline{k}\right) \leftrightarrow \text{Env}_{\text{right}}\right]\right)\\
&\leq 1 - \left|\text{Prob}\left[\hat{\mathcal{B}}\left(\overline{k}\right) \leftrightarrow \text{Env}_{\text{left}}\right] - \text{Prob}\left[\hat{\mathcal{B}}\left(\overline{k}\right) \leftrightarrow \text{Env}_{\text{right}}\right]\right|
\end{aligned}$$

so

$$\begin{aligned}
\frac{1}{5 \cdot k^j} &= 2 \cdot \frac{1}{10 \cdot k^j}\\
&\leq \left|\text{Prob}\left[\hat{\mathcal{B}}\left(\overline{k}\right) \leftrightarrow \text{Env}_{\text{left}}\right] - \text{Prob}\left[\hat{\mathcal{B}}\left(\overline{k}\right) \leftrightarrow \text{Env}_{\text{right}}\right]\right|
\end{aligned}$$

Since $p_{\text{invalid}} \leq 6/(2^k)$, that means

$$\frac{1}{5 \cdot k^j} - \frac{6}{2^k} \leq \left|\text{Prob}\left[\hat{\mathcal{B}}\left(\overline{k}\right) \overset{v}{\leftrightarrow} \text{Env}_{\text{left}}\right] - \text{Prob}\left[\hat{\mathcal{B}}\left(\overline{k}\right) \overset{v}{\leftrightarrow} \text{Env}_{\text{right}}\right]\right|$$

Now $\dfrac{6}{2^k}$ is negligible and $\dfrac{1}{5 \cdot k^j}$ is noticeable, so $\dfrac{1}{5 \cdot k^j} - \dfrac{6}{2^k}$ is noticeable.

Furthermore, $\hat{\mathcal{B}}$ is feasible if $\mathcal{A}$ is: Env_{left} and $\text{Env}_{\text{right}}$ are efficient, and $\hat{\mathcal{B}}$ simulates polynomially many interactions of them with $\mathcal{B}\left(\overline{k}, \overline{\ell}\right)$, all of which have $\ell \leq p(k)$, and then does a polynomial amount of extra work. Also, $\mathcal{B}\left(\overline{k}, \overline{\ell}\right)$ is either

$\mathcal{B}_{\text{prf}}\left(\overline{k}, \overline{\ell}\right)$ or $\mathcal{B}_{\text{witenc}}\left(\overline{k}, \overline{\ell}\right)$. In turn, by inspection, $\mathcal{B}_{\text{prf}}$ and $\mathcal{B}_{\text{witenc}}$ are feasible if $\mathcal{A}$ is.

Thus, by the security of the PRFF and the witness encryption scheme, there are only finitely many k such that

$$\frac{1}{5 \cdot k^j} - \frac{6}{2^k} \leq \left| \text{Prob}\left[\hat{\mathcal{B}}\left(\overline{k}\right) \overset{v}{\leftrightarrow} \text{Env}_{\text{left}}\right] - \text{Prob}\left[\hat{\mathcal{B}}\left(\overline{k}\right) \overset{v}{\leftrightarrow} \text{Env}_{\text{right}}\right] \right|$$

Since this inequality holds whenever k satisfies equation (5.3) there are only finitely many such k. Furthermore, this holds for all positive integers j, so β and γ are both negligible. Therefore

$$\begin{aligned} &\max_{1 \leq \ell \leq p(k)} \left| \text{Prob}_{C \leftarrow \mathcal{G}_k, z \leftarrow \text{auxG}(C, \overline{\ell})} \left[\mathcal{A}^{\langle\langle C \rangle\rangle}(\overline{k}, z)\, \text{pred}(C)\right] - \frac{1}{2} \right| \\ &= \max_{1 \leq \ell \leq p(k)} \left| \text{Prob}\left[\text{Expt}_{\text{learn}}\right] - \frac{1}{2} \right| = \beta + \gamma \end{aligned}$$

is negligible.

This holds for all feasible adversaries $\mathcal{A}$, so black-box unlearnability holds. This completes the proof. $\square$

Bibliography

[1] S. Agrawal and A. Pellet-Mary. Indistinguishability obfuscation without maps: Attacks and fixes for noisy linear fe. Cryptology ePrint Archive, Report 2020/415, 2020. `https://eprint.iacr.org/2020/415`.

[2] B. Barak and O. Goldreich. Universal arguments and their applications. *SIAM J. Comput.*, 38(5):1661–1694, 2008/09.

[3] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. Vadhan, and K. Yang. On the (im)possibility of obfuscating programs. *J. ACM*, 59(2):Art. 6, 48, 2012.

[4] B. Barak and M. Mahmoody-Ghidary. Merkle puzzles are optimal—an $O(n^2)$-query attack on any key exchange from a random oracle. In *Advances in cryptology—CRYPTO 2009*, volume 5677 of *Lecture Notes in Comput. Sci.*, pages 374–390. Springer, Berlin, 2009.

[5] A. Beimel, Y. Ishai, E. Kushilevitz, and T. Malkin. One-way functions are essential for single-server private information retrieval. In *Annual ACM Symposium on*

Theory of Computing (Atlanta, GA, 1999), pages 89–98. ACM, New York, 1999.

[6] N. Bitansky, R. Canetti, Y. T. Kalai, and O. Paneth. On virtual grey box obfuscation for general circuits. *Algorithmica*, 79(4):1014–1051, 2017.

[7] I. Bonacina and N. Talebanfard. Improving resolution width lower bounds for k-CNFs with applications to the strong exponential time hypothesis. *Inform. Process. Lett.*, 116(2):120–124, 2016.

[8] C. Calabro, R. Impagliazzo, V. Kabanets, and R. Paturi. The complexity of Unique k-SAT: an isolation lemma for k-CNFs. *J. Comput. System Sci.*, 74(3):386–393, 2008.

[9] M. Chase and I. Visconti. Secure database commitments and universal arguments of quasi knowledge. In *Advances in cryptology—CRYPTO 2012*, volume 7417 of *Lecture Notes in Comput. Sci.*, pages 236–254. Springer, Heidelberg, 2012.

[10] M. Chase and I. Visconti. Secure database commitments and universal arguments of quasi knowledge. In *Advances in cryptology—CRYPTO 2012*, volume 7417 of *Lecture Notes in Comput. Sci.*, pages 236–254. Springer, Heidelberg, 2012.

[11] R. P. Chris Calabro, Russell Impagliazzo. A duality between clause width and clause density for sat. 2006.

[12] P. Dongxue, L. Hongda, and N. Peifang. Candidate differing-inputs obfuscation from indistinguishability obfuscation and auxiliary-input point obfuscation. Cryp-

tology ePrint Archive, Report 2018/1055, 2018. https://eprint.iacr.org/2018/1055.

[13] S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. *SIAM J. Comput.*, 45(3):882–929, 2016.

[14] C. Gentry. Applications of (indistinguishablity) obfuscation. *Notes from Cryptography Boot Camp, Simons Institute*, 2015.

[15] O. Goldreich. A uniform-complexity treatment of encryption and zero-knowledge. *J. Cryptology*, 6(1):21–53, 1993.

[16] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *J. Assoc. Comput. Mach.*, 33(4):792–807, 1986.

[17] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *J. Assoc. Comput. Mach.*, 33(4):792–807, 1986.

[18] I. Haitner, J. J. Hoch, O. Reingold, and G. Segev. Finding collisions in interactive protocols—tight lower bounds on the round and communication complexities of statistically hiding commitments. *SIAM J. Comput.*, 44(1):193–242, 2015.

[19] I. Haitner, O. Reingold, and S. Vadhan. Efficiency improvements in constructing pseudorandom generators from one-way functions. In *STOC'10—Proceedings of the 2010 ACM International Symposium on Theory of Computing*, pages 437–446. ACM, New York, 2010.

[20] I. Haitner, O. Reingold, S. Vadhan, and H. Wee. Inaccessible entropy. In *STOC'09—Proceedings of the 2009 ACM International Symposium on Theory of Computing*, pages 611–620. ACM, New York, 2009.

[21] T. Hertli. Breaking the PPSZ barrier for unique 3-SAT. In *Automata, languages, and programming. Part I*, volume 8572 of *Lecture Notes in Comput. Sci.*, pages 600–611. Springer, Heidelberg, 2014.

[22] C. Kingsford. Sat, coloring, hamiltonian cycle, tsp. 2017.

[23] K. Makino, S. Tamaki, and M. Yamamoto. Derandomizing the HSSW algorithm for 3-SAT. *Algorithmica*, 67(2):112–124, 2013.

[24] R. A. Moser and D. Scheder. A full derandomization of Schöning's k-SAT algorithm. In *STOC'11—Proceedings of the 43rd ACM Symposium on Theory of Computing*, pages 245–251. ACM, New York, 2011.

[25] M. Naor. Bit commitment using pseudo-randomness (extended abstract). In *Advances in cryptology—CRYPTO '89 (Santa Barbara, CA, 1989)*, volume 435 of *Lecture Notes in Comput. Sci.*, pages 128–136. Springer, New York, 1990.

[26] S. J. Ong and S. Vadhan. Zero knowledge and soundness are symmetric. In *Advances in cryptology—EUROCRYPT 2007*, volume 4515 of *Lecture Notes in Comput. Sci.*, pages 187–209. Springer, Berlin, 2007.

[27] R. Paturi, P. Pudlák, M. E. Saks, and F. Zane. An improved exponential-time algorithm for k-SAT. *J. ACM*, 52(3):337–364, 2005.

[28] P. Rogaway and T. Shrimpton. A provable-security treatment of the key-wrap problem. In *Advances in cryptology—EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Comput. Sci.*, pages 373–390. Springer, Berlin, 2006.

[29] H. Rogers, Jr. *Theory of recursive functions and effective computability*. MIT Press, Cambridge, MA, second edition, 1987.

[30] R. R. Williams. Strong ETH breaks with Merlin and Arthur: short non-interactive proofs of batch evaluation. In *31st Conference on Computational Complexity*, volume 50 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages Art. No. 2, 17. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2016.

[31] D. Y. Xiao. The evolution of expander graphs. *BA Thesis, Harvard*, 2003.