Quantum and Post-Quantum Cryptography

Venkata Koppula[*]

January 2nd, 2023

Contents

[*]kvenkata@cse.iitd.ac.in

Many thanks to Ramprasad Saptarishi for this LaTeX template.

Course Introduction

Lecture 1:
January 3rd, 2023

COL759 Recap

Let us start with a quick recap of a few basic principles of provable security from COL759. For any cryptographic primitive, recall the four-step recipe for a provably secure construction:

1. (Syntax and security definition) The first step is to define the syntax. This involves defining the various algorithms involved, as well as the correctness condition. For example, in the case of public key encryption, there is a setup algorithm that outputs a public key and a secret key, an encryption algorithm that uses the public key to encrypt a message (producing a ciphertext), and a decryption algorithm that uses the secret key to decrypt the ciphertext. Correctness requires that if a message m is encrypted using a public key, then decryption of the ciphertext using the corresponding secret key must produce m.

   Once the syntax is defined, we define security for the cryptographic primitive. This is usually done via a security game. For public key encryption, we defined the semantic security game between a challenger and an adversary. The challenger runs the setup algorithm to sample a secret key and a public key. It sends the public key to the adversary. Next, the adversary picks two distinct messages m_0 and m_1 (adversarially), and sends them to the challenger. The challenger encrypts one of them (using the public key) and sends the resulting ciphertext to the adversary. In order to win the game, the adversary must guess whether m_0 was encrypted, or m_1. An encryption scheme is secure if no polynomial time adversary can win this game with noticeable advantage.

2. (Choosing an appropriate cryptographic assumption) Most cryptography is based on cryptographic assumptions. These are computational problems which are believed to be 'hard'. For example, we saw computational problems such as RSA, DDH. The second step, therefore, is to choose an appropriate cryptographic assumption.

3. (Proposing a construction) Once we have chosen a cryptographic assumption, we propose a construction. Depending on the structure present in our cryptographic assumption, the constructions can be very different. For instance, compare the RSA based PKE scheme and the El-Gamal encryption scheme.

4. (Proof of security) The final step ties together the first three steps. We show that if there exists a polynomial time adversary that wins the security game (defined in Step 1) against our construction (given in Step 3), then there exists a polynomial time algorithm that solves the 'hard' computation problem (chosen in Step 2).

This recipe has worked very well so far (at least in theory). Decades of research in algorithms and complexity theory have given us many hard compu-

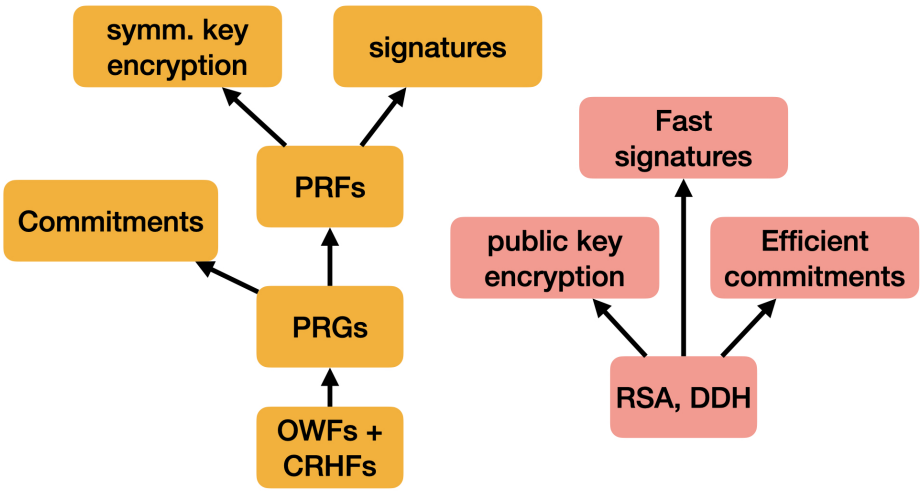


Figure 1: COL759: summarized in one figure. Some cryptographic primitives that can be built from OWFs and CRHFs. In practice, we have very efficient OWFs, PRGs, PRFs and CRHFs. Others such as public key encryption and key agreement rely on computational number-theoretic problems such as RSA or DDH.

tational problems, and some of them are well suited for building cryptographic primitives. Thanks to these hard computational problems, we have good confidence in our security systems.

What happens if someone finds an efficient algorithm for one of these hard problems? The natural idea then is to use a different hard problem, and hope that (a) it is structurally very different so that it doesn't succumb to the same attack (b) it still has enough structure to be useful for cryptography. This is the reason why we have different constructions for the same cryptographic primitive under a diverse set of assumptions.

Quantum computers are coming

In the recent years, there has been tremendous progress in the area of quantum computing. A couple of years ago, Google announced a 53 qubit quantum computer. Even though this quantum device was extremely noisy, Google claimed that it could solve certain computational problems within seconds, which would require much more time on a classical computer. While this demonstration is exciting news (and quantum computing has received much media attention due to this), this is bad news for cybersecurity. A lot of our modern protocols rely on number-theoretic problems such as RSA and DDH, and there exist efficient quantum algorithms for solving RSA and DDH. As a result, if we have a crypto primtive whose security is based on RSA/DDH, the security proof is useless in the presence of quantum adversaries.

Since then, other research groups have built quantum processors with 100+ qubits.

Fortunately, we do have computational problems which (a) are potentially quantum-resilient, (b) can be used for building cryptography (see Part I for

more details). In Section 1, we discuss how to build quantum resilient one-way functions and quantum-resilient collision resistant hash functions. In Section 2, we discuss how to build quantum-resilient public key encryption schemes. As a result, have we successfully dealt with all quantum adversaries (assuming the computational problems used in Section 1 and 2)? For instance, we saw (in the last two lectures of COL759) that OWFs + CRHFs suffice for building (classical) zero knowledge proofs. Does that mean that quantum-resilient OWFs + quantum-resilient CRHFs imply quantum-resilient zero knowledge proofs? Similarly, it is easy to show that the existence of secure public key encryption implies the existence of secure commitment schemes. Does this mean that quantum-resilient PKE implies quantum-resilient commitment schemes? The next section explores these questions in a bit more detail, and most of this course will revolve around such questions.

Challenges in the presence of quantum adversaries

This section will briefly discuss two scenarios where a classical proof/definition is not very useful in the presence of quantum adversaries.

Rewinding issue in quantum zero knowledge proofs

Towards the end of COL759, we saw the notion of zero knowledge proofs, and also saw that public key encryption can be used to build a secure zero knowledge protocol for the 3COL problem, and therefore we get a zero knowledge protocol for all of NP.

We will discuss these protocols in detail next week.

Suppose now we want a zero knowledge protocol in the presence of quantum adversaries. Let us focus on the zero-knowledge property. This involves cheating verifiers. Suppose we want to deal with cheating quantum verifiers. The security definition stays the same, except for a minor modification — we will need to allow the simulator to be a quantum polynomial time algorithm.

Definition Attempt (ZK property wrt quantum poly. time adversaries, informal)**.** *For every quantum polynomial time verifier $\mathbf{V}^*$, there exists a quantum polynomial time simulator S such that the verifier's view in the real protocol is indistinguishable from the simulator's output.*

Given this reasonable looking definition, can we replicate the classical proof for zero-knowledgeness? Unfortunately, no. The proof crucially relied on rewinding the verifier, and in the quantum setting, one needs to be very careful with regard to such arguments (and in particular, many of the classical proofs simply don't have an analogue in the quantum setting).

The curious case of commitments

In the last section, we saw that proofs of security may not translate to the quantum setting. For some cryptographic primitives, we might need new security definitions too. We will illustrate this using cryptographic commitments.

Recall the following toy motivation for commitments: there is a professor who gives very difficult assignments. The students want some 'proof' that the professor can solve the assignment. Commitments can be used in this scenario. The professor, when giving out the assignment, also produces a 'commitment'

to the solutions. This commitment must not reveal the committed solutions. Additionally, we also want the commitment to be binding. After the assignment deadline, the professor must produce an 'opening' which proves that he/she indeed committed to the solutions. Informally, we don't want the following scenario: the students submit solutions, and the professor uses one of those (correct) solutions to produce a matching opening. Let us formally define the syntax for commitments, and focus on the binding security game.

Last semester, we defined commitments as a one-round interactive protocol. Here, we are looking at a slightly weaker notion which is easier to work with. There is a honestly chosen public key, and the sender/receiver must use this public key. This allows us a 'non-interactive' commitment scheme.

NON-INTERACTIVE COMMITMENTS WITH SETUP : A non-interactive commitment scheme with setup consists of the following two algorithms:

- $\mathsf{Setup}(1^n)$: The setup algorithm takes as input the security parameter, and outputs a public key pk.
- $\mathsf{Commit}(\mathsf{pk}, m; r)$: The commitment algorithm takes as input a public key pk, message m and randomness r. It outputs a commitment com. The randomness r is used as the opening.

Last semester, we defined the following (strong) security game for binding:

Binding-Game

- Challenger chooses a public key pk using Setup and sends it to the adversary.
- Adversary produces a commitment com, and two messages m_0, m_1 and randomness r_0, r_1. The adversary wins if $\mathsf{Commit}(\mathsf{pk}, m_0; r_0) = \mathsf{Commit}(\mathsf{pk}, m_1; r_1) = \mathsf{com}$.

Figure 2: The Binding Security Game

Note, for our toy scenario, we only require a weaker security requirement for binding. I'll call this 'prof-binding':

In our informal lingo, weaker security requirement implies the adversary's job is harder, and therefore our 'expectations' from the commitment scheme are 'weaker'.

Prof-binding is not a standard notion; please don't use this outside our classroom :) We will give this a proper name when we discuss this in more detail.

Prof-Binding-Game

- Challenger chooses a public key pk using Setup and sends it to the adversary.
- Adversary produces a commitment com.
- Challenger picks a random message m and sends it to the adversary.
- Adversary produces randomness r and wins if $\mathsf{Commit}(\mathsf{pk}, m; r) = \mathsf{com}$.

Figure 3: The Prof-Binding Security Game

Prof-binding a weaker security requirement than (standard) binding, because if a commitment scheme satisfies binding, then it also satisfies prof-binding.

Claim. *Let $\mathcal{C} = (\mathsf{Setup}, \mathsf{Commit})$ be a commitment scheme. If there exists a p.p.t. adversary that wins the* Prof-Binding-Game *w.r.t. $\mathcal{C}$, then there exists a p.p.t. adversary that wins the* Binding-Game *w.r.t. $\mathcal{C}$.*

Sketch of Proof. Suppose there exists an adversary that wins the Prof-Binding-Game w.r.t. $\mathcal{C}$. The reduction algorithm works as follows:

- The reduction algorithm receives public key pk from the challenger, which it forwards to the adversary.
- Next, the adversary sends a commitment com. The reduction algorithm picks a uniformly random message m and sends to the adversary. The adversary sends opening r.
- The reduction then 'rewinds' the adversary to the point immediately after it sends com. The reduction chooses a new random message m' and sends it to this rewinded adversary. It receives r' as the new opening. Note that if the adversary successfully wins the Prof-Binding-Game, then $\mathsf{Commit}(\mathsf{pk}, m; r) = \mathsf{Commit}(\mathsf{pk}, m'; r') = \mathsf{com}$.
- The reduction algorithm sends (m, r) and (m', r') to the challenger, and wins the Binding-Game.

□

Note: It is important to rewind the adversary in this proof, and not restart the adversary's execution. This is because we want two (message, randomness) pairs *for the same commitment* com.

This claim, together with the claim you proved in Assignment 1 of COL759, give us the following result:

Claim. *Assuming the existence of secure PRGs, there exists a commitment scheme that satisfies binding security. Using the above claim, there also exists a commitment scheme that satisfies prof-binding security (assuming the existence of secure PRGs).*

Let us now focus our attention on the quantum analogues of these security definitions. The definition is quite natural — replace the 'p.p.t. adversary' with a 'quantum polynomial time adversary'. Your COL759 Assignment 1 proof can be adapted to get a commitment scheme that satisfies binding security w.r.t. quantum adversaries, assuming quantum-secure PRGs. However, the existence of quantum-secure PRGs **does not** immediately imply the existence of commitment schemes that satisfies prof-binding security w.r.t. quantum adversaries. Again, the issue is that we cannot rewind a quantum adversary. This brings us to the following questions:

What is the 'correct' definition for binding security in the presence of quantum adversaries? And if we care about prof-binding, then how do we construct such commitment schemes?

Going beyond classical messages

So far, we have considered classical crypto primitives in the presence of quantum adveraries. What happens if the messages themselves are *qubits* instead of classical bit strings? Can we commit to a quantum state, or encrypt a quantum state? Can we commit to a quantum state using only classical communication? These are some of the questions that we will consider in the final segment of our course.

Part I: Lattice-based Cryptography

Introduction to Lattice-based Cryptography

Notations We will use the following notations for this chapter (and most other chapters, unless specified otherwise).

- For any two integers $a < b$, $[a, b]$ denotes the set of integers $\{a, a+1, \ldots, b\}$.
- For a finite set S, Unif_S denotes the uniform distribution over S.
- For a finite set $\mathcal{S}$, $x \leftarrow \mathcal{S}$ denotes a uniformly random element drawn from $\mathcal{S}$. Similarly, for a distribution $\mathcal{D}$ over $\mathcal{S}$, $x \leftarrow \mathcal{D}$ denotes an element drawn from distribution $\mathcal{D}$.
- For any two distributions $\mathcal{D}_1, \mathcal{D}_2$ over a finite set $\mathcal{X}$, the statistical distance between $\mathcal{D}_1$ and $\mathcal{D}_2$, denoted by $\mathsf{SD}(\mathcal{D}_1, \mathcal{D}_2)$ is defined as follows:

  $$\mathsf{SD}(\mathcal{D}_1, \mathcal{D}_2) = \frac{1}{2}\left(\sum_{a \in \mathcal{X}} \left| \Pr_{x \leftarrow \mathcal{D}_1}[x = a] - \Pr_{x \leftarrow \mathcal{D}_2}[x = a] \right|\right)$$

  We say that two distributions are statistically indistinguishable, denoted by $\mathcal{D}_1 \approx_s \mathcal{D}_2$ if $\mathsf{SD}(\mathcal{D}_1, \mathcal{D}_2)$ is negligible in $n = \log|\mathcal{X}|$.

  If two distributions are statistically indistinguishable, then no algorithm can distinguish between these two distributions with non-negligible advantage.
- Let $\{\mathcal{X}_n\}_n$ be a family of finite sets, where $|\mathcal{X}_n| \leq 2^{\mathrm{poly}(n)}$ for some fixed polynomial $\mathrm{poly}(\cdot)$. For any two efficiently sampleable distributions $\mathcal{D}_1, \mathcal{D}_2$ over $\mathcal{X}_n$, the shorthand notation $\mathcal{D}_1 \approx_c \mathcal{D}_2$ indicates that the two distributions are computationally indistinguishable. That is, for any p.p.t. adversary $\mathcal{A}$, there exists a negligible function $\mu(\cdot)$ such that for all n,

  $$\Pr[\mathcal{A}(x) = 1 : x \leftarrow \mathcal{D}_1] - \Pr[\mathcal{A}(x) = 1 : x \leftarrow \mathcal{D}_2] \leq \mu(n).$$

1 The Short-Integer-Solutions Problem

The first (potentially) quantum-resilient assumption of this course is the Short Integer Solutions problem.

Computational Problem 1 (Short Integer Solution Problem ($\mathsf{SIS}_{n,m,q}$))**.** *Given a uniformly random matrix $\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}$, output a nonzero vector $\mathbf{x} \in \{-1, 0, 1\}^m$ such that $\mathbf{A} \cdot \mathbf{x} \pmod q = \mathbf{0}$.*

The 'interesting' regime for this problem is when $n^{1.5} \leq q \leq \exp(n)$ and $m \geq \Omega(n \log q)$. If we did not have the constraint that $\mathbf{x} \in \{-1, 0, 1\}$, then this problem can be solved easily using Gaussian elimination. However, once we

add this restriction for $\mathbf{x}$, this problem is no longer easy. If m is large enough, then such a 'short integer solution' is guaranteed to exist (see exercise below).

Exercise 1.1. *Show that if $2^m > q^n$, then for every matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, there exists a vector $\mathbf{x} \in \{-1, 0, 1\}^m$ such that $\mathbf{A} \cdot \mathbf{x} \pmod q = \mathbf{0}$.*[a]

[a] An earlier version of this exercise had a 3^m instead of 2^m. As pointed out by one of the students in class, the old version was incorrect. Thanks for the correction.

However, finding such a short integer solution is believed to be hard (even given a quantum computer). Ajtai [Ajt96] showed that, in the interesting regime, if there exists an efficient algorithm for the SIS problem, then there exists an efficient algorithm for certain lattice problems in the *worst-case*.

The main parameters that govern the hardness of SIS: n and the size of modulus q. The exact form of q does not matter. The parameter m should be polynomial in n and $\log q$, but the exact value of m does not alter the hardness of SIS.

1.1 One-Way Functions and Collision Resistance from SIS

Recall, a collision resistant hash function family is a set of keyed functions such that, given a random function from this family, it is hard to find two inputs that map to the same output. Let q be a power of 2, and $m > 2n \log q$. Consider the following function family $\mathcal{H} = \left\{ H_{\mathbf{A}} : \{0,1\}^m \to \mathbb{Z}_q^n \right\}_{\mathbf{A} \in \mathbb{Z}_q^{n \times m}}$ where

$$H_{\mathbf{A}}(\mathbf{x}) = \mathbf{A} \cdot \mathbf{x} \pmod q$$

This is a compressing function since $m > 2n \log q$, and if there exist distinct bit-vectors $\mathbf{x}$ and $\mathbf{y}$ such that $\mathbf{A} \cdot \mathbf{x} = \mathbf{A} \cdot \mathbf{y}$, then $\mathbf{A} \cdot (\mathbf{x} - \mathbf{y}) = \mathbf{0}$, and $\mathbf{x} - \mathbf{y} \in \{-1, 0, 1\}^m$.[1]

Since this function family is sufficiently compressing, collision resistance implies one-wayness.

Exercise 1.2. *Show that if $\mathcal{H} = \left\{ H_k : \{0,1\}^{2n} \to \{0,1\}^n \right\}_{k \in \mathcal{K}}$ is a collision-resistant hash function family, then $\mathcal{H}$ is also a one-way function family.*

However, this does not hold true if $\mathcal{H}$ is only mildly-compressing. That is, suppose $\mathcal{H} = \left\{ H_k : \{0,1\}^{n+1} \to \{0,1\}^n \right\}_{k \in \mathcal{K}}$ is a collision-resistant hash function family. Construct a new hash function family $\mathcal{H}'$ that is also collision-resistant, but $\mathcal{H}'$ is not one-way.

1.2 Commitment Scheme from SIS

In this section, we will present a non-interactive commitment scheme with setup, based on SIS. For this construction, we will consider a variant of SIS, called 'inhomogeneous SIS' (iSIS).

Computational Problem 2 (Inhomogeneous Short Integer Solution Problem ($\mathsf{iSIS}_{n,m,q}$))**.** *Given a uniformly random matrix $\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}$ and a uniformly random vector $\mathbf{b} \leftarrow \mathbb{Z}_q^n$, output a nonzero vector $\mathbf{x} \in \{-1, 0, 1\}^m$ such that $\mathbf{A} \cdot \mathbf{x} \pmod q = \mathbf{b}$.*

[1] All these equalities are modulo q, I will ignore $\pmod q$ when it is clear from the context.

It is easy to show that this variant is as hard as SIS.

Exercise 1.3. *Show that there exists a p.p.t. adversary that solves* $\mathsf{iSIS}_{n,m,q}$ *with non-negligible probability, if and only if there exists a p.p.t. adversary that solves* $\mathsf{SIS}_{n,m,q}$ *with non-negligible probability.*

The iSIS gives a direct proof of one-wayness of $\{H_{\mathbf{A}} \equiv \mathbf{A} \cdot \mathbf{x}\}_{\mathbf{A}}$. This proof goes via the *leftover hash lemma*, a useful lemma that we'll encounter multiple times in this course. Essentially, this lemma says that if m is sufficiently larger than $n \log q$, then given a random matrix $\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}$, the vector $\mathbf{A} \cdot \mathbf{x}$, where $\mathbf{x}$ is drawn from a distribution with enough randomness, looks like a uniformly random vector in $\mathbb{Z}_q^n$. We will not prove this lemma here (you can refer to [AB09] for a proof of the lemma).

Fact 1 (Leftover Hash Lemma)**.** *Let $S \subset \mathbb{Z}_q$, $|S| = B$. If $m \geq n \log_B q + n$, then the statistical distance between the following distributions is a negligible funtion of n:*

$$\mathcal{D}_1 = \left\{ (\mathbf{A}, \mathbf{A} \cdot \mathbf{x}) : \begin{array}{c} \mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}, \\ \mathbf{x} \leftarrow S^m \end{array} \right\} \qquad \mathcal{D}_2 = \left\{ (\mathbf{A}, \mathbf{u}) : \begin{array}{c} \mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}, \\ \mathbf{u} \leftarrow \mathbb{Z}_q^n \end{array} \right\}$$

This is a simplified version of the leftover hash lemma. The 'full version' states that for any distribution $\mathcal{D}$ over $\mathbb{Z}_q^m$ with sufficient entropy, if $\mathbf{x} \leftarrow \mathcal{D}$, then $(\mathbf{A}, \mathbf{A} \cdot \mathbf{x}) \approx_s (\mathbf{A}, \mathbf{U})$.

A short discussion on LHL: First, let us understand where the name of this lemma comes from. The 'hash' in the name arises because the function $\mathbf{x} \to \mathbf{A} \cdot \mathbf{x}$ is a pairwise independent hash function mapping $\{0,1\}^m$ to $\mathbb{Z}_q^n$. The 'leftover' in the name comes from the following scenario that arises often in crypto/-complexity: suppose you have a source of randomness X that outputs t bits of randomness, and you know that there is an adversary that has somehow learnt k out of t bits. You don't know which bits the adversary knows. Given that the adversary has only k bits of information, you still have $t - k$ bits of randomness *left over*. The leftover hash lemma says that by choosing an appropriate hash function, you can indeed extract somme pure randomness out of this corrupted source.

A more concrete scenario: Alice and Bob are performing the Diffie-Hellman key exchange protocol. Alice chooses a group generator g, an integer $a \leftarrow \mathbb{Z}_q$ and sends (g, g^a) to Bob. Bob chooses $b \leftarrow \mathbb{Z}_q$ and sends g^b to Alice. Both can now compute g^{ab} (and therefore both have $\log q$ random bits). But suppose the adversary can learn t bits about b. Such things can happen in practice, and are known as *side channel attacks*. For instance, maybe the adversary is close to Bob's computer and is monitoring the electromagnetic radiations when Bob is computing g^b. Now we cannot conclude that Bob has no information about the shared key g^{ab}. To take care of this situation, Alice also sends an appropriate hash function (say a matrix $\mathbf{A}$). Instead of using g^{ab} as the shared key, both Alice and Bob must first express g^{ab} as a bit string $\mathbf{x}$, and then use the hash of $\mathbf{x}$ (that is, $\mathbf{A} \cdot \mathbf{x}$) as the shared key.

Exercise 1.4. *Let $m = n$. Compute a lower bound on the statistical distance*

between the following distributions:

$$\mathcal{D}_1 = \left\{ (\mathbf{A}, \mathbf{A}\cdot\mathbf{x}) : \begin{array}{c} \mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m}, \\ \mathbf{x} \leftarrow \{0,1\}^m \end{array} \right\} \quad \mathcal{D}_2 = \left\{ (\mathbf{A}, \mathbf{u}) : \begin{array}{c} \mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m}, \\ \mathbf{u} \leftarrow \mathbb{Z}_q^{n} \end{array} \right\}$$

Let us put the leftover hash lemma to good use. Recall, last semester we had discussed that OWFs can be used to build PRGs, and in one of your assignments, you had constructed a commitment scheme using PRGs. Therefore, following this approach, we have a commitment scheme whose security is based on SIS. There is a simpler direct construction based on SIS, which is described below.

Lecture 2: January 6th, 2023

Construction 1.1 (Commitment scheme from SIS)**.** *The message space for our commitment scheme is $\{0,1\}^m$, and the sender's commitment algorithm uses m bits of randomness.*

- $\mathsf{Setup}(1^n)$: *The setup algorithm samples two matrices $\mathbf{A}_1, \mathbf{A}_2 \leftarrow \mathbb{Z}_q^{n\times m}$ and sets $\mathsf{pk} = (\mathbf{A}_1, \mathbf{A}_2)$.*
- $\mathsf{Commit}(\mathsf{pk}, \mathsf{msg})$: *The commit algorithm chooses a uniformly random bit string $\mathbf{r} \leftarrow \{0,1\}^m$ and sets $\mathsf{s\text{-}msg} = \mathbf{A}_1 \cdot \mathsf{msg} + \mathbf{A}_2 \cdot \mathbf{r}$.*

$\Diamond$

The binding property is similar to the collision-resistance of $\mathcal{H}$ (discussed in Section 1.1 above). The reduction algorithm receives a matrix $\mathbf{A} = [\mathbf{A}_1 \mid \mathbf{A}_2]$ from the SIS challenger. It sends $\mathsf{pk} = (\mathbf{A}_1, \mathbf{A}_2)$ and sends it to the adversary. The adversary then produces $\mathsf{msg}_0, r_0, \mathsf{msg}_1, r_1$ (each of which is an m bit string) and wins if $\mathbf{A}_1 \cdot \mathsf{msg}_0 + \mathbf{A}_2 \cdot r_0 = \mathbf{A}_1 \cdot \mathsf{msg}_1 + \mathbf{A}_2 \cdot r_1$. The reduction algorithm sends $\mathbf{x} = [\mathsf{msg}_0 \mid r_0]^\top - [\mathsf{msg}_1 \mid r_1]^\top$ to the SIS challenger.

For hiding, note that the commitment $\mathsf{com} = \mathbf{A}_1 \cdot \mathsf{msg} + \mathbf{A}_2 \cdot r$. Using the leftover hash lemma, it follows that $\left\{ (\mathbf{A}_2, \mathbf{A}_2 \cdot r) : \mathbf{A}_2 \leftarrow \mathbb{Z}_q^{n\times m}, r \leftarrow \{0,1\}^m \right\} \approx_s \left\{ (\mathbf{A}_2, \mathbf{u}) : \mathbf{A}_2 \leftarrow \mathbb{Z}_q^{n\times m}, \mathbf{u} \leftarrow \mathbb{Z}_q^n \right\}$. As a result, com is statistically indistinguishable from a uniformly random vector, and as a result, even an exponential time adversary cannot recover the message given the commitment.

Summing it up, we have the following claim.

Claim 1.2. *Let $m = 2n \log q$. Assuming $\mathsf{SIS}_{n,2m,q}$ is computationally hard, the non-interactive commitment scheme with setup described in Construction 1.1 is computationally binding (that is, no polynomial time adversary can win* Binding-Game *defined in Figure 2) and statistically hiding (no adversary can win the hiding game defined in Figure* **??**).

Exercise 1.5. *The commitment scheme in Construction 1.1 involves a honest setup. In COL759 Assignment 1, you had built a stronger commitment scheme. In that scheme, the receiver chooses the first message (there is no trusted setup). Would the above construction satisfy that stronger definition?*

Another way to remove honest setup is to allow the sender to run the setup (in this case, choose matrices $\mathbf{A}_1, \mathbf{A}_2$). Would that be secure?

Exercise 1.6. *The commitment scheme described in Construction 1.1 is computationally binding and statistically hiding. Similarly, one can define and construct commitment schemes that are statistically binding and computationally hiding.*

Prove that it is impossible to construct a commitment scheme that is statistically binding and statistically hiding.

2 The Learning-with-Errors Problem

The SIS problem can be used to build OWFs, CRHFs, but the real quantum threat is for public key encryption (and other crypto primitives that rely on RSA/DDH). Fortunately, we have another computational problem that is also believed to be quantum-resilient, and has enough structure to give us public key encryption (and much more). This problem is closely related to the SIS problem, and is called the 'Learning with Errors' problem. The (decisional) Learning with Errors problem is parameterized by matrix dimensions n, m, modulus q and a 'noise distribution' χ over $\mathbb{Z}_q$.

For the purpose of this course, it suffices to think of q as a $\sqrt{n}$-bit prime and χ as uniform distribution over $\{-B, B\}$, where B is much smaller than q. For instance, take $B = \sqrt{q}$.

Computational Problem 3 (Learning with Errors Problem ($\mathsf{LWE}_{n,m,q,\chi}$))**.** *Distinguish between the following distributions:*

$$\mathcal{D}_1 = \left\{ (\mathbf{A}, \mathbf{s}^\top \cdot \mathbf{A} + \mathbf{e}^\top) : \begin{array}{c} \mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m} \\ \mathbf{s} \leftarrow \mathbb{Z}_q^n \\ \mathbf{e} \leftarrow \chi^m \end{array} \right\} \quad \mathcal{D}_2 = \left\{ (\mathbf{A}, \mathbf{u}^\top) : \begin{array}{c} \mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m} \\ \mathbf{u} \leftarrow \mathbb{Z}_q^m \end{array} \right\}$$

We will call samples from $\mathcal{D}_1$ as 'LWE pairs'.

A few noteworthy points about this problem:

- The vector $\mathbf{e}$ is the 'error vector'. If there was no error vector, then it is easy to distinguish between the two distributions (using Gaussian elimination).
- There is a corresponding 'search' version of this problem, where given $(\mathbf{A}, \mathbf{s}^\top \cdot \mathbf{A} + \mathbf{e}^\top)$, one must learn $\mathbf{s}$ (hence the name 'learning with errors'). We have a search-to-decision reduction for LWE, and therefore it suffices to restrict our attention to the decision version.
- Suppose $q = O(2^{\sqrt{n}})$, and let χ be the uniform distribution over $[-\sqrt{q}, \sqrt{q}]$. Let us compute the amount of randomness needed to get one sample from each of the distributions. In the distribution $\mathcal{D}_2$, we require $(n \cdot m + m) \cdot \log q \approx (n \cdot m + m) \cdot \sqrt{n}$ bits of randomness (per sample). In $\mathcal{D}_1$, the amount of randomness (per sample) is $n \cdot m \cdot \sqrt{n} + n \log q + m \cdot \log B \approx n \cdot m \cdot \sqrt{n} + n \cdot \sqrt{n} + m \cdot \sqrt{n}/2$. As a result, if m is larger than $2n$, these two distributions are statistically far-apart (and hence an unbounded adversary can distinguish between these two distributions). However, a computationally bounded adversary cannot distinguish between the two distributions (assuming $\mathsf{LWE}_{n,m,q,\chi}$ is hard).

- Just like the SIS problem, LWE is interesting only when m is larger than n. For instance, if $m \leq n$, then the two distributions are statistically indistinguishable.

The Learning with Errors problem was introduced in the landmark paper of Regev [Reg09]. The key contributions of this paper are as follows:

- Regev showed that the search version of LWE is as hard as some well studied lattice problems. His reduction was a **quantum** reduction. That is, he showed that if there exists an algorithm (classical or quantum) that solves the search version of LWE, then there exists a quantum algorithm for solving all instances of hard lattice problems.

The main parameters that govern the hardness of LWE: size of q and the modulus/noise ratio. Again, it does not depend much on the parameter m. Similarly, it does not depend on the form of q.

- Next, he showed that decisional LWE is as hard as the search version. His reduction was for modulus having certain properties, but this was later extended to work for all moduli.
- Finally, Regev showed that this is a very useful assumption for cryptography. Regev showed how to build public key encryption from decisional LWE. Soon after, researchers realized that this is a very useful assumption for crypto.

Thanks to one of the students for raising this question in class.

If the error is very small, say each coordinate is bounded by some constant value B, then there is a polynomial time attack on LWE, shown by Arora and Ge [AG11]. There are other subexponential time attacks for specialized parameter settings. However, for the version described above, currently there is no known algorithm that runs in time $o(2^n)$.

2.1 *LWE and SIS*

SIS and LWE are closely related problems. Both these problems, with minor modification, become 'easy' problems via Gaussian elimination. In SIS, if we remove the restriction that $x \in \{-1, 0, 1\}^m$, then the problem can be solved efficiently. Similarly, in LWE, if we remove the error, then this problem can also be solved efficiently.

The following observation shows that SIS is as hard as LWE.

Observation 2.1 (SIS $\geq$ LWE)**.** *If there exists a p.p.t. algorithm that solves* $\mathsf{SIS}_{n,m,q}$ *(as defined in Section 1), then there exists a p.p.t. algorithm that solves* $\mathsf{LWE}_{n,m,q,\chi}$*.*

Sketch of Proof. The reduction algorithm receives $(\mathbf{A}, \mathbf{b})$ from the LWE challenger. It sends $\mathbf{A}$ to the SIS adversary, and receives $\mathbf{x} \in \{-1, 0, 1\}^m$. If $\mathbf{A} \cdot \mathbf{x} = \mathbf{0}$, then the reduction checks if $\mathbf{b}^\top \cdot \mathbf{x}$ is in $[0, q/4] \cup [3q/4, q]$. If so, it concludes that $(\mathbf{A}, \mathbf{b})$ is an LWE pair. Else it concludes that $(\mathbf{A}, \mathbf{b})$ are uniformly random. □

The other direction (LWE $\geq$ SIS) is interesting. Currently, we don't know a classical reduction, but a quantum reduction follows from Regev's work. I do not plan to discuss this reduction in this course. However, if there's sufficient interest in seeing this reduction, I am happy to include it in one of the later lectures.

2.2 *Cryptographic Primitives from LWE*

LWE is a very versatile cryptographic assumption. When it was proposed by Regev, it was introduced as a post-quantum hardness assumption that implies public key encryption. However, over the next few years, researchers showed various advanced cryptographic primitives that can be built using LWE. We will see a few of them in this course.

First, note that LWE immediately gives us a pseudorandom generator. This is because the LWE distribution uses fewer random bits than the uniform distribution, yet is computationally indistinguishable from the uniform distribution. PRGs imply PRFs, which in turn imply semantically secure symmetric-key encryption. However, let us look at a direct construction of symmetric key encryption based on LWE. This will serve as a warm-up for public-key encryption.

Lecture 3: January 10th, 2023

Symmetric Key Encryption using LWE

Construction 2.2 (Symmetric key encryption scheme based on LWE)**.** *Let $q = 2^{\sqrt{n}}$, $m = O(n \log q)$, $\chi \equiv \mathsf{Unif}_{[-\sqrt{q},\sqrt{q}]}$ be the LWE parameters. The message space is $\{0,1\}$, and the secret key is a vector $\mathbf{s} \in \mathbb{Z}_q^n$.*

- $\mathsf{Enc}(\mathbf{s}, \mathsf{msg})$*: Choose a random vector $\mathbf{a} \leftarrow \mathbb{Z}_q^n$ and $e \leftarrow \chi$, output* $\mathsf{ct} = \left(\mathbf{c}_1 = \mathbf{a}, c_2 = \mathbf{s}^\top \cdot \mathbf{a} + e + \mathsf{msg} \cdot \frac{q}{2}\right)$.
- $\mathsf{Dec}(\mathbf{s}, \mathsf{ct} = (\mathbf{c}_1, c_2))$*: If $c_2 - \mathbf{s}^\top \cdot \mathbf{c}_1$ is in the range $[q/4, 3q/4]$, then output* 1*, else output* 0*.*

$\Diamond$

Perfect correctness: The correctness of this scheme follows immediately. Since χ is the uniform distribution over $[0, \sqrt{q}] \cup [q - \sqrt{q}, q]$, if $e \leftarrow \chi$, then $e + q/2 \pmod q$ will be in the range $[q/4, 3q/4]$.

Semantic security proof sketch: Suppose an adversary $\mathcal{A}$ makes at most t queries and breaks the semantic security of this scheme. Then there exists a reduction algorithm that breaks the $\mathsf{LWE}_{n,t,q,\chi}$ assumption. The reduction receives an $n \times t$ matrix $\mathbf{A}$ and a vector $\mathbf{b} \in \mathbb{Z}_q^t$ from the LWE challenger. For the i^{th} encryption query by $\mathcal{A}$, the reduction uses the i^{th} column of $\mathbf{A}$ and the i^{th} entry of $\mathbf{b}$.

Additive homomorphism The above scheme has the following property (which is easy to verify): we can 'add' ciphertexts, and the underlying messages get added $\pmod 2$. Given encryption $\mathsf{ct} = (\mathbf{c}_1, c_2)$ of message msg and encryption $\mathsf{ct}' = (\mathbf{c}_1', c_2')$ of message msg', we can produce an encryption of $\mathsf{msg} \oplus \mathsf{msg}'$. This is simply $(\mathbf{c}_1 + \mathbf{c}_1', c_2 + c_2')$.[2] Note that the error grows slightly, but perfect decryption still holds.

[2] Again, I am ignoring the $\bmod q$ here.

Similarly, given an encryption ct of msg, we can also produce an encryption of the negation $1 \oplus \mathsf{msg}$ (without knowing msg).

I will refer to this as 'homomorphic negation'.

Later in the course, we will see how to perform arbitrary computations over ciphertexts.

Another approach for building symmetric key encryption from LWE The following scheme was proposed in class:

Construction 2.3 (Another SKE scheme based on LWE)**.** *Let $q = 2^{\sqrt{n}}$, $m = O(n \log q)$, $\chi \equiv \mathsf{Unif}_{[-\sqrt{q},\sqrt{q}]}$ be the LWE parameters. The message space is $\{0,1\}$, and the secret key is a vector $\mathbf{s} \in \mathbb{Z}_q^n$.*

- $\mathsf{Enc}(\mathbf{s}, \mathsf{msg})$*: Choose two random matrices $\mathbf{A}_0, \mathbf{A}_1$ of dimensions $n \times n$ and an error vector $\mathbf{e} \leftarrow \chi^n$. Compute $\mathbf{c} = \mathbf{s}^\top \cdot \mathbf{A}_{\mathsf{msg}} + \mathbf{e}$ and output $(\mathbf{A}_0, \mathbf{A}_1, \mathbf{c})$ as the ciphertext.*
- $\mathsf{Dec}(\mathbf{s}, \mathsf{ct} = (\mathbf{A}_0, \mathbf{A}_1, \mathbf{c}))$*: If $\|\mathbf{c} - \mathbf{s}^\top \cdot \mathbf{A}_0\| \le q/4$, output 0, else output 1.*

$\Diamond$

Suppose the adversary makes at most t queries. This scheme is semantically secure, assuming $\mathsf{LWE}_{n,nt,q,\chi}$ is hard. However, this scheme has a small (negligible) decryption error. Negligible decryption error is mostly fine. However, for certain applications (see Section 2.2), we require perfect correctness. Also, this scheme does not seem to have the homomorphism property (which will be useful in the following section).

Why is it reasonable to assume that we know a bound on the adversary's running time/number of queries? The reduction can always 'guess' a bound on the number of queries. As long as the adversary's view is unaffected by this guess, we are fine.

Exercise 2.1. *Modify the above construction (Construction 2.3) to achieve perfect correctness.*

Public Key Encryption using LWE

Let us now extend Construction 2.2 to build a public key encryption scheme. We will first propose an abstract framework for going from SKE to PKE, and then present a concrete LWE-based construction. Suppose there is a symmetric key encryption scheme $\mathcal{E}_{\mathrm{SKE}} = (\mathsf{Setup}_{\mathrm{SKE}}, \mathsf{Enc}_{\mathrm{SKE}}, \mathsf{Dec}_{\mathrm{SKE}})$ with perfect correctness, and supporting additive homomorphism and homomorphic negation. The message space is $\{0,1\}$, and the ciphertext space is $\{0,1\}^\ell$.

$\mathcal{E}_{\mathrm{SKE}} \to \mathcal{E}_{\mathrm{PKE}}$ framework: In our public key encryption scheme, the setup algorithm runs the symmetric key setup $\mathsf{Setup}_{\mathrm{SKE}}$, generating a secret key sk. Next, it produces *many* encryptions of 0. How many depends on the ciphertext size (which is ℓ). Suppose it produces m ciphertexts. The public key consists of these m encryptions of zero.

To encrypt a bit msg using this public key, we do the following: if $\mathsf{msg} = 0$, pick a random subset S of these ciphertexts, and 'add' them together. Thanks to the additive homomorphism, this resulting ciphertext will also be an encryption of

0. If $\mathsf{msg} = 1$, we will again pick a random subset S of these ciphertexts, and add them together. Finally, we will homomorphically negate the resulting ciphertext, resulting in an encryption of 1.

It follows, from the additive homomorphism and perfect correctness, that the resulting public key encryption scheme is also perfectly correct. Why is it secure? Here again, leftover hash lemma is our friend. Note, we are mapping the random subset S (which can be described using m bits) to an ℓ bit string. If this mapping is a 'nice' hash such that LHL can be applied, then the resulting ciphertext looks like a uniformly random string.

An instantiation of the above framework - Regev's encryption scheme Below we discuss the public key encryption scheme proposed by Regev.

In Regev's scheme, the modulus was polynomial, and hence there was a small decryption error.

Construction 2.4 (Regev's PKE scheme based on LWE)**.** *Let $q = 2^{\sqrt{n}}$, $m = O(n \log q)$, $\chi \equiv \mathsf{Unif}_{[-\sqrt{q},\sqrt{q}]}$ be the LWE parameters.*
Regev encryption scheme for message space $\{0,1\}$ can be specified as follows:

- $\mathsf{Setup}(1^n)$*: It sets m, q, χ as above. It sample a random matrix $\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}$, a random secret vector $\mathbf{s} \leftarrow \mathbb{Z}_q^n$, and a random error vector $\mathbf{e} \leftarrow \chi^m$. It outputs the public-secret key pair as:*

$$\mathsf{pk} = (\mathbf{A}, \mathbf{b}) \text{ where } \mathbf{b}^\top = \mathbf{s}^\top \cdot \mathbf{A} + \mathbf{e}^\top, \quad \mathsf{sk} = \mathbf{s}.$$

- $\mathsf{Enc}(\mathsf{pk}, \mathsf{msg})$*: Let $\mathsf{pk} = (\mathbf{A}, \mathbf{b})$. It samples a random vector $\mathbf{r} \leftarrow \{0,1\}^m$. It outputs the ciphertext as:*

$$\mathsf{ct} = \left(\mathbf{A} \cdot \mathbf{r}, \mathbf{b}^\top \cdot \mathbf{r} + \mathsf{msg} \cdot \frac{q}{2}\right).$$

- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$*: Let the ciphertext $\mathsf{ct} = (\mathbf{c}_1, c_2)$. It outputs the message bit as:*

$$\mathsf{round}^{q/2}(c_2 - \mathbf{c}_1^\top \cdot \mathbf{s}),$$

where $\mathsf{round}^{q/2}(z)$ rounds an element $z \in \mathbb{Z}_q$ to 1 if $z \in [q/4, 3q/4]$ and 0 otherwise.

$\Diamond$

Perfect correctness: The correctness of this scheme relies on the following observation: the quantity $\mathbf{e}^\top \cdot \mathbf{r} \bmod q$ will never be in the range $[q/4, 3q, 4]$. This is because $\mathbf{e} \in [-\sqrt{q}, \sqrt{q}]^m$ and $\mathbf{r} \in \{0,1\}^m$.

Similarly, $(q/2 + \mathbf{r}^\top \cdot \mathbf{e}) \bmod q$ will never be in the range $[0, q/4] \cup [3q/4, q]$. Therefore, $\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, \mathsf{msg})) = \mathsf{msg}$ for $\mathsf{msg} \in \{0,1\}$.

Semantic security proof sketch: Using LWE, we can first switch $\mathbf{b}$ to be a uniformly random vector. Once we have done that, note that using the leftover hash lemma (Fact 1), $(\mathbf{A} \cdot \mathbf{r}, \mathbf{b}^\top \cdot \mathbf{r})$ looks like a uniformly random vector in $\mathbb{Z}_q^{n+1}$. This concludes our proof sketch.

Viewing Regev's scheme via the $\mathcal{E}_{\text{SKE}} \to \mathcal{E}_{\text{PKE}}$ framework: The public key in Regev's scheme is a matrix $\mathbf{A}$ and vector $\mathbf{b}^\top = \mathbf{s}^\top \cdot \mathbf{A} + \mathbf{e}^\top$. We can view the public key as m encryptions of 0, using the secret vector $\mathbf{s}$.

To encrypt a message bit msg, Regev's scheme computes $\mathbf{A} \cdot \mathbf{r}$ and $\mathbf{b}^\top \cdot \mathbf{r} + \mathsf{msg} \cdot \frac{q}{2}$, where $\mathbf{r} \leftarrow \{0,1\}^m$. Note that $\mathbf{r}$ is used to sample a subset of the columns of $\mathbf{A}$ (and similarly the same subset of the entries of $\mathbf{b}$). As a result, we are taking a random subset of the zero encryptions present in the public key, and adding them together. If $\mathsf{msg} = 0$, then this is our final ciphertext. Otherwise we add $\frac{q}{2}$.

Finally, we need to argue security. The random set (viewed as an m bit string) is mapped to a ciphertext (which consists of $n+1$ integers in $\mathbb{Z}_q$). Therefore, if m is sufficiently larger than $n \log q$, then there's some hope of using LHL. The hashing from the set to $n+1$ integers uses $(\mathbf{A}, \mathbf{b})$, but they're not uniformly random. However, using the LWE assumption, we argue that $\mathbf{A}, \mathbf{b}$ are computationally indistinguishable from uniformly random, and once that is done, we can use LHL.

Thanks to one of the students for raising this question in class.

Exercise 2.2. *Suppose an encryption scheme has message space $\{0,1\}^t$ and ciphertext space $\{0,1\}^\ell$. The* encryption rate *of this scheme is defined as t/ℓ. Clearly, high-rate would be desirable for efficiency reasons.*

Note that the rate of Regev's encryption scheme (as described in Construction 2.7) is $\Theta(1/n \log q) = 1/n\sqrt{n}$. Modify the construction to improve the rate to $\Theta(1/\sqrt{n})$.

Non-interactive commitments (without setup) from LWE

In Construction 1.2, we built a commitment scheme (based on SIS) that required an honest setup. In this section, we will see a simple construction that does not require any setup or interaction. This commitment scheme can be built *generically from any PKE with perfect correctness*. Therefore, using Regev's encryption scheme, we have a noninteractive commitment scheme without setup.

Construction 2.5 (Noninteractive commitments without setup)**.** *Let $\mathcal{E}_{\text{PKE}} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ be a public key encryption scheme with message space $\mathcal{M}$, and having perfect correctness.*

- $\mathsf{Commit}(\mathsf{msg} \in \mathcal{M}, 1^n)$*: The commitment algorithm takes as input the message* msg *and security parameter n. It chooses* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^n)$*, computes* $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathsf{msg})$ *and sends* $(\mathsf{pk}, \mathsf{ct})$ *as the commitment.*

The opening for the commitment is the randomness used by Setup *and* Enc. ◊

The hiding property follows immediately from the semantic security of $\mathcal{E}_{\text{PKE}}$. However, we need to be careful with the binding property. Recall, in Construction 1.1, when we allowed the sender to choose $(\mathbf{A}_1, \mathbf{A}_2)$, then we could not guarantee the binding property. Here also, the same issue could arise: *what if the adversarial sender can choose 'malicious' public key* pk *and ciphertext* ct *such that there exist two secret keys* $\mathsf{sk}_0, \mathsf{sk}_1$ *with the property that both can be tied to* pk*, and* $\mathsf{Dec}(\mathsf{sk}_b, \mathsf{ct}) = \mathsf{msg}_b$*?*

Here, we will use the perfect correctness of $\mathcal{E}_{\text{PKE}}$. Since the scheme is perfectly correct, for every $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^n)$, every message $\mathsf{msg} \in \mathcal{M}$ and every $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, m)$, $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) = \mathsf{msg}$. Suppose an adversary can break the binding property. Then it can output a public key pk, a ciphertext ct and randomness pairs $(r_{\mathsf{Setup},0}, r_{\mathsf{Enc},0})$ and $(r_{\mathsf{Setup},1}, r_{\mathsf{Enc},1})$ such that

- $\mathsf{Setup}(1^n; r_{\mathsf{Setup},b}) = (\mathsf{pk}, \mathsf{sk}_b)$
- $\mathsf{Enc}(\mathsf{pk}, \mathsf{msg}_0; r_{\mathsf{Enc},0}) = \mathsf{Enc}(\mathsf{pk}, \mathsf{msg}_1; r_{\mathsf{Enc},1}) = \mathsf{ct}$.

This would violate the perfect correctness. There exists a public key/secret key pair $(\mathsf{pk}, \mathsf{sk}_0)$, a message msg_1, a ciphertext $\mathsf{ct} = \mathsf{Enc}(\mathsf{pk}, \mathsf{msg}_1; r_{\mathsf{Enc},1})$ such that $\mathsf{Dec}(\mathsf{sk}_0, \mathsf{ct}) = \mathsf{msg}_0$.

2.3 *Variants of LWE*

Just like we saw for SIS, there are a number of variants of LWE that are as hard as vanilla LWE. Here, we discuss one variant where the secret vector, instead of being a uniformly random vector, is drawn from the noise distribution. This is called *small-secrets LWE*, since the secret vector consists of small entries.

Computational Problem 4 (Small Secrets Learning with Errors Problem ($\mathsf{ss\text{-}LWE}_{n,m,q,\chi}$))**.** *Distinguish between the following distributions:*

$$\mathcal{D}_1 = \left\{ (\mathbf{A}, \mathbf{s}^\top \cdot \mathbf{A} + \mathbf{e}^\top) : \begin{array}{c} \mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m} \\ \mathbf{s} \leftarrow \chi^n \\ \mathbf{e} \leftarrow \chi^m \end{array} \right\} \quad \mathcal{D}_2 = \left\{ (\mathbf{A}, \mathbf{u}^\top) : \begin{array}{c} \mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m} \\ \mathbf{u} \leftarrow \mathbb{Z}_q^{m} \end{array} \right\}$$

Clearly, LWE is as hard as ssLWE, since $\mathsf{uniform} + \chi \equiv \mathsf{uniform}$. The following claim shows that small-secret LWE is as hard as LWE (albeit with a slight increase in m - the number of columns).

Claim 2.6 ($\mathsf{ss\text{-}LWE} \geq \mathsf{LWE}$)**.** *Let q be a prime. If there exists a p.p.t. algorithm $\mathcal{A}$ that solves $\mathsf{ss\text{-}LWE}_{n,m-n,q,\chi}$, then there exists a p.p.t. algorithm $\mathcal{B}$ that solves $\mathsf{LWE}_{n,m,q,\chi}$.*

Sketch of Proof. The reduction algorithm receives $(\mathbf{A}, \mathbf{b})$ from the LWE challenger. Let $\mathbf{A}_1$ denote the first n columns of $\mathbf{A}$, and $\mathbf{A}_2$ the last $m - n$ columns. Since $\mathbf{A}$ is chosen uniformly at random, $\mathbf{A}_1$ is invertible (with high probability). Let $\mathbf{b}_1$ denote the first n entries of $\mathbf{b}$, and $\mathbf{b}_2$ the remaining $m - n$ entries. If $(\mathbf{A}, \mathbf{b})$ is a $\mathsf{LWE}_{n,m,q,\chi}$ pair and $\mathbf{b}^\top = \mathbf{s}^\top \cdot \mathbf{A} + \mathbf{e}^\top$, then $\mathbf{b}_1^\top = \mathbf{s}^\top \cdot \mathbf{A}_1 + \mathbf{e}_1^\top$ and $\mathbf{b}_2^\top = \mathbf{s}^\top \cdot \mathbf{A}_2 + \mathbf{e}_2^\top$.

Computing the inverse of $\mathbf{A}_1$ is the only place where we've used the primality of q.

Let $\mathbf{A}' = -\mathbf{A}_1^{-1} \cdot \mathbf{A}_2$. This is a matrix of dimension $(m - n) \times n$. Since $\mathbf{A}_1$ and $\mathbf{A}_2$ are uniformly random matrices, $\mathbf{A}'$ is also uniformly random.

Let $\mathbf{b}' = \mathbf{b}_1^\top \cdot \mathbf{A}' + \mathbf{b}_2^\top$ be a vector of dimension $(m - n)$. If $(\mathbf{A}, \mathbf{b})$ is a $\mathsf{LWE}_{n,m,q,\chi}$ pair, then $(\mathbf{A}', \mathbf{b}')$ is a $\mathsf{ss\text{-}LWE}_{n,m-n,q,\chi}$ pair, since $\mathbf{b}' = \mathbf{e}_1^\top \cdot \mathbf{A}' + \mathbf{e}_2^\top$. If $\mathbf{b}$ is uniformly random, then so is $\mathbf{b}'$. Therefore the adversary $\mathcal{A}$ can be used to decide whether $(\mathbf{A}', \mathbf{b}')$ is an ssLWE pair or not. □

Here is another LWE variant that was proposed in the lecture, where the secret is a binary vector, and noise is same as above. This variant is also as hard as LWE (although in this case, the LWE parameters will be worse; there will be a $\log q$ factor loss in n).

Thanks to one of the students for bringing this up. I had mistakenly said that we don't know a hardness proof for this variant.

Computational Problem 5 (Binary Secrets Learning with Errors Problem ($\mathsf{bin\text{-}LWE}_{n,m,q,\chi}$)). *Distinguish between the following distributions:*

$$\mathcal{D}_1 = \left\{ (\mathbf{A}, \mathbf{s}^\top \cdot \mathbf{A} + \mathbf{e}^\top) : \begin{array}{c} \mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m} \\ \mathbf{s} \leftarrow \{0,1\}^n \\ \mathbf{e} \leftarrow \chi^m \end{array} \right\} \quad \mathcal{D}_2 = \left\{ (\mathbf{A}, \mathbf{u}^\top) : \begin{array}{c} \mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m} \\ \mathbf{u} \leftarrow \mathbb{Z}_q^{m} \end{array} \right\}$$

One can show that $\mathsf{bin\text{-}LWE}_{n,m,q,\chi}$ is as hard as $\mathsf{LWE}_{n/\log q,m,q,\chi}$, see this paper [Mic18] for reference. I might include a simplified version in one of the future problem sets.

Regev's encryption scheme with smaller public keys

Recall, the public key in Regev's encryption scheme consists of a matrix of dimension $n \times m$, where $m \geq n \log q$. Can we use an $n \times n$ matrix instead? As you may have noticed, this immediately leads to a problem: we cannot use LHL (since LHL requires $m \geq n \log q + n$). LHL was needed to argue that $(\mathbf{A} \cdot \mathbf{r}, \mathbf{b}^\top \cdot \mathbf{r})$ is indistinguishable from a uniformly random vector in $\mathbb{Z}_q^{n+1}$. However, instead of a statistical hammer (LHL), can we use a computational assumption?

This leads us to the following attempt: set the public key as $(\mathbf{A}, \mathbf{s}^\top \cdot \mathbf{A} + \mathsf{noise})$ where $\mathbf{A}$ is an $n \times n$ matrix. To encrypt a bit msg, we choose a low-norm vector $\mathbf{r}$, and output $\mathbf{c}_1 = \mathbf{A} \cdot \mathbf{r} + \mathsf{noise}$, $c_2 = \mathbf{b}^\top \cdot \mathbf{r} + \mathsf{noise}$.

As you may have noticed, decryption will not be correct. This is because there is noise in the ciphertext, and $\mathbf{s}^\top \cdot \mathsf{noise}$ will blow up. The fix, therefore, is to choose $\mathbf{s}$ also from the noise distribution (and use ssLWE instead of LWE). We will have to reduce the noise bound slightly. The full construction is described below.

Note that the error bound here is slightly less than $q^{0.5}$. Thanks for pointing out during the lecture.

Construction 2.7 (LWE-based PKE scheme with smaller public keys). *Let $q = 2^{\sqrt{n}}$, $m = n$, $\chi \equiv \mathsf{Unif}_{[-q^{0.4},q^{0.4}]}$ be the ssLWE parameters.*
The encryption scheme for message space $\{0,1\}$ can be specified as follows:

- $\mathsf{Setup}(1^n)$*: It sample a random matrix $\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times n}$, a random secret vector $\mathbf{s} \leftarrow \chi^n$, and a random error vector $\mathbf{e} \leftarrow \chi^n$. It outputs the public-secret key pair as:*

$$\mathsf{pk} = (\mathbf{A}, \mathbf{b}) \text{ where } \mathbf{b}^\top = \mathbf{s}^\top \cdot \mathbf{A} + \mathbf{e}^\top, \quad \mathsf{sk} = \mathbf{s}.$$

- $\mathsf{Enc}(\mathsf{pk}, \mathsf{msg})$*: Let $\mathsf{pk} = (\mathbf{A}, \mathbf{b})$. It samples a random vector $\mathbf{r} \leftarrow \chi^n$, error vector $\mathbf{e}_1 \leftarrow \chi^n$ and $e_2 \leftarrow \chi$. It outputs the ciphertext as:*

$$\mathsf{ct} = \left(\mathbf{A} \cdot \mathbf{r} + \mathbf{e}_1, \mathbf{b}^\top \cdot \mathbf{r} + e_2 + \mathsf{msg} \cdot \frac{q}{2} \right).$$

- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$*: Let the ciphertext $\mathsf{ct} = (\mathbf{c}_1, c_2)$. It outputs the message bit as:*

$$\mathsf{round}^{q/2}(c_2 - \mathbf{c}_1^\top \cdot \mathbf{s}),$$

where $\mathsf{round}^{q/2}(z)$ *rounds an element* $z \in \mathbb{Z}_q$ *to* 1 *if* $z \in [q/4, 3q/4]$ *and* 0 *otherwise.*

$\Diamond$

The scheme satisfies perfect correctness. For semantic security, we first switch the public key to uniformly random using the ss-$\mathsf{LWE}_{n,n,q,\chi}$ assumption. Next, we switch $(\mathbf{A} \cdot \mathbf{r} + \mathbf{e}_1, \mathbf{b}^\top \cdot \mathbf{r} + e_2)$ to uniformly random using the ss-$\mathsf{LWE}_{n,n+1,q,\chi}$ assumption.

Can we further reduce the public key size? One natural question is whether we need a uniformly random matrix? Can we choose one column vector $\mathbf{a}$, and deterministically generate the remaining columns from $\mathbf{a}$? This is an active area of research. The resulting security cannot be based on LWE, but instead relies on a family of assumptions called Ring-LWE. Unfortunately this is beyond the scope of this course.

Part II: Interactive Proofs

In the previous part, we saw two post-quantum hardness assumptions, and used them to build a few post-quantum cryptographic primitives. In this part, we will see the first scenario where a classical proof technique doesn't translate to the quantum setting. This is in the context of zero knowledge proofs (ZKPs). We will spend a couple of lectures on a more general primitive called interactive proofs. Besides being a very useful and well-studied generalization of ZKPs, interactive proofs will also be useful later in the semester, when we discuss *quantum cryptography*.

Post-quantum security refers to security of classical crypto primitives in the presence of a quantum adversary. The syntax and security definition stays the same, except that 'p.p.t. adversary' is replaced with 'efficient quantum adversary'.

Introduction to Interactive Proofs

The complexity class NP consists of problems (such as SAT, 3COL) whose solutions are efficiently and deterministically verifiable. For every problem, there is a deterministic verification process $\mathbf{V}$ such that if an instance is a 'yes' instance, then there exists a 'certificate' that $\mathbf{V}$ can check efficiently. For example, let us consider 3COL. The 'yes' instances consist of all graphs that can be colored using three distinct colors such that no two neighboring vertices have the same color. As a result, for every 'yes' instance, there is a very simple certificate: an assignment of colors to the vertices. Given a yes instance graph G and such a valid coloring, the verifier can check that only three colors are used, and that no neighboring vertices in the graph have the same color.

However, what if someone wants to prove that there exists **no** three coloring for a graph? It is unlikely that such statements will have short certificates. The lack of short certificates does not mean that such statements cannot be efficiently verified. A series of remarkable works showed that such statements (and much harder problems) can be verified if we allow interaction and a small error probability.

An interesting (non-technical) summary of the rich history of interactive proofs can be found here.

3 Interactive Proofs for Problems outside NP

Let us start with the formal definitions, and some notations that will be useful for this section.

Notations

- An interactive protocol consists of interactive algorithms $\mathbf{P}, \mathbf{V}$ (the prover and verifier) sending messages back and forth, and at the end, the verifier outputs a single bit. For a k message protocol, if the prover sends the first message, think of these as k Turing machines $\mathbf{P}_1, \mathbf{V}_2, \mathbf{P}_3, \ldots, \mathbf{P}_k$. The prover consists of $(\mathbf{P}_1, \mathbf{P}_3, \ldots, \mathbf{P}_k)$, and the verifier consists of $(\mathbf{V}_2, \ldots, \mathbf{V}_{k-1})$. On input x, $\mathbf{P}_1(x)$ outputs the first message m_1. The verifier uses x, m_1 and outputs $m_2 \leftarrow \mathbf{V}_2(x, m_1)$. Next, the prover outputs $m_3 = \mathbf{P}_3(x, m_1, m_2)$ and so on.

- Let $\text{out}\langle \mathbf{P}, \mathbf{V}\rangle(x)$ denote the final output of $\mathbf{V}$. The verifier Turing machines are randomized, and therefore this output is a random variable.
- The transcript of the protocol, denoted by $\text{trans}\langle \mathbf{P}, \mathbf{V}\rangle(x)$ consists of all the messages exchanged between $\mathbf{P}$ and $\mathbf{V}$. Again, this is a random variable.

Definition 3.1 (Interactive Proofs)**.** *Let $\mathsf{L} = (\mathsf{L}_{\text{yes}}, \mathsf{L}_{\text{no}})$ denote a language, where L_{yes} consists of all the 'yes' instances, and L_{no} consists of all the 'no' instances. An interactive proof system for L with completeness error c and soundness error s consists of a pair of interactive algorithms $(\mathbf{P}, \mathbf{V})$ where $\mathbf{V}$ is a polynomial time algorithm, and the following properties hold:*

- *Completeness: For every $x \in \mathsf{L}_{\text{yes}}$, $\Pr[\text{out}\langle \mathbf{P}, \mathbf{V}\rangle(x) = 1] \geq 1 - c$*
- *Soundness: For every $x \in \mathsf{L}_{\text{no}}$ and every prover $\mathbf{P}^*$ (including malicious provers), $\Pr[\text{out}\langle \mathbf{P}^*, \mathbf{V}\rangle(x) = 1] \leq s$.*

◊

For example, consider the *graph nonisomorphism* problem. Here the 'yes' instances set L_{yes} consists of graph pairs (G_0, G_1) that are not isomorphic, and the 'no' instances set L_{no} consists of graph pairs (G_0, G_1) such that G_0 and G_1 are isomorphic.

A few comments regarding this definition:

- If we have completeness error c and soundness error s where $1 - c$ and s are at least $1/\text{poly}$ apart, then we can boost the gap by sequential/parallel repetition, resulting in a protocol with negligible completeness and soundness errors.

Error reduction for sequential repetition is easy to prove; the case of parallel repetition requires a careful argument. See [Gol98], Appendix C for a formal proof.

- Using the above definition of interactive protocols, we can define the complexity class IP, which is the set of all languages that have probabilistic polynomial time verifiers. Note that in the definition of IP, the prover is allowed to run for unbounded time. There are also other complexity classes that capture protocols that have prover/verifier with certain resource restrictions.

3.1 *Interactive proof for Graph Non-Isomorphism*

Lecture 4: January 13th, 2023

The first interactive protocol (for a problem outside NP) will be for the graph nonisomorphism problem. Recall, the 'yes' instances here are pairs of graphs (on the same set of vertices) that are not isomorphic. The protocol for graph non-isomorphism is described below.

Common Input: graphs $G_0 = (V, E_0)$, $G_1 = (V, E_1)$.

Claim to prove: G_0 and G_1 are not isomorphic.

Protocol 1: Graph Non-Isomorphism: Private Coins Protocol

1 Verifier chooses a bit $b \leftarrow \{0,1\}$ and a uniformly random permutation $\pi \leftarrow S_n$. It computes a new graph $H = (V, E_H)$ where H is a random isomorphism of the graph G_b. More formally, $(\pi(u), \pi(v)) \in E_H$ if and only if $(u, v) \in E_b$. Verifier sends H to the prover.
2 Prover checks which of G_0 or G_1 is isomorphic to H (since the prover is unbounded, it can iterate over all permutations in S_n). It sends the corresponding bit $b' \in \{0,1\}$.
3 Verifier outputs 1 if $b = b'$, else it outputs 0.

S_n is the set of all permutations of $[n]$.

Completeness: If G_0 and G_1 are not isomorphic, then the graph H is isomorphic to exactly one of the two graphs. The prover can correctly find the graph that is isomorphic to H.

Soundness: For soundness, we require that if G_0 and G_1 are both isomorphic to each other, then H is a random isomorphism of both G_0 and G_1. Therefore, the graph H contains no information about the bit b sampled by the verifier. As a result, the prover's guess b' is correct with probability equal to $1/2$.

Reducing the soundness error: The k parallel repetition of the above protocol results in soundness error being $1/2^k$.

3.2 *Public-coins Interactive proof for Graph Non-Isomorphism*

The protocol described in Section 3.1 is a 'private coins' protocol. In that protocol, it was crucial that the verifier's randomness remains hidden from the prover. A surprising result by Goldwasser and Sipser [GS86] showed that any interactive protocol can be transformed into one where the verifier reveals all its randomness. Such protocols are called 'public coins' protocol. Public coins protocols are interesting because they have a nice structural property: for any constant k, a k round public coins protocol can be transformed into a two round public coins protocol.

In this section, we will discuss a public coins protocol for graph nonisomorphism. This protocol uses pairwise independent hash functions, which are defined below.

Definition 3.2 (Pairwise Independent Hash Functions)**.** *Let $\mathcal{H}$ be a family of hash functions with domain $\mathcal{X}$ and co-domain $\mathcal{Y}$. We say that $\mathcal{H}$ is a pairwise independent hash function family if the following properties hold:*

- *for any $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, $\Pr_{h \leftarrow \mathcal{H}}[h(x) = y] = 1/|\mathcal{Y}|$.*
- *for any distinct $x, x' \in \mathcal{X}$ and $y, y' \in \mathcal{Y}$, $\Pr_{h \leftarrow \mathcal{H}}[h(x) = y \text{ and } h(x') = y'] = 1/|\mathcal{Y}|^2$*

◊

The second property says that knowing the value at x tells you no information about the value at x'. A uniformly random function from $\mathcal{X}$ to $\mathcal{Y}$ would be k-wise independent (for any k). However, sampling and storing a uniformly random function is very expensive. Instead, for many applications, just pairwise independence suffices, in which case we can use a pairwise independent hash function family. These hash functions can be sampled/stored efficiently.

Example 3.3 (Pairwise Independent Hash Function)**.** *Let m, n be integers, $m > n$. Consider the hash family $\left\{f_{\mathbf{A},\mathbf{b}} : \{0,1\}^m \to \{0,1\}^n\right\}_{\mathbf{A}\in\mathbb{Z}_2^{n\times m},\mathbf{b}\in\mathbb{Z}_2^n}$ where*

$$f_{\mathbf{A},\mathbf{b}}(\mathbf{x}) = \mathbf{A}\cdot\mathbf{x} + \mathbf{b} \pmod 2.$$

Check that this satisfies the two requirements of pairwise independent hash functions. ◊

Back to our public coins protocol for GNI: the key idea here is to consider the set of graphs that are isomorphic to either G_0 or G_1. Intuitively, in the 'yes' case, the size of this set is larger than the size of the set in the 'no' case. A natural first guess is that the size will be $2n!$ in the 'yes' case, and $n!$ in the 'no' case.

However, strictly speaking, this statement is not true. Consider for instance, a 'yes' instance on three vertices, where G_0 is the triangle graph, and G_1 is the empty graph on three vertices. These two graphs are non-isomorphic. However, the set of graphs that are isomorphic to one of these two has only two graphs. On the other hand, if we take a 'no' instance where G_0 and G_1 are both path graphs on three vertices, then there are three graphs that are isomorphic to either G_0 or G_1.

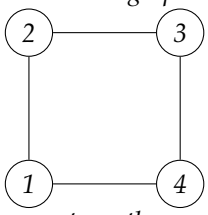
The main issue is that for certain graphs (such as the triangle and empty graphs), the set of graphs isomorphic to such graphs are identical to the graph. Such permutations are called *automorphisms* of the graph.

Definition 3.4 (Automorphism of a graph)**.** *Given a graph $G = (V, E)$ on n vertices, an automorphism of G is a permutation $\pi \in S_n$ such that for all (u, v) pairs,*

$$(u, v) \in E \iff (\pi(u), \pi(v)) \in E.$$

◊

Example 3.5 (Automorphisms of the 4-cycle)**.** *Consider the cycle graph on 4 vertices. Check that this graph has 8 automorphisms:*



$(1\to1, 2\to2, 3\to3, 4\to4)$ $(1\to1, 2\to4, 3\to3, 4\to2)$
$(1\to2, 2\to1, 3\to4, 4\to3)$ $(1\to2, 2\to3, 3\to4, 4\to1)$
$(1\to3, 2\to2, 3\to1, 4\to4)$ $(1\to3, 2\to4, 3\to1, 4\to2)$
$(1\to4, 2\to3, 3\to2, 4\to1)$ $(1\to4, 2\to1, 3\to2, 4\to3)$

There are two other graphs isomorphic to the 4-vertex graph.

◊

For any graph G, the set $\mathsf{aut}(G)$ consists of all automorphisms of G. Note that this set is non-empty, since $\mathsf{aut}(G)$ always contains the identity permutation. Similarly, let $\mathsf{iso}(G)$ denote the set of all graphs isomorphic to G (this set includes G). From the definitions of $\mathsf{aut}(G)$ and $\mathsf{iso}(G)$, we get the following observation for graphs on n vertices:

Observation 3.6. *For any graph on n vertices, $|\mathsf{iso}(G)| \cdot |\mathsf{aut}(G)| = n!$.*

This observation follows because there is a bijection between S_n (the set of all permutations over $[n]$) and $\mathsf{iso}(G) \times \mathsf{aut}(G)$.

The above observation gives us the following distinction between the 'yes' and 'no' instances. Instead of looking at all graphs that are isomorphic to either G_0 or G_1, we should look at the following set:

$$\mathsf{S}_{G_0,G_1} = \left\{ (H,\pi) : \begin{array}{c} H \text{ is isomorphic to } G_0 \text{ or } G_1 \\ \pi \in \mathsf{aut}(H) \end{array} \right\}$$

For the 'yes' case, this set has size $2n!$, while in the 'no' case, it has size $n!$.

The protocol

With all tools in place, we are ready to see the public-coins protocol for graph non-isomorphism. The set S_{G_0,G_1} has two properties:

- Given graphs (G_0, G_1), the set S_{G_0,G_1} has size $2n!$ if G_0 and G_1 are non-isomorphic, and size $n!$ if the sets are isomorphic.
- Given graphs (G_0, G_1), in both the 'yes' and 'no' case, for every $(H,\pi) \in \mathsf{S}_{G_0,G_1}$, there exists a short 'certificate' for proving that (H,π) is an element in S_{G_0,G_1}. The certificate simply consists of a bit b and a permutation σ mapping G_b to H. Given b, σ, one can efficiently verify that H is an isomorphism of G_b. (Checking that $\pi \in \mathsf{aut}(H)$ can be performed efficiently without any witness).

Let $\mathcal{U}$ denote the 'universe set' containing S_{G_0,G_1} (take $\mathcal{U} = \{\text{ALL GRAPHS} \times S_n\}$ for instance). The verifier picks a random hash function h mapping $\mathcal{U}$ to a suitable co-domain, and a random point y in the co-domain. The hope is the following:

- if we are in the 'yes' case, then with high probability, there exists a point x in the set S_{G_0,G_1} such that $h(x) = y$.
- in the 'no' case, since S_{G_0,G_1} is a small set, y will have no preimage in S_{G_0,G_1}.

Of course, it is important to choose the size of our co-domain carefully. Consider the first extreme: we choose the co-domain to be very large, say equal to $\mathcal{U}$). Note that S_{G_0,G_1} is much smaller than $\mathcal{U}$, and therefore a random point in the co-domain will not have a preimage in S_{G_0,G_1}. On the other hand, if we choose the co-domain to be very small, then y will have a preimage in S_{G_0,G_1} even in the 'no' case.

Let ℓ be an integer such that $2^{\ell-2} < 2n! < 2^{\ell-1}$, and let $\mathcal{H}$ denote a pairwise independent hash function family with domain $\mathcal{U}$ and co-domain $\{0,1\}^\ell$.

Common Input: Graphs (G_0, G_1)

Claim to prove: $|\mathsf{S}_{G_0,G_1}| = 2n!$

Protocol 2: Graph Non-Isomorphism : Public-coins Protocol

1 Verifier chooses $h \leftarrow \mathcal{H}$ and $y \leftarrow \{0,1\}^\ell$. It sends (h, y) to the prover.
2 Prover finds an $x = (H, \pi) \in \mathsf{S}_{G_0,G_1}$ such that $h(x) = y$. If no such x exists, prover outputs $\perp$. Else, it provides a certificate (b, σ), certifying that $H = \sigma(G_b)$.
3 Verifier, on receiving $(x = (H, \pi), (b, \sigma))$, checks that $h(x) = y$, $H = \sigma(G_b)$ and $\pi \in \mathsf{aut}(H)$. If these checks pass, it outputs 1, else outputs 0.

Clearly, this is a public-coins two-round protocol. Let us now analyze the completeness and soundness error probabilities.

Completeness: This bound follows from inclusion-exclusion, and uses the pairwise independence property.

$$\begin{aligned}
&\mathsf{Pr}_{h,y}\left[\exists x \in \mathsf{S}_{G_0,G_1} \text{ s.t. } h(x) = y\right] \\
&\geq \sum_{x \in \mathsf{S}_{G_0,G_1}} \mathsf{Pr}_{h,y}\left[h(x) = y\right] - \sum_{\substack{x,x' \in \mathsf{S}_{G_0,G_1} \\ x \neq x'}} \mathsf{Pr}_{h,y}\left[h(x) = h(x') = y\right] \\
&= \left(\frac{2n!}{2^\ell}\right) \cdot \left(\frac{(2n!)(2n!-1)}{2 \cdot 2^{2\ell}}\right) = \left(\frac{2n!}{2^\ell}\right) \cdot \left[1 - \left(\frac{(2n!-1)}{2 \cdot 2^\ell}\right)\right] > \left(\frac{2n!}{2^\ell}\right) \cdot \left(\frac{3}{4}\right)
\end{aligned}$$

The first step follows from inclusion-exclusion, and the second step follows from the pairwise independence property.

Soundness: The soundness bound is easier to prove. Note that in the 'no' case, $|\mathsf{S}_{G_0,G_1}| = n! < 2^{\ell-2}$. As a result, with probability at least $1 - n!/2^\ell$, a randomly picked string $y \leftarrow \{0,1\}^\ell$ has no preimage in S_{G_0,G_1}. The soundness error is at most $n!/2^\ell$.

A few remarks about the protocol:

- Note there is a gap in the completeness and soundness error probabilities. Since the gap is at least $1/\mathrm{poly}$, this can be amplified by parallel repetition.
- It is possible to make the completeness 'perfect' (that is, in the 'yes' case, the verifier always accepts).

Exercise 3.1. *Modify the above protocol so that the prover chooses the hash function h.*

4 Interactive Proofs for Very Hard Problems

In this section, we will present an interactive proof for a large class of very hard problems. Recall the NP-complete problem 3SAT. Given an instance ϕ, we want

to know if there *exists* a satisfying assignment. What if we want to count the number of satisfying solutions? Here, we will present an interactive protocol between an unbounded prover and a polynomial time verifier for verifying that a 3SAT instance ϕ has exactly t satisfying inputs.

Note, for instance, this protocol can be used to prove that a formula ϕ has no satisfying inputs. For quite some time, it was believed that such a protocol is not possible, and therefore it was a big surprise when this protocol was proposed.

4.1 *Arithmetization*

The first key idea in this protocol is to *arithmetize* the SAT instance. Given a 3SAT instance ϕ over n variables, one can efficiently produce an n-variate polynomial g_ϕ over $\mathbb{Z}_q$ (for some large q) such that

All coefficients of g_ϕ are from $\mathbb{Z}_q$, and when we evaluate g_ϕ on any $(x_1, \ldots, x_n)$, the evaluation is modulo q.

(*) for any input $(x_1, \ldots, x_n) \in \{0,1\}^n$, $\phi(x_1, \ldots, x_n) = g_\phi(x_1, \ldots, x_n)$

Note that this polynomial g_ϕ needs to agree with ϕ over the Boolean hypercube, but the polynomial can be evaluated at other points too.

The formula ϕ consists of m clauses. For the i^{th} clause over variables $x_{i_1}, x_{i_2}, x_{i_3}$, we will produce a 3-variate polynomial g_i over variables $x_{i_1}, x_{i_2}, x_{i_3}$ such that if clause is satisfied, $g_i(x_{i_1}, x_{i_2}, x_{i_3}) = 1$, otherwise $g_i(x_{i_1}, x_{i_2}, x_{i_3}) = 0$. If we can produce such a polynomial for each clause, then note that $g_\phi(x_1, \ldots, x_n) = \prod_i g_i(x_{i_1}, x_{i_2}, x_{i_3})$ satisfies (*).

Arithmetizing a clause: Given a clause $C(x, y, z)$, a *valid arithmetization* of the clause is a polynomial p_C over $\mathbb{Z}_q$ such that $C(x, y, z) = p_C(x, y, z)$ for all $(x, y, z) \in \{0,1\}^3$. Let $C_i = (x_{i_1} \vee x_{i_2} \vee x_{i_3})$ be a clause. Check that the polynomial

$$g_i(x_{i_1}, x_{i_2}, x_{i_3}) = 1 - (1 - x_{i_1}) \cdot (1 - x_{i_2}) \cdot (1 - x_{i_3})$$

is equal to 1 if at least one of the variables is set to 1. If all the variables are set to 0 in C_i, then g_i evaluates to 0. If one or more of the variables are negated, then we can replace the $(1 - x)$ with x. For example, if $C_i = (\neg x_{i_1} \vee x_{i_2} \vee x_{i_3})$, then check that the following polynomial

$$g_i(x_{i_1}, x_{i_2}, x_{i_3}) = 1 - x_{i_1} \cdot (1 - x_{i_2}) \cdot (1 - x_{i_3})$$

is a valid arithmetization of this clause.

We will call this the *canonical arithmetization* of the formula ϕ.

Important note: Given a formula ϕ, one can efficiently 'write down' the canonical arithmetization g_ϕ. The polynomial g_ϕ can also be efficiently evaluated at any point $(x_1, \ldots, x_n) \in \mathbb{Z}_q^n$.

Example 4.1. *Consider the formula $\phi(x_1, x_2, x_3) = (x_1 \vee x_2) \wedge (\neg x_1 \vee x_3)$. The canonical arithmetization is*

$$g_\phi(x_1, x_2, x_3) = (1 - (1 - x_1) \cdot (1 - x_2)) \cdot (1 - x_1 \cdot (1 - x_3))$$

Note that this is not the unique 3-variate polynomial that agrees with ϕ on the Boolean hypercube. Verify that the simplified polynomial $g'_\phi = x_2 - x_1 x_2 + x_1 x_3$ also agrees with ϕ on the Boolean hypercube. ◊

Observation 4.2. *For any* 3SAT *instance ϕ over n variables and having m clauses, the canonical arithmetization g_ϕ has degree at most $3m$. The degree of g_ϕ in any variable x is at most m (assuming none of the clauses have repeated variables).*

4.2 *The sum-check protocol*

Now that we have seen how to arithmetize a formula, we can cast the original 'logic' problem (counting the number of satisfying inputs) as an 'arithmetic problem' : compute $\sum g_\phi(x_1, \ldots, x_n)$ where the sum is over all $(x_1, \ldots, x_n) \in \{0,1\}^n$. As a result, in our protocol, if a prover needs to prove that ϕ has exactly t satisfying solutions, then it suffices for the prover to prove that $\sum g_\phi(x_1, \ldots, x_n) = t$. Thus, this is a *sum-check* protocol - the verifier must efficiently check that the sum of $g_\phi(x_1, \ldots, x_n)$ over all $(x_1, \ldots, x_n) \in \{0,1\}^n$ is indeed t.

The verifier can efficiently compute g_ϕ at any point $(x_1, \ldots, x_n)$. Using this protocol, it can check the sum of g_ϕ over all $\{0,1\}^n$.

Protocol intuition: First, let us start with the easy case: g_ϕ is a univariate polynomial. In this case, the verifier doesn't need the prover. It can simply check $g_\phi(0) + g_\phi(1) = t$. This suggests that if we have a means of reducing the number of variables one-by-one, then the problem can be solved easily when base case $n = 1$.

Reducing the number of variables: Suppose our verifier could check, for all $(n-1)$ variate polynomials $\tilde{g}$ and integers $\tilde{t}$, if $\sum_{(x_1,\ldots,x_{n-1}) \in \{0,1\}^{n-1}} \tilde{g}(x_1, \ldots, x_{n-1})$ is equal to $\tilde{t}$. Here is the first natural idea to reduce the n-variate case to the $(n-1)$ variate case: given $g(x_1, \ldots, x_n)$, the verifier can ask the prover for the values $t_0 = \sum_{(x_2,\ldots,x_n)} g(0, x_2, \ldots, x_n)$ and $t_1 = \sum_{(x_2,\ldots,x_n)} g(1, x_2, \ldots, x_n)$. Note that $h_0(x_2, \ldots, x_n) \equiv g(0, x_2, \ldots, x_n)$ and $h_1(x_2, \ldots, x_n) \equiv g(1, x_2, \ldots, x_n)$ are both $(n-1)$ variate polynomials, and given these values, the verifier needs to check:

- $t = t_0 + t_1$
- $t_0 = \sum_{x_2,\ldots,x_n} h_0(x_2, \ldots, x_n)$
- $t_1 = \sum_{x_2,\ldots,x_n} h_1(x_2, \ldots, x_n)$

While the first check is easy, there are two recursive calls, and therefore the verifier's work doubles up in each recursive call.

As mentioned by one of the students in class, if the verifier proceeds this way, it is essentially asking the prover for $g_\phi(x_1, \ldots, x_n)$ for all $(x_1, \ldots, x_n) \in \{0,1\}^n$.

Instead of asking the prover for t_0 and t_1 as above, the verifier can ask the prover for a univariate polynomial g_1 such that

$$g_1(X) \equiv \sum_{(x_2,\ldots,x_n) \in \{0,1\}^{n-1}} g(X, x_2, \ldots, x_n)$$

Given this polynomial, the verifier can compute $t_0 = g_1(0)$ and $t_1 = g_1(1)$ and check $t_0 + t_1 = t$. However, the verifier also needs to check that this polynomial is identical to $\sum_{(x_2,\ldots,x_n) \in \{0,1\}^{n-1}} g(X, x_2, \ldots, x_n)$. Can this be done using a single recursive call to sum-check over $(n-1)$ variables? Checking that two polynomials are identical can be very expensive. Note that one polynomial (g_1) is given explicitly in coefficient representation, but the other one $(\sum_{(x_2,\ldots,x_n)} g(X, x_2, \ldots, x_n))$ is implicit, and the verifier cannot compute its coefficient representation.

This brings us to the main observation of the protocol: both these polynomials are low-degree polynomials. Their degree is bounded by m, and *if two low-degree*

polynomials are not identical, they can agree on at most m points. As a result, the verifier picks a random number θ_1, and asks the prover to prove that

Given two degree d polynomials p_1, p_2, consider the polynomial $p = p_1 - p_2$. If $p_1 \neq p_2$, p has at most d roots.

$$\mathsf{Stmt}_1 : \sum_{(x_2,\ldots,x_n)\in\{0,1\}^{n-1}} g(\theta_1, x_2, \ldots, x_n) = g_1(\theta_1).$$

If the prover had lied in the first step (by sending an incorrect g_1), with high probability, it will get caught when it proves Stmt_1. Note that Stmt_1 is a sum-check statement over $(n-1)$ variables, and we can use recursion now.

Before stating the full protocol, we will describe the high-level template. The reader is encouraged to figure out the protocol based on the template.

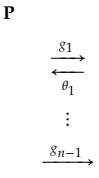
Common Input: 3CNF formula ϕ and integer $t \leq 2^n$. Equivalently, we can think of the common input as a prime $q > 2^n$, a number $t \leq 2^n$ and the canonical arithmetization g_ϕ, a polynomial over $\mathbb{Z}_q$.

Claim to prove: $\sum_{(x_1,\ldots,x_n)\in\{0,1\}^n} g_\phi(x_1, \ldots, x_n) = t \mod q$

Protocol 3: Sum-Check Protocol

1 Set $\gamma_0 = t$ and statement $\mathsf{Stmt}_0 : \sum_{(x_1,\ldots,x_n)\in\{0,1\}^n} g_\phi(x_1, \ldots, x_n) = \gamma_0$;
2 **for** $i = 1$ *to* $n-1$ **do**
3 Prover needs to prove statement Stmt_{i-1}. It sends a *univariate* polynomial g_i of degree at most m.
4 Verifier checks that g_i has degree at most m and $g_i(0) + g_i(1) = \gamma_{i-1}$.
5 **if** *check passes* **then**
6 verifier samples a random $\theta_i \leftarrow \mathbb{Z}_q$, sets $\gamma_i = g_i(\theta_i)$ and sends θ_i to the prover. The statement Stmt_i for the next round is $\sum_{(x_{i+1},\ldots,x_n)\in\{0,1\}^{n-i}} g_\phi(\theta_1, \ldots, \theta_i, x_{i+1}, \ldots, x_n) = \gamma_i$.
7 **else**
8 Verifier rejects and outputs 0;
9 Finally, verifier checks $g_\phi(\theta_1, \ldots, \theta_{n-1}, 0) + g_\phi(\theta_1, \ldots, \theta_{n-1}, 1) = \gamma_{n-1}$ and outputs 1 if this check passes, else outputs 0.

A few points to note before we discuss the protocol details:



- This is a 'public coins' protocol. Note that the verifier's messages $\{\theta_i\}_{i\in[n-1]}$ are all random numbers sampled from $\mathbb{Z}_q$. Other than this, the verifier uses no other randomness.
- In our protocol description, the verifier also sends θ_{n-1} in the last round. However, this is not needed.
- The verifier's work mainly consists of evaluating polynomials of degree at most m, and finally, in the last step, the verifier needs to compute $g_\phi(\theta_1, \ldots, \theta_{n-1}, b)$ for $b \in \{0,1\}$. The 3CNF formula ϕ can only be evaluated on bit strings, but g_ϕ can be evaluated on any $z \in \mathbb{Z}_q^n$. Also, note that the verifier does not need to 'open up' the n-variate polynomial. For instance, the polynomial corresponding to $\phi(x_1, x_2, x_3) = (x_1 \vee x_2) \wedge (\neg x_1 \vee x_3)$ is

$(1-(1-x_1)\cdot(1-x_2))\cdot(1-x_1\cdot(1-x_3))$. However, the verifer should not express this as a sum of monomials (there can be exponentially many monomials). Instead, the verifier substitutes values for $x_1,\ldots,x_n$ in the above expression.

The main step that is left to describe is Step 3 of the protocol.

Step 3 of the Sum-Check Protocol

In the i^{th} iteration of the protocol, the prover needs to prove statement

$$\mathsf{Stmt}_{i-1}: \sum_{x_i\in\{0,1\}} \sum_{(x_{i+1},\ldots,x_n)\in\{0,1\}^{n-i}} g_\phi(\theta_1,\ldots,\theta_{i-1},x_i,\ldots,x_n) = \gamma_{i-1}$$

Note that the prover has $g_1,\theta_1,\ldots,g_{i-1},\theta_{i-1}$ from the previous rounds, and can compute γ_{i-1} given θ_{i-1}.

Consider the univariate polynomial

$$g_i(X) = \sum_{(x_{i+1},\ldots,x_n)\in\{0,1\}^{n-i}} g_\phi(\theta_1,\ldots,\theta_{i-1},X,x_{i+1},\ldots,x_n)$$

This polynomial has degree at most $3m$ (since the overall degree of g_ϕ is $3m$, the degree of any variable is also at most $3m$). Moreover, if statement Stmt_{i-1} is true, then $g_i(0)+g_i(1)=\gamma_{i-1}$. Hence the verifier's check in Step 4 passes.

From the definition of g_i, it follows that for all $\theta\in\mathbb{Z}_q$,

$$g_i(\theta) = \sum_{(x_{i+1},\ldots,x_n)\in\{0,1\}^{n-i}} g_\phi(\theta_1,\ldots,\theta_{i-1},\theta,x_{i+1},\ldots,x_n)$$

Therefore statement Stmt_i (which is defined using a random $\theta_i\leftarrow\mathbb{Z}_q$) is true.

Soundness Error

Before we analyze the behavior of malicious provers, let us see where the honest prover gets stuck if we try to prove an incorrect statement. Let us consider the formula in Example 4.1. There are four satisfying assignments for $\phi(x_1,x_2,x_3)=(x_1\vee x_2)\wedge(\neg x_1\vee x_3)$. Suppose we try to prove the false statement $\mathsf{Stmt}_0: \sum_{(x_1,x_2,x_3)\in\{0,1\}^3} g_\phi(x_1,x_2,x_3)=3$, and suppose the prime $q=43$.

Nothing special about 43, picked an arbitrary large prime.

In the first iteration, the prover sends the univariate polynomial g_1 obtained by summing over x_2 and x_3. This polynomial is

$$g_1(x) = 2+x-x^2$$

and recall $\gamma_0=3$. In the first round itself, the verifier can realize that this statement is incorrect, since $g_1(0)+g_1(1)=4\neq\gamma_0$.

Suppose that the prover is malicious and sends $g_1'(x)=2-x^2$. Now, when the verifier checks $g_1'(0)+g_1'(1)=\gamma_0$, the first check passes. The verifier then picks a random number θ_1, say 5. The statement for the next round is

$$\mathsf{Stmt}_1: \sum_{(x_2,x_3)\in\{0,1\}^2} g_\phi(5,x_2,x_3) = g_1'(5) = -23 = 20$$

If the prover is honest in the next round, then it must send

$$g_2(x) = 12x - 15$$

However, with this polynomial, the prover will get caught, since $g_2(0) + g_2(1) \neq 20$. Therefore, the prover must again lie. Suppose it sends $g_2'(x) = 2x + 9$. Now, check that $g_2'(0) + g_2'(1) = 20$. The verifier again chooses a random number θ_2, say 3, and checks if $g_\phi(5,3,0) + g_\phi(5,3,1) = g_2'(3)$. At this point, the prover is caught, since the LHS is 21 and the RHS is 15.

Exercise 4.1. *In the sum-check protocol, it is essential that in the i^{th} round, θ_i is sent to the prover **after** the prover sends g_i. What would happen if θ_i is sent before the prover sends g_i?*

Exercise 4.2. *In the sum-check protocol, it is essential that the verifier checks the degree bound on each g_i sent by the prover. Show that the prover can cheat successfully if this check is removed.*

Formal proof of soundness The formal proof of soundness relies on a simple mathematical fact: any degree d polynomial has at most d roots. A useful corollary of this simple fact: any two distinct degree d polynomials can agree on at most d points. As a result, if we have two distinct polynomials, and we pick a random integer $\theta \leftarrow \mathbb{Z}_q$, the two polynomials will not agree at θ with probability at least $1 - d/q$.

In class, we saw a slightly stronger bound: for a false statement, the verifier outputs 0 with probability at least $\left(1 - \frac{d}{q}\right)^n$.

Lemma 4.3. *Suppose $g_\phi \in \mathbb{Z}_q[x_1, \ldots, x_n]$ is a degree d polynomial (over n variables), and the statement $\mathsf{Stmt}_0 : \sum_{(x_1,\ldots,x_n)\in\{0,1\}^n} g_\phi(x_1, \ldots, x_n) = \gamma_0$ is false. Then*

$$\Pr\left[\mathbf{V} \textit{ outputs } 1 \textit{ at end of } n \textit{ rounds }\right] \leq \frac{nd}{q}$$

Lecture 5: January 17th, 2023

Sketch of Proof.

If the verifier outputs 1 at the end of the protocol, Stmt_{n-1} must be a true statement. Since we started with a false statement Stmt_0, there exists some index i such that Stmt_{i-1} was a false statement, but Stmt_i was a true statement. Let g_i be the 'correct' polynomial corresponding to Stmt_{i-1}, and g_i' the polynomial sent by prover in round i, and let θ_i be the random integer chosen by the verifier in round i. If g_i and g_i' are identical polynomials, then the verifier outputs 0 in round i. This is because Stmt_{i-1} is incorrect, and $g_i(0) + g_i(1) \neq \gamma_{i-1}$.

Therefore, in order to succeed, the prover must send g_i' such that $g_i'(0) + g_i'(1) = \gamma_{i-1}$. Since g_i and g_i' are of degree at most m, the random integer θ_i satisfies $g_i(\theta_i) = g_i'(\theta_i)$ with probability at most d/q. If $g_i(\theta_i) \neq g_i'(\theta_i)$, then Stmt_i is a false statement. Therefore, for each i,

$$\Pr_{\theta_i}\left[\begin{array}{c}\mathsf{Stmt}_{i-1} \text{ is false} \\ \wedge \text{ Check in Step 4 passes} \\ \wedge\ \mathsf{Stmt}_i \text{ is true}\end{array}\right] \leq \frac{d}{q}$$

□

5 Zero Knowledge Proofs

In the last section, we saw that interaction and randomness can be useful for verifying computation efficiently. In this section, we will add a 'security requirement' to interactive proofs. What if we want the computation to be efficiently verifiable, but the verifier must not learn any 'secrets' possessed by the prover? Let us take the following scenario (this is a simplification of a common real-world setting). A client is interacting with a server, and every time the client accesses to the server, it needs to 'prove' its digital identity. Here is one way this is done in practice: the client has a `password`, and the server stores a hash of `password`. Every time the client interacts with the server, it sends `password`, and the server can compute the hash, and be convinced that it is interacting with the correct client. However, this is clearly insecure — an eavesdropping attacker can learn `password`, and therefore impersonate the client. This brings us to the following question:

How can the client 'prove' its identity to the server without revealing `password`*?*

Note that the above statement is an NP statement. Abstractly, we are looking for protocols where a prover (having a witness for an instance) can prove to a verifier that it has a witness, without revealing the witness. Before we get to proving possession of a witness, we will discuss protocols for proving that there *exists* a witness (without revealing the witness). Even this notion has a lot of applications in cryptography, in scenarios where we have multiple interacting parties, and we want to enforce honest behavior.

5.1 *Defining the 'Zero Knowledge' Property*

Let L be an NP language. We want a protocol with the following informal properties:

- completeness: if x is a 'yes' instance, then the verifier should accept with high probability.
- soundness: if x is a 'no' instance, then the verifier should reject with high probability.
- zero knowledgeness: at the end of the protocol, the verifier should not 'learn' anything.

We encountered the first two properties when we discussed interactive proofs, but how to formally capture that the verifier learns nothing at the end of the protocol? There are multiple ways of doing this, and zero-knowledgeness is one of the strongest definitions for this.

First, we define the 'view' of a verifier in a protocol. Note, a protocol defines the behavior of an honest prover and an honest verifier. However, either of these parties can deviate arbitrarily. In the definition of a verifier's view, we are considering honest provers who interact with possibly malicious verifiers. This will be needed for the definition of zero knowledgeness.

Definition 5.1 (View of the verifier)**.** *Let* $(\mathbf{P}, \mathbf{V})$ *be an interactive proof system for language* $\mathsf{L} = (\mathsf{L}_{\text{yes}}, \mathsf{L}_{\text{no}})$*. For any probabilistic polynomial time verifier* $\mathbf{V}^*$*, the view of* $\mathbf{V}^*$ *on input* x *in the protocol, denoted by* $\mathsf{view}\langle \mathbf{P}, \mathbf{V}^* \rangle(x)$ *consists of all the messages exchanged between* $\mathbf{P}$ *and* $\mathbf{V}^*$*, together with the randomness used by* $\mathbf{V}^*$*.* ◊

We are now ready to define the zero knowledge property. Intuitively, it says that the adversarial verifier's view in the real-world (when it interacts with a prover) can be simulated efficiently, without interacting with the prover.

Definition 5.2 (Perfect/Statistical/Computational Zero knowledge)**.** *Let* $(\mathbf{P}, \mathbf{V})$ *be an interactive proof system for language* $\mathsf{L} = (\mathsf{L}_{\text{yes}}, \mathsf{L}_{\text{no}})$*. We say that the proof system satisfies perfect/statistical/computational zero knowledge if for every (possibly malicious) probabilistic polynomial time verifier* $\mathbf{V}^*$*, there exists a probabilistic polynomial time simulator* $\mathbf{S}$ *with the following properties:*

- *for every* $x \in \mathsf{L}_{\text{yes}}$*,* $\mathbf{S}$ *outputs either* $\perp$ *(indicating that it could not simulate), or a string in* $\{0,1\}^*$*. Moreover, there exists a negligible function* $\mu(\cdot)$ *such that for every* $x \in \mathsf{L}_{\text{yes}}$*,* $\Pr[\mathbf{S} \text{ outputs } \perp] \leq \mu(|x|)$*.*
- *For every* $x \in \mathsf{L}_{\text{yes}}$*, the following distributions are indistinguishable:*

  $$\mathcal{D}_1 : \{\mathsf{view}\langle \mathbf{P}, \mathbf{V}^* \rangle(x)\}$$

  $$\mathcal{D}_2 : \{\mathbf{S}(x) \mid \mathbf{S}(x) \neq \perp\}$$

  If the two distributions are identical, then we say that the interactive proof is perfect zero knowledge. If the two distributions are statistically indistinguishable, then the zero knowledge property is statistical, else it is computational zero knowledge.

  In distribution $\mathcal{D}_2$*, we are considering the simulator's output conditioned on it being non-*$\perp$*.*

◊

Let us break down this definition. When a malicious verifier is participating in a protocol, it sees the messages sent by the prover, and it can see its own randomness. This is what constitutes the verifier's 'knowledge' in the real world. If a protocol is zero knowledge, then instead of interacting with the prover, the malicious verifier can simply use the simulator. The real world distribution is identical/statistically indistinguishable/computationally indistinguishable from the simulated distribution.

In COL759, we often described security in terms of a game. The same can be done here as well. Let us take computational zero knowledge. It can be described via a game between a challenger and an adversary. The adversary first sends a string $x \in \mathsf{L}_{\text{yes}}$ and a description of malicious verifier $\mathbf{V}^*$. The challenger then tosses a coin $b \leftarrow \{0,1\}$. If $b = 0$, it runs an interaction between $\mathbf{P}$ and $\mathbf{V}^*$ and sends $\mathsf{view}\langle \mathbf{P}, \mathbf{V}^* \rangle(x)$ to the adversary. If $b = 1$, it sends $\mathbf{S}(x)$ to the adversary. The adversary must guess b.

The definition also allows the simulator to output $\perp$ *with negligible probability. Strictly speaking, this is only needed for the definition of perfect ZK. For the other two, this 'negligible' term gets absorbed in the statistical/-computational indistinguishability guarantee.*

Why this definition captures our intuition for zero knowledge: The above is just a formal definition. Let us consider a couple of 'attack scenarios' where the verifier learns something non-trivial about the statement being proven, and let us see if a zero knowledge protocol prevents these attack scenarios.

Verifier learning the witness of an NP complete problem: Suppose we have an interactive protocol between a prover and a verifier for 3SAT, such that the verifier, for every satisfiable 3SAT instance ϕ, can learn a satisfying witness for ϕ after the protocol. Then this protocol is not zero-knowledge (unless $\mathsf{NP} \subseteq \mathsf{BPP}$).

Verifier trying to 'reproduce' a zero knowledge proof after completing its interaction with prover: Let x be an instance of an NP-complete language L, and let $(\mathbf{P}, \mathbf{V})$ be an interactive protocol for L. A malicious verifier $\mathbf{V}^*$ interacts with a prover on input x, and records its view after the interaction. Can it use this view to prove to someone else that $x \in \mathsf{L}_{\text{yes}}$? If the protocol is zero knowledge, then $\mathbf{V}^*$ cannot do this for every instance of L (unless $\mathsf{NP} \subseteq \mathsf{BPP}$).

There was a question in class: how can we show that a protocol is not zero knowledge? We don't have a characterization of protocols that are not zero-knowledge. However, if there is some information that a real-world interaction will provide which cannot be simulated efficiently, then this shows that the protocol is not zero-knowledge.

Exercise 5.1. *Consider the following interactive protocol for an* NP *language* L*. Given an instance x, the prover first sends the number of witnesses for x, and then runs the sum-check protocol to prove the above. Show that a malicious verifier can learn a witness for x by interacting with the prover (and therefore this protocol is not zero-knowledge).*

5.2 *Lower Bound for Zero Knowledge Proofs*

In this section, we will show that both interaction and randomness are essential for zero knowledge protocols.

Interaction is necessary First, let us start with interaction. Is it possible to have a *truly* non-interactive zero-knowledge protocol for NP complete languages? Note that if the zero-knowledge requirement was not there, then there exists a simple non-interactive protocol : the prover simply sends a witness w that certifies $x \in \mathsf{L}_{\text{yes}}$.

Lemma 5.3. *Let $(\mathbf{P}, \mathbf{V})$ be a zero knowledge proof system for language L with completeness c and soundness error s. Moreover, this zero knowledge protocol is truly non-interactive: the prover, on receiving the instance x, outputs a proof π and there is no further communication between the prover and verifier. Then $\mathsf{L} \in \mathsf{BPP}$.*

For yes instances, the verifier outputs 1 with probability at least c, and for no instances it outputs 1 with probability at most s.

Proof. Let $\mathbf{S}$ be the simulator corresponding to the honest verifier $\mathbf{V}$, and let $x \in \mathsf{L}_{\text{yes}} \cup \mathsf{L}_{\text{no}}$. We want to use $\mathbf{S}$ to decide whether $x \in \mathsf{L}_{\text{yes}}$ or $x \in \mathsf{L}_{\text{no}}$. Our probabilistic algorithm $\mathcal{B}$ for deciding x works as follows:

1. It first runs the simulator and computes $(\pi, r) \leftarrow \mathbf{S}(x)$.
2. Next, it samples a uniformly random r' and runs $\mathbf{V}(x, \pi; r')$ (using r' as randomness). If the output is 1, it concludes that $x \in \mathsf{L}_{\text{yes}}$, else it concludes that $x \in \mathsf{L}_{\text{no}}$.

Claim 5.4. *For any $x \in \mathsf{L}_{\text{yes}}$, the above algorithm outputs 1 with probability at least $c - \text{negl}$.*

Proof. Suppose, on the contrary, $\mathcal{B}$ outputs 1 with probability at most $c - \epsilon$ for some non-negligible function ϵ. Then there exists a p.p.t. distinguisher that breaks the zero-knowledge property with advantage ϵ.

In the real-world, the prover's message π is accepted with probability at least c. Therefore, if the simulated π is accepted with probability at most $c - \epsilon$, then this can be used to distinguish the real-world from the simulated world.

□

Claim 5.5. *For any $x \in \mathsf{L}_{\text{no}}$, the above algorithm outputs 1 with probability at most s.*

Proof. Suppose $\mathsf{Pr}_{r'}[\mathbf{V}(x, \pi; r') : (\pi, r) \leftarrow \mathbf{S}(x)] > s$. Then there exists a prover that breaks soundness with probability strictly greater than s. The prover computes $(\pi, r) \leftarrow \mathbf{S}(x)$ and sends π. □

□

Exercise 5.2. *In the proof of Lemma 5.3, the simulator also outputs randomness r. However the algorithm $\mathcal{B}$ sampled its own randomness. Why?*

The reader is encouraged to try and extend this proof for two-message protocols: the verifier sends a string msg_1, and the prover sends msg_2. Can we have zero-knowledge protocols for hard languages with just two messages?

Verifier Randomness is Necessary Next we show that it is necessary to have verifier's randomness. This one is fairly simple (similar to why we require verifier's randomness for nontrivial interactive proofs).

Observation 5.6. *Let $(\mathbf{P}, \mathbf{V})$ be a zero knowledge proof system for language L with completeness c and soundness error s. Moreover, the verifier is deterministic. Then $\mathsf{L} \in \mathsf{BPP}$.*

Proof. This can be reduced to Lemma 5.3. Since the verifier is deterministic, the prover can compute the verifier's messages. Therefore, it becomes a truly non-interactive protocol, and therefore Lemma 5.3 applies here. □

5.3 *Characterizing the Verifier, Prover and Simulator*

Lecture 6: January 20th, 2023

The prover: So far, in our definition of zero-knowledge protocols, we have not placed any restrictions on the prover (that is, the prover is allowed to run in unbounded time).

The verifier: In this part of the course, the verifier is restricted to probabilistic polynomial time. Suppose we have a $2k - 1$ message protocol and the prover sends the first message. We will think of the verifier as consisting of k Turing machines $\mathbf{V} = (\mathbf{V}_2, \ldots, \mathbf{V}_{2k})$ where the i^{th} verifier machine $\mathbf{V}_{2i}$ takes as input the message m_{2i-1} (coming from the prover), state st_{2i-2} (coming from $\mathbf{V}_{2i-2}$), randomness r_{2i} and outputs the message msg_{2i} (for the prover) and state st_{2i} (for the next verifier $\mathbf{V}_{2i+2}$). The first verifier machine $\mathbf{V}_2$ also gets input x, and the last verifier machine outputs the final decision.

The number of messages in a protocol need not be odd. However, the prover always sends the last message and the verifier accepts/rejects.

In the zero-knowledge property, we consider malicious p.p.t. verifiers. Again, we assume $\mathbf{V}^* = \left(\mathbf{V}_2^*, \mathbf{V}_4^*, \ldots, \mathbf{V}_{2k}^*\right)$, but we will not assume anything about the internal workings of $\mathbf{V}_i^*$. In particular, maybe the malicious verifier takes in randomness as input, but doesn't use it.

The simulator: In most of the zero-knowledge proofs that we will see in this course, the simulator will need only **black-box access** to the malicious verifier. That is, the simulator can feed any inputs/randomness to each of the malicious verifier machines. The simulator can also run the verifier machines with different inputs/randomness and observe/analyze the outputs of the verifier machine. This gives us a notion of zero knowledge proofs where we have one fixed simulator that can simulate the verifier's view by simply interacting with the verifier.

Definition 5.7 (Black-box Zero Knowledgeness)**.** *An interactive proof system* $(\mathbf{P}, \mathbf{V})$ *for language* $\mathsf{L} = (\mathsf{L}_{\text{yes}}, \mathsf{L}_{\text{no}})$ *satisfies statistical (resp. computational)black-box zero knowledge if there exists a p.p.t. simulator* $\mathbf{S}$ *such that for all p.p.t. verifiers* $\mathbf{V}^*$*, for all inputs* $x \in \mathsf{L}_{\text{yes}}$*, the following distributions are statistically (resp. computationally) indistinguishable:*

$$\mathcal{D}_1 : \{\mathsf{view}\langle \mathbf{P}, \mathbf{V}^* \rangle(x)\}$$

$$\mathcal{D}_2 : \left\{ \mathbf{S}^{\mathbf{V}^*}(x) \right\}$$

◊

For a long time, all known protocols had black-box simulation. Barak [Bar01] gave the first protocol where the simulator made non-black-box use of the verifier. Barak's protocol was especially influential because it bypasses certain impossibility results for black-box simulation.

5.4 *Zero Knowledge Proof for Graph Isomorphism*

In this section, we will see the first nontrivial zero knowledge protocol. This is for the graph isomorphism problem. The protocol was given by Goldreich, Micali and Wigderson [GMW91], who also showed (in the same paper) a zero-knowledge protocol for all NP languages.

Recall, the 'yes' instances here are pairs of graphs $G_0 = (V, E_0)$ and $G_1 = (V, E_1)$ such that there exists a permutation $\sigma : V \to V$ such that $(u, v) \in E_0 \iff (\sigma(u), \sigma(v)) \in E_1$.

Common Input: graphs $G_0 = (V, E_0)$, $G_1 = (V, E_1)$.
Prover's Private Input: Isomorphism $\sigma \in S_n$ such $G_1 = \sigma(G_0)$. [a]

Claim to prove: G_0 and G_1 are isomorphic.

Protocol 4: Graph Isomorphism: GMW Protocol [GMW91]

1. $\mathbf{P}_1$: Prover picks a uniformly random permutation $\pi \leftarrow S_n$, computes $H = \pi(G_1)$. It sends H to the verifier.
2. $\mathbf{V}_2$: Verifier sends a uniformly random bit $b \leftarrow \{0, 1\}$.
3. $\mathbf{P}_3$: If $b = 0$, prover sends $\tau = \pi \circ \sigma$, else it sends $\tau = \pi$.
4. $\mathbf{V}_4$: Verifier outputs 0 if $\tau(G_b) \neq H$.

[a] The set of all permutations over $[n]$ is denoted by S_n. For a graph $G = (V, E)$ and a permutation π, let $\pi(G)$ denote the graph obtained by mapping label i to $\pi(i)$.

Clearly, if $G_0 \cong G_1$, then the verifier outputs 1 with probability 1. If $G_0 \ncong G_1$, then the malicious prover can succeed with probability at most 1/2 (since $G_0 \ncong G_1$, the graph H is isomorphic to at most one of the two input graphs).

Zero Knowledge Property. We will show that this protocol satisfies black-box simulation. The simulator $\mathbf{S}$ is defined as follows. Let $\mathbf{V}^* = (\mathbf{V}_2^*, \mathbf{V}_4^*)$ be any p.p.t. malicious verifier for the graph isomorphism protocol, and assume it takes t bits of randomness.

Simulator:

1 **for** $i = 1$ *to* n **do**
2 Simulator $\mathbf{S}$ picks a uniformly random bit $b_i \leftarrow \{0,1\}$, uniformly random permutation $\tau_i \leftarrow S_n$ and $r_{2,i} \leftarrow \{0,1\}^t$.
3 Let $H_i = \tau_i(G_{b_i})$. It computes $b' = \mathbf{V}_2^*\left(x = (G_0, G_1), H_i, r_{2,i}\right)$.
4 If $b_i = b'$, it sets $b = b_i$, $\tau = \tau_i$, $H = H_i$, $r_2 = r_{2,i}$, `success = true` and quits the loop.
5 If `success` $\neq$ `true`, $\mathbf{S}$ outputs $\perp$. Else $\mathbf{S}$ picks a uniformly random string r_4 and outputs $((H, b, \tau), (r_2, r_4))$.

The proof of zero-knowledge property relies on the following observation (which we also used for our graph non-isomorphism protocol in Section 3.2).

Observation 5.8. *Let G_0 and G_1 be two isomorphic graphs. Then the following distributions are identical:*

$$\mathcal{D}_0 = \left\{ H : \begin{array}{c} \pi \leftarrow S_n \\ H = \pi(G_0) \end{array} \right\} \qquad \mathcal{D}_1 = \left\{ H : \begin{array}{c} b \leftarrow \{0,1\} \\ \pi \leftarrow S_n \\ H = \pi(G_b) \end{array} \right\}$$

Claim 5.9. *In the above protocol, the simulator outputs $\perp$ with probability at most $1/2^n$. Conditioned on the simulator not outputting $\perp$, the simulator's output distribution is identical to the honest prover's output distribution.*

The real-world represents the case where $\mathbf{V}^*$ interacts with $\mathbf{P}$, while the ideal-world is the output of the simulator.

Proof. The view consists of a graph H, bit b, permutation τ and strings r_2, r_4. For any (H, b, τ, r_2, r_4), let $p_{\text{real}}(H, b, \tau, r_2, r_4)$ (resp. $p_{\text{ideal}}(H, b, \tau, r_2, r_4)$) denote the probability of this view in the real (resp. ideal) world. All the terms below are with respect to common input $x = (G_0, G_1)$, and prover's witness σ. The following are some immediate observations:

1. if $H \notin \text{iso}(G_1)$, then $p_{\text{real}}(H, b, \tau, r_2, r_4) = p_{\text{ideal}}(H, b, \tau, r_2, r_4) = 0$.
2. For any $b \in \{0,1\}$ and $H \in \text{iso}(G_1)$, $p_{\text{real}}(H, b, \tau, r_2, r_4) = 0$ if $\tau(G_b) \neq H$.

Next, let us compute $p_{\perp,i}$, the probability that during simulation, `success` $\neq$ `true` after i iterations.

$$
\begin{aligned}
p_{\perp,i} &= \mathsf{Pr}_{\{b_j,\tau_j,r_{2,j}\}_{j\le i}}\left[\begin{array}{c}\texttt{success} \neq \texttt{true} \text{ after } i-1 \text{ iterations} \\ \wedge\, b_i \neq \mathbf{V}^*(x,H_i,r_{2,i})\end{array}\right] \\
&= \mathsf{Pr}_{\{b_j,\tau_j,r_{2,j}\}_{j<i}}\left[\begin{array}{c}\texttt{success} \neq \texttt{true} \text{ after } i-1 \text{ iterations}\end{array}\right] \\
&\quad \cdot \mathsf{Pr}_{b_i,\tau_i,r_{2,i}}\left[\begin{array}{c} b_i \neq \mathbf{V}^*(x,H_i,r_{2,i})\end{array}\right] \\
&= \left(p_{\perp,i-1}\right)\cdot\left(\frac{1}{2}\right)
\end{aligned}
$$

Therefore, the simulator outputs $\perp$ with probability $1/2^n$.

Someone please check these calculations, and let me know if something is incorrect or needs further explanation.

Next, we will show that the verifier's view in the real and ideal worlds are identical. First, we compute $p_{\text{real}}(H,b,\tau,r_2,r_4)$.

$$
p_{\text{real}}(H,b,\tau,r_2,r_4) = \begin{cases} \frac{1}{n!\cdot 2^{2t}} \text{ if } H=\tau(G_b), b=\mathbf{V}_2^*(x,H,r_2) \\ 0 \text{ otherwise}\end{cases}
$$

Next, we compute $p_{\text{ideal}}(H,b,\tau,r_2,r_4)$. We can break down p_{ideal} into n summands, depending on the iteration in which `success` is set to `true`.

$$
p_{\text{ideal}}(H,b,\tau,r_2,r_4) = \frac{\sum_{j=1}^{n} \mathsf{Pr}_{\{(\tau_i,b_i,r_{2,i},r_{4,i})\}_{i\le j}}\left[\begin{array}{c}\texttt{success = true} \text{ in } i^{\text{th}} \text{ iteration} \\ H=\tau_i(G_{b_i}), b=\mathbf{V}_2^*(x,H,r_{2,i}) \\ \tau=\tau_i, r_2=r_{2,i}, r_4=r_{4,i}\end{array}\right]}{1-1/2^n}
$$

Conditioned on $\tau_i, r_{2,i}, r_{4,i}$, there is at most one value of b_i such that $H=\tau_i\left(G_{b_i}\right)$, and hence

$$
\mathsf{Pr}_{\{(\tau_i,b_i,r_{2,i},r_{4,i})\}_{i\le j}}\left[\begin{array}{c}\texttt{success = true} \text{ in } i^{\text{th}} \text{ iteration} \\ H=\tau_i(G_{b_i}), b=\mathbf{V}_2^*(x,H,r_{2,i}) \\ \tau=\tau_i, r_2=r_{2,i}, r_4=r_{4,i}\end{array}\right] = \frac{1}{n!\cdot 2^{2t+i}}
$$

This concludes our proof.

□

Exercise 5.3. *Here are a few modifications to the above simulator. Check whether the resulting simulator suffices for the zero knowledge proof.*

- *Simulator always chooses $b_i = 1$.*
- *Simulator sets $b_i = 1$ with probability $1/3$, and 0 with probability $2/3$.*
- *Before the start of first iteration, simulator picks $\sigma \leftarrow S_n$, and uses the same σ for all n iterations (that is, $\sigma_i = \sigma$ for all $i \in [n]$).*
- *In the last step, simulator picks r_2 and r_4, which it includes in the final output. (In the above simulator's description, r_2 is obtained from the 'for' loop).*

Exercise 5.4. *Modify the GMW graph-isomorphism protocol as follows: The prover picks a random permutation π (same as the given protocol). It sends $H = \pi(G_1)$, together with a commitment to $\pi \circ \sigma$ and π. The verifier sends a bit b. If $b = 0$, the prover provides an opening to the first commitment, else it sends an opening to the second commitment.*

Show that this protocol satisfies computational zero knowledge.

5.5 *Composition of Zero-Knowledge Proofs*

In the graph isomorphism protocol described above, we saw that sequential repetition reduces the soundness error to negligible, while preserving zero-knowledgeness. This, however, does not hold in general. One can construct contrived 'base' protocols satisfying computational zero knowledge where repeating the protocol twice violates the zero-knowledge property. This was formally shown by Goldreich and Krawczyk [GK96b]. Here is a high-level description of the contrived protocol: the verifier first sends a random string. The prover checks whether this is an encryption of a number k such that the string "`hello`", when iteratively hashed k times, results in a string that starts with x (the common instance). If so, it outputs the witness (and protocol ends). Otherwise, they execute the 'base' zero knowledge protocol. Finally, the prover sends an encryption of the smallest k such that the string "`hello`", when iteratively hashed k times, results in a string that starts with x.

Inspiration comes from blockchains :)

The encryption at the end ensures that if the base protocol is zero knowledge, then (hopefully) so is the new protocol. However, if the new protocol is repeated twice, then there exists a malicious verifier that can learn the witness!

I have not checked formally if the above is a zero knowledge protocol. However, similar ideas were used by [GK96b] for showing that sequential composition doesn't always preserve zero-knowledgeness.

The issue here is that our definition of zero knowledge (Definition 5.2) is not strong enough to guarantee security when the zero knowledge protocol is used as a building block inside a bigger application/protocol. For instance, maybe the adversary has some auxiliary information about the NP statement. A good definition for zero knowledge would guarantee that the adversary does not learn anything new (other than the auxiliary information which it already had). This brings us to the following definition of zero-knowledgeness. For simplicity, we will only define it for statistical/computational zero knowledgeness.

Definition 5.10 (Statistical/Computational Zero knowledge with Auxiliary Information)**.** *Let $(\mathbf{P}, \mathbf{V})$ be an interactive proof system for language $\mathsf{L} = (\mathsf{L}_{\text{yes}}, \mathsf{L}_{\text{no}})$. We say that the proof system satisfies statistical/computational zero knowledge if for every (possibly malicious) probabilistic polynomial time verifier $\mathbf{V}^*$, there exists a probabilistic polynomial time simulator $\mathbf{S}$ such that for every $x \in \mathsf{L}_{\text{yes}}$ and every string $z \in \{0,1\}^*$, the following distributions are statistically/computationally indistinguishable:*

$$\mathcal{D}_1 : \{\mathsf{view}\langle \mathbf{P}, \mathbf{V}^*(z)\rangle(x)\}$$

$$\mathcal{D}_2 : \{\mathbf{S}(x, z)\}$$

$\Diamond$

Check that the contrived protocol described above does not work, since one can set z to be the encryption of the appropriate integer k. This definition of zero

knowledge is good enough for supporting sequential composition, as stated in the lemma below.

This notion of security is also *closed under sequential composition*. Formally, this is captured by the following lemma.

Lemma 5.11. *Let* $(\mathbf{P}, \mathbf{V})$ *be an interactive protocol with* r *messages, completeness* 1, *soundness* $1/2$ *and satisfying zero-knowledgeness w.r.t. auxiliary information. Then, repeating this protocol sequentially* n *times results in an interactive protocol with* $n \cdot r$ *messages, completeness* 1, *soundness* $1/2^n$ *and satisfying zero-knowledgeness w.r.t. auxiliary information.*

The proof follows via a careful hybrid argument (it is not very complicated, but has lots of cumbersome details). See [Gol01], Lemma 4.3.11 for a detailed proof.

If anyone is interested in writing up a simplified version of Goldreich's proof, please let me know. I'll be happy to assign 5 scribe-marks for the same.

5.6 *Zero Knowledge Proofs for* NP

Lecture 7: January 24th, 2023

In the last section, we saw a zero-knowledge proof for the graph-isomorphism problem. This is a *non-trivial* protocol since we don't know if GraphIso $\in$ BPP. However, it is unlikely that GraphIso is NP-complete. In this section, we will present a zero-knowledge protocol for GraphHam, which is an NP-complete problem. Unlike the previous protocol, this one satisfies computational zero knowledge (recall, the GMW protocol satisfies perfect zero knowledge). In hindsight, the need for cryptographic assumptions is not surprising. Fortnow [For87] showed that if a language has a perfect zero-knowledge protocol, then it cannot be NP-complete (unless something very unexpected happens in the complexity world). Later works [AH91] extended this to show that NP-complete languages cannot have even statistical zero-knowledge protocols.

The first zero-knowledge protocol for NP-complete languages was given by Goldreich, Micali and Widgerson [GMW91]. Their protocol was for the 3COL problem, and used cryptographic (stat. binding, comp. hiding) commitments. Here, we will present a protocol for GraphHam proposed by Blum [Blu86]. Structurally, this protocol is similar to the GMW protocol for 3COL. It also uses (stat. bind, comp. hiding) commitments, and is described below.

Common Input: graphs $G = (V, E)$
Prover's Private Input: Hamiltonian cycle $(v_1, \ldots, v_n)$ such that $\forall i \in [n-1]$, $(v_i, v_{i+1}) \in E$ and $(v_n, v_1) \in E$.

Claim to prove: G has a Hamiltonian cycle.

Protocol 6: Graph Hamiltonicity: Blum's Protocol [Blu86]

1 $\mathbf{P}_1$: Prover picks a uniformly random permutation $\pi \leftarrow S_n$, computes $H = \pi(G)$. Let $\mathbf{A}_H$ denote the adjacency matrix corresponding to H. The prover chooses randomness $r_{i,j}$ and computes commitment $c_{i,j} = \mathsf{Commit}(\mathbf{A}_H[i,j]; r_{i,j})$ for all $i, j \in [n]$. Prover sends $\{c_{i,j}\}_{i,j\in[n]}$

2 $\mathbf{V}_2$: Verifier sends a uniformly random bit $b \leftarrow \{0,1\}$.

3 $\mathbf{P}_3$: Prover's response depends on the bit b. If $b = 0$, prover sends π together with $\{r_{i,j}\}_{i,j\in[n]}$. Else, the prover sends $(\pi(v_1), \ldots, \pi(v_n))$ together with the randomness corresponding to these edges. That is, it sends $\left(r_{\pi(v_1),\pi(v_2)}, \ldots, r_{\pi(v_n),\pi(v_1)}\right)$.

4 $\mathbf{V}_4$: **if** $b = 0$ **then**

5 the verifier gets a permutation π and n^2 strings $\{r_{i,j}\}_{i,j}$. It computes the adjacency matrix A_H of $H = \pi(G)$. Next, it checks that $c_{i,j} = \mathsf{Commit}(A_H[i,j]; r_{i,j})$ for all $i, j \in [n]$

6 **else**

7 the verifier gets a cycle $(w_1, w_2, \ldots, w_n)$ and n strings $(r_1, \ldots, r_n)$. It checks that for all $i < n$, $c_{w_i,w_{i+1}} = \mathsf{Commit}(1; r_i)$. Finally, it checks that $c_{w_n,w_1} = \mathsf{Commit}(1; r_n)$.

This protocol has perfect completeness. For soundness, note that if the commitment scheme is statistically binding, then the prover will fail in one of the two cases. Hence the soundness error is at most 1/2 (and this can again be reduced to negl using sequential repetition).

Exercise 5.5. *Show that if the commitment scheme is only computationally binding, then an unbounded cheating prover can break soundness of the above protocol.*

Zero-knowledge property: The simulator's description is very similar to the simulator for the graph isomorphism protocol (Protocol 4).

Simulator:

for $k = 1$ *to* n **do**
  Simulator **S** picks a uniformly random bit $b \leftarrow \{0,1\}$, uniformly random permutation $\pi \leftarrow S_n$ and $r_2 \leftarrow \{0,1\}^t$. If $b = 0$, the simulator computes $H = \pi(G)$, $c_{i,j} = \mathsf{Commit}(A_H[i,j]; r_{i,j})$. Else, it computes $c_{i,j} = \mathsf{Commit}(1; r_{i,j})$ for all i, j.
  It computes $b' = \mathbf{V}_2^*\left(x = G, \left\{c_{i,j}\right\}, r_2\right)$.
  If $b = b'$, it sets `success = true` and quits the loop.
If `success` $\neq$ `true`, **S** outputs $\perp$. Else, if $b = 0$, the simulator sets $\mathsf{op} = \left(\pi, \left(r_{i,j}\right)\right)$. If $b = 1$, it chooses a random cycle $(w_1, w_2, \ldots, w_n)$, sets $\mathsf{op} = \left(\left(w_i\right), \left(r_{w_i,w_{i+1}}\right)\right)$. It outputs $\left(\left(\left(c_{i,j}\right), b, \mathsf{op}\right), (r_2, r_4)\right)$, where r_4 is a uniformly random string.

As in our proof for Protocol 4, we need to first prove that **S** outputs $\perp$ with negligible probability. Next, we need to show that conditioned on the output being non-$\perp$, the real and ideal-world distributions are indistinguishable.

Claim 5.12. *Assuming* Commit *is a computationally hiding commitment scheme,* **S** *outputs* $\perp$ *with probability at most* $(2/3)^n$.

Sketch of Proof. Note that it is necessary for the commitment scheme to be secure, otherwise the verifier can learn the bit picked by the simulator, and send the opposite bit as the challenge.

Here, 2/3 is a loose bound, we can use $(1/2 + 1/\mathrm{poly})$. The key idea is that if the output of $\mathbf{V}_2^*$ depends on the bit b chosen by the simulator, then $\mathbf{V}_2^*$ can distinguish between the commitment to A_H and the commitment to $\mathbb{1}^{n\times n}$. Therefore, it can break the hiding property of the commitment scheme. The formal proof will be included in Section 5.13. □

Claim 5.13. *For any p.p.t. verifier* $\mathbf{V}^*$, *input* $x \in \mathsf{L}$ *with witness* w, *and auxiliary information* z, $\mathsf{view}\langle \mathbf{P}(w), \mathbf{V}^*(z)\rangle(x) \approx_c \mathbf{S}^{\mathbf{V}^*}(x, z)$.

This proof also relies on the hiding property of the commitment scheme. However it is more involved than the previous claim. The detailed proof will be included in Section 5.13, here we provide a high-level overview.

Sketch of Proof. The proof uses a neat hybrid structure that uses the following hybrid-simulator:

Hybrid-Simulator: Simulator for Hybrid-world, has witness $\mathbf{w} = (v_1, \ldots, v_n)$

1 **for** $k = 1$ *to* n **do**
2 Hybrid-Simulator **HybS** picks a uniformly random bit $b \leftarrow \{0,1\}$, uniformly random permutation $\pi \leftarrow S_n$ and $r_2 \leftarrow \{0,1\}^t$. The simulator computes $H = \pi(G)$, $c_{i,j} = \mathsf{Commit}(A_H[i,j]; r_{i,j})$.
3 It computes $b' = \mathbf{V}_2^*\left(x = G, \{c_{i,j}\}, r_2\right)$.
4 If $b = b'$, it sets `success = true` and quits the loop.
5 If `success` $\neq$ `true`, **HybS** outputs $\perp$. Else, if $b = 0$, the simulator sets $\mathsf{op} = \left(\pi, \left(r_{i,j}\right)\right)$. If $b = 1$, it uses the witness $(v_1, v_2, \ldots, v_n)$, sets $w_i = \pi(v_i)$, $\mathsf{op} = \left((w_i), \left(r_{w_i, w_{i+1}}\right)\right)$. It outputs $\left(\left(\left(c_{i,j}\right), b, \mathsf{op}\right), (r_2, r_4)\right)$, where r_4 is a uniformly random string.

First, we can show that $\mathbf{HybS}^{\mathbf{V}^*}(x, \mathbf{w})$ is statistically indistinguishable from $\mathsf{view}\langle \mathbf{P}(\mathbf{w}), \mathbf{V}^* \rangle(x)$. This argument is fairly simple (if we ignore the small probability of $\perp$, then these two distributions are identical).

The more interesting step is arguing that $\mathbf{HybS}^{\mathbf{V}^*}(x, \mathbf{w})$ is computationally indistinguishable from $\mathbf{S}^{\mathbf{V}^*}(x)$. If the distribution of verifier's challenge bit b is different in the two scenarios, then we have an attack on the computational hiding of the commitment scheme (similar to proof of Claim 5.12).

Let us assume for simplicity that the distribution of b is identical in both scenarios (say $\mathbf{V}^*$ sends a uniformly random bit b). If $b = 0$, then both $\mathbf{HybS}^{\mathbf{V}^*}(x, \mathbf{w})$ and $\mathbf{S}^{\mathbf{V}^*}(x)$ are identical. The case that remains is for $b = 1$. Suppose there exists a verifier $\mathbf{V}^*$, instance x, witness $\mathbf{w}$ and a distinguisher $\mathcal{D}$ such that $\mathbf{HybS}^{\mathbf{V}^*}(x, \mathbf{w})$ and $\mathbf{S}^{\mathbf{V}^*}(x)$ can be distinguished by $\mathcal{D}$. We will use $\mathcal{D}$, together with x and $\mathbf{w}$ to break the computational hiding of the commitment scheme. The proof details will be available in Section 5.13.

□

5.7 *Parallel Repetition of GMW protocol/Blum's Protocol are not BB-ZK*

Lecture 8: January 27th, 2023 Thanks to Aman Verma for reviewing this lecture

In this section, we will show that parallel repetition of all protocols seen so far (the GMW protocol for graph isomorphism, described in Section 5.4 and Blum's protocol for Hamiltonicity, described in Section 5.6) are not black-box zero-knowledge (unless GraphIso and GraphHam are in BPP). First, one should check that the naïve simulation does not work, because the simulator will now have to guess all the challenge bits correctly. But maybe there exists a better simulator that can do this?

We will prove a stronger statement: *any constant-round protocol for non-trivial languages (with negligible soundness error) must either be non-black-box, or use private-coins.*

Here are a few assumptions that we will make about the verifier and simulator. These can be made without loss of generality:

- The last verifier machine is deterministic. That is, if we consider three message protocols, then we assume $\mathbf{V}_4$ is deterministic. If $\mathbf{V}_4$ is randomized,

then we can consider a five-message protocol, where the fourth message is the randomness used by $\mathbf{V}_4$, and the fifth message is empty. The 'impossibility' result discussed in this section can be extended for any constant message protocol.

- The black-box simulator $\mathbf{S}$ interacts with $\mathbf{V}^* = (\mathbf{V}_2^*, \mathbf{V}_4^*)$. We assume that $\mathbf{S}$ runs $\mathbf{V}_2^*$ t times, each time with a different input $(x, \mathsf{msg}_1^i, r_2^i)$. It receives msg_2^i in response. Finally, it outputs transcript $(\mathsf{msg}_1^*, \mathsf{msg}_2^*, \mathsf{msg}_3^*)$ and randomness r_2^* where $(\mathsf{msg}_1^*, r_2^*) = (\mathsf{msg}_1^i, r_2^i)$ and $\mathsf{msg}_2^* = \mathsf{msg}_2^i$ for some $i \in [t]$.

  A general simulator can output a transcript $(\mathsf{msg}_1^*, \mathsf{msg}_2^*, \mathsf{msg}_3^*)$ and randomness r_2^* such that $(\mathsf{msg}_1^*, r_2^*)$ was never present in any of the queries to $\mathbf{V}_2^*$. However, given such a simulator, we can construct a new simulator that always queries on $(\mathsf{msg}_1^*, r_2^*)$. Moreover, note that the distinguisher can always check that $\mathbf{V}_2^*(\mathsf{msg}_1^*, r_2^*) = \mathsf{msg}_2^*$. Therefore, without loss of generality, we can assume that there exists an index $i \in [t]$ such that $(\mathsf{msg}_1^*, r_2^*) = (\mathsf{msg}_1^i, r_2^i)$ and $\mathsf{msg}_2^* = \mathsf{msg}_2^i$.

Lemma 5.14. *Suppose there exists a three-round, public-coin interactive protocol $(\mathbf{P}, \mathbf{V} = (\mathbf{V}_2, \mathbf{V}_4), \mathbf{S})$ with black-box zero-knowledge for language L, satisfying perfect completeness and negligible soundness error. Then $\mathsf{L} \in \mathsf{BPP}$.*

The proof idea is very simple: suppose there exists a simulator $\mathbf{S}$ that makes t queries to the verifier, and outputs a transcript. This simulator can be used to (probabilistically) decide whether x is in the language or not. *Our algorithm $\mathcal{B}$ will run the simulator on input x. For every query by the simulator, our algorithm outputs a uniformly random string. Finally, the simulator outputs the transcript, which will be checked by the last-stage verifier $\mathbf{V}_4$.*

If x is not in the language, then the transcript should be rejected. Otherwise such a protocol will not have negligible soundness error. A cheating prover can always use such a simulator to break soundness of the underlying protocol with non-negligible probability.

If x is in the language, then we would want to argue that the simulator's transcript, when it interacts with our algorithm $\mathcal{B}$ is always accepted. We will prove this in three steps. First, we will show that the simulator's interaction with $\mathcal{B}$ ($\mathbf{S} \leftrightarrow \mathcal{B}$) is similar to the simulator's interaction with some verifier $\mathbf{V}_2^*$ ($\mathbf{S} \leftrightarrow \mathbf{V}_2^*$). Next, we will argue (using the ZK property) that the simulator's interaction with $\mathbf{V}_2^*$ is similar to the interaction of $\mathbf{V}_2^*$ with an honest prover ($\mathbf{P} \leftrightarrow \mathbf{V}_2^*$). Finally, we will argue that $\mathbf{V}_4$ will accept the final transcript of $\mathbf{P} \leftrightarrow \mathbf{V}_2^*$ (and therefore, the simulator's interaction with $\mathcal{B}$ is also accepted).

What verifier $\mathbf{V}_2^*$ must we use for the above? Recall, the algorithm $\mathcal{B}$ sends a fresh random string for every new query by the simulator. This looks similar to what the honest verifier $\mathbf{V}_2$ does in the protocol (since this is a public-coins protocol). However, note that the simulator could feed bad randomness to $\mathbf{V}_2$, and if the simulator feeds bad randomness, we cannot conclude that $\mathbf{S} \leftrightarrow \mathcal{B}$ is similar to $\mathbf{S} \leftrightarrow \mathbf{V}_2$.

Therefore, we will instead consider the simulator's interaction with a malicious verifier $\mathbf{V}_2^*$ that does not use the simulator's randomness. Instead, it uses a PRF to generate the response to simulator's messages. Now, we can argue three things: (a) this verifier's interaction with the honest prover is accepted

by the last-stage verifier (using security of PRF and completeness of the base protocol); (b) this verifier's interaction with the simulator is also accepted by the last-stage verifier (using ZK property); (c) this verifier's interaction with the simulator is indistinguishable from the simulator's interaction with $\mathcal{B}$. *The interactions $\mathbf{S} \leftrightarrow \mathbf{V}_2^*$ and $\mathbf{S} \leftrightarrow \mathcal{B}$ are indistinguishable because of the PRF security, and here we crucially use the fact that this is a black-box simulator.*

Putting all these together, we get that if $x \in \mathsf{L}_{\text{yes}}$, then the algorithm $\mathcal{B}$ correctly guesses that $x \in \mathsf{L}_{\text{yes}}$. In the proof below, we use a t-wise independent hash function instead of a PRF (since the number of queries of the simulator are bounded by some polynomial t).

Proof. For simplicity, let ℓ denote the length of each message exchanged (recall, we are assuming $\mathbf{V}_4$ is deterministic). Let t denote the total number of queries by the simulator. We will use a t-wise independent hash function (which works as a perfect pseudorandom function if there are at most t queries).

The algorithm $\mathcal{B}$ for determining membership in L works as follows:

Algorithm: $\mathcal{B}$ for deciding membership in L

1. The algorithm $\mathcal{B}$ chooses t uniformly random strings $\{r_i\}_{i\in[t]}$. These t strings will be used to respond to the simulator's queries. It runs the simulator $\mathbf{S}$, and responds to the queries as below:
2. **for** *each of the simulator's queries* **do**
3. For the i^{th} unique query from the simulator, $\mathcal{B}$ sends r_i. If it is a repeat query, $\mathcal{B}$ sends the same response as before.
4. Finally, the simulator outputs $(\mathsf{msg}_1, \mathsf{msg}_2, \mathsf{msg}_3)$. The algorithm $\mathcal{B}$ accepts x if $\mathbf{V}_4(x, \mathsf{msg}_1, \mathsf{msg}_2, \mathsf{msg}_3) = 1$.

We will now analyze the two cases, depending on whether $x \in \mathsf{L}_{\text{yes}}$ or not.

Claim 5.15. *If $x \in \mathsf{L}_{\text{yes}}$, then $\mathcal{B}$ outputs 1 with probability $1 - \text{negl}(|x|)$.*

Sketch of Proof. Consider the following transcripts:

- $\mathsf{trans}\langle \mathbf{P}, \mathbf{V}\rangle(x)$ - the transcript of the honest execution
- $\mathsf{trans}\langle \mathbf{P}, \mathbf{V}^*\rangle(x)$ - let $\mathbf{V}^*[h]$ be a verifier that uses a hash function h chosen from a t-wise independent hash family $\mathcal{H}$. It computes $\mathsf{msg}_2 = h(\mathsf{msg}_1)$ and sends it to the prover.

  Think of this as a perfect PRF.
- The transcript output by $\mathbf{S}$ when interacting with $\mathbf{V}^*[h]$.
- The transcript output by $\mathbf{S}$ when interacting with $\mathcal{B}$.

The first and second transcript are identical (using properties of t-wise independent hash). The second and third transcript are indistinguishable (using the zero-knowledge property). The third and fourth transcript are identical (using properties of t-wise independence, and the fact that $\mathbf{S}$ only has black-box access).

Finally, note that since $\mathbf{V}_4$ accepts the first transcript, it also accepts the final transcript. □

Claim 5.16. *If $x \in \mathsf{L}_{\text{no}}$, then $\mathcal{B}$ outputs 1 with probability* $\text{negl}(|x|)$.

Sketch of Proof. Suppose, on the contrary, there exists an $x \in \mathsf{L}_{\text{no}}$ such that $\mathcal{B}$ outputs 1 with non-negligible probability ϵ. We will show that there exists a malicious prover that breaks soundness of the 'base' interactive protocol $(\mathbf{P}, \mathbf{V})$ using $\mathcal{B}$ with non-negligible probability. The malicious prover $\mathbf{P}^*$ interacts with the simulator (pretending to be a verifier), and the (honest) verifier $\mathbf{V}$.

The malicious prover breaks the soundness with non-negligible probability. This is where we use the guarantee that the 3-message 'base' protocol has negligible soundness error.

Recall, the simulator makes t queries to $\mathbf{V}_2^*$. The prover picks an index $i \leftarrow [t]$. It guesses the index corresponding to the query that will be included in the final transcript (recall, we assumed without loss of generality that the transcript messages $(\mathsf{msg}_1, \mathsf{msg}_2, \mathsf{msg}_3)$ must be such that msg_1 was one of the queried messages, and msg_2 was the response). For all other queries, the prover responds by sending a uniformly random string. For the i^{th} query, it sends the query to $\mathbf{V}$ and receives msg_2^i, which it forwards to the simulator.

With probability $1/t$, the prover's guess was correct, and the simulator outputs $(\mathsf{msg}_1^i, \mathsf{msg}_2^i, \mathsf{msg}_3)$ as the final transcript. The prover sends msg_3 as the final message.

Note that from the simulator's perspective, $(\mathbf{P}^* + \mathbf{V})'s$ behavior is identical to that of $\mathcal{B}$. Since $\mathcal{B}$ outputs 1 with probability ϵ, the cheating prover succeeds with probability ϵ/t. □

Using the above two claims, it follows that $\mathcal{B}$ can decide membership in L_{yes}, and therefore $\mathsf{L} \in \mathsf{BPP}$.

□

The above result can be generalized in two ways:

- The same argument can be easily extended for any k-message public-coins protocol, where k is a constant. There is a factor $t^{O(k)}$ loss in the soundness, since the prover will need to make $O(k)$ guesses to be forwarded to the honest verifier $\mathbf{V}$.
- For three-round protocols, one can show that even private-coins are not useful. If there exists a three-message protocol for L with black-box simulation and negligible soundness, then $\mathsf{L} \in \mathsf{BPP}$. See [GK96b] for further details.

5.8 *Weaker Notions of Zero-Knowledge*

In the last few lectures, we discussed a few 'lower bounds' for (non-trivial) zero-knowledge protocols.

- Any zero-knowledge protocol must have at least three messages (that is, two-message ZK is impossible; see Assignment 2 for more details). Furthermore, if we require black-box simulation, then we can't do it in three messages.
- ZK protocols for NP-complete problems cannot have statistical security.

We know very little about non-black-box simulation. For instance, while 3-round BB-ZK is impossible (for non-trivial languages), it seems difficult to show that $O(\log^2 n)$ parallel repetitions of GMW protocol cannot be simulated. Such an impossibility would imply certain circuit lower bounds for SAT. See [BLV04] for more details.

Given these implausibility results, a natural question to ask is whether there exist meaningful relaxations of the zero-knowledge property that can bypass these lower bounds. In this section, we present one such relaxation (Assignment 2 will have another such relaxation). In this security notion, we assume the verifier is honest, and the real-world transcript should be indistinguishable from the simulated transcript. *Note that the description of* $\mathbf{V}$ *is fixed once the protocol is fixed, and therefore the simulator only needs to simulate the transcript corresponding to this fixed verifier.*

Definition 5.17 (Honest Verifier Zero Knowledge)**.** *Let* $(\mathbf{P}, \mathbf{V})$ *be an interactive proof system for language* $\mathsf{L} = (\mathsf{L}_{\text{yes}}, \mathsf{L}_{\text{no}})$*. We say that the proof system satisfies statistical (resp. computational) honest verifier zero knowledge if there exists a probabilistic polynomial time simulator* $\mathbf{S}$ *such that for every* $x \in \mathsf{L}_{\text{yes}}$*, the following distributions are statistically (resp. computationally) indistinguishable:*

$$\mathcal{D}_1 : \{\mathsf{view}\langle \mathbf{P}, \mathbf{V}\rangle(x)\}$$

$$\mathcal{D}_2 : \{\mathbf{S}(x)\}$$

◊

Note that there is no 'auxiliary information' here, since we are dealing with the honest verifier's view in both real and ideal world. An honest verifier does not use the auxiliary information.

Why study HVZK protocols?

- There could be real-world scenarios where you are not worried about the verifier learning something from the interaction, but instead are worried about an eavesdropper who is seeing the prover-verifier interaction, and trying to gather some information about the witness, based on this transcript. Therefore, in such settings, the verifier is behaving honestly, but there is an eavesdropper (the distinguisher in our definition) who tries to learn some information from the transcript.
- HVZK is a useful stepping stone for building ZK protocols. Further, we have generic transformations from HVZK to ZK.
- HVZK protocols preserve zero-knowledge property even under parallel repetition (see Lemma 5.18) below.
- Finally, HVZK protocols are extremely useful in practice. This is because, using the random oracle heuristic, we can transform a constant-round public-coin HVZK to a non-interactive ZK protocol (whose security is proven in the random oracle model).

Lemma 5.18. *Let* $(\mathbf{P}, \mathbf{V})$ *be an interactive protocol with completeness* 1*, soundness* s *and satisfying honest-verifier zero-knowledge property. Consider the parallel repeated version* $(\mathbf{P}^t, \mathbf{V}^t)$*. This protocol has completeness* 1*, soundness* s^t *and satisfies honest-verifier zero-knowledge.*

Sketch of Proof. The proof follows via a simple hybrid argument. In the real-world, the verifier's view is

$$\mathcal{D}_{\text{real}} \equiv \mathsf{view}\langle \mathbf{P}, \mathbf{V}\rangle(x) \mid \mathsf{view}\langle \mathbf{P}, \mathbf{V}\rangle(x) \mid \ldots \mid \mathsf{view}\langle \mathbf{P}, \mathbf{V}\rangle(x)$$

Recall, $\mathsf{view}\langle \mathbf{P}, \mathbf{V}\rangle(x)$ denotes the output of interaction between two probabilistic machines, and in the parallel-repeated version, we have t samples from this distribution.

The simulator for this parallel-repeated protocol simply runs the base simulator t times, and outputs

$$\mathcal{D}_{\text{ideal}} \equiv \mathbf{S}(x) \mid \mathbf{S}(x) \mid \ldots \mid \mathbf{S}(x)$$

Using t intermediate hybrids, we can gradually switch $\mathcal{D}_{\text{real}}$ to $\mathcal{D}_{\text{ideal}}$. □

It is typically easier to prove that a protocol satisfies HVZK. Since we are talking about honest verifiers, the simulator does not need to rewind the verifier anymore, and therefore the proofs are significantly simpler.

Honest-Verifier Zero-Knowledge Protocol for Graph Hamiltonicity

Here, we will show that Blum's protocol (as described in Protocol 6) satisfies honest-verifier zero-knowledgeness. The simulator first chooses a uniformly random bit b. Next, based on the bit b, it sets the first round message as either commitments to A_H, or commitments to $\mathbf{1}^{n\times n}$. It can accordingly produce the required openings.

The key idea is the following: here we care about an honest verifier's transcript in the real and ideal world. In the ideal world, it does not matter (as far as the final transcript is concerned) whether the verifier chooses the random bit or the simulator does.

If there exists a distinguisher that can distinguish between these two commitments, then the distinguisher can be used to break hiding security of the commitment scheme.

5.9 *Proofs of Knowledge*

Lecture 9: January 31st, 2023
Thanks to Aman Verma and Soumil Aggarwal for reviewing this lecture

So far, we have seen protocols for proving the validity of NP statements. However, this doesn't suffice for many real-world applications. For instance, we discussed the client-server problem at the start of this section — a client wishes to prove to a server that it *knows* a certain password. How can the client prove this without actually revealing the password? This is not captured by the soundness property, since soundness just proves that a correct password *exists*. Intuitively, we want that if a prover succeeds in a protocol, then it must 'know' some relevant secrets used to succeed in the protocol. This is formally captured via an 'extraction' based definition — if a prover succeeds in a protocol, then it can be used to extract a witness. We first discuss a few attempts towards defining this, before giving the correct definition.

Definition Attempt (Proofs of Knowledge - Attempt 1)**.** *An interactive protocol* $(\mathbf{P}, \mathbf{V})$ *for* NP *language* L *with relation* R_{L} *is a proof of knowledge if for every prover*

In class, we saw a slightly restricted version, where we only considered poly. time provers who are given a witness. Here we will present the general definition.

$\mathbf{P}^*$, *there exists an extractor* Ext *such that for every* $x \in \mathsf{L}_{\text{yes}}$, *the following holds:*

$$\Pr\left[\text{out}\langle \mathbf{P}^*, \mathbf{V}\rangle(x) = 1\right] \geq \epsilon \implies \Pr\left[(x,w) \in R_{\mathsf{L}} \ : \ w \leftarrow \mathsf{Ext}(x)\right] \geq \epsilon$$

Intuitively, the above definition says the following: if a cheating prover convinces a verifier that $x \in \mathsf{L}_{\text{yes}}$, then there exists an extractor that can output a witness. While this may seem reasonable (similar to our simulator-based definition for zero-knowledge), this definition is not very useful. This is because there always exists an extractor that is not related to the prover, and on input x, outputs a witness w.

The next definition attempt gets us closer — if there exists a prover that can convince a verifier, then an extractor can extract a witness by making black-box queries to the prover.

Definition Attempt (Proofs of Knowledge - Attempt 2)**.** *An interactive protocol* $(\mathbf{P}, \mathbf{V})$ *for* NP *language* L *with relation* R_{L} *is a proof of knowledge if there exists a p.p.t. extractor* Ext *such that for every prover* $\mathbf{P}^*$, *every* $x \in \mathsf{L}_{\text{yes}}$, *the following holds:*

$$\Pr\left[\text{out}\langle \mathbf{P}^*, \mathbf{V}\rangle(x) = 1\right] \geq \epsilon \implies \Pr\left[(x,w) \in R_{\mathsf{L}} \ : \ w \leftarrow \mathsf{Ext}^{\mathbf{P}^*}(x)\right] \geq \epsilon$$

This definition is also not good enough. Recall, in the graph hamiltonicity protocol, there exists a cheating prover that can always succeed with probability 1/2 (by choosing a bit b', ensuring that it succeeds on b', and hoping that the challenge is equal to b'). Such a prover does not have a witness, and therefore the extractor will not be able to extract a witness. We need to allow for a small probability of error, which is called the *knowledge error*.

Definition 5.19 (Proofs of Knowledge)**.** *An interactive protocol* $(\mathbf{P}, \mathbf{V})$ *for* NP *language* L *with relation* R_{L} *is a proof of knowledge with knowledge error* κ *if there exists a p.p.t. extractor* Ext *such that for every prover* $\mathbf{P}^*$, *every* $x \in \mathsf{L}_{\text{yes}}$, *the following holds:*

$$\Pr\left[\text{out}\langle \mathbf{P}^*, \mathbf{V}\rangle(x) = 1\right] \geq \epsilon \implies \Pr\left[(x,w) \in R_{\mathsf{L}} \ : \ w \leftarrow \mathsf{Ext}^{\mathbf{P}^*}(x)\right] \geq \epsilon - \kappa$$

◊

Note that this is a soundness property, and has nothing to do with the zero-knowledge property. However, the knowledge extraction property may seem to be at odds with the zero-knowledge property — if the extractor can extract the witness by simply interacting with the prover, can't the verifier also do the same (or learn some information about a witness), thereby violating the zero-knowledge property. The key difference is that the *extractor can rewind the prover*, while the verifier cannot do so in a regular interaction with the prover. This rewinding may allow the extractor to obtain two different (but correlated) transcripts from the prover, which can be useful for extracting the witness.

PoK for Graph Hamiltonicity

In this section, we will show that Blum's protocol is a proof-of-knowledge with knowledge error 1/2. Recall, the prover first sends a set of commitments. Let us call this timestamp t_0. The extractor sends bit $b = 0$ to the prover, and receives some of the openings. Next, it rewinds the prover to t_0, and now sends bit $b = 1$. If the prover succeeds in both these executions, then the extractor has a

permutation of the original graph, and a permutation of a Hamiltonian cycle. Using these two, it can recover a Hamiltonian cycle in the graph.

The above argument can be made formal to show that this protocol has knowledge error 1/2. The formal proof will be available in Section 5.13.

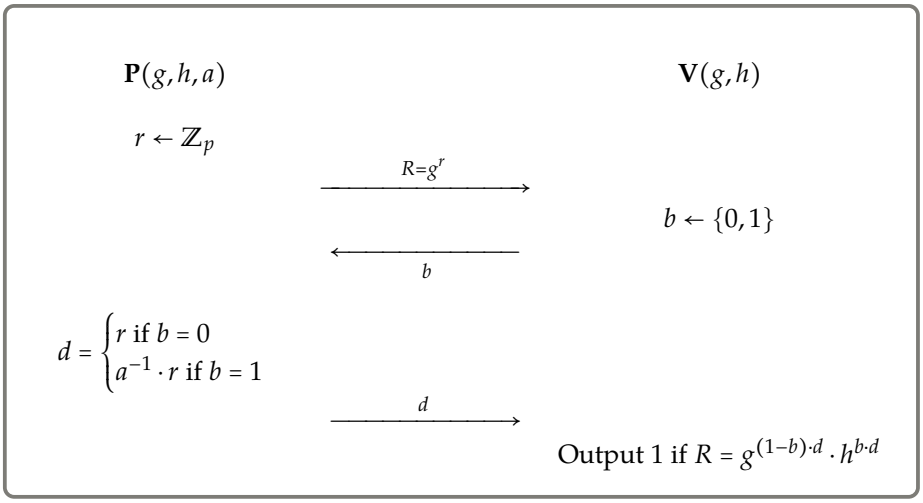
Sequential Repetition Reduces Knowledge Error

Similar to the case of soundness, knowledge error can also be reduced by sequential repetition of the base protocol. The formal proof will be included in Section 5.13.

ZK/HVZK PoK for Discrete Log Problem

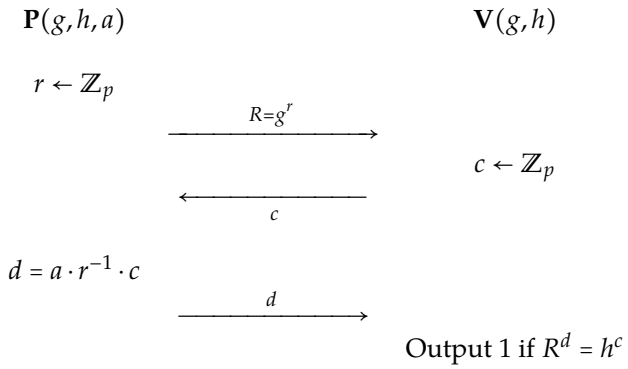
Let $\mathbb{G}$ be a group of prime order p. We want a zero-knowledge protocol by which a prover can prove that, given group elements g and h, it *knows* an exponent a such that $h = g^a$. Such protocols are widely used as *identification protocols*.

As a first attempt, the following protocol was proposed in class.



This protocol is a zero-knowledge proof-of-knowledge protocol, and can be repeated sequentially to obtain negligible knowledge error (while still maintaining zero-knowledgeness). Can we get a constant round, honest-verifier zero-knowledge proof-of-knowledge? One way to do that would be via parallel-repetition. Can we get a more efficient, constant-round, HVZK PoK?

Here is the first attempt proposed in class.



This protocol satisfies completeness. It is also honest-verifier zero-knowledge. A simulator, given (g,h), can first sample $c \leftarrow \mathbb{Z}_p$, choose a random $d \leftarrow \mathbb{Z}_p$, set $R = (h^c)^{1/d}$ and outputs (R,c,d) as the transcript. Note that this is perfectly identical to the honest verifier's view when it interacts with an honest prover. However, this protocol does not satisfy proof-of-knowledge. A prover can succeed in this protocol without knowing the discrete log, by first sending $R = h^{1/r}$, then sends $d = r \cdot c$.

Schnorr's protocol: The following protocol (proposed in class as the second attempt) was first given by Schnorr [Sch91] in the context of signature schemes. It is an HVZK PoK for the discrete log problem.

HVZK PoKs can be viewed as 'interactive signatures'. It is possible to make them non-interactive via the random oracle heuristic. More on this later in the course.

Common Input: Group elements g, h
Prover's Private Input: $a \in \mathbb{Z}_p$ such that $h = g^a$

Claim to prove: Prover knows a

Protocol 10: Discrete Log HVZK PoK: Schnorr's Protocol [Sch91]

1. $\mathbf{P}_1$: Prover picks a uniformly random $t \leftarrow \mathbb{Z}_p$ and sends $u = g^t$.
2. $\mathbf{V}_2$: Verifier sends a uniformly random $c \leftarrow \mathbb{Z}_p$.
3. $\mathbf{P}_3$: Prover sends $w = t + c \cdot a$
4. $\mathbf{V}_4$: Verifier, on receiving w, checks if $u \cdot h^c = g^w$.

Completeness is immediate.

Proof of Knowledge: The knowledge extractor runs the prover. On receiving u, it chooses a uniformly random c_1 and sends it to the prover. It receives w_1. Next, it rewinds the prover (to the point after it receives u), picks a uniformly random c_2 and sends it to the prover. It receives w_2. Finally, it outputs $(w_2 - w_1)/(c_2 - c_1)$. If the prover succeeds in the protocol with non-negligible probability ϵ, then the knowledge extractor extracts a with non-negligible probability. The exact probability computation is part of Assignment 2 (for a related problem - the decisional Diffie-Hellman problem).

HVZK: The HVZK simulator picks $c \leftarrow \mathbb{Z}_p$ and $w \leftarrow \mathbb{Z}_p$. It sets $u = g^w/h^c$ and outputs (u, c, w) as the transcript. This is identical to the transcript in the real-world.

It is not clear how to prove zero-knowledgeness for the above protocol. The issue is that the challenge space is now exponential (in the security parameter), and a malicious verifier could always choose a challenge that depends (deterministically) upon u. As a result, rewinding doesn't help.

5.10 *Zero-Knowledge Protocols for* IP

Lecture 10: February 3rd, 2023

So far, we have seen zero-knowledge protocols for languages in NP. In this section, we will present zero-knowledge interactive proofs for languages outside NP. Let us start with the sum-check protocol. We saw that the sum-check protocol is not zero-knowledge because the polynomials sent by the prover leak extra information. Using two simple observations, we can transform the sum-check protocol into one that is also zero-knowledge:

- the sum-check protocol is a public coins protocol. The verifier picks random elements from the field and sends them to the prover.
- At the end of the protocol, the final-stage verifier $\mathbf{V}_{\text{final}}$ uses all prior messages for checking the transcript. Note that this is an efficient deterministic algorithm (that is, it is in P).

By public-coins, we mean that the verifier is only choosing uniformly random strings in each round, which it sends to the prover.

Here is the recipe to convert any public-coins protocol into a zero-knowledge public-coins protocol:

- For the i^{th} message, the prover sends a commitment to the message (instead of the actual message). Namely, if the prover's i^{th} message is msg_i, in the modified protocol, the prover sends $c_i = \mathsf{Commit}(\mathsf{msg}_i; r_i)$, where Commit is a perfectly-binding computationally hiding commitment scheme.
- Suppose the base protocol is a $2k-1$ message protocol, where the prover sends $\{\mathsf{msg}_{2i-1}\}_{i\in[k]}$ and the verifier sends $\{\mathsf{msg}_{2i}\}_{i\in[k-1]}$. At the end, the verifier of the base protocol outputs $\mathbf{V}_{\text{final}}\left(\{\mathsf{msg}_i\}_{i\in[2k-1]}\right)$.

  Consider the following NP statement:

  $$\exists \{\mathsf{msg}_{2i-1}, r_{2i-1}\}_{i\in[k]} \text{ s.t.}$$
  $$c_i = \mathsf{Commit}(\mathsf{msg}_{2i-1}; r_{2i-1}) \text{ for all } i \in [k] \text{ and } \mathbf{V}_{\text{final}}\left(\{\mathsf{msg}_i\}_{i\in[2k-1]}\right) = 1$$

  The prover gives a zero-knowledge proof for the validity of this NP statement.

Note that any interactive protocol can be converted into a public-coins interactive protocol (with just two extra messages), and therefore, we have zero-knowledge proofs for all of IP. Soundness follows from the soundness of the base protocol, as well as the soundness of the zero-knowledge protocol used at the end. For zero-knowledge, we need to define a p.p.t. simulator. The simulator commits to garbage strings, and gives a simulated zero-knowledge proof at the end. Proof follows via a hybrid argument: we first switch to a hybrid

simulator that honestly commits to the correct messages (during the k rounds of interaction), but gives a simulated zero-knowledge proof at the end. This is indistinguishable from the real-world prover-verifier interaction using security of zero-knowledge protocol. Next, we switch the commitments one-by-one, and these are indistinguishable using the hiding guarantee of commitment scheme.

5.11 *Constant Round Zero-Knowledge Protocols*

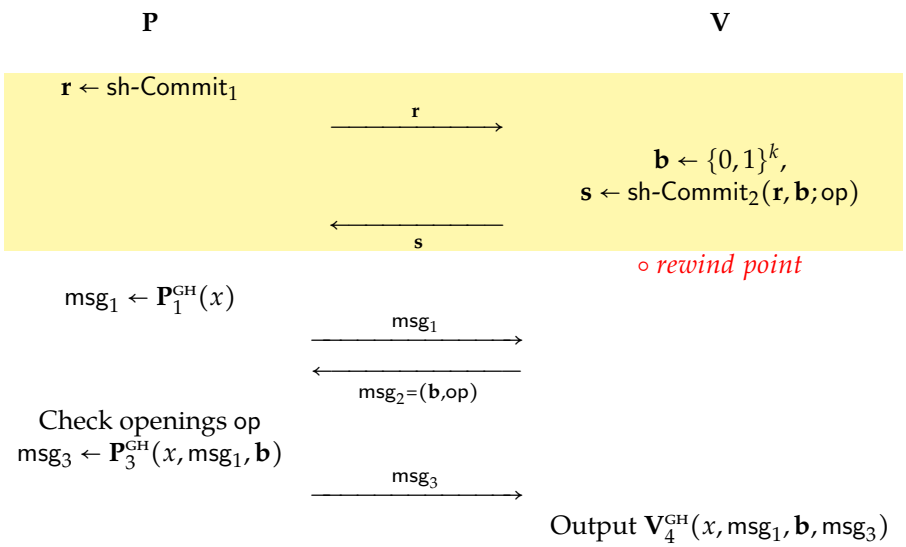
Recall, in the lower bound discussed in Section 5.7, the malicious verifier applies the PRF on the prover's first message, which means the verifier's response was depending on the whole 'first message'.

So far, we have seen constant round zero-knowledge protocols with constant soundness error. Parallel repetition reduces the soundness error, but does not preserve zero-knowledge. Let us recall the main issue, using Protocol 6 as the guiding example. Suppose we repeat this protocol k times in parallel. The prover commits to k different permutations of the graph. The verifier, on receiving these permutations, sends k bits (the i^{th} bit serves as a 'challenge' for the i^{th} parallel repetition). While the honest verifier chooses these k bits uniformly at random, the malicious verifier can choose these bits *depending on all the k commitments by the challenger*.

Even if we only cared about p.p.t. cheating provers, we cannot use comp. hiding commitments.

A natural idea to constrain the malicious verifier: ask the verifier to commit to the 'challenge bits' before the prover sends the commitments. Note that this commitment needs to be *statistically hiding* (since we want soundness against unbounded provers). The resulting protocol is described below. For clarity, let $\mathsf{sh\text{-}Commit}$ be a statistically hiding, computationally binding commitment scheme. There exist two-message protocols for such commitment schemes, where the receiver sends the first message (using $\mathsf{sh\text{-}Commit}_1$), and the sender sends the second message (using $\mathsf{sh\text{-}Commit}_2$). Similarly, let $\mathsf{sb\text{-}Commit}$ be a non-interactive statistically binding (computationally hiding) commitment scheme. Finally, let $(\mathbf{P}^{\text{GH}}, \mathbf{V}^{\text{GH}}, \mathbf{S}^{\text{GH}})$ be the k-parallel repetition of Blum's commit-and-open protocol. We assume that $\mathbf{P}_1^{\text{GH}}$ outputs the first message (which consists of k commitments), then $\mathbf{V}_2^{\text{GH}}$ outputs the challenge vector, and finally $\mathbf{P}_3^{\text{GH}}$ computes the third message.

Will hopefully include stat. hiding commitments in in Assignment 3, please assume this primitive for now. Construction for perfectly binding commitments can be found in Section 2.2.



This five-message protocol has perfect completeness, and negligible soundness (assuming $k = \omega(\log n)$). The interesting part is zero-knowledgeness. For this protocol, there is a very natural simulator. The simulator interacts with the verifier up to the fourth message (that is, when it receives $\mathbf{b}, \mathsf{op}$). After this, it rewinds the verifier, and changes msg_1 (depending on $\mathbf{b}$). Recall, msg_1 consists of k commitments, and using the hiding property of this commitment scheme, we can argue that the distribution of $\mathbf{b}$ does not change (ignoring negligible differences). At this point, we might be tempted to conclude that we have a security proof for zero-knowledgeness.

An important subtlety : Aborting verifiers

There is a subtlety that we have not discussed so far in the course, but is important to discuss here. What happens if the verifier refuses to produce $\mathbf{b}$ together with a valid opening? That is, suppose the verifier *intentionally* aborts the protocol after the third message. At first sight, it appears that this shouldn't be an issue : if the verifier aborts with probability p, then the honest prover would also end the protocol, and therefore this should be easy to simulate - the simulator can abort whenever the adversary aborts. Note: by 'aborting verifier', we mean that msg_2 is not a valid opening for $\mathbf{s}$. For ease of notation, we say that $\mathsf{msg}_2 = \perp_{\mathbf{V}}$.

Below, we describe a few natural approaches to try, and why the approach doesn't work. This will lead us to the correct simulator. In all these attempts, we assume a simulator $\mathbf{S}^{\mathrm{GH}}$ for the 'base' graph hamiltonicity protocol which takes as input a k bit vector $\mathbf{b}$ and outputs $(\mathsf{msg}_1, \mathsf{msg}_3) \leftarrow \mathbf{S}^{\mathrm{GH}}(\mathbf{b})$.

Simulator aborts if verifier aborts: In our first attempt, let us consider a simulator that aborts if the verifier aborts. If the verifier doesn't abort, the simulator rewinds the verifier and sends a message based on the bit vector $\mathbf{b}$. Again, if the verifier aborts, the simulator aborts, else it outputs the transcript.

Simulation Attempt: Simulator 1

1 Simulator chooses $\mathbf{r}$, computes $(\mathbf{s}, \mathsf{st}_2) \leftarrow \mathbf{V}_2^*(\mathbf{r})$.
2 Next, it produces $\mathsf{msg}_1 \leftarrow \mathbf{P}^{\mathrm{GH}}(x)$.
3 It computes $\mathsf{msg}_2 \leftarrow \mathbf{V}_4^*(\mathsf{msg}_1, \mathsf{st}_2)$. **if** $\mathsf{msg}_2 = \perp_{\mathbf{V}}$ **then**
4 the simulator outputs $(\mathbf{r}, \mathbf{s}, \mathsf{msg}_1, \perp_{\mathbf{V}})$.
5 **else**
6 Let $\mathsf{msg}_2 = (\mathbf{b}, \mathsf{op})$. The simulator uses $\mathbf{S}^{\mathrm{GH}}$ to compute msg_1' and msg_3' based on $\mathbf{b}$. That is, it computes $(\mathsf{msg}_1', \mathsf{msg}_3') \leftarrow \mathbf{S}^{\mathrm{GH}}(\mathbf{b})$.
7 It computes $\mathsf{msg}_2' \leftarrow \mathbf{V}_4^*(\mathsf{msg}_1', \mathsf{st}_2)$. **if** $\mathsf{msg}_2' = \perp_{\mathbf{V}}$ **then**
8 Simulator outputs $(\mathbf{r}, \mathbf{s}, \mathsf{msg}_1, \perp_{\mathbf{V}})$.
9 **else**
10 ...

In the above protocol, Step 10 is not specified, since this simulator's output is *distinguishable* from the real-world transcript. Suppose, in the real-world, a malicious verifier outputs $\perp_{\mathbf{V}}$ with probability p (irrespective of the message msg_1). Check that in the above simulated output, the simulator outputs $\perp_{\mathbf{V}}$ with probability $2p - p^2$.

Exercise 5.6. *Consider the following simulator for the above protocol.*

Simulation Exercise: Simulator that doesn't rewind

1 *Simulator chooses* $\mathbf{r}$, *computes* $(\mathbf{s}, \mathsf{st}_2) \leftarrow \mathbf{V}_2^*(\mathbf{r})$.
2 *Next, it produces* $\mathsf{msg}_1 \leftarrow \mathbf{P}^{GH}(x)$.
3 *It computes* $\mathsf{msg}_2 \leftarrow \mathbf{V}_4^*(\mathsf{msg}_1, \mathsf{st}_2)$. **if** $\mathsf{msg}_2 = \perp_{\mathbf{V}}$ **then**
4 *the simulator outputs* $(\mathbf{r}, \mathbf{s}, \mathsf{msg}_1, \perp_{\mathbf{V}})$.
5 **else**
6 *Let* $\mathsf{msg}_2 = (\mathbf{b}, \mathsf{op})$. *The simulator computes* $(\mathsf{msg}_1', \mathsf{msg}_3') \leftarrow \mathbf{S}^{GH}(\mathbf{b})$ *and outputs* $(\mathbf{r}, \mathbf{s}, \mathsf{msg}_1', \mathsf{msg}_2, \mathsf{msg}_3')$.

The intuition for this simulator: since the commitment scheme used for the first two messages is comp. binding, if the verifier produces a non-$\perp_{\mathbf{V}}$ *message in the fourth message, then it has to be* $\mathbf{b}$. *Therefore we don't need to re-run the verifier if we get a non-*$\perp_{\mathbf{V}}$ *output (in Step 3).*
What is the problem with this simulator?

The problem in the first attempt is that the simulator is aborting twice. Even if the verifier doesn't output $\perp_{\mathbf{V}}$ after msg_1, there is a possibility that the simulator outputs $\perp_{\mathbf{V}}$ (because it rewinds the verifier and outputs $\perp_{\mathbf{V}}$ if the verifier outputs $\perp_{\mathbf{V}}$ in the second try). To fix this, consider the following simulator. This does not abort if the verifier does not abort in Step 3.

Simulation Attempt: Simulator 2

1 Simulator chooses $\mathbf{r}$, computes $(\mathbf{s}, \mathsf{st}_2) \leftarrow \mathbf{V}_2^*(\mathbf{r})$.
2 Next, it produces $\mathsf{msg}_1 \leftarrow \mathbf{P}^{\mathrm{GH}}(x)$.
3 It computes $\mathsf{msg}_2 \leftarrow \mathbf{V}_4^*(\mathsf{msg}_1, \mathsf{st}_2)$. **if** $\mathsf{msg}_2 = \perp_{\mathbf{V}}$ **then**
4 the simulator outputs $(\mathbf{r}, \mathbf{s}, \mathsf{msg}_1, \perp_{\mathbf{V}})$.
5 **else**
6 **while** `success` $\neq$ `true` **do**
7 Let $\mathsf{msg}_2 = (\mathbf{b}, \mathsf{op})$. The simulator computes $(\mathsf{msg}_1', \mathsf{msg}_3') \leftarrow \mathbf{S}^{\mathrm{GH}}(\mathbf{b})$.
8 It computes $\mathsf{msg}_2' \leftarrow \mathbf{V}_4^*(\mathsf{msg}_1', \mathsf{st}_2)$. **if** $\mathsf{msg}_2' \neq \perp_{\mathbf{V}}$ **then**
9 It sets `success= true`. Output $(\mathbf{r}, \mathbf{s}, \mathsf{msg}_1', \mathsf{msg}_2', \mathsf{msg}_3')$.

Before discussing whether the simulator's output is indistinguishable from the real-world output, let us first check the simulator's running time. Clearly, this simulator's running time (in the worst case) can be exponential. However, can we show that this simulator has *polynomial expected running time*? Let $p_{\mathbf{V}^*}$ denote the probability that the malicious verifier does not output $\perp_{\mathbf{V}}$ when given $\mathsf{msg}_1 \leftarrow \mathbf{P}^{\mathrm{GH}}(x)$. The expected running time of this simulator is

$$((1 - p_{\mathbf{V}^*}) + p_{\mathbf{V}^*} \cdot \text{ expected number of iterations of the while loop}) \cdot \mathrm{poly}(|x|)$$

The expected number of iterations of the while loop depend upon the verifier not outputting $\perp_{\mathbf{V}}$ when given msg_1' (where $(\mathsf{msg}_1', \mathsf{msg}_3') \leftarrow \mathbf{S}^{\mathrm{GH}}(\mathbf{b})$). Intuitively, by the computational hiding property of the commitment scheme, we know that this probability, denoted by $q_{\mathbf{V}}$, should be negligibly close to $p_{\mathbf{V}}$. This probability $q_{\mathbf{V}}$ governs the number of iterations in the while loop, and the expected number

of iterations is $1/q_{\mathbf{V}}$. However, does that guarantee that the expected running time is poly($|x|$)?

Consider the following scenario: let $n = |x|$, $p_{\mathbf{V}^*} = 2^{-n}$ and $q_{\mathbf{V}^*} = 2^{-2n}$. The probabilities $p_{\mathbf{V}^*}$ and $q_{\mathbf{V}^*}$ are negligibly close, but the expected running time of $\mathbf{S}$ is not polynomial. Thus, the expected running time of the above simulator is not polynomial in $|x|$.

In this attempt, we are rewinding the verifier too many times (compare this to the previous attempt, where we were rewinding only once). The fix here is to rewind sufficiently many times, but also have a bound on the number of rewinds so that the expected running time is still polynomial.

Expected poly. time vs strict poly. time: Before we move on to the correct simulation, note that we are aiming for the simulator's *expected* running time to be polynomial in n. As pointed out by one of the students in class, we can modify the definition of perfect zero-knowledge (Definition 5.2) to allow for expected poly. time simulation, in which case we don't need the simulator to output $\perp$.

Barak and Lindell [BL02] show that if L has a const. round ZKP with negl. soundness error and strict-poly. time BB simulation, then $\mathsf{L} \in \mathsf{BPP}$.

Definition 5.20 (Zero knowledge with Expected Poly Time Simulation)**.** *Let $(\mathbf{P}, \mathbf{V})$ be an interactive proof system for language $\mathsf{L} = (\mathsf{L}_{\text{yes}}, \mathsf{L}_{\text{no}})$. We say that the proof system satisfies statistical/computational zero knowledge with expected poly. time simulation if for every (possibly malicious) probabilistic polynomial time verifier $\mathbf{V}^*$, there exists an expected p.p.t. simulator $\mathbf{S}$ such that for every $x \in \mathsf{L}_{\text{yes}}$ and every string $z \in \{0,1\}^*$, the following distributions are statistically/computationally indistinguishable:*

$$\mathcal{D}_1 : \{\mathsf{view}\langle \mathbf{P}, \mathbf{V}^*(z)\rangle(x)\}$$

$$\mathcal{D}_2 : \{\mathbf{S}(x, z)\}$$

◊

An algorithm can have expected poly. running time in one model of computation, but superpolynomial expected running time in another computational model.

Strict polynomial time simulation is more desirable than expected poly. time simulation because the notion of expected polynomial time computation is dependent on the model of computation (also see discussion below Claim 5.22). However, for NP-complete languages, strict polynomial time, black-box simulation is unlikely for protocols that have constant rounds and negligible soundness [BL02].

Exercise 5.7. *Suppose we have an algorithm $\mathcal{A}$ for solving a problem with the following guarantees:*

- *for every input, the algorithm always gives the correct output.*
- *for every input, the expected running time of the algorithm is polynomial in the input size.*

Show that we can construct a new algorithm $\mathcal{B}$ for solving the same problem with the following guarantees:

- *for every input, the algorithm always runs in strict polynomial time. That is, there exists a polynomial* poly($\cdot$) *such that the running time on input x is at most* poly($|x|$).

- *for every input, if the input is a 'yes' instance, then the algorithm always outputs* 1. *If the input is a 'no' instance, then the algorithm outputs* 0 *with probability at least* 1/2.

Why can't we do a similar transformation for ZKP simulators?

Simulation by estimating the abort probability

The key idea is to estimate the probability $p_{\mathbf{V}^*}$, and limit the number of iterations of the while loop. The probability estimation should happen only if $\mathsf{msg}_2 \neq \perp_{\mathbf{V}}$ in Step 3.

Simulation: Simulator for constant round protocol [GK96a]

1 Simulator chooses $\mathbf{r}$, computes $(\mathbf{s}, \mathsf{st}_2) \leftarrow \mathbf{V}_2^*(\mathbf{r})$.
2 Next, it produces $\mathsf{msg}_1 \leftarrow \mathbf{P}^{\mathrm{GH}}(x)$.
3 It computes $\mathsf{msg}_2 \leftarrow \mathbf{V}_4^*(\mathsf{msg}_1, \mathsf{st}_2)$. **if** $\mathsf{msg}_2 = \perp_{\mathbf{V}}$ **then**
4 the simulator outputs $(\mathbf{r}, \mathbf{s}, \mathsf{msg}_1, \perp_{\mathbf{V}})$ together with the verifier's randomness.
5 **else**
6 Let $\mathsf{msg}_2 = (\mathbf{b}, \mathsf{op})$, $T = 0$.
 /* The simulator estimates $p_{\mathbf{V}^*}$: the prob. of $\mathsf{msg}_2 \neq \perp_{\mathbf{V}}$. */
7 **while** $T \leq 100 \cdot n$ **do**
8 The simulator samples $\mathsf{msg}_1 \leftarrow \mathbf{P}^{\mathrm{GH}}(x)$.
9 It computes $\mathsf{msg}_2' \leftarrow \mathbf{V}_4^*(\mathsf{msg}_1, \mathsf{st}_2)$.
10 **if** $\mathsf{msg}_2' = (\mathbf{b}', \mathsf{op}') \neq \perp_{\mathbf{V}}$ **then**
11 **if** $\mathbf{b} \neq \mathbf{b}'$ **then**
12 Output 'comp. binding property violated'.
13 **else**
14 $T = T + 1$.
15 The simulator sets est $= (100 \cdot n)/T$.
 /* With overwhelming probability, est is close to $p_{\mathbf{V}^*}$. */
16 **for** count $= 1$ *to* n*:* **do**
 /* Suppose $q_{\mathbf{V}^*}$ is probability of non-$\perp_{\mathbf{V}}$ when $\mathbf{V}^*$ is given simulated first message. If $q_{\mathbf{V}^*}$ is close to $p_{\mathbf{V}^*}$ and est is a good estimate, then with prob. $\geq 1 - 1/n$, the for-loop below produces an output. This follows from Markov inequality. Therefore repeating this n times ensures that simulation fails with probability at most $1/n^n$. We additionally need a bound of 2^n to handle the case when est is a bad estimate. */
17 **for** $j = 1$ *to* $\min(2^n, n/\mathsf{est})$*:* **do**
18 Simulator computes $(\mathsf{msg}_1', \mathsf{msg}_3') \leftarrow \mathbf{S}^{\mathrm{GH}}(\mathbf{b})$, $\mathsf{msg}_2' \leftarrow \mathbf{V}_4^*(\mathsf{msg}_1', \mathsf{st}_2)$.
19 **if** $\mathsf{msg}_2' = (\mathbf{b}', \mathsf{op}') \neq \perp_{\mathbf{V}}$ **then**
20 **if** $\mathbf{b}' \neq \mathbf{b}$ **then**
21 Output 'comp. binding property violated'.
22 **else**
23 Output $(\mathbf{r}, \mathbf{s}, \mathsf{msg}_1', \mathsf{msg}_2', \mathsf{msg}_3')$ together with the verifier's randomness.
24 Output 'Simulation failed'

Claim 5.21. *The expected running time of the simulator is polynomial in the input size.*

Proof. Clearly, the running time of Steps 1 to 3 is polynomial in $n = |x|$. Suppose the verifier outputs $\perp_{\mathbf{V}}$ with probability $1 - p_{\mathbf{V}}$. The expensive computation is in the '*else*' part, starting with Step 6, and this part is performed with probability $p_{\mathbf{V}}$. The expected running time of the '*while*' loop (Steps 7 to 14) is $(100 \cdot n)/p_{\mathbf{V}}$. Using Chernoff bounds for geometric random variables (see [Lin13], Lemma 2.1 for the exact tail bound), one can show that for any constant $c > 1$, the '*while*'

loop terminates in $(c \cdot 100 \cdot n)/p_{\mathbf{V}}$ steps with probability $1 - \exp(c \cdot n)$. Finally, note that the rest of the '*else*' part's running time depends polynomially on the number of executions of the while loop.

Using this, we can conclude that the overall expected running time is polynomial in n. □

Claim 5.22. *If the simulator outputs* `'comp. binding property violated'` *with non-negligible probability, then there exists a p.p.t. algorithm for breaking the computational binding property of* sh-Commit.

The above claim follows immediately from the description of the simulator. Note that one needs to be careful with the running time of the p.p.t. algorithm, since our simulator's running time is expected p.p.t instead of strict p.p.t. Reductions do not preserve expected polynomial running time. For instance, you can have an algorithm that solves a particular problem $\mathcal{X}$ in expected polynomial time. And suppose you have a polynomial time reduction that reduces some problem $\mathcal{Y}$ to $\mathcal{X}$. This does not guarantee that you have an expected polynomial time algorithm for $\mathcal{Y}$.

Claim 5.23. *Assuming* sb-Commit *is computationally hiding, the simulator outputs* `'Simulation failed'` *with negligible probability.*

Sketch of Proof. Simulation fails only in the 'else' part. As before, let $p_{\mathbf{V}}$ denote the probability of $\mathbf{V}^*$ not outputting $\perp_{\mathbf{V}}$ after receiving $\mathsf{msg}_1 \leftarrow \mathbf{P}^{\mathrm{GH}}(x)$, and let $q_{\mathbf{V}}$ denote the probability of $\mathbf{V}^*$ not outputting $\perp_{\mathbf{V}}$ after receiving $\mathsf{msg}_1 \leftarrow \mathbf{S}(\mathbf{b})$. If $q_{\mathbf{V}}$ and $p_{\mathbf{V}}$ are far-apart, then we have a p.p.t. algorithm that breaks the computational hiding property. If $p_{\mathbf{V}}$ and $q_{\mathbf{V}}$ are close, then there are two possibilities:

- our estimate of $p_{\mathbf{V}}$ (and therefore $q_{\mathbf{V}}$) is not good. The probability of this can again be bounded using Chernoff bounds.
- our estimate of $p_{\mathbf{V}}$ (and therefore $q_{\mathbf{V}}$) is within a constant factor of $q_{\mathbf{V}}$. Then, the expected number of attempts to get a non-$\perp_{\mathbf{V}}$ output from $\mathbf{V}_4^*(\mathsf{msg}_1', \mathsf{st}_2)$ (where $(\mathsf{msg}_1', \mathsf{msg}_3') \leftarrow \mathbf{S}^{\mathrm{GH}}(\mathbf{b})$) is $1/q_{\mathbf{V}}$. As a result, using Markov inequality, the probability of simulation failing during the '*for*' loop from Steps 17 to 23 is $O(1/n)$. Therefore, repeating this '*for*' loop n times (using independent random coins) ensures that there will be at least one successful run with overwhelming probability.

□

Claim 5.24. *Assuming* sb-Commit *is comp. hiding and* sh-Commit *is comp. binding, the simulator's output is indistinguishable from the real-world interaction between an honest prover and* $\mathbf{V}^*$.

This proof roughly follows the proof in Section 5.13. However, as in the proof of Claim 5.22, we need to be careful with regard to the 'expected running time' issue. The reader can refer to [GK96a] (Claim 4) for a formal argument of this claim ([GK96a] proves this for the 3COL problem, but the same argument should also work for graph hamiltonicity).

Concluding Remarks This protocol illustrates some of the subtleties that arise in the security analysis of protocols. The reader is encouraged to revisit Blum's protocol 6, and check if we also need to separately handle aborting verifiers for that protocol. Also, check that the protocol of Goldreich-Kahan is not a proof-of-knowledge.

A few years after the protocol of Goldreich-Kahan, Rosen [Ros04] gave a 7-round protocol with a much simpler analysis. In this protocol, the verifier first picks a uniformly random 'challenge string' $\mathbf{b}$ (as in the [GK96a] protocol). It also picks n random strings $\mathbf{b}_{1,0}, \ldots, \mathbf{b}_{n,0}$, sets $\mathbf{b}_{j,1} = \mathbf{b} \oplus \mathbf{b}_{j,1}$, and commits to all $\{\mathbf{b}_{j,\beta}\}_{j\in[n],\beta\in\{0,1\}}$ using a two-round statistically hiding commitment scheme. Next, the prover sends an n bit string $\mathbf{t}$, and the verifier produces $\left\{\mathbf{b}_{j,\mathbf{t}_j}\right\}_j$ together with the corresponding openings. Note, up to this point, the bit string $\mathbf{b}$ is still hidden from the prover (but, looking ahead, this allows the simulator to extract $\mathbf{b}$ by rewinding and querying on two different strings). After this, the prover and the verifier participate in a three-round protocol (using $\mathbf{P}^{\text{GH}}, \mathbf{V}^{\text{GH}}$), and the simulator can simulate this since it would have extracted the challenge string.

Simulator for Rosen's protocol: The simulator first sends $\mathbf{r}$, receives $2n$ commitments. Next, it sends a uniformly random string $\mathbf{t}$. If the verifier sends a non-$\perp_{\mathbf{V}}$ message, the verifier rewinds and sends a different (uniformly random) $\mathbf{t}'$, and repeats this until it receives a non-$\perp_{\mathbf{V}}$ message again. Using the two non-$\perp_{\mathbf{V}}$ messages, it extracts $\mathbf{b}$ which is used to simulate the remaining three messages. The key difference between this protocol and Goldreich-Kahan's protocol is that in this one, the distribution of simulator's message after rewinding does not change. Even after rewinding, it sends a uniformly random n-bit string. As a result, the probabilities $p_{\mathbf{V}}$ and $q_{\mathbf{V}}$ (as defined in the analysis of Goldreich-Kahan's protocol) are equal.

Lindell [Lin13] showed a simple modification to the Goldreich-Kahan protocol to make it a proof-of-knowledge. Apparently a similar modification also works for Rosen's protocol, although a formal security proof would be interesting.

5.12 *Chapter Summary, and Related Results*

- **Interactive Proofs**
  - The complexity class BPP consists of all problems that can be efficiently solved (that is, in polynomial time) with bounded error. The complexity class NP consists of all problems whose solutions can be efficiently and deterministically verified. We can also consider problems whose solutions can be efficiently verified using a randomized verifier, and this is the complexity class MA.
  - Goldwasser, Micali and Rackoff [GMR85] and Babai [Bab85] introduced the notion of interactive proofs. A language L has an interactive protocol (with an efficient verifier) if
    (a) a honest prover can convince a honest verifier that $x \in \mathsf{L}_{\text{yes}}$ by following the protocol. This is the completeness requirement.
    (b) if $x \in \mathsf{L}_{\text{no}}$, then even a cheating prover cannot convince a honest verifier. This is the soundness requirement.

MA *stands for 'Merlin-Arthur'. This class consists of problems where, given any instance x, an all-powerful prover Merlin can convince a prob. poly. time verifier Arthur if $x \in \mathsf{L}_{\text{yes}}$, but cannot fool Arthur if $x \in \mathsf{L}_{\text{no}}$.*
AM *stands for 'Arthur-Merlin'. Here, Arthur first sends a random string, then Merlin sends a proof, which Arthur can then verify. This is a two round pub. coins protocol. One can show that any const. round pub. coins protocol can be converted to a two round pub. coins protocol.*

Here, we observed that both randomness and interaction are necessary to go beyond the complexity class MA.

- If we allow both randomness and interaction, then we can have protocols for languages that are believed to be outside MA. We saw the example of graph non-isomorphism. We also discussed that any such protocol can be made a public-coins protocol, where the verifier only sends random coins. This brings us to the complexity class AM, which consists of all problems that can be verified by a *constant-round*, public-coins protocol. For quite some time, it was believed that this is all that can be done with interactive proofs.
- The sumcheck protocol, proposed by [LFKN92], showed that interactive protocols can be used to verify much harder problems.
- Interactive proofs will show up again in the course, when we discuss quantum protocols. Here, we will allow the prover to be a quantum machine (running in polynomial time), and the verifier will either be prob. poly. time, or a machine with *very limited* quantum capabilities.

- **Zero-Knowledge Proofs**
  - In zero-knowledge proofs, other than completeness and soundness, we have an additional requirement — the verifier should not learn anything 'new' from the interactive proof, other than the validity of the statement. This is formally defined using a simulation-based definition.
  - Next, we saw a three-message protocol for graph isomorphism that is perfectly zero-knowledge.
  - After the graph isomorphism protocol, we discussed a zero-knowledge protocol for graph hamiltonicity (Blum's protocol). Since graph hamiltonicity is NP-complete, this gives us zero-knowledge protocols for all problems in NP. Both these proofs *crucially rely on the ability to rewind a classical device*. Looking ahead, rewinding will be tricky when dealing with quantum adversaries.
  - The above protocols had constant soundness error. This can be reduced by sequentially repeating the protocol. Parallel repetition can also reduce the soundness error, but it does not preserve the zero-knowledge property.
  - More generally, in a three-message protocol with negligible soundness error, a malicious verifier's view cannot be simulated via black-box access to the verifier. In class, we discussed a similar lower bound - NP-complete problems cannot have a public-coins, constant-round, black-box zero-knowledge proof with negligible soundness error.
  - Zero-knowledge protocols can be expensive in practice, and therefore, weaker notions of zero-knowledge proofs are studied. One such notion is called *honest-verifier zero knowledge* (HVZK) proofs. Here, the simulator should be able to simulate the interaction between an honest prover and an honest verifier. As a result, the simulator does not need to rewind the verifier — it can feed the randomness to the verifier, and can prepare the simulated transcript based on this randomness.

- Next, we studied a different soundness property. Here, we want the prover to demonstrate that it knows a witness, and protocols satisfying this notion are called *proofs of knowledge* (PoK). This is formally captured via an extractor who, given a prover that can convince the verifier, can 'extract' a witness from such a prover. We saw a ZK PoK protocol for graph hamiltonicity (with constant knowledge-of-soundness error), and a HVZK PoK protocol for the discrete log problem.
- Finally, we saw a constant-round zero-knowledge protocol for NP. The protocol was a simple modification of Blum's protocol repeated in parallel. At the start of the protocol, the verifier commits to its challenge bits, then the prover and verifier participate in the parallel-repetition of Blum's protocol. The analysis was not-so-simple, since we had to deal with aborting verifiers. This protocol required expected polynomial time simulation (and strict polynomial time simulation is believed to be unlikely).

5.13 *Missing Proofs*

First, we present a formal definition of zero-knowledge proofs (with auxiliary information).

Definition 5.25 (Zero knowledge w.r.t Auxiliary Information, Formal)**.** *Let* $(\mathbf{P}, \mathbf{V})$ *be an interactive proof system for language* $\mathsf{L} = (\mathsf{L}_{\text{yes}}, \mathsf{L}_{\text{no}})$*. We say that the proof system satisfies computational zero knowledge if for every (possibly malicious) p.p.t. verifier* $\mathbf{V}^*$ *and polynomial* p*, there exists a p.p.t. simulator* $\mathbf{S}$ *such that for every distinguisher* $\mathbf{D}^*$*, there exists a negligible function* μ *such that for all* $x \in \mathsf{L}_{\text{yes}}$*, every* $z \in \{0,1\}^{p(|x|)}$*,*

$$\Pr\left[\mathbf{D}^*\left(\mathsf{view}\langle \mathbf{P}, \mathbf{V}^*(z)\rangle(x)\right) = 1\right] - \Pr\left[\mathbf{D}^*\left(\mathbf{S}(x,z)\right) = 1\right] \leq \mu(|x|)$$

where the probabilities are taken over the randomness used by $\mathbf{P}, \mathbf{V}^*, \mathbf{S}$ *and* $\mathbf{D}^*$*.* ◊

We now prove Claim 5.12 by showing that if the probability of $\mathbf{S}$ outputting $\perp$ is not negligibly close to $(1/2)^n$, then we can construct an adversary $\mathcal{A}$ which can be used to break the hiding property of the commitment scheme.

Proof. Let us assume, that the probability of $\mathbf{S}$ outputting $\perp$ is $(1/2)^n + \epsilon$ for some verifier $\mathbf{V}^*$, where ϵ is not a negligible function in n. Note that $\mathbf{S}$ ouputs $\perp$ iff $b' \neq b$ for each of the n iterations of $\mathbf{S}$. Using this, we can show that the probability of $\mathbf{V}_2^*$ outputting $1 - b$ is significantly different from $1/2$, say $1/2 + \epsilon'$. Note that this probability will be the same for each $k \in [n]$ since b and r_2 are uniformly sampled. This can be easily be shown by contradiction and using binomial expansion for the product of probabilities.

Infact, the probability of $b' \neq b$ will be negligibly close to 1. Readers are encouraged to work out the details.

Thus, the probability of $\mathbf{S}$ outputting $\perp$ is negligibly close to $(1/2)^n$. This is a contradiction to ϵ not being a negligible function in n and therefore, $\Pr[b' \neq b]$ must be $1/2 + \epsilon'$. Now, we show that we can construct an adversary $\mathcal{A}$ that can break the hiding property of the commitment scheme with probability $1/2 + \epsilon'$,

Adversary $\mathcal{A}$: Description of adversary that breaks the hiding property of commitment scheme using $\mathbf{S}$ and $\mathbf{V}_2^*$

1. $\mathcal{A}$ picks a uniformly random permutation $\pi \leftarrow S_n$ and computes $H = \pi(G)$.
2. $\mathcal{A}$ sends $m_0 = \{A_H[i,j]\}_{i\in[n],j\in[n]}$ and $m_1 = \{1\}_{i\in[n],j\in[n]}$ to the challenger and receives $\{c_{i,j}\}_{i\in[n],j\in[n]}$ as the response.
3. $\mathcal{A}$ samples a uniformly random $r_2 \leftarrow \{0,1\}^t$ and computes $b' = \mathbf{V}_2^*\left(x = G, \{c_{i,j}\}, r_2\right)$ and sends $1 - b'$ to the challenger.

Now, the probability of $\mathcal{A}$ winning the hiding game is given as,

$$\begin{aligned}\mathsf{Pr}\left[\mathcal{A} \text{ wins hiding game}\right] &= \mathsf{Pr}\left[\mathbf{V}_2^* \text{ outputs } b' \neq b\right] \\ &= \frac{1}{2} + \epsilon'\end{aligned}$$

Thus, we have shown a contradiction to our initial assumption that the commitment scheme is computationally hiding (since the probability of $\mathcal{A}$ winning the hiding game is not negligibly close to $1/2$). Therefore, the probability of $\mathbf{S}$ outputting $\perp$ must be negligiby close to $(1/2)^n$. This completes the proof of Claim 5.12. □

Thus, we have shown that the simulator $\mathbf{S}$ only outputs $\perp$ with negligible probability. Now we need to show that the transcript outputted by $\mathbf{S}^{\mathbf{V}^*}$ on inputs x, z is computationally indistinguishable from $\mathsf{view}\langle\mathbf{P}(w), \mathbf{V}^*(z)\rangle(x)$, i.e., Claim 5.13.

Proof. We use the `Hybrid-Simulator` described in the proof-sketch of Claim 5.13. We first show that $\mathsf{view}\langle\mathbf{P}(w), \mathbf{V}^*(z)\rangle(x)$ is statistically indistinguishable from $\mathbf{HybS}^{\mathbf{V}^*}(x, \mathbf{w})$. Note that $\mathbf{HybS}$ only outputs $\perp$ if $b' \neq b$ for all $k \in [n]$. This only happens with probability $(1/2)^n$. In the case when $\mathbf{HybS}$ does not output $\perp$, the transcript is exactly same as in the case of $\mathsf{view}\langle\mathbf{P}(w), \mathbf{V}^*(z)\rangle(x)$. Therefore, any distinguisher $\mathcal{D}$ only has a negligible advantage when it attempts to distinguish between $\mathbf{HybS}$ and $\mathsf{view}\langle\mathbf{P}(w), \mathbf{V}^*(z)\rangle(x)$.

Now, we consider the computational distance between $\mathbf{HybS}^{\mathbf{V}^*}$ and $\mathbf{S}^{\mathbf{V}^*}$. For simplicity of the argument, we will show that they are computationally indistinguishable when $n = 1$. The same argument works for any n since the change in n only leads to a decrease in the probability of the simulators outputting $\perp$.

We will show that if there exists a distinguisher $\mathcal{D}(x, \mathbf{w})$ that can distinguish between $\mathbf{HybS}^{\mathbf{V}^*}(x, \mathbf{w})$ and $\mathbf{S}^{\mathbf{V}^*}(x)$ with probability $1/2 + \epsilon''$ then we can construct an adversary that can break the computational hiding of the commitment scheme with probability $1/2 + \epsilon'' \cdot (1/2 - \mu(n))$. The adversary $\mathcal{A}$ is defined as,

Adversary $\mathcal{A}(x, \mathbf{w} = (v_1, v_2, \ldots, v_n))$**:** Description of adversary that breaks the hiding property of commitment scheme using distinguisher $\mathcal{D}(x, \mathbf{w})$

1. $\mathcal{A}$ picks a uniformly random permutation $\pi \leftarrow S_n$ and computes $H = \pi(G)$.
2. $\mathcal{A}$ computes $w_i = \pi(v_i)$ for $i \in [n]$ and the set $E' = ([n] \times [n]) \setminus \{(w_i, w_{i+1}) | i \in [n-1]\}$
3. $\mathcal{A}$ sends $m_0 = \{A_H[i,j]\}_{(i,j)\in E'}$ and $m_1 = \{1\}_{(i,j)\in E'}$ to the challenger and receives $\{c_{i,j}\}_{i\in[n],j\in[n]}$ as the response.
4. $\mathcal{A}$ computes $c_{w_i,w_{i+1}} = \mathsf{Commit}(A_H[w_i, w_{i+1}]; r_{w_i,w_{i+1}})$ for $i \in [n-1]$ and sets $\mathsf{op} = \left\{\left(w_i, r_{w_i,w_{i+1}}\right) | i \in [n-1]\right\}$.
5. $\mathcal{A}$ samples a uniformly random $r_2 \leftarrow \{0,1\}^t$ and computes $b' = \mathbf{V}_2^*\left(x = G, \{c_{i,j}\}, r_2\right)$.
6. If $b' = 1$, $\mathcal{A}$ forwards $\left(\left(\{c_{i,j}\}, 1, \mathsf{op}\right), (r_2, r_4)\right)$ to $\mathcal{D}$ and receives b'' which it forwards to the challenger. Else $\mathcal{A}$ sends a uniformly random bit to the challenger.

We use the result of Claim 5.12 to get the probability of $b' = 1$ to be at least $1/2 - \mu(n)$. Also note that $\mathcal{A}$ only uses the distinguisher for the transcripts which have $b = b' = 1$ since the transcripts are exactly identical for then other cases and hence the probability of any distinguisher guessing the distribution correctly will be 1/2. Therefore, the advantage that $\mathcal{D}$ has comes only from the case when $b = 1$ and that is why $\mathcal{A}$ uses the distinguisher only for those cases Now, consider the probability of $\mathcal{A}$ winning the hiding game,

$$\begin{aligned}
\mathsf{Pr}\left[\mathcal{A} \text{ wins hiding game}\right] &= \mathsf{Pr}\left[b' = 1\right] \cdot \mathsf{Pr}\left[\mathcal{D} \text{ distinguishes correctly}\right] \\
&\quad + \mathsf{Pr}\left[b' = 0\right] \cdot \frac{1}{2} \\
&\geq \left(\frac{1}{2} - \mu(n)\right) \cdot \left(\frac{1}{2} + \epsilon''\right) + \left(\frac{1}{2} + \mu(n)\right) \cdot \frac{1}{2} \\
&\geq \frac{1}{2} + \epsilon''\left(\frac{1}{2} - \mu(n)\right)
\end{aligned}$$

Thus, this is a contradiction to our assumption that the commitment scheme is computationally hiding and therefore we cannot have any computationaly distinguisher that can distinguish between the two transcripts generated by $\mathbf{HybS}^{\mathbf{V}^*}$ and $\mathbf{S}^{\mathbf{V}^*}$. Therefore, the advantage any distinguisher has when distinguishing between $\mathsf{view}\langle \mathbf{P}(\mathbf{w}), \mathbf{V}^*(z)\rangle(x)$ and $\mathbf{S}^{\mathbf{V}^*}(x, z)$ will be,

$$\begin{aligned}
&\mathsf{Pr}\left[\mathbf{D}^*\left(\mathsf{view}\langle \mathbf{P}(w), \mathbf{V}^*(z)\rangle(x)\right) = 1\right] - \mathsf{Pr}\left[\mathbf{D}^*\left(\mathbf{S}(x)\right) = 1\right] \\
&= \mathsf{Pr}\left[\mathbf{D}^*\left(\mathsf{view}\langle \mathbf{P}(w), \mathbf{V}^*(z)\rangle(x)\right) = 1\right] - \mathsf{Pr}\left[\mathbf{D}^*\left(\mathbf{HybS}^{\mathbf{V}^*}(x,z)\right) = 1\right] \\
&\quad + \mathsf{Pr}\left[\mathbf{D}^*\left(\mathbf{HybS}^{\mathbf{V}^*}(x,z)\right) = 1\right] - \mathsf{Pr}\left[\mathbf{D}^*\left(\mathbf{S}(x,z)\right) = 1\right] \\
&= \left(\frac{1}{2^n} + \mu_1(n)\right) + \left(\frac{1}{2^n} + \mu_2(n)\right) \\
&= \mu_3(n)
\end{aligned}$$

Therefore, we have shown that the two distributions are negligibly close for

any distinguisher and therefore, the proposed protocol for Graph Hamiltonicity satisfies zero-knowledge property and hence is a ZKP. □

Part III: Dealing with Quantum Adversaries

6 Introduction to Quantum Computing

In this section, we present a quick overview of quantum computing. These notes are not complete or comprehensive (but hopefully should suffice for our lectures). For a more detailed introduction to quantum computing, please refer to these notes or these lecture videos.

6.1 *Postulates of Quantum Computing*

Lecture 11: February 10th, 2023

We will begin this section with the postulates of quantum computing.

1. Any quantum state over n qubits can be represented as a unit-norm vector $\mathbf{v} \in \mathbb{C}^{2^n}$. We will use $|v\rangle$ to represent this (column) vector, and $\langle v|$ to represent $\mathbf{v}^\dagger$. These are known as *pure states*. We can also have a *mixture* of states. In a mixture, we have a set of quantum states, and each state has an associated probability. For now, we will represent this as follows: $\{|\psi_i\rangle, p_i\}_{i\in[r]}$, where $p_i \geq 0$ and $\sum_i p_i = 1$.

   If our state is $|\psi_1\rangle$ with probability p and $|\psi_2\rangle$ with probability $1-p$, then we cannot think of this state as $p|\psi_1\rangle + (1-p)|\psi_2\rangle$. This quantity may not even be a unit-norm vector.

2. If $|\psi_1\rangle$ is a state over n_1 qubits and $|\psi_2\rangle$ is a state over a separate set of n_2 qubits, then the combined state (over $n_1 + n_2$ qubits) is $|\psi_1\rangle \otimes |\psi_2\rangle$. Not every quantum state over $n_1 + n_2$ qubits can be decomposed into this form. For instance, consider the state $|\mathsf{EPR}\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$. This state cannot be written as $|\psi_1\rangle \otimes |\psi_2\rangle$, where $|\psi_1\rangle$ and $|\psi_2\rangle$ are single-qubit states. States which cannot be decomposed in this manner are called *entangled states*.

   The EPR pair is named after Einstein-Podolsky-Rosen.

3. There are two kinds of quantum operations: unitary operation and measurements. A unitary operation is represented using a unitary matrix $\mathbf{U}$, and if this operation is applied on a state $|\psi\rangle$, the resulting state after the operation is $|\psi'\rangle = \mathbf{U}|\psi\rangle$. Since unitaries are invertible, given the state $|\psi'\rangle$, we can recover $|\psi\rangle$.

   Measurements: let us first consider measurements on single qubits. A measurement operation applied on a qubit $|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle$ outputs a classical bit. It outputs 0 with probability $|\alpha_0|^2$ and 1 with probability $|\alpha_1|^2$. After the measurement, if the classical output was bit b, then the quantum state *collapses* to $|b\rangle$. We can think of the resulting state as a mixed state $\{(|0\rangle, |\alpha_0|^2), (|1\rangle, |\alpha_1|^2)\}$.

   More generally, consider the measurement operation on an n qubit state $\sum_x \alpha_x |x\rangle$. This outputs an n-bit classical string $\mathbf{x}$ with probability $|\alpha_x|^2$. If the classical string output is $\mathbf{x}$, then the quantum state collapses to $|x\rangle$. Again, we can think of the final state as a mixed state $\{|x\rangle, |\alpha_x|^2\}_{x\in\{0,1\}^n}$.

   We can also consider *partial measurements*, where we measure a subset of the qubits. Since it is a bit complicated (notationally) to describe the most general version for n-qubit systems, I will just describe it for single-qubit measurement in an n-qubit system. Let $|\psi\rangle = \sum_x \alpha_x |x\rangle$ be an n-qubit

(pure) state. If we measure just the i^{th} qubit, then we see a classical outcome (which is 0 or 1), and the state collapses to $|\psi_0\rangle$ or $|\psi_1\rangle$ described as follows.

Let S_0 (resp. S_1) denote the set of all n-bit strings where the i^{th} bit is 0 (resp. 1).

$$\text{outcome } b \text{ w.p. } \textstyle\sum_{x\in S_b} |\alpha_x|^2; |\psi_b\rangle = \frac{\sum_{x\in S_b} \alpha_x |x\rangle}{\sqrt{\sum_{x\in S_b} |\alpha_x|^2}}$$

4. An operation applied on one set of qubits does not affect the remaining qubits. Therefore, if operation U is applied on the first n_1 qubits, and operation V is applied on the remaining qubits, then these two operations commute.

Exercise 6.1. *Consider the following game between a challenger and an adversary. The adversary sends a quantum state $|\psi\rangle$ (of its choice). The challenger tosses a coin $b \leftarrow \{0,1\}$. If $b = 0$, it sends back $|\psi\rangle$, else it performs a measurement on $|\psi\rangle$ and sends the mixed state. The adversary must guess b. What is the adversary's probability of success?*

Exercise 6.2. *Consider the following game between a challenger and an adversary. The challenger tosses a coin $b \leftarrow \{0,1\}$. If $b = 0$, it sends $|0\rangle$, else it sends $|+\rangle$. The adversary must guess b. What is the adversary's probability of success?*

6.2 *Quantum Circuits*

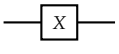
Lecture 12: February 17th, 2023

In this section, we will present a few commonly used quantum gates on one,two and three qubits. We will often describe these gates in terms of their action on the *computational basis states* $\{|x\rangle\}_{x\in\{0,1\}^n}$. Suppose a unitary maps $|x\rangle$ to $|\psi_x\rangle$ for all $x \in \{0,1\}^n$. Then note that the unitary is equal to $\sum_x |\psi_x\rangle\langle x|$.

Recall, $\langle x|$ denotes $(|x\rangle)^\dagger$.

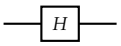
Single Qubit Gates

- X gate: this is analogous to the 'NOT' gate. It maps $|0\rangle$ to $|1\rangle$ and $|1\rangle$ to $|0\rangle$. The matrix corresponding to this gate is



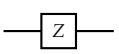
$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = |1\rangle\langle 0| + |0\rangle\langle 1|$$

- H gate: It maps $|0\rangle$ to $|+\rangle$ and $|1\rangle$ to $|-\rangle$. The matrix corresponding to this gate is



$$H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = |+\rangle\langle 0| + |-\rangle\langle 1|$$

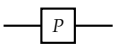
- Z gate: It maps $|0\rangle$ to $|0\rangle$ and $|1\rangle$ to -$|1\rangle$. The matrix corresponding to this



gate is

$$Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = |0\rangle\langle 0| - |1\rangle\langle 1|$$

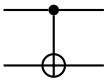
- P gate: It maps $|0\rangle$ to $|0\rangle$ and $|1\rangle$ to $i|1\rangle$. The matrix corresponding to this gate is



$$P = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix} = |0\rangle\langle 0| + i|1\rangle\langle 1|$$

Two Qubit Gates

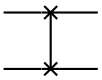
- CNOT gate: this is the 'controlled NOT' gate. For the computational basis states, it flips the second bit if the first one is 1.



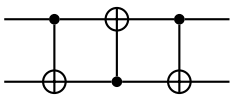
$$\text{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} = |00\rangle\langle 00| + |01\rangle\langle 01| + |11\rangle\langle 10| + |10\rangle\langle 11|$$

- SWAP gate: This gate is represented by the matrix

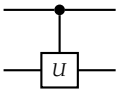
$$|00\rangle\langle 00| + |10\rangle\langle 01| + |01\rangle\langle 10| + |11\rangle\langle 11|$$



Note that the SWAP gate can be implemented using three CNOT gates as follows:

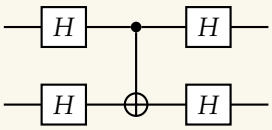


- Controlled U gate: on the computational basis states, it applies U on the second qubit if the first qubit is 1.

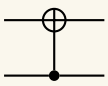


Exercise 6.3. *Implement the controlled Z gate using* CNOT *and H gates.*

Exercise 6.4. *Consider the following quantum circuit:*



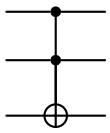
Show that this circuit is identical to



Three Qubit Gates

The most popular three qubit gate is the Toffoli gate. On the computational basis states, it flips the third bit only if the first two are equal to 1. The matrix corresponding to this gate is

$$|000\rangle\langle 000| + |001\rangle\langle 001| + |010\rangle\langle 010| + |011\rangle\langle 011| \\ + |100\rangle\langle 100| + |101\rangle\langle 101| + |111\rangle\langle 110| + |110\rangle\langle 111|$$



Exercise 6.5. *This exercise is related to the task of encrypting a quantum state using classical bits.*
Consider the following game between a challenger and an adversary. The adversary prepares a single qubit state $|\psi\rangle$ and sends it to the challenger. The challenger picks two uniformly random bits a, b, and does the following:
If $a = b = 0$, the challenger sends $|\psi\rangle$.
If $a = 0$, $b = 1$, the challenger sends $Z|\psi\rangle$.
If $a = 1$, $b = 0$, the challenger sends $X|\psi\rangle$.
Finally, if $a = b = 1$, the challenger sends $ZX|\psi\rangle$.
The adversary, on receiving this state, applies a unitary of its choice, followed by measurement. What is the probability of seeing 0 after measurement?

Universal Quantum Computation

Classically, we know that any constant-arity deterministic function can be expressed using NAND gates. In the quantum setting, such exact simulation is not possible. We cannot have a fixed set of quantum gates that can exactly simulate all unitaries that act on a single qubit. Instead, we aim for approximate simulation, and this is possible with a fixed set of gates. This is captured by the following theorem (which we will not prove).

Even in the randomized setting, we cannot perfectly simulate all distributions. Take, for instance, a distribution that outputs 1 with probability $1/\pi$. We cannot simulate this with a randomness source that outputs 0 or 1 w.p. 1/2.

Theorem 6.1. *Let $S = \{\mathsf{Toffoli}, H, P\}$. For any unitary U acting on a constant number of qubits, there exists a quantum circuit C of size $\mathrm{polylog}(1/\epsilon)$ composed of gates from S such that U and C are ϵ-close (in matrix norm).*

Such gate sets (known as *universal gate sets*) allow us to define *efficient* quantum computation. If a quantum computation that can be approximated using polynomially many gates from a universal gate set, then we say that this computation is efficient.

6.3 *Basic Quantum Algorithms*

Lecture 13: February 21st, 2023

In this section, we will see a few basic quantum algorithms that outperform classical algorithms. Most of these results will be in certain *idealized* computational models. The objective of this section is to see some commonly used 'quantum tricks'.

The Oracle Model of Computation: In this computational model, the algorithms have oracle access to a classical $f : \{0,1\}^n \to \{0,1\}^m$. The algorithm can send inputs to f, and it receives the corresponding outputs.

- If the algorithm is classical, then it sends an input x, and receives $f(x)$.
- In the quantum setting, we assume that the algorithm prepares a quantum state $|\psi_{\text{in}}\rangle = \sum_{x,y} \alpha_{x,y} |x\rangle |y\rangle$ over two quantum registers and sends these two registers to the oracle. The oracle sends back $|\psi_{\text{out}}\rangle = \sum_{x,y} \alpha_{x,y} |x\rangle |y \oplus f(x)\rangle$. In other words, we are defining a unitary U_f that maps $|x\rangle |y\rangle$ to $|x\rangle |y \oplus f(x)\rangle$, and the oracle, on receiving $|\psi_{\text{in}}\rangle$, sends back the quantum state after applying U_f. We will call these *superposition queries*.

The objective here is to learn some property about the function f, by making (superposition) queries to f. The efficiency measure (based on which we will compare the classical and quantum algorithms) in this model is the number of queries made to the oracle.

Deutsch-Josza Problem

The first problem in this category is the Deutsch-Josza problem. Here, we have a function $f : \{0,1\} \to \{0,1\}$, and we wish to determine whether $f(0) = f(1)$. Classically, we will require two queries, but a quantum algorithm can determine this using only one superposition query!

Since we are allowed only one superposition query, we don't have much option: we can prepare a general state

$$|\psi_{\text{in}}\rangle = \alpha_{00} |0\rangle |0\rangle + \alpha_{01} |0\rangle |1\rangle + \alpha_{10} |1\rangle |0\rangle + \alpha_{11} |1\rangle |1\rangle$$

send this as a query, then finally apply some unitary followed by measurement.

First, let us note the state after the query:

$$|\psi_{\text{out}}\rangle = \alpha_{00} |0\rangle |f(0)\rangle + \alpha_{01} |0\rangle \left|\overline{f(0)}\right\rangle + \alpha_{10} |1\rangle |f(1)\rangle + \alpha_{11} |1\rangle \left|\overline{f(1)}\right\rangle$$

Depending on whether $f(0) = f(1)$ or not, these terms can be grouped together differently (and this will guide us in setting the amplitudes appropriately).

If $f(0) = f(1)$, then

$$|\psi_{\text{out}}\rangle = (\alpha_{00} |0\rangle + \alpha_{10} |1\rangle) |f(0)\rangle + (\alpha_{01} |0\rangle + \alpha_{11} |1\rangle) \left|\overline{f(0)}\right\rangle$$

On the other hand, if $f(0) \neq f(1)$, then

$$|\psi_{\text{out}}\rangle = (\alpha_{00} |0\rangle + \alpha_{11} |1\rangle) |f(0)\rangle + (\alpha_{01} |0\rangle + \alpha_{10} |1\rangle) \left|\overline{f(0)}\right\rangle$$

The grouping above suggests what might be a good candidate for distinguishing between these two scenarios. Suppose we set $\alpha_{00} = \alpha_{11} = -\alpha_{10} = -\alpha_{01} = 1/\sqrt{2}$, then we get the following: If $f(0) = f(1)$, then

$$|\psi_{\text{out}}\rangle = (|0\rangle - |1\rangle) \left(|f(0)\rangle - \left|\overline{f(0)}\right\rangle\right)$$

On the other hand, if $f(0) \neq f(1)$, then

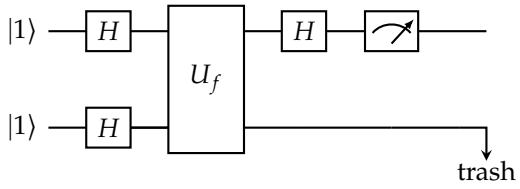
$$|\psi_{\text{out}}\rangle = (|0\rangle + |1\rangle) \left(|f(0)\rangle - \left|\overline{f(0)}\right\rangle\right)$$

Now, it is clear what we must do to distinguish the two scenarios. We can apply the Hadamard gate (H) on the first register and measure it. Depending on the outcome, we can decide whether $f(0) = f(1)$ or not.

Finally, we also need to describe how to create the initial state

$$|\psi_{\text{in}}\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle|0\rangle - |0\rangle|1\rangle - |1\rangle|0\rangle + |1\rangle|1\rangle\right) = |-\rangle|-\rangle .$$

This brings us to our single-query quantum circuit (described in terms of U_f):



This quantum circuit is yet another example where our 'classical intuition' is misleading. If we think about this circuit classically, then the first register is the input register, the second one is the output register, and the unitary U_f 'writes' the answer $f(x)$ in the output register. However, we are not using the second register at all! After the evaluation U_f, we are ignoring the second register (and just applying Hadamard on the first register, followed by measurement).

The phase-kickback trick: The following observation will perhaps help to explain this strange situation. Let $f : \{0,1\}^n \to \{0,1\}$ be any classical function, and consider the action of U_f on $|x\rangle|-\rangle$.

$$U_f|x\rangle|-\rangle = |x\rangle\left(|f(x)\rangle - \left|\overline{f(x)}\right\rangle\right) = (-1)^{f(x)}|x\rangle|-\rangle$$

Trick #1: the function evaluation can be put in the phase.

By applying U_f on $|x\rangle|-\rangle$, we are essentially putting the f evaluation in the phase, while the second register remains unchanged. Therefore, having access to U_f is equivalent to having oracle access to a 'phase oracle' P_f that maps $|x\rangle$ to $(-1)^{f(x)}|x\rangle$. This also explains why we didn't use the second register. It was used to put the answer in-the-phase, and once that is done, we can restrict ourselves to the first register.

1-bit Deutsch-Josza via the Phase Oracle: Let us solve the same problem using the phase oracle. Now, we don't need two qubits. The general single-qubit state is $|\psi_{\text{in}}\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle$. When the phase oracle for function f is applied to $|\psi_{\text{in}}\rangle$, the resulting state is

$$|\psi_{\text{out}}\rangle = \alpha_0(-1)^{f(0)}|0\rangle + \alpha_1(-1)^{f(1)}|1\rangle$$

If $f(0) = f(1)$, then the resulting state is

$$(-1)^{f(0)}\left(\alpha_0|0\rangle + \alpha_1|1\rangle\right)$$

If $f(0) \neq f(1)$, then the resulting state is

$$(-1)^{f(0)} (\alpha_0 |0\rangle - \alpha_1 |1\rangle)$$

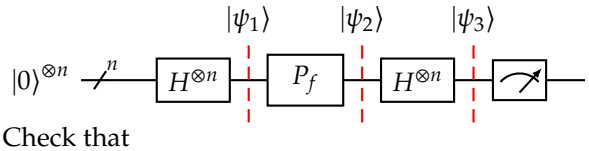
After applying the phase oracle, we need to apply an appropriate unitary followed by measurement. The unitary followed by measurement should distinguish between $\alpha_0 |0\rangle + \alpha_1 |1\rangle$ and $\alpha_0 |0\rangle - \alpha_1 |1\rangle$ (note that we are ignoring the 'global phase' $(-1)^{f(0)}$).At this point, I hope it is clear what our unitary should be.

We can ignore the global phase because $|x\rangle$ and $-|x\rangle$ represent the same quantum state.

The n-variate Deutsch-Josza Problem

Let us now consider an n-variate generalization of the above problem. We have a function $f : \{0,1\}^n \to \{0,1\}$, with the guarantee that either f is a constant function, or f maps exactly half the inputs to 0, and the remaining half to 1 (that is, f is a *balanced* function). Any deterministic classical algorithm will require at least $2^{n-1} + 1$ queries (in the worst case). Any randomized classical algorithm that gives the correct answer with probability $\Omega(1 - 1/n)$ requires superconstant queries to the oracle. However, there exists a simple quantum circuit that solves this problem (with probability 1) by making exactly one query.

Here, we will assume the algorithm has access to phase oracle P_f corresponding to function f.



Check that

$$|\psi_2\rangle = \frac{1}{2^{n/2}} \left(\sum_x (-1)^{f(x)} |x\rangle \right)$$

If f is a constant function, then this state is $|+\rangle^{\otimes n}$, and therefore, applying $H^{\otimes n}$ results in $|0\rangle^{\otimes n}$. Therefore, if f is a constant function, then the measurement outputs 0^n with probability 1.

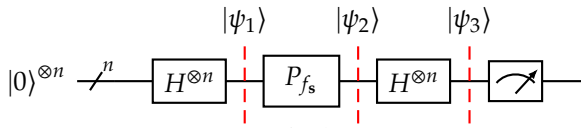
If f is a balanced function, then note that the state $|\psi_2\rangle$ is orthogonal to $|+\rangle^{\otimes n}$. Since $H^{\otimes n}$ is a unitary matrix, it preserves orthogonality, and therefore $|\psi_3\rangle$ is orthogonal to $|0\rangle^{\otimes n}$. As a result, upon measurement, with probability 1, we will not get 0^n.

Bernstein-Vazirani Problem

Lecture 14: February 24th, 2023

In this problem, we have access to an oracle $f_{\mathbf{s}} : \{0,1\}^n \to \{0,1\}$ that maps $\mathbf{x}$ to $\mathbf{x} \cdot \mathbf{s}$ for some bit-vector s. The objective is to learn $\mathbf{s}$ given oracle access to $f_{\mathbf{s}}$. A classical algorithm requires $\Theta(n)$ queries to recover $\mathbf{s}$. However, there exists a quantum algorithm that can achieve this using just one query (and therefore, a polynomial speedup in the oracle model).

Our quantum circuit will be exactly identical to the one used for (n-variate) Deutsch-Josza problem.



Let us consider the state $|\psi_2\rangle$. This is equal to

$$|\psi_2\rangle = \frac{1}{2^{n/2}}\left(\sum_{\mathbf{x}}(-1)^{\mathbf{s}\cdot\mathbf{x}}|x\rangle\right)$$

We apply the Hadamard gate to each of these wires. The following claim describes the action of $H^{\otimes n}$ on $|x\rangle$.

Claim 6.2. *For any* $\mathbf{x} \in \{0,1\}^n$,

$$H^{\otimes n}|x\rangle = \frac{1}{2^{n/2}}\left(\sum_{\mathbf{y}}(-1)^{\mathbf{x}\cdot\mathbf{y}}|y\rangle\right)$$

The proof of this follows via induction on n. Using this claim, we can write down the state $|\psi_3\rangle$. However, there is a simple observation using which we can avoid the calculations for $|\psi_3\rangle$. Note that

$$H^{\otimes n}|s\rangle = \frac{1}{2^{n/2}}\left(\sum_{x}(-1)^{\mathbf{s}\cdot\mathbf{x}}|x\rangle\right) = |\psi_2\rangle$$

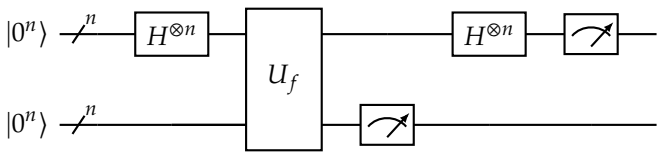
and also note that the inverse of $H^{\otimes n}$ is $H^{\otimes n}$. As a result, $|\psi_3\rangle = |s\rangle$, and therefore measurement of these n qubits outputs $\mathbf{s}$ as desired.

Simon's Problem

The next algorithm in this segment will illustrate an exponential speedup. Let $f_{\mathbf{s}}$ be an oracle which maps n bits to n bits, with the following guarantee: for any $\mathbf{x}, \mathbf{y} \in \{0,1\}^n$, $f_{\mathbf{s}}(\mathbf{x}) = f_{\mathbf{s}}(\mathbf{y})$ if and only if $\mathbf{y} = \mathbf{x} \oplus \mathbf{s}$. There exists a classical randomized algorithm that can find $\mathbf{s}$ by making $\sqrt{2^n}$ queries to the oracle. The algorithm queries the oracle on uniformly random inputs until it finds a collision. In fact, this is the best possible algorithm for this problem (in the oracle model). However, using superposition queries to U_f, we can compute s using polynomially many queries. At a high level, the quantum algorithm finds uniformly random vectors orthogonal to $\mathbf{s}$. Given sufficiently many such linearly independent equations, we can compute $\mathbf{s}$.

Finding a vector orthogonal to $\mathbf{s}$: Consider the circuit described in Figure 4.

Figure 4: Quantum Circuit to obtain one equation $x \cdot k_0 = 0$



Suppose the classical output after measuring the second register is $\mathbf{y}$. Then the state of the first register is $\sum_{\mathbf{x}:f(\mathbf{x})=\mathbf{y}} |x\rangle$. Since there are exactly two strings $\mathbf{x}_0, \mathbf{x}_1 = \mathbf{x}_0 \oplus \mathbf{s}$ such that $f(\mathbf{x}_0) = f(\mathbf{x}_1) = \mathbf{y}$, the state of the first register is

$$|\psi_1\rangle = \frac{1}{\sqrt{2}} \left(|x_0\rangle + |x_0 \oplus \mathbf{s}\rangle \right).$$

If we just measure this register, then we will get a string that maps to $\mathbf{y}$. However, that is not enough to get $\mathbf{s}$. If we repeat the entire experiment again, then the measurement of the second register may ouptut a different string $\mathbf{y}'$. Instead, we apply Hadamard to the first register. This gives us the following state:

$$\begin{aligned} |\psi_2\rangle &= \frac{1}{\sqrt{2 \cdot 2^n}} \sum_{\mathbf{z}} \left((-1)^{\mathbf{x}_0 \cdot \mathbf{z}} |z\rangle + (-1)^{\mathbf{x}_0 \cdot \mathbf{z} + \mathbf{s} \cdot \mathbf{z}} |\mathbf{z}\rangle \right) \\ &= \frac{1}{\sqrt{2 \cdot 2^n}} \sum_{\mathbf{z}} \left((-1)^{\mathbf{x}_0 \cdot \mathbf{z}} + (-1)^{\mathbf{x}_0 \cdot \mathbf{z} + \mathbf{s} \cdot \mathbf{z}} \right) |\mathbf{z}\rangle . \end{aligned}$$

Note that the only terms in this superposition are the ones that are orthogonal to $\mathbf{s}$. As a result, when we measure $|\psi_2\rangle$, we get a uniformly random vector (in $\mathbb{Z}_2^n$) orthogonal to $\mathbf{s}$. Suppose we have obtained k such linearly independent equations. There are at most 2^k vectors (in $\mathbb{Z}_2^n$) that are linearly dependent on these k vectors. However, since we get a uniformly random orthogonal vector in each run, with probability at least $1/2$, we get a new linearly independent vector.

We are working in the vector space $\mathbb{Z}_2^n$ over $\mathbb{Z}_2$.

6.4 *Mixed States*

Lecture 16: March 3rd, 2023

So far, we have mostly dealt with pure quantum states. We have also encountered mixed states, where we have a set of pure states $|\psi_i\rangle$, and associated probabilities p_i such that $p_i \geq 0$ and $\sum_i p_i = 1$. However, this representation is not unique. For instance, consider the mixed states $\mathfrak{P}_0 = \left(\left(\frac{1}{2}, |0\rangle\right), \left(\frac{1}{2}, |1\rangle\right) \right)$ and $\mathfrak{P}_1 = \left(\left(\frac{1}{2}, |+\rangle\right), \left(\frac{1}{2}, |-\rangle\right) \right)$. Pick any unitary U acting on a single qubit. When we apply U on either of these states, followed by measurement, the probability of seeing 0 is identical in both cases.

We encountered something similar for pure states — multiplying a global phase (that is, $e^{i\theta}$ for some θ) does not change the pure state.

> **Exercise 6.6.** *Consider the mixed states $\mathfrak{P}_0 = \{(1/3, |0\rangle), (2/3, |1\rangle)\}$ and $\mathfrak{P}_1 = \{(1/3, |+\rangle), (2/3, |-\rangle)\}$. Is there a unitary followed by measurement that can distinguish between these two states?*

More generally, let $\mathfrak{P}_0 = \{(p_i, |\psi_i\rangle)\}_{i=1}^{k_0}$ and $\mathfrak{P}_1 = \{(q_i, |\phi_i\rangle)\}_{i=1}^{k_1}$ be two mixed states. In this notation, it is not immediately clear how to check if two states are identical. In this section, we will present a different way to express mixed states, one that will give a *unique representation* for every mixed state. Given any mixed state $\mathfrak{P} = \{(p_i, |\psi_i\rangle)\}_i$, we can consider the matrix $\sum_i p_i |\psi_i\rangle\langle\psi_i|$. In this section, we will show that this matrix representation gives every mixed state a unique representation, and this is referred to as the *density matrix* for the mixed state. For starters, check that if $|\psi\rangle$ is a pure state, then $|\psi'\rangle = e^{i\theta} |\psi\rangle$ and $|\psi\rangle$ have the same density matrix $|\psi\rangle\langle\psi|$ (hence, we can ignore the global

phase $e^{i\theta}$). Next, check that the density matrices for $\{(1/2, |0\rangle), (1/2, |1\rangle)\}$ and $\{(1/2, |+\rangle), (1/2, |-\rangle)\}$ are identical. Finally, check that the density matrices for $\mathfrak{P}_0 = \{(1/3, |0\rangle), (2/3, |1\rangle)\}$ and $\mathfrak{P}_1 = \{(1/3, |+\rangle), (2/3, |-\rangle)\}$ are not identical, and hence they are distinguishable.

Basic Properties of Density Matrices:

1. Hermitian: for any density matrix ρ, $\rho^\dagger = \rho$. Suppose $\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i|$. Note that $(|\psi_i\rangle\langle\psi_i|)^\dagger = |\psi_i\rangle\langle\psi_i|$, and since all p_i are non-negative reals, the conjugate-transpose of ρ is equal to ρ.

2. positive semi-definite (psd): A Hermitian matrix is positive semi-definite if $\langle v|\rho|v\rangle \geq 0$ for all $|v\rangle \in \mathbb{C}^{2^n}$. Suppose $\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i|$, then $\langle v|\rho|v\rangle = \sum_i p_i \langle v||\psi_i\rangle\langle\psi_i||v\rangle$, and note that $\langle v||\psi_i\rangle\langle\psi_i||v\rangle = |\langle v|\psi_i\rangle|^2$ is non-negative. Hence density matrices are positive semi-definite.

   Since density matrices are psd, they inherit all the 'nice' properties of psd matrices. We will use some of them later in the lecture.

3. Unit trace: The trace of a matrix is the sum of its diagonal entries. The trace operator, defined by $\mathsf{Tr}(\cdot)$ is linear. Therefore, if $\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i|$, then $\mathsf{Tr}(\rho) = \sum_i p_i \mathsf{Tr}(|\psi_i\rangle\langle\psi_i|)$. Finally, we use the fact that $\mathsf{Tr}(\mathbf{A}\cdot\mathbf{B}) = \mathsf{Tr}(\mathbf{B}\cdot\mathbf{A})$, and hence $\mathsf{Tr}(|\psi_i\rangle\langle\psi_i|) = \mathsf{Tr}(\langle\psi_i|\psi_i\rangle) = \mathsf{Tr}(1) = 1$.

In fact, the converse is also true: any matrix with the above properties (positive semi-definite and unit trace) is the density matrix for a mixed state. This is because any psd matrix ρ has non-negative eigenvalues $\{\lambda_i\}_i$, and moreover, there is an orthonormal basis $\{|v_i\rangle\}_i$ such that $\rho \cdot |v_i\rangle = \lambda_i |v_i\rangle$. Note that the trace of a matrix is equal to the sum of its eigenvalues. Since each $\lambda_i \geq 0$ and $\sum_i \lambda_i = 1$, we can think of $\{\lambda_i\}_i$ as probabilities and ρ as the mixed state $\{(\lambda_i, |v_i\rangle)\}_i$.

Exercise 6.7. *Consider the mixed state $\mathfrak{P} = \{(1/4, |0\rangle), (1/4, |1\rangle), (1/2, |+\rangle)\}$. Express $\mathfrak{P}$ as a mixture of at most two pure states.*

Operations on Mixed States

We discussed the effects of unitary and measurement operations on pure states. There is a natural extension of those effects when the state is a mixed state. Let $\mathfrak{P} = \{(p_i, |\psi_i\rangle)\}_i$ be a mixed state. On applying a unitary U, the new mixed state is $\mathfrak{P}' = \{(p_i, U|\psi_i\rangle)\}_i$. Similarly, suppose we perform a measurement on a mixed state. Then the probability of getting output $\mathbf{x}$ is $\sum_i p_i |\langle x|\psi_i\rangle|^2$.

In this section, we will define these operations on density matrices. This will also illustrate the uniqueness of the density matrix representation.

Unitary Operation: Let $\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i|$ be the density matrix corresponding to $\mathfrak{P} = \{(p_i, |\psi_i\rangle)\}_i$. As discussed above, when a unitary U is applied, the new mixed state is $\mathfrak{P}' = \{(p_i, U|\psi_i\rangle)\}_i$. Let us consider the density matrix representation for $\mathfrak{P}'$. This is simply

$$\rho' = \sum_i p_i U|\psi_i\rangle\langle\psi_i|U^\dagger = U\left(\sum_i p_i |\psi_i\rangle\langle\psi_i|\right)U^\dagger = U\rho U^\dagger$$

Note that this part shows that if two mixed states have the same density matrix, then after applying a unitary, they both will still have the same density matrix.

Measurement Operation: In this part, we will show that if two mixed states have the same density matrix, then they cannot be distinguished by any measurement (and hence the density matrix is a unique representation of the state).

Let $\mathfrak{P} = \{(p_i, |\psi_i\rangle)\}_i$ and $\mathfrak{P}' = \left\{\left(p'_i, |\psi'_i\rangle\right)\right\}_i$ be two mixed states with the same density matrix. As mentioned earlier, when $\mathfrak{P}$ is measured, we get output string $\mathbf{x}$ with probability $\sum_i p_i |\langle x|\psi_i\rangle|^2$. Similarly, when ρ' is measured, we get $\sum_i p'_i \left|\langle x|\psi'_i\rangle\right|^2$. Below, we will express these in terms of the density matrix, and therefore, if the density matrix is the same, then the measurement distribution is also identical.

$$\begin{aligned}
\Pr[\text{outcome is } \mathbf{x}] &= \sum_i p_i |\langle x|\psi_i\rangle|^2 \\
&= \sum_i p_i \langle x|\psi_i\rangle \cdot \langle \psi_i|x\rangle \\
&= \langle x| \left(\sum_i p_i |\psi_i\rangle\langle\psi_i| \right) |x\rangle \\
&= \langle x| \rho |x\rangle
\end{aligned}$$

Therefore, if two mixed states have the same density matrix, then any unitary operation followed by measurement results in identical output distribution. Hence, if two mixed states have the same density matrix, then they are indistinguishable. We can show a similar thing for partial measurements too (this will be included in one of the upcoming assignments).

Thanks to one of the students in class for pointing this out.

The converse is also true: if two mixed states $\mathfrak{P}_0$ and $\mathfrak{P}_1$ have identical distributions after applying a unitary followed by measurement, then the corresponding density matrices ρ_0 and ρ_1 are identical. The diagonal entries must be equal (otherwise there exists some output which has different post-measurement probabilities). For any off-diagonal position, we can find an appropriate unitary that ensures that the corresponding entries in ρ_0 and ρ_1 are identical.

Exercise 6.8. *Consider two density matrices ρ_0, ρ_1 of dimension 2 (that is, the mixed state is over a single qubit) that have identical diagonal entries, but different off-diagonal entries. Design an experiment (unitary followed by measurement) to differentiate between the two states.*

Note to reviewer: is the above explanation (together with the exercise) sufficient?

Purification of Mixed States

In one of the initial lectures, we mentioned that when we perform (partial) measurement on a pure state, we get a mixed state. Here, we will show that any mixed state can be obtained by performing partial measurement on a larger

system. Let $\mathfrak{P} = \{(p_i, |\psi_i\rangle)\}_{i=1}^{D}$ be a mixed state over n qubits, and let ρ be the corresponding density matrix. Since ρ is psd Hermitian, there exists an orthonormal basis $|v_i\rangle$ and associated eigenvalues λ_i such that $\rho = \sum_{j=1}^{2^n} \lambda_i |v_i\rangle\langle v_i|$.

Consider the following pure state on $2n$ qubits:

$$|\psi\rangle = \sum_{i=1}^{2^n} \sqrt{\lambda_i} |v_i\rangle |\mathsf{bin}(i)\rangle$$

When we measure the last n qubits, the resulting mixed state is $\{(\lambda_i, |v_i\rangle)\}$, and therefore the density state representation is ρ. The pure state $|\psi\rangle$ is not unique. For instance, consider $|\psi'\rangle = \sum_{i=1}^{2^n} \sqrt{\lambda_i} |v_i\rangle |v_i\rangle$. Here also, when we measure the last n qubits, the resulting density matrix is ρ.

Exercise 6.9. *Let $\mathfrak{P} = \{(1/2, |0\rangle), (1/2, |+\rangle)\}$. Following the above recipe, consider the state $|\psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle|0\rangle + |+\rangle|+\rangle)$. Does measuring the last qubit produce $\mathfrak{P}$?*

Projective Measurements

Lecture 17 (part 1): March 14th, 2023

So far, we have seen (full) measurements, and partial measurements. Partial measurements are a special case of projective measurements, which we describe next. The main idea behind this generalisation is to partition the space $\{0,1\}^n$ into m subsets $\{S_i\}_{i\in[m]}$. These subsets can be used to define m 'outcomes'. The i^{th} outcome happens if the measurement is in the set S_i. Such measurements are captured using m projective matrices $\{\mathbf{P}_i\}_i$, where

$$\mathbf{P}_i = \sum_{k\in S_i} |k\rangle\langle k|.$$

Note that $\mathbf{P}_i^2 = \mathbf{P}_i$, and $\sum_i \mathbf{P}_i = \mathbf{I}$. When a projective measurement is applied to a density matrix ρ, the probability of seeing outcome i is

$$\Pr[\text{ outcome} = i] = \sum_{k\in S_i} \langle k|\rho|k\rangle = \mathsf{Tr}(\mathbf{P}_i\rho).$$

Similar to partial measurements, when a projective measurement is applied to a quantum state, we can describe the residual state in terms of the original state and the measurement outcome. Suppose a projective measurement $\mathcal{P} = \{\mathbf{P}_i\}_i$ is applied to a pure state $|\psi\rangle$, and the resulting outcome is i. Then the residual state is a pure state, given by

$$|\psi_i\rangle = \frac{\mathbf{P}_i|\psi\rangle}{\|\mathbf{P}_i|\psi\rangle\|_2}.$$

Exercise 6.10. *Suppose a projective measurement $\mathcal{P} = \{\mathbf{P}_i\}_i$ is applied to a mixed state described by density matrix ρ. Conditioned on the measurement outcome being i, what is the residual mixed state?*

Note that there is nothing special about the basis $\{|k\rangle\}_{k\in\{0,1\}^n}$. We could have taken any orthonormal basis $\{\mathbf{u}_k\}_{k\in\{0,1\}^n}$, and defined sets $\{S_i\}_{i\in[m]}$, and projective matrices $\mathbf{P}_i = \sum_{k\in S_i} |u_k\rangle\langle u_k|$. In other words, we can take any set of m projective matrices such that $\sum_i \mathbf{P}_i = \mathbf{I}$, $\mathbf{P}_i \cdot \mathbf{P}_j = \mathbf{0}$, and define a projective measurement using these m projective matrices. The probability of seeing outcome i on measuring ρ is $\mathsf{Tr}(\mathbf{P}_i\rho)$.

'Quantum Probability'

Mixed states can be seen as a generalization of probability distributions. In this section, we will discuss some similarities between classical probability theory, and probability theory arising from quantum states. We will restrict our attention to discrete probability, where we have a universe $\Omega = \{0,1\}^n$. Given this universe, we define a probability distribution over Ω as a vector $\mathbf{p} \in \mathbb{R}^{|\Omega|}$ such that $\mathbf{p} \geq \mathbf{0}$ and $\|\mathbf{p}\|_1 = 1$. The probability distribution can also be represented as a psd diagonal matrix $\mathbf{D} = \mathsf{Diag}(\mathbf{p})$ such that $\mathsf{Tr}(\mathbf{D}) = 1$. In the quantum setting, a general mixed state is described using a psd matrix $\rho \in \mathbb{C}^{2^n\times 2^n}$ such that $\mathsf{Tr}(\rho) = 1$.

Events vs Projective Measurements: Next, in classical discrete probability, we can define a set of mutually exclusive, collectively exhaustive events $(E_1, \ldots, E_m)$ such that each $E_i \subseteq \Omega$, $E_i \cap E_j = \varnothing$ and $\bigcup_i E_i = \Omega$. We can describe the event E_i using a projection matrix $\mathbf{P}_i = \sum_{k\in E_i} |e_k\rangle\langle e_k|$ ($\mathbf{e}_i$ is the indicator vector where $\mathbf{e}_i[i] = 1$, and $\mathbf{e}_i[j] = 0$ for all $j \neq i$). Since the events are mutually exclusive, $\mathbf{P}_i \cdot \mathbf{P}_j = \mathbf{0}$ if $i \neq j$. Moreover, since the events are collectively exhaustive, $\sum_i \mathbf{P}_i = \mathbf{I}$.

Using this projective matrix notation, we can check that $\mathsf{Pr}_{\mathbf{p}}[E_i] = \|\mathbf{P}_i \cdot \mathbf{p}\|_1$. If we use the matrix representation of a probability distribution, then $\mathsf{Pr}_{\mathbf{p}}[E_i] = \mathsf{Tr}(\mathbf{P}_i \cdot \mathbf{D})$, where $\mathbf{D}$ is the diagonal matrix corresponding to $\mathbf{p}$. Hence, classical events can be described using special projective matrices $\{\mathbf{P}_1, \ldots, \mathbf{P}_m\}$ that are mutually orthogonal, and add up to $\mathbf{I}$.

Similarly, quantum events (which we call projective measurements) can be described using m projective matrices $\{\mathbf{P}_i\}$ where $\mathbf{P}_i \cdot \mathbf{P}_j = \mathbf{0}$ and $\sum_i \mathbf{P}_i = \mathbf{I}$. When applied to a density matrix ρ, the probability of seeing measurement i is $\mathsf{Tr}(\mathbf{P}_i \cdot \rho)$.

Statistical Distance vs Trace Distance: Given two probability distributions $\mathbf{p}_1$ and $\mathbf{p}_2$, we can define the statistical distance between these two distributions, which also captures the maximum distinguishing advantage. The statistical distance can also be defined using the matrix representation of probability distributions. Let $\mathbf{D}_b = \mathsf{Diag}(\mathbf{p}_b)$, then

$$\mathsf{SD}(\mathbf{p}_1, \mathbf{p}_2) = \frac{1}{2}\mathsf{Tr}(|\mathbf{D}_1 - \mathbf{D}_2|).$$

Similarly, for quantum states, we have the notion of 'trace distance'. More formally, given two density matrices ρ_1, ρ_2, the 'trace distance' between ρ_1 and ρ_2 is

$$\mathsf{TD}(\rho_1, \rho_2) = \frac{1}{2}\mathsf{Tr}(|\rho_1 - \rho_2|).$$

Just like the classical setting, this gives an upper bound on the probability of distinguishing between ρ_1 and ρ_2, and there exist measurements that achieve this bound (see Theorem 9.1 in [NC16] for a proof of this statement).

7 Quantum Adversaries against Symmetric Key Primitives

Lecture 15: February 28th, 2023

We discussed this topic in Lecture 15, before discussing mixed states in Lecture 16. However, in the notes, I have included mixed states in the previous section.

In COL759, we saw some symmetric key primitives such as pseudorandom functions/permutations, MACs, private key encryption. In all these primitives, the adversary had access to an oracle that received queries, and responded using a secret key. For instance, in the case of PRFs, the adversary could query on any input of its choice, and it received the PRF evaluation at that point. Finally, after polynomially many queries, the adversary had to decide whether it was interacting with a pseudorandom function or a truly random function.

In the quantum adversary setting, our algorithms are still classical (that is, if we are talking about pseudorandom functions, we still have a classical function F that takes a key K, an input string x and produces a classical bit string $y = F(K, x)$). However, we can have different security definitions. In the first definition, we still allow only classical queries to the oracle, but the adversary, after all its queries, can perform some quantum post-processing to decide whether the function is pseudorandom or truly random. A stronger definition is where we allow the adversary to make *superposition queries* to the oracle. Are these two definitions equivalent?

In this section, we show that these two definitions are **not** equivalent. Many popularly used constructions for these primitives would be secure as per the first definition (under some reasonable assumptions), but the construction is broken when an adversary is allowed superposition queries. These attacks use Simon's algorithm (discussed in Section 6.3).

The Even-Mansour Cipher

Let $\pi : \{0,1\}^n \to \{0,1\}^n$ be a publicly known permutation. Consider the following keyed permutation

$$F_\pi((k_1, k_2), x) = \pi(x \oplus k_1) \oplus k_2$$

This permutation operates over $\{0,1\}^n$, and is the building block for widely used keyed permutations such as AES. In the classical setting, the security of this construction is proven in the *ideal permutation model*. More formally, π is modeled as a uniformly random permutation. The adversary is given oracle access to π, together with oracle access to either $F_\pi((k_1, k_2), \cdot)$ or a uniformly random permutation, and it must guess whether it has oracle access to a pseudorandom permutation or a uniformly random permutation. If the adversary makes at most q queries to π and the pseudorandom/truly random oracle, then its distinguishing advantage is at most $O(q^2/2^n)$.

See [BS23] Section 4.7.3 for more details on the Even-Mansour cipher.

See Theorem 4.14 in [BS23] for the formal statement/proof.

In the quantum setting, we can ask the following analogous question: suppose an adversary is given superposition access to π and either $F_\pi((k_1, k_2), \cdot)$ or a uniformly random permutation. Let U_π denote the unitary corresponding to π, and U_P the unitary corresponding to either $F_\pi((k_1, k_2), \cdot)$ or a uniformly random

permutation. Every superposition query to U_π or U_P counts as one query.

$$U_\pi |x\rangle y = |x\rangle |y \oplus \pi(x)\rangle$$
$$U_P |x\rangle y = |x\rangle |y \oplus P(x)\rangle$$

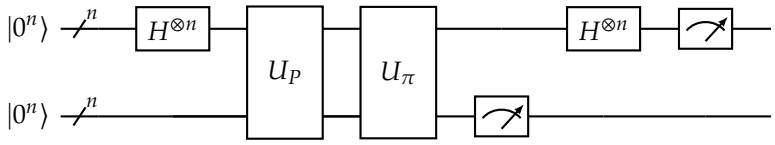
Using Simon's algorithm, we can construct a quantum circuit that makes $O(n)$ superposition queries to U_π and U_P, and if $P = F_\pi((k_0, k_1), \cdot)$, the algorithm recovers k_0 with overwhelming probability (and therefore distinguishes F_π from a uniformly random permutation). The main idea: *use π and $F_\pi((k_1, k_2), \cdot)$ to construct a classical function $f : \{0,1\}^n \to \{0,1\}^n$ such that $f(x) = f(y) \iff x \oplus y \in \{\mathbf{0}^n, k_0\}$.* By making superposition queries to U_f, we can extract k_0.

Superposition Attack on Even-Mansour Cipher: Consider the function

$$f(x) = \pi(x) \oplus P(x)$$

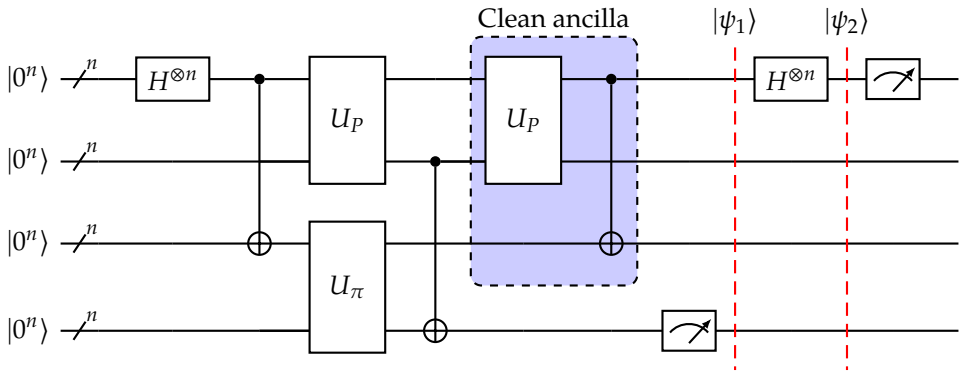
If $P = F_\pi((k_0, k_1), x)$, then $f(x) = \pi(x) \oplus \pi(x \oplus k_0) \oplus k_1 = f(x + k_0)$. Note that we don't have the exact same promise as Simon's problem (since there could exist some x, x' such that $\pi(x) \oplus \pi(x \oplus k_0) = \pi(x') \oplus \pi(x' \oplus k_0)$). However, we will prove that Simon's solution also works in this setting. We describe two different circuits: one that was proposed in class (described in Figure 5). We also present another circuit (described in Figure 6). While it is more complicated than the one in Figure 5, it illustrates the need for cleaning up ancillas.

Figure 5: Quantum Circuit to obtain one equation $x \cdot k_0 = 0$: Simpler circuit



Remark: In the circuit described below (Figure 6), we need two queries to P for each equation. This is to ensure that we do not have any entangled garbage in the second and third registers.

Figure 6: Quantum Circuit to obtain one equation $x \cdot k_0 = 0$: Alternate circuit



Analysis: Let us first consider the case where $P \equiv F_\pi((k_0, k_1), \cdot)$. Suppose the measurement of the last wire output a string y. The (unnormalized) state corresponding to the remaining three registers is

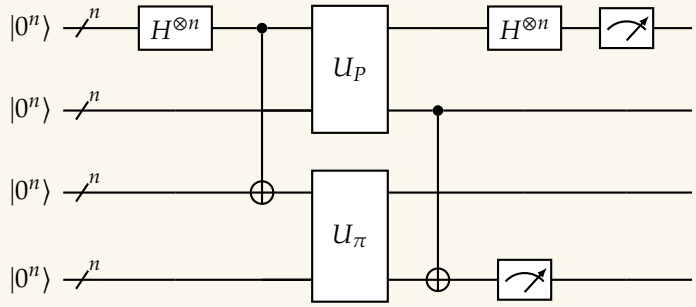
$$|\psi_1\rangle = \sum_{\substack{x:\pi(x)\oplus\pi(x\oplus k_0)\oplus k_1 = y \\ x<x\oplus k_0}} (|x\rangle + |x\oplus k_0\rangle) \otimes |0^{2n}\rangle$$

On applying Hadamard to the first register, we get

$$|\psi_2\rangle = \left(\sum_{\substack{x:\pi(x)\oplus\pi(x\oplus k_0)\oplus k_1 = y \\ x<x\oplus k_0}} \sum_z \left((-1)^{x_i\cdot z} + (-1)^{(x_i\oplus k_0)\cdot z} \right) |z\rangle \right) \otimes |0^{2n}\rangle$$

Similar to our analysis for Simon's algorithm, on measuring the first register, we get a measurement z such that $z \cdot k_0 = 0$.

Exercise 7.1. *Consider the circuit described below. This is similar to the circuit in Figure 6, but without the second U_P gate and the third* CNOT *operation. Do we get the desired result in this case?*



References, Related Works and Recent Progress The above attack was shown by Hidenori and Morii [KM10]. This attack crucially exploits the structure of Even-Mansour cipher. There are other constructions of PRPs (such as $F(k, x) \equiv \pi(k\|x)$ which was proposed in class) for which we don't have any quantum attacks currently.

Classical access to F, quantum access to π: Very recently, Alagic et. al. [ABKM22] showed that the Even-Mansour cipher is secure against quantum adversaries that make only classical queries to F, but are allowed superposition queries to π. This model of security, where the adversary gets classical access to the keyed functions, but superposition queries to the public function modeled as an oracle, is a very natural model to study. For instance, consider a (classical) signature scheme that is proven secure in the random oracle model. In this case, even if the adversary is quantum, for many applications, it suffices to only provide classical access to the signing oracle. However, we must allow superposition queries to the public function (modeled as an oracle). Recall that the random oracle/ideal-permutation model are used to rule out generic real-world attacks.

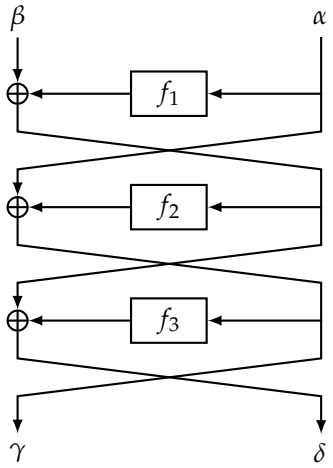
In the real-world, the adversary can run the public function over superposition of inputs, and hence in the quantum random oracle model, we should allow superposition queries.

Note for reviewer: Please let me know if I should elaborate on the above paragraph.

Luby-Rackoff PRP Construction

In this section, we consider another popular PRP construction - the Luby-Rackoff construction. Suppose we have a pseudorandom permutation F with domain and range $\{0,1\}^n$. Using F, we would like to build a new pseudorandom permutation with a larger domain/range. The construction below shows how to do this using three calls to F. Let k_1, k_2, k_3 be three different keys, $f_i \equiv F(k_i, \cdot)$. Let $F_{\text{LR}-3}$ denote the three-round Luby-Rackoff construction.

Last semester, we had studied this construction for building a PRP using PRFs.



Classically, we know that if F is a secure PRP/PRF, then $F_{\text{LR}-3}$ is a secure pseudorandom permutation. However, if the adversary is allowed superposition queries to the PRP, then this construction is not secure! We will show the attack for the case where $F(\cdot,\cdot)$ is a pseudorandom permutation (if F is a PRF, a similar argument works).

The output of $F_{\text{LR}-3}$ is a $2n$ bit string. Let $F_{\text{LR}-3}((k_1,k_2,k_3),(\alpha,\beta))_1$ denote the first n bits, denoted by γ in the figure above. Formally, the unitary U_{LR} is defined as follows:

$$\forall \alpha,\beta,y_0,y_1 \in \{0,1\}^n, U_{\text{LR}} |\alpha\rangle |\beta\rangle |y_0\rangle |y_1\rangle = |\alpha\rangle |\beta\rangle |y_0 \oplus \gamma\rangle |y_1 \oplus \delta\rangle$$
$$\text{where } F_{\text{LR}-3}((k_1,k_2,k_3),(\alpha,\beta)) = (\gamma,\delta).$$

Main idea: Consider the function

Note that the 0^n and 1^n are swapped in the two cases.

$$H(b,x) = \begin{cases} 1^n \oplus F_{\text{LR}-3}((k_1,k_2,k_3),(0^n,x))_1 \text{ if } b = 0 \\ 0^n \oplus F_{\text{LR}-3}((k_1,k_2,k_3),(1^n,x))_1 \text{ if } b = 1 \end{cases}$$

This function maps $n+1$ bits to n bits. If $H(b,x) = H(b',x')$, then either $(b,x) = (b',x')$, or $b' = b \oplus 1$ and $x' = x \oplus f_1(0^n) \oplus f_1(1^n)$.

Observation 7.1.

$$H(b,x) = H(b',x') \iff \big((b,x) = (b',x')\big) \text{ or } \big((b' = b \oplus 1) \text{ and } x' = x \oplus f_1(0^n) \oplus f_1(1^n)\big)$$

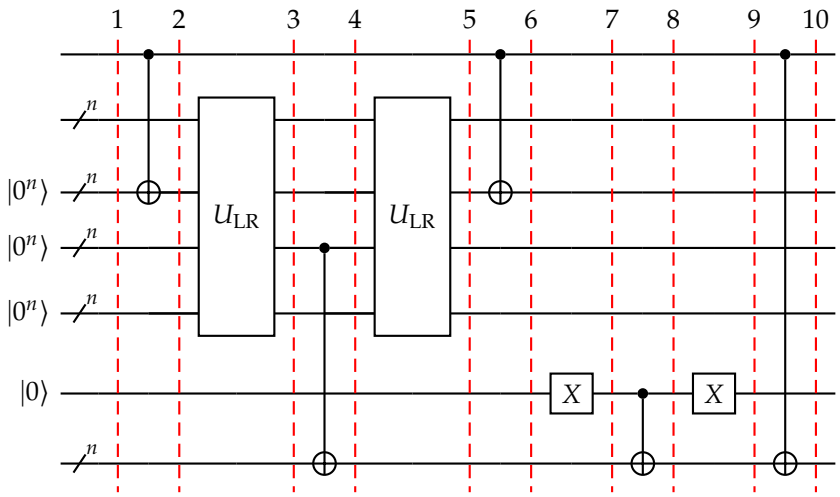
Note for scribes: add short proof for above observation.

Using this observation, we can run Simon's algorithm on H multiple times. In one case (when the oracle is pseudorandom), we will always get the same output $(f_1(0^n) \oplus f_1(1^n))$, while in the other case we will get different outputs in each run.

Implementing the unitary U_H using U_{LR}: The description of function H is defined above. We will first show an implementation of the corresponding unitary. We are allowed to use additional qubits. *However, at the end of the computation, all the additional qubits must be returned to their original state.* Formally, we want a unitary U_H such that the following holds:

$$\forall b \in \{0,1\}, x \in \{0,1\}^n, y \in \{0,1\}^n, U_H |b\rangle |x\rangle |y\rangle \left|0^t\right\rangle = |b\rangle |x\rangle |y \oplus H(b,x)\rangle \left|0^t\right\rangle \tag{7.2}$$

Figure 7: Circuit for U_H



The circuit for U_H is described in Figure 7. Let $|b\rangle |x\rangle \left|0^{3n+1}\right\rangle |y\rangle$ be the initial state. One can check that U_H satisfies Equation 7.2.

Note for scribes: describe states $|\psi_2\rangle, |\psi_4\rangle, |\psi_5\rangle, |\psi_6\rangle, |\psi_9\rangle$ and $|\psi_{10}\rangle$.

Let us consider Simon's algorithm, with oracle U_H.

$$\sum_{\substack{b\in\{0,1\}\\ x\in\{0,1\}^n}} |b\rangle |x\rangle |0^n\rangle \xrightarrow{U_H} \sum_{b,x} |b\rangle |x\rangle \left|\left(\overline{b}\right)^n \oplus F_{\mathrm{LR}-3}(b^n,x)_1\right\rangle = \sum_{b,x} |b\rangle |x\rangle |1^n \oplus f_2(x \oplus f_1(b^n))\rangle$$

On measuring the last n qubits, suppose the output is y. Then the state in the first $n+1$ qubits is

Here, we use the fact that f_2 is a permutation.

$$|0\rangle|x_0\rangle + |1\rangle|x_0 \oplus f_1(0^n) \oplus f_1(1^n)\rangle$$

where $1^n \oplus f_2(x_0 \oplus f_1(0^n)) = 1^n \oplus f_2(x_0 \oplus f_1(0^n) \oplus f_1(1^n)) = y$. Therefore, applying the Hadamard on the first $n+1$ qubits, followed by measurement, produces $\mathbf{z} \in \{0,1\}^{n+1}$ such that $\mathbf{z} \cdot (1, f_1(0^n) \oplus f_1(1^n)) = 0$.

Let us now consider the case where $F_{\mathrm{LR}-3}$ is replaced with a random permutation P in the function H. In this case, the unitary U_H works as follows:

$$U_H|b\rangle|x\rangle|y\rangle = |b\rangle|x\rangle\left|y \oplus \left(\bar{b}\right)^n \oplus P(b^n, x)\right\rangle$$

Here, after applying the circuit for Simon's algorithm, we get a measurement y, and the state in the first $n+1$ qubits either collapses to $|0\rangle|x_0\rangle + |1\rangle|x_1\rangle$, or collapses to $|b\rangle|x_0\rangle$. In the former case, $y = \overline{P(0^n, x_0)_1} = P(1^n, x_1)_1$. In the latter case, $y = \left(\bar{b}\right)^n \oplus P(b^n, x_b)$. In either of these cases, since P is a uniformly random permutation, in different runs of the Simon's algorithm, we will get a different answer.

Here, by one run of the Simon's algorithm, we mean that the algorithm finds n linearly independent equations, and computes the solution.

Note for reviewer: I am not including the formal calculations for this part. Let me know if this is needed.

References, Related Works and Recent Progress The above result was shown by Hidenori and Morii [KM10]. A follow-up work by Santoli and Schaffner [SS17] shows a similar result where F is a keyed function (instead of a permutation).

Hosoyamada and Iwata [HI19] recently showed that the 4-round Luby-Rackoff construction is secure even against superposition attacks. Ito et al. [IHM+19] showed that the 4-round Luby-Rackoff construction is insecure if the adversary has quantum access to both the permutation and its inverse. Currently, it is an open question if 5 (or more) rounds suffice for security when the adversary has superposition access to both the permutation and its inverse (classically, we know that 4 rounds are sufficient to obtain this stronger notion of security).

8 Post-Quantum Zero Knowledge

Lecture 17 (part 2): March 14th, 2023

In this section, we present a classical zero-knowledge protocol (that is, a protocol with classical prover and verifier) that is secure against polynomial time quantum adversaries. Let us recall Definition 5.10, and discuss some simplifications that can be made without loss of generality.

Definition 8.1 (Post-Quantum Zero knowledge with Auxiliary Information)**.** *Let $(\mathbf{P}, \mathbf{V})$ be an interactive proof system (with completeness 1 and soundness s) for an* NP *language $\mathsf{L} = (\mathsf{L}_{\mathrm{yes}}, \mathsf{L}_{\mathrm{no}})$. We say that the proof system satisfies statistical (resp. computational) zero knowledge if for every (possibly malicious) quantum polynomial time verifier $\mathbf{V}^* = \{\mathbf{V}^*_i\}_{i\in\mathbb{N}}$, there exists a quantum polynomial time simulator $\mathbf{S} = \{\mathbf{S}_i\}_{i\in\mathbb{N}}$ such that for every $x \in \mathsf{L}_{\mathrm{yes}}$ and every (mixed) state ρ, the following mixed states are statistically (resp. computationally) indistinguishable:*

$$\mathcal{D}_1 : \left\{\mathsf{view}\langle \mathbf{P}, \mathbf{V}^*_{|x|}(\rho)\rangle(x)\right\}$$

$$\mathcal{D}_2 : \left\{\mathbf{S}_{|x|}(x, \rho)\right\}$$

◊

First, note that here, we are talking about two quantum states being statistically/computationally indistinguishable, since the output of the malicious verifier/simulator can be a quantum state. In the statistical setting, informally, we require that the two density matrices are 'close' (we will make this more precise later in the lecture), while in the computational setting, we require that no q.p.t. algorithm can distinguish between the two density matrices.

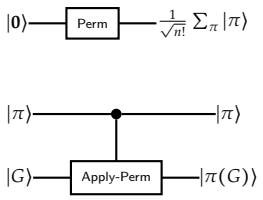
Next, we can assume the auxiliary information is a pure state (since every mixed state can be purified by increasing the number of qubits). Finally, we can assume the adversary, in each round, first applies a unitary, and then measures the first few qubits (which are sent to the prover). The remaining qubits are passed as 'internal state' to the next round's verifier. The last verifier measures everything and outputs the resulting measurement.

8.1 *Post-quantum ZK protocol for graph isomorphism*

We will show that Protocol 4 also satisfies Definition 8.1. In this protocol, the prover chooses a random permutation π, applies it to G_0 and sends the resulting graph H to the verifier. The verifier, using this graph H and its auxiliary information z, computes a bit b. If $b = 0$, the prover sends π, else it sends $\pi \circ \sigma$, where $\sigma(G_1) = G_0$.

First, we will describe the entire protocol (interaction between an honest prover and a malicious verifier) as a quantum circuit.

The first message (from the prover): Let Perm denote a quantum gate that takes sufficiently many qubits, initialised to $|\mathbf{0}\rangle$ and outputs $\frac{1}{\sqrt{n!}}\sum_\pi |\pi\rangle$. Let Apply-Perm denote a quantum gate that has graph G_0 hardwired, it takes as input two registers - one containing a permutation π, another containing $|\mathbf{0}\rangle$, and outputs the permutation π together with $\pi(G_0)$. More formally, for all permutations π, Apply-Perm $|\pi\rangle|\mathbf{0}\rangle = |\pi\rangle|\pi(G_0)\rangle$.



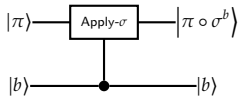
The prover operates on two registers $\mathsf{M}_3, \mathsf{M}_1$. It applies Perm to M_1, then applies Apply-Perm to $(\mathsf{M}_3, \mathsf{M}_1)$, and finally measures M_1 and sends this as the first message to the verifier.

The second message (from the verifier): The verifier operates on three registers: $\mathsf{M}_1, \mathsf{M}_2, \mathsf{Z}$ representing the first message, second message and the register containing aux. info respectively.[3] On receiving a graph H (in register M_1), it applies unitary $\mathbf{U}_H$ on registers M_2, Z. Let $\mathbf{U}$ denote the unitary that maps

$$|H\rangle_{\mathsf{M}_1}|0\rangle_{\mathsf{M}_2}|\psi\rangle_{\mathsf{Z}} \xrightarrow{\mathbf{U}} |H\rangle_{\mathsf{M}_1} \otimes \left(\mathbf{U}_H|0,z\rangle_{\mathsf{M}_2,\mathsf{Z}}\right).$$

After applying the unitary $\mathbf{U}$, it measures the register M_2, and this is sent as the second message.

The third message (from the prover): Finally, the prover uses the challenge message (in register M_2) to compute the third message. This is the part that uses the witness σ such that $\sigma(G_1) = G_0$. We will represent this computation

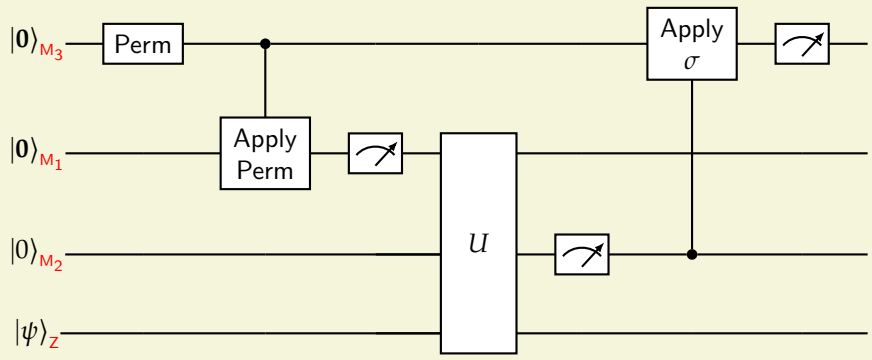


[3] We assume the aux. info contains sufficiently many ancilla bits for the computation.

using a controlled Apply-σ gate. This operation takes as input two registers, one containing a bit b and another containing a permutation π. It outputs $|b\rangle |\pi \circ \sigma^b\rangle$. The prover applies this operation to registers $(\mathsf{M}_3, \mathsf{M}_2)$. After applying this operation, the prover measures register M_3.

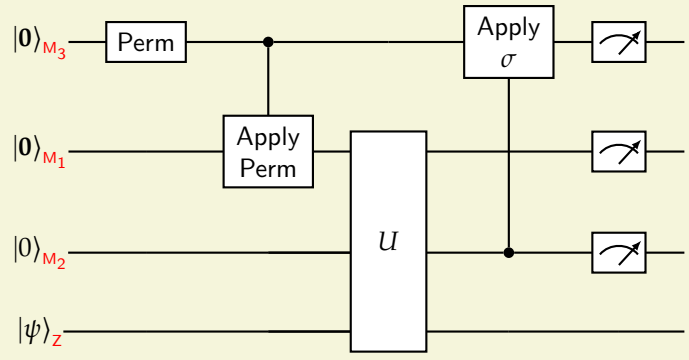
The distinguisher gets the registers $(\mathsf{M}_1, \mathsf{M}_2, \mathsf{M}_3, \mathsf{Z})$. The full circuit is described in Figure 8.

Figure 8: Quantum Circuit for the Graph-Isomorphism Protocol



Next, note that the above circuit is equivalent to the circuit described in Figure 9. In this circuit, all measurements happen at the end. The two circuits are equivalent because the measurement of M_1 commutes with the operation $\mathbf{U}$, and the measurement of M_2 commutes with the controlled Apply-σ operation. This equivalent circuit allows us to talk about the state of the circuit just before all measurements are performed.

Figure 9: Equivalent Quantum Circuit for the Graph-Isomorphism Protocol



Suppose

$$U_H |0, \psi\rangle_{\mathsf{M}_2,\mathsf{Z}} = \sum_{z',b} \alpha_{z',b}^{H,\psi} |b\rangle_{\mathsf{M}_2} |z'\rangle_{\mathsf{Z}}$$

where the amplitudes $\left\{\alpha_{z',b}^{H,\psi}\right\}_{z',b}$ depend on the graph H and input ψ. The state of circuit described in Figure 9, just before the measurements, is

(8.2)

$$|\phi_{\text{real}}\rangle_{\mathsf{M}_3,\mathsf{M}_1,\mathsf{M}_2,\mathsf{Z}} = \frac{1}{\sqrt{n!}}\left(\sum_{H\in\text{iso}(G_0)}\sum_{\substack{\pi:\\ \pi(G_0)=H}}\sum_{z',b}\alpha_{z',b}^{H,\psi}\left|\pi\circ\sigma^b\right\rangle_{\mathsf{M}_3}|H\rangle_{\mathsf{M}_1}|b\rangle_{\mathsf{M}_2}\left|z'\right\rangle_{\mathsf{Z}}\right)$$

The simulator

We will now describe our first attempt at defining the simulator. The first attempt is similar to what we did classically. Recall, in the classical setting, the simulator picks a uniformly random bit b, then simulates a transcript $(\mathsf{msg}_1 = \pi(G_b), b, \mathsf{msg}_3 = \pi)$ according to b. It sends msg_1 to the verifier. If the verifier sends $\mathsf{msg}_2 = b$, then the simulator sends msg_3, else it restarts this conversation.

In the quantum setting, we will have a new wire/register S for the simulator. Here, the simulator generates a uniform superposition of $|0\rangle$ and $|1\rangle$, and proceeds similar to the classical case. In the register M_1, it computes a random permutation of G_b. Therefore, the contents of the register M_1 depend on the contents of register M_3 as well as register S, and are computed using a 'controlled' Apply-Perm gate. More formally, this controlled Apply-Perm gate has G_0, G_1 hardwired, and it maps $|b\rangle|\pi\rangle|\mathbf{0}\rangle$ to $|b\rangle|\pi\rangle|\pi(G_b)\rangle$.

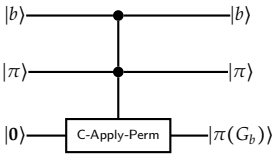
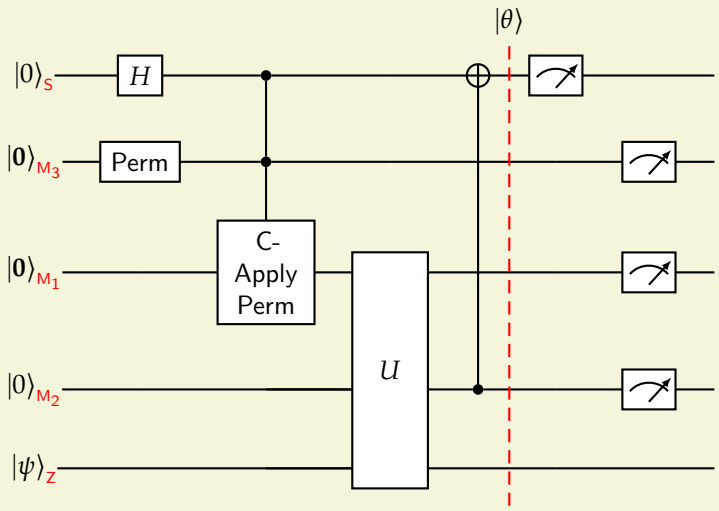


Figure 10: Simulator - Attempt 1



Recall, in the classical setting, the simulator, with probability 1/2, computes a transcript that is identical to the real-world transcript. We can prove a similar result in the quantum setting too.

Claim 8.3. *When the register* S *is measured, it outputs* 0 *and* 1 *with equal probability. Moreover, when the output is* 0*, the mixed state of the remaining registers is identical to the mixed state in the real-world (that is, the state* $|\phi_{\text{real}}\rangle$ *described in Equation 8.2).*

Proof. Let us consider the state $|\theta\rangle$ in Figure 10. This state is

$$\begin{aligned}
|\theta\rangle = & \frac{1}{\sqrt{2n!}}\left(\sum_{H\in\mathsf{iso}(G_0)}\sum_{b'}\sum_{\substack{\pi:\\ \pi(G_{b'})=H}}\sum_{z',b}\alpha^{H,\psi}_{z',b}\left|b'\oplus b\right\rangle_{\mathsf{S}}|\pi\rangle_{\mathsf{M}_3}|H\rangle_{\mathsf{M}_1}|b\rangle_{\mathsf{M}_2}\left|z'\right\rangle_{\mathsf{Z}}\right)\\
= & \frac{1}{\sqrt{2n!}}\left(\sum_{H\in\mathsf{iso}(G_0)}\sum_{b}\sum_{\substack{\pi:\\ \pi(G_{b})=H}}\sum_{z'}\alpha^{H,\psi}_{z',b}|0\rangle_{\mathsf{S}}|\pi\rangle_{\mathsf{M}_3}|H\rangle_{\mathsf{M}_1}|b\rangle_{\mathsf{M}_2}\left|z'\right\rangle_{\mathsf{Z}}\right)\\
& +\frac{1}{\sqrt{2n!}}\left(\sum_{H\in\mathsf{iso}(G_0)}\sum_{b}\sum_{\substack{\pi:\\ \pi(G_{\overline{b}})=H}}\sum_{z'}\alpha^{H,\psi}_{z',b}|1\rangle_{\mathsf{S}}|\pi\rangle_{\mathsf{M}_3}|H\rangle_{\mathsf{M}_1}|b\rangle_{\mathsf{M}_2}\left|z'\right\rangle_{\mathsf{Z}}\right)
\end{aligned}$$

The probability of seeing bit β after measuring the S register is equal to

$$p_\beta = \sum_{H\in\mathsf{iso}(G_0)}\sum_{b}\sum_{\substack{\pi:\\ \pi(G_{b+\beta})=H}}\sum_{z'}\frac{\left|\alpha^{H,\psi}_{z',b}\right|^2}{2n!}$$

The number of permutations mapping G_0 to H is equal to the number of permutations mapping G_1 to H, and the amplitudes $\alpha^{H,\psi}_{z',b}$ do not depend on the exact permutation mapping G_b to H. Hence $p_0 = p_1$. However, note that the residual states are very different. If the measurement outputs 0, the state of registers $(\mathsf{M}_3,\mathsf{M}_1,\mathsf{M}_2,\mathsf{Z})$ is

$$\left|\phi_{0,\psi}\right\rangle = \frac{1}{\sqrt{p_0}}\left(\sum_{H\in\mathsf{iso}(G_0)}\sum_{b}\sum_{\substack{\pi:\\ \pi(G_{b})=H}}\sum_{z'}\alpha^{H,\psi}_{z',b}|\pi\rangle_{\mathsf{M}_3}|H\rangle_{\mathsf{M}_1}|b\rangle_{\mathsf{M}_2}\left|z'\right\rangle_{\mathsf{Z}}\right)$$

while, if the measurement outputs 1, the state of registers is

$$\left|\phi_{1,\psi}\right\rangle = \frac{1}{\sqrt{p_1}}\left(\sum_{H\in\mathsf{iso}(G_0)}\sum_{b}\sum_{\substack{\pi:\\ \pi(G_{1-b})=H}}\sum_{z'}\alpha^{H,\psi}_{z',b}|\pi\rangle_{\mathsf{M}_3}|H\rangle_{\mathsf{M}_1}|b\rangle_{\mathsf{M}_2}\left|z'\right\rangle_{\mathsf{Z}}\right)$$

Finally, note that for any $b\in\{0,1\}$, the following two vectors are identical:

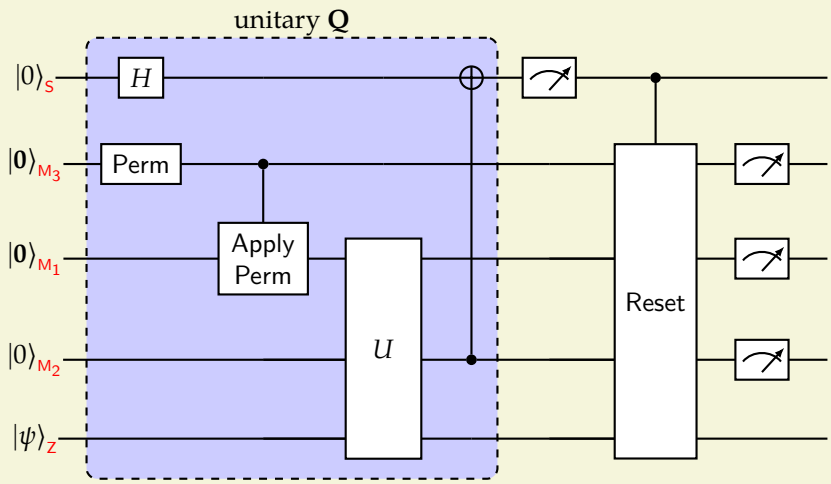
$$\sum_{\substack{\pi:\\ \pi(G_0)=H}}\left|\pi\circ\sigma^b\right\rangle = \sum_{\substack{\pi:\\ \pi(G_b)=H}}|\pi\rangle$$

Using this, we can conclude that the state $\left|\phi_{0,\psi}\right\rangle$ is equal to the state described in (8.2). □

This solves half our problem. When the register S is measured, we get 0 or 1 with equal probability, and conditioned on this output being 0, the state of the remaining registers is identical to the real-world distribution. However, what happens when the measurement outputs 1? Unlike the classical case, we cannot restart the computation — the aux. info. $|\psi\rangle$ has been 'consumed' (and due to the *no-cloning* theorem, we cannot make multiple copies of $|\psi\rangle$).

The key idea is to somehow 'reset' the state of the remaining registers if the measurement of S produces 1. For this particular protocol, it turns out that one appropriate 'reset' is sufficient, however in other cases we might need several such resets.

Figure 11: Simulator - Attempt 2



Resetting the state when measurement outcome is 0

Let us consider the simulator described in Figure 11 abstractly. It takes as input $|\psi\rangle_Z$ together with some ancilla bits $|\mathbf{0}\rangle_{A,Y}$, and applies a unitary $\mathbf{Q}$. For simplicity, let us assume Z is an n qubit register, A is a single qubit register and Y is an $n-1$ qubit register. The unitary $\mathbf{Q}$ depends on the verifier's unitary $\mathbf{U}$. The state after applying $\mathbf{Q}$ is

$$\mathbf{Q}\,|\mathbf{0}\rangle_{A,Y}\,|\psi\rangle_Z = \frac{1}{\sqrt{2}}\,|0\rangle_A\,|\phi_{0,\psi}\rangle_{Y,Z} + \frac{1}{\sqrt{2}}\,|1\rangle_A\,|\phi_{1,\psi}\rangle_{Y,Z}$$

Note that this holds true for all aux. info. $|\psi\rangle$, and $|\phi_{0,\psi}\rangle_{Y,Z}$ is the quantum state we want to output. In this section, we will discuss how to transform the state $|1\rangle_A\,|\phi_{1,\psi}\rangle_{Y,Z}$ to $|0\rangle_A\,|\phi_{0,\psi}\rangle_{Y,Z}$ using an appropriately defined Reset unitary.

Defining the Reset unitary: In order to reset the circuit, we will need to use the inverse of $\mathbf{Q}$. Let us consider the effect of applying $\mathbf{Q}^\dagger$ on $|0\rangle_A\,|\phi_{0,\psi}\rangle_{Y,Z}$ and $|1\rangle_A\,|\phi_{1,\psi}\rangle_{Y,Z}$. First, let Γ be some orthonormal basis that contains $|\psi\rangle$ as one of

Note that $|\psi\rangle$ need not be one of the computational basis states $\{|z\rangle\}_{z\in\{0,1\}^n}$. It can be some superposition of these vectors. Therefore, we consider some orthonormal basis that contains $|\psi\rangle$ as one of the basis vectors.

the basis vectors. Let

$$\begin{aligned}
&\mathbf{Q}^\dagger |b\rangle_A |\phi_{b,\psi}\rangle_{Y,Z} \\
=&\mathbf{Q}^\dagger \left(\frac{1}{2} \left(|0\rangle_A |\phi_{0,\psi}\rangle_{Y,Z} + |1\rangle_A |\phi_{1,\psi}\rangle_{Y,Z} \right) + \frac{(-1)^b}{2} \left(|0\rangle_A |\phi_{0,\psi}\rangle_{Y,Z} - |1\rangle_A |\phi_{1,\psi}\rangle_{Y,Z} \right) \right) \\
=&\frac{1}{\sqrt{2}} |\mathbf{0}\rangle_{A,Y} |\psi\rangle_Z + \frac{(-1)^b}{2} \mathbf{Q}^\dagger \left(|0\rangle_A |\phi_{0,\psi}\rangle_{Y,Z} - |1\rangle_A |\phi_{1,\psi}\rangle_{Y,Z} \right)
\end{aligned}$$

We will now make a few observations about the vector $\mathbf{Q}^\dagger \left(|0\rangle_A |\phi_{0,\psi}\rangle_{Y,Z} - |1\rangle_A |\phi_{1,\psi}\rangle_{Y,Z} \right)$. First, note that this is orthogonal to $|\mathbf{0}\rangle_{A,Y} |\psi\rangle_Z$. This is because $\mathbf{Q}^\dagger$ is a unitary and $\left(|0\rangle_A |\phi_{0,\psi}\rangle_{Y,Z} - |1\rangle_A |\phi_{1,\psi}\rangle_{Y,Z} \right)$ is orthogonal to $\left(|0\rangle_A |\phi_{0,\psi}\rangle_{Y,Z} + |1\rangle_A |\phi_{1,\psi}\rangle_{Y,Z} \right)$. In fact, a stronger statement holds true: this vector is orthogonal to $|\mathbf{0}\rangle_{A,Y} |\eta\rangle_Z$ for any $\eta \in \Gamma$. We will prove this in Claim 8.4 below, but first let us see why this claim is useful.

From the above discussion, it follows that

$$\begin{aligned}
\mathbf{Q}^\dagger |0\rangle_A |\phi_{0,\psi}\rangle_{Y,Z} &= \frac{1}{\sqrt{2}} |\mathbf{0}\rangle_{A,Y} |\psi\rangle_Z + \sum_{\substack{z \neq \mathbf{0} \\ \tau \in \Gamma}} \beta_{0,z,\tau} |z\rangle_{A,Y} |\tau\rangle_Z \\
\mathbf{Q}^\dagger |1\rangle_A |\phi_{1,\psi}\rangle_{Y,Z} &= \frac{1}{\sqrt{2}} |\mathbf{0}\rangle_{A,Y} |\psi\rangle_Z - \sum_{\substack{z \neq \mathbf{0} \\ \tau \in \Gamma}} \beta_{0,z,\tau} |z\rangle_{A,Y} |\tau\rangle_Z
\end{aligned}$$

To finish our argument, it suffices to have a unitary that acts as identity on $\left\{ |\mathbf{0}\rangle_{A,Y} |\psi\rangle_Z \right\}_\psi$, but negates every state of the form $|z\rangle_{A,Y} |\tau\rangle_Z$, where $z \neq \mathbf{0}$. This is reflection about the subspace spanned by $\left\{ |\mathbf{0}\rangle_{A,Y} |\tau\rangle_Z \right\}_{\tau \in \Gamma}$. Check that $\mathsf{Flip} = 2 \left(|\mathbf{0}\rangle\langle\mathbf{0}|_{A,Y} \otimes \mathbf{I}_Z \right) - \mathbf{I}_{A,Y,Z}$ is the required unitary.

Hence, for all $|\psi\rangle$,

$$\left(\mathbf{Q} \cdot \mathsf{Flip} \cdot \mathbf{Q}^\dagger \right) |1\rangle_A |\phi_{1,\psi}\rangle_{Y,Z} = |0\rangle_A |\phi_{0,\psi}\rangle_{Y,Z},$$

and this concludes our proof. We will now state and prove our missing claim.

Claim 8.4. *For any $\tau \in \Gamma$, $|\mathbf{0}\rangle_{A,Y} |\tau\rangle_Z$ is orthogonal to $\mathbf{Q}^\dagger \left(|0\rangle_A |\phi_{0,\psi}\rangle_{Y,Z} - |1\rangle_A |\phi_{1,\psi}\rangle_{Y,Z} \right)$.*

Proof. To prove this claim, we will use the assumption that for all $|\tau\rangle$,

$$\mathbf{Q} |\mathbf{0}\rangle_{A,Y} |\tau\rangle_Z = \frac{1}{\sqrt{2}} \left(|0\rangle_A |\phi_{0,\tau}\rangle_{Y,Z} + |1\rangle_A |\phi_{1,\tau}\rangle_{Y,Z} \right) \text{ where } \left\| |\phi_{0,\tau}\rangle_{Y,Z} \right\| = \left\| |\phi_{1,\tau}\rangle_{Y,Z} \right\| = 1$$

Suppose, on the contrary, $\mathbf{Q}^\dagger \left(|0\rangle_A |\phi_{0,\psi}\rangle_{Y,Z} - |1\rangle_A |\phi_{1,\psi}\rangle_{Y,Z} \right)$ is not orthogonal to $|\mathbf{0}\rangle_{A,Y} |\tau\rangle_Z$. Then $\mathbf{Q} \cdot |\mathbf{0}\rangle_{A,Y} |\tau\rangle_Z = \frac{1}{\sqrt{2}} \left(|0\rangle_A |\phi_{0,\tau}\rangle_{Y,Z} + |1\rangle_A |\phi_{1,\tau}\rangle_{Y,Z} \right)$, and $\langle \phi_{0,\tau} | \phi_{0,\psi} \rangle \neq \langle \phi_{1,\tau} | \phi_{1,\psi} \rangle$. Note that this implies one of $\langle \phi_{0,\tau} | \phi_{0,\psi} \rangle$ or $\langle \phi_{1,\tau} | \phi_{1,\psi} \rangle$ is non-zero.

Consider some superposition of $|\mathbf{0}\rangle_{A,Y} |\tau\rangle_Z$ and $|\mathbf{0}\rangle_{A,Y} |\psi\rangle_Z$, say $\frac{1}{\sqrt{2}} \left(|\mathbf{0}\rangle_{A,Y} |\tau\rangle_Z + |\mathbf{0}\rangle_{A,Y} |\psi\rangle_Z \right)$,

and apply $\mathbf{Q}$ to this vector. The resulting pure state is

$$\frac{|0\rangle_{\mathsf{A}}}{\sqrt{2}}\left(\frac{|\phi_{0,\psi}\rangle_{\mathsf{Y},\mathsf{Z}}}{\sqrt{2}}+\frac{|\phi_{0,\tau}\rangle_{\mathsf{Y},\mathsf{Z}}}{\sqrt{2}}\right)+\frac{|1\rangle_{\mathsf{A}}}{\sqrt{2}}\left(\frac{|\phi_{1,\psi}\rangle_{\mathsf{Y},\mathsf{Z}}}{\sqrt{2}}+\frac{|\phi_{1,\tau}\rangle_{\mathsf{Y},\mathsf{Z}}}{\sqrt{2}}\right)$$

$$\left\|\frac{|\phi_{b,\psi}\rangle_{\mathsf{Y},\mathsf{Z}}}{\sqrt{2}}+\frac{|\phi_{b,\tau}\rangle_{\mathsf{Y},\mathsf{Z}}}{\sqrt{2}}\right\|_2^2=1+\langle\phi_{b,\psi}|\phi_{b,\tau}\rangle$$

Since at least one of $\langle\phi_{0,\tau}|\phi_{0,\psi}\rangle$ or $\langle\phi_{1,\tau}|\phi_{1,\psi}\rangle$ is non-zero, we arrive at a contradiction.

□

Proof Summary Below, we break down the zero-knowledge proof into key claims. The reader is encouraged to prove these claims to reconstruct the argument.

1. First, we expressed the protocol as a quantum circuit, with all measurements deferred to the end.
2. Next, we defined the simulator-based circuit (following the classical ZK proof).
3. Using the simulator-based circuit, we can define a unitary $\mathbf{Q}$ with the following property:

$$\forall\,|\psi\rangle\,,\mathbf{Q}\,|\mathbf{0}\rangle\,|\psi\rangle=\frac{1}{\sqrt{2}}\left(|0\rangle\,|\phi_{0,\psi}\rangle+|1\rangle\,|\phi_{1,\psi}\rangle\right)$$

   where $|\phi_{b,\psi}\rangle$ are unit-norm vectors. Moreover, $|\phi_{0,\psi}\rangle$ is the desired state (that is identical to the verifier's view in the real-world).
4. If we measure the first qubit, we get $|0\rangle\,|\phi_{0,\psi}\rangle$ and $|1\rangle\,|\phi_{1,\psi}\rangle$ with equal probability. If the measurement is 0, then we are done. If the measurement outcome is 1, then the resulting state is $|\phi_{1,\psi}\rangle$.
5. We define a reset procedure which can map $|1\rangle\,|\phi_{1,\psi}\rangle$ to $|0\rangle\,|\phi_{0,\psi}\rangle$. Here, the main idea is to observe the effect of applying $\mathbf{Q}^\dagger$ on these two states.
6. There exists a simple unitary (which we called Flip) that maps $\mathbf{Q}^\dagger\,|1\rangle\,|\phi_{1,\psi}\rangle$ to $\mathbf{Q}^\dagger\,|0\rangle\,|\phi_{0,\psi}\rangle$. Hence, to map $|1\rangle\,|\phi_{1,\psi}\rangle$ to $|0\rangle\,|\phi_{0,\psi}\rangle$, we first apply $\mathbf{Q}^\dagger$, then apply Flip, and finally apply $\mathbf{Q}$.

8.2 *Post-quantum ZK for graph isomorphism via alternating projectors*

Lecture 18: March 17th, 2023

We will now see a different simulator for the same protocol. This simulator will be weaker than the one we saw in the previous section (in particular, this simulator will have a negligible probability of failure). However, the ideas used in the analysis of this simulator have been used in many different contexts.

The starting point for our simulator will be similar to the one in Figure 11. We have a unitary $\mathbf{Q}$ such that, for any aux. info. state $|\psi\rangle$,

$$\mathbf{Q}\,|\mathbf{0}\rangle_{\mathsf{A},\mathsf{Y}}\,|\psi\rangle_{\mathsf{Z}} = \sqrt{0.5}\,|0\rangle_{\mathsf{A}}\,\left|\phi_{0,\psi}\right\rangle_{\mathsf{Y},\mathsf{Z}} + \sqrt{0.5}\,|1\rangle_{\mathsf{A}}\,\left|\phi_{1,\psi}\right\rangle_{\mathsf{Y},\mathsf{Z}} \tag{8.5}$$

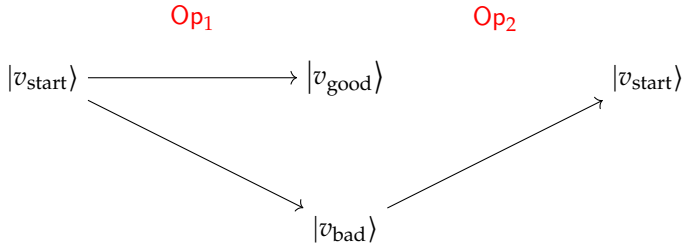
where $|b\rangle_{\mathsf{A}}\left|\phi_{b,\psi}\right\rangle_{\mathsf{Y},\mathsf{Z}}$ are both quantum states over $2n$ qubits. We also observed

$$\mathbf{Q}^{\dagger}\,|b\rangle_{\mathsf{A}}\left|\phi_{b,\psi}\right\rangle_{\mathsf{Y},\mathsf{Z}} = \sqrt{0.5}\,|\mathbf{0}\rangle_{\mathsf{A},\mathsf{Y}}\,|\psi\rangle_{\mathsf{Z}} + \sqrt{0.5}\left(\sum_{\substack{z\neq\mathbf{0}\\ \tau\in\Gamma}}\beta_{b,z,\tau}\,|z\rangle_{\mathsf{A},\mathsf{Y}}\,|\tau\rangle_{\mathsf{Z}}\right) \tag{8.6}$$

Consider the following operations: the first one (let's call it Op_1) applies $\mathbf{Q}$ and measures the register A. The second operation (let's call it Op_2) applies $\mathbf{Q}^{\dagger}$, and **measures** the first n qubits (in the registers A,Y). From the above equation, we know that if this measurement is 0^n, then our state is back to $|\mathbf{0}\rangle_{\mathsf{A},\mathsf{Y}}\,|\psi\rangle_{\mathsf{Z}}$, in which case we can apply Op_1 again.

Let $|v_{\text{start}}\rangle = |\mathbf{0}\rangle\,|\psi\rangle$, $|v_{\text{good}}\rangle = |0\rangle_{\mathsf{A}}\left|\phi_{0,\psi}\right\rangle_{\mathsf{Y},\mathsf{Z}}$, $|v_{\text{bad}}\rangle = |1\rangle_{\mathsf{A}}\left|\phi_{1,\psi}\right\rangle_{\mathsf{Y},\mathsf{Z}}$.

Figure 12: Transition diagram for $\mathsf{Op}_1, \mathsf{Op}_2$



In this figure, the arrows represent transition with probability 1/2.

Let p denote the probabilty of eventually reaching $|v_{\text{good}}\rangle$, starting at $|v_{\text{start}}\rangle$. Then, using the above transition diagram, we get the following recurrence:

$$p = 1/2 + p/4,$$

which gives us a simulation strategy that succeeds with probability $\approx 2/3$. Can we do better?

The transition diagram in Figure 13 gives us some hope of reaching the target state $|\mathbf{0}\rangle\,|\psi\rangle$ even if the first measurement (from Op_1) produces a 1 outcome. One key idea is the following:

When applying Op_2, instead of measuring the first n qubits, we can simply check if these n qubits are equal to $\mathbf{0}$ or not.

Such a 'check' would be a two-outcome projective measurement. The benefit of doing a two-outcome measurement instead of measuring n qubits is that the two-outcome measurement improves our chances of getting back $|\mathbf{0}\rangle\,|\psi\rangle$ (even if the two-outcome measurement did not produce $\mathbf{0}$).

This idea will be clearer once we spell out some more details.

Our simulator will apply two partial measurements repeatedly (interspersed by some carefully chosen unitaries), and we will view these partial measurements as projective measurements.

1. The first partial measurement (and a natural partial measurement to be used here) is the measurement of the first qubit. This is a two-outcome projective measurement, formally represented using the following projective matrices:

$$\mathbf{\Pi}' = \left(\mathbf{\Pi}'_0 = |0\rangle\langle 0|_{\mathsf{A}} \otimes \mathbf{I}_{\mathsf{Y},\mathsf{Z}} \ , \ \mathbf{\Pi}'_1 = \mathbf{I}_{\mathsf{A},\mathsf{Y},\mathsf{Z}} - \mathbf{\Pi}'_0 \right)$$

2. The second projective measurement that we will use will check if the first n qubits (in registers A,Y) are set to $|\mathbf{0}\rangle$ or not. Our hope is that this two-outcome measurement will somehow 'reset' the state to $|\mathbf{0}\rangle_{\mathsf{A},\mathsf{Y}} |\ldots\rangle_{\mathsf{Z}}$. Of course, at this point, it is not at all clear why this measurement will be useful, for the following two reasons:

   - Maybe it will output 1 most of the times (in which case, we are not resetting to $|\mathbf{0}\rangle |\ldots\rangle$).
   - Even if it outputs 0, it might reset to $|\mathbf{0}\rangle |\eta\rangle$ for some $\eta \neq \psi$, in which case this is not useful for simulation.

   The second concern was addressed in Claim 8.4.

   This measurement is defined below:

$$\mathbf{\Delta} = \left(\mathbf{\Delta}_0 = |\mathbf{0}\rangle\langle\mathbf{0}|_{\mathsf{A},\mathsf{Y}} \otimes \mathbf{I}_{\mathsf{Z}} \ , \ \mathbf{\Delta}_1 = \mathbf{I}_{\mathsf{A},\mathsf{Y},\mathsf{Z}} - \mathbf{\Delta}_0 \right)$$

With these two projectors and the unitary $\mathbf{Q}$, we are ready to describe our simulator. Recall, the simulator takes as input $|\mathbf{0}\rangle_{\mathsf{A},\mathsf{Y}} |\psi\rangle_{\mathsf{Z}}$.

Quantum Simulator:

for $i = 1$ *to* n **do**
  Simulator applies $\mathbf{Q}$, followed by projective measurement $\mathbf{\Pi}'$. If the output of the measurement is 0, then the simulator sets `success = true` and exits the loop.
  Simulator applies $\mathbf{Q}^\dagger$, followed by $\mathbf{\Delta}$.
If `success` $\neq$ `true`, $\mathbf{S}$ outputs $\perp$. Else it outputs the registers $(\mathsf{A},\mathsf{Y},\mathsf{Z})$.

We can express this simulator in terms of alternating projections by 'absorbing' the $\mathbf{Q}$ and $\mathbf{Q}^\dagger$ into one of the projections. More formally, let

$$\mathbf{\Pi} = \left(\mathbf{\Pi}_0 = \mathbf{Q}^\dagger \cdot (|0\rangle\langle 0|_{\mathsf{A}} \otimes \mathbf{I}_{\mathsf{Y},\mathsf{Z}}) \cdot \mathbf{Q} \ , \ \mathbf{\Pi}_1 = \mathbf{I}_{\mathsf{A},\mathsf{Y},\mathsf{Z}} - \mathbf{\Pi}_0 \right)$$

Note that $\mathbf{\Pi}_0 = \mathbf{Q}^\dagger \cdot \mathbf{\Pi}'_0 \cdot \mathbf{Q}$. Then our simulator can be described as follows.

Quantum Simulator with alternating projections:

1 **for** $i = 1$ *to* n **do**
2 Simulator applies proj. measurement $\mathbf{\Pi}$. If the output of the measurement is 0, then the simulator sets `success = true`, and exits the loop.
3 Simulator applies proj. measurement $\mathbf{\Delta}$.
4 If `success` $\neq$ `true`, $\mathbf{S}$ outputs $\perp$. Else it applies $\mathbf{Q}$ to the registers $\mathsf{A}, \mathsf{Y}, \mathsf{Z}$ and outputs the registers $(\mathsf{A}, \mathsf{Y}, \mathsf{Z})$.

We will show that the above simulator, with overwhelming probability, outputs $|0\rangle_{\mathsf{A}} \left|\phi_{0,\psi}\right\rangle_{\mathsf{Y},\mathsf{Z}}$. In order to prove this, we will need to understand how alternating projectors interact.

Alternating Projectors

To start, let us consider two projective measurements $\mathbf{\Pi} = (|v\rangle\langle v|, \mathbf{I} - |v\rangle\langle v|)$ and $\mathbf{\Delta} = (|w\rangle\langle w|, \mathbf{I} - |w\rangle\langle w|)$. Here, the 0-outcome measurements $(\mathbf{\Pi}_0, \mathbf{\Delta}_0)$ project onto one-dimensional subspaces (spanned by $|v\rangle$ and $|w\rangle$ respectively). Suppose our starting state is $|w\rangle$, and we apply $\mathbf{\Delta}$ and $\mathbf{\Pi}$ alternately n times (starting with $\mathbf{\Pi}$). What is the probability that we see outcome $= 0$ for $\mathbf{\Pi}$ at some point?

When the measurement $\mathbf{\Pi}$ is applied to $|w\rangle$, the measurement outcome is either 0 (in which case the state is $|v\rangle$) or 1, in which case the state $|v^\perp\rangle$ is $\mathbf{\Pi}_1 |w\rangle = |w\rangle - \langle v|w\rangle |v\rangle$ (appropriately normalized). Note that $|v^\perp\rangle$ is orthogonal to $|v\rangle$.

Similarly, when the measurement $\mathbf{\Delta}$ is applied to $|v\rangle$, the measurement outcome is either 0 (in which case the state is $|w\rangle$) or 1, in which case the state is $|w^\perp\rangle = \mathbf{\Delta}_1 |v\rangle = |v\rangle - \langle w|v\rangle |w\rangle$ (appropriately normalized). Again, $|w^\perp\rangle$ is orthogonal to $|w\rangle$.

An immediate observation is that after $\mathbf{\Pi}$ (resp. $\mathbf{\Delta}$) is applied, the state is either $|v\rangle$ or $|v^\perp\rangle$ (resp. $|w\rangle$ or $|w^\perp\rangle$). Moreover, after k rounds, the probability of seeing '0' for a $\mathbf{\Pi}$ measurement can be obtained by following the transition graph.

Back to our simulator's analysis: Note that in the above example scenario, the intermediate states were always contained in a two-dimensional space (spanned by $|v\rangle$ and $|w\rangle$), and this used the fact that $\mathbf{\Pi}_0$ and $\mathbf{\Delta}_0$ are projections onto a one-dimensional subspace. The projection matrices involved in our simulation do not project onto one-dimensional spaces.

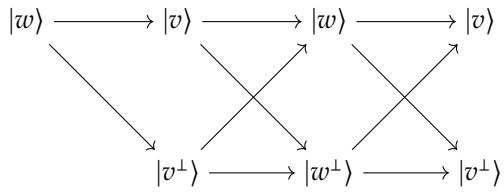
We will require some additional structure that is present in our projective matrices $\mathbf{\Pi}_0$ and $\mathbf{\Delta}_0$. Note that, for any $|\psi\rangle$, the state $|\mathbf{0}\rangle |\psi\rangle$ is an eigenvector of $\mathbf{\Delta}_0$ (with eigenvalue 1), and an eigenvector of $\mathbf{\Delta}_0 \cdot \mathbf{\Pi}_0 \cdot \mathbf{\Delta}_0$ with eigenvalue 1/2 (follows from Equations 8.5 and 8.6). Consider the subspace $\mathcal{S}$ spanned by $\{|w\rangle = |\mathbf{0}\rangle |\psi\rangle, |v'\rangle = \mathbf{\Pi}_0 |w\rangle\}$, and let $|v\rangle = \mathbf{\Pi}_0 |w\rangle / \|\mathbf{\Pi}_0 |w\rangle\|$ (that is, normalization of $|v'\rangle$). Let $|v^\perp\rangle, |w^\perp\rangle$ be vectors in $\mathcal{S}$ orthogonal to $|v\rangle$ and $|w\rangle$ respectively.

Using the fact that $\mathbf{\Delta}_0 \cdot \mathbf{\Pi}_0 \cdot \mathbf{\Delta}_0 |w\rangle = \frac{|w\rangle}{2}$, we can conclude that $\langle v|w\rangle = \frac{1}{\sqrt{2}}$.

Next, we note that $\mathbf{\Pi}_0$ and $\mathbf{\Delta}_0$ are one-dimensional projectors when restricted to $\mathcal{S}$. For any vector $|u\rangle = \alpha |v\rangle + \beta |w\rangle$, $\mathbf{\Pi}_0 |u\rangle = (\alpha + \beta)\mathbf{\Pi}_0 |w\rangle \in \mathrm{span}(|v\rangle)$, $\mathbf{\Delta}_0 |u\rangle =$

$\left(\frac{\alpha}{(2\|\mathbf{\Pi}_0|w\rangle\|} + \beta\right)|w\rangle \in \text{span}(|w\rangle)$. Therefore, the simulator's state, while it is in the 'for' loop, oscillates between the four vectors $|v\rangle, |v^\perp\rangle, |w\rangle$ and $|w^\perp\rangle$. The transition graph is as follows.

Figure 13: Transition diagram for $\mathbf{\Pi}_0, \mathbf{\Delta}_0$



In this figure, the arrows represent transition with probability 1/2.

From here, it is easy to conclude that with overwhelming probability, in n iterations, we will get a '0' outcome on performing a $\mathbf{\Pi}$ measurement, and when this happens, the resulting state will be the desired state.

Jordan's Lemma In the above scenario, we saw that our starting vector lies in a two-dimensional subspace $\mathcal{S}$, and both projectors project onto a one-dimensional subspace in $\mathcal{S}$. Consider any arbitrary projections $\mathbf{\Pi}, \mathbf{\Delta}$, their interaction is a generalization of the scenario encountered above. This is captured via *Jordan's lemma*, which we describe below. Jordan's lemma states that, given any two projective matrices $\mathbf{\Pi}, \mathbf{\Delta}$, there exist t one/two dimensional subspaces $\left\{\mathcal{S}_j\right\}_{j\in[t]}$ such that for each j, both $\mathbf{\Pi}$ and $\mathbf{\Delta}$ project $\mathcal{S}_j$ onto a one-dimensional subspace of $\mathcal{S}_j$. Note that if $\mathcal{S}_j$ is a two-dimensional subspace, then $\mathbf{\Pi}$ and $\mathbf{\Delta}$ can project onto different one-dimensional subspaces within $\mathcal{S}_j$. A formal statement of the lemma, as well as the proof, can be found here.

9 Quantum Proofs of Knowledge

Lecture 20: March 28th, 2023

We discussed classical proofs of knowledge in Section 5.9. Let us recall the Graph Hamiltonicity protocol.

Review of Blum's Graph Hamiltonicity Protocol This protocol uses statistically binding, computationally hiding commitments. The prover has a graph G and a Hamiltonian cycle H.

- **P1**: The prover chooses a uniformly random permutation π, computes a commitment com_0 to π and a commitment com_1 to $G' = \pi(G)$. Note that com_1 consists of n^2 commitments, one for each entry of the adjacency matrix of $\pi(G)$.

  It sends $(\mathsf{com}_0, \mathsf{com}_1)$ to the verifier.

- **V1**: The verifier chooses a uniformly random bit b.

- **P2**: If the bit $b = 0$, the prover sends openings $\mathsf{op}_0, \mathsf{op}_1$ for $\mathsf{com}_0, \mathsf{com}_1$ respectively.

If $b = 1$, the prover opens n out of the n^2 commitments in com_1, where the n locations are derived from the Hamiltonian cycle H.

- **V2**: If $b = 0$, the verifier checks that $\mathsf{op}_0, \mathsf{op}_1$ are valid openings for $\mathsf{com}_0, \mathsf{com}_1$.

  If $b = 1$, the verifier checks openings corresponding to the n locations in com_1, and checks that these n locations form an n-cycle.

  Let $\mathbf{V}_2$ denote this last-step verifier. It takes as input the first message msg_1, a bit b, the third message msg_3 and outputs $\mathbf{V}_2(\mathsf{msg}_1, b, \mathsf{msg}_3)$.

Blum's protocol satisfies special soundness - given two different accepting transcripts with the same first message, the witness can be extracted. This was used for the knowledge extraction.

We argued that if there exists a (cheating) prover that convinces the (honest) verifier with probability $1/2 + \epsilon$, then there exists an extractor that has black-box access to the prover, and extracts a Hamiltonian cycle in the graph. The classical extractor works as follows: it first runs the prover and obtains the first message msg_1. Next, it sends bit $b = 0$, receives msg_3, rewinds the prover and sends $b = 1$, receives msg_3'. If both msg_3 and msg_3' are valid messages, then using the perfect binding guarantee of the commitment scheme, we can extract a Hamiltonian cycle in the graph.

Analysis of classical extractor Let $\mathbf{P}$ be a prover that succeeds with probabity $1/2 + \epsilon$. First, let us classify the first message as 'good' or 'bad', depending on the success probabilities conditioned on this first message.

Definition 9.1. *Let* msg_1 *be any string. We say that* msg_1 *is* ϵ-good *if*

$$\sum_b \Pr\left[\mathbf{P} \text{ succeeds } : (\mathsf{msg}_1, b) \text{ are first two messages }\right] \geq 1 + \epsilon.$$

Here, the probabilities are taken over the choice of randomness used by the prover for computing the third message. ◊

First, note that since the prover succeeds with probility at least $1/2 + \epsilon$, we can show a lower bound on the probability of the first message being ϵ-good.

Observation 9.2.

$$\Pr\left[\mathsf{msg}_1 \text{ is } \epsilon\text{-good}\right] \geq \Omega(\epsilon).$$

Next, conditioned on the first message being ϵ-good, we can show a lower bound on the success probability of the extractor.

Observation 9.3.

$$\Pr\left[\mathbf{V}_2(\mathsf{msg}_1, 0, \mathsf{msg}_3) = 1 \wedge \mathbf{V}_2(\mathsf{msg}_1, 1, \mathsf{msg}_3) = 1 \ : \ \mathsf{msg}_1 \text{ is } \epsilon\text{-good}\right] \geq \Omega(\epsilon).$$

Putting these two observations together, we get that the extractor succeeds with probability at least $\Omega(\epsilon^2)$.

Note to scribe: please add proofs for the above observations.

9.1 *Quantum extraction*

As we saw in the case of post-quantum zero-knowledge, rewinding is an issue in the quantum setting.

Characterising a quantum malicious prover We can describe the malicious prover using three unitaries $\mathbf{P}_1, \mathbf{U}_0, \mathbf{U}_1$. Similar to the case of zero knowledge, since the prover is malicious, we allow the prover to take some auxiliary quantum state $|\phi\rangle$. Let t denote the length of the first and third messages. The prover works as follows:

- It takes as input some auxiliary information $|\phi\rangle$ over ℓ qubits, computes $\mathbf{P}_1 \cdot |\phi\rangle$ and measures the first t qubits. Let c be the measurement. This is the first message sent by the prover. The residual state is $|\phi_c\rangle$, which will be used by the next-stage prover.
- The prover receives a bit $b \in \{0,1\}$. It applies $\mathbf{U}_b$ to $|\phi_c\rangle$, and measures the first t qubits. This is the third message sent by the prover.

In class, I had initially said that w.l.o.g., we can assume $|\phi_c\rangle$ is embedded inside $\mathbf{U}_0$ and $\mathbf{U}_1$. As correctly pointed out by one of the students in class, we cannot say this without loss of generality. The prover will need two copies of $|\phi_c\rangle$, and may not be able to produce these two copies to be embedded inside $\mathbf{U}_0$ and $\mathbf{U}_1$.

Extractors for two easy cases

The extractor has black-box access to $\mathbf{P}_0, \mathbf{U}_0, \mathbf{U}_0^\dagger, \mathbf{U}_1, \mathbf{U}_1^\dagger$. A natural definitional question to ask: why does the extractor get black-box access to $\mathbf{U}_0^\dagger$ and $\mathbf{U}_1^\dagger$. We can also define quantum proofs of knowledge where the extractor has only black-box access to the prover (and not its constituent unitaries). This would be a stronger definition of quantum PoK, and it is not clear how to achieve this notion. Moreover, the version that we are using (where extractor has access to the inverses) suffices for most applications of PoK.

Easy case 1: First, let us consider the following prover: it always outputs an ϵ-good first message. Moreover, if the challenge bit is 0, it succeeds with probability 1, else it succeeds with probability ϵ.

A natural idea is the following: apply $\mathbf{U}_0$, measure the first t qubits, then apply $\mathbf{U}_0^\dagger$ followed by $\mathbf{U}_1$, and finally measure the first t qubits again. The hope is that the first measurement will not disturb the state, and therefore applying $\mathbf{U}_0^\dagger$ will be like starting the computation afresh. Therefore, we can hope that this extractor will succeed with probability ϵ (since the first measurement will produce message a valid msg_3 corresponding to challenge 0 with probability 1, and the second measurement will produce a valid msg_3' for challenge bit 1 with probability ϵ (assuming that we return to the original state after applying $\mathbf{U}_0^\dagger$).

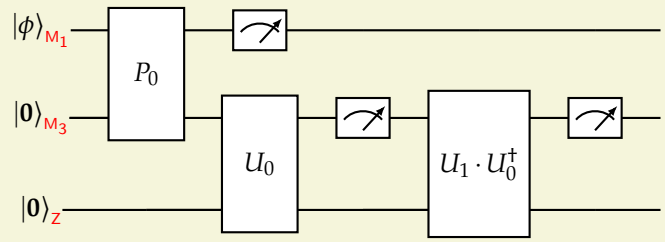
Unfortunately, our hope that the first measurement will not disturb the prover's state is not accurate, even if the underlying commitment scheme is perfectly binding. The problem is that perfect binding only says that the message is unique given the commitment; however there could be multiple openings corresponding to the same message. One way to make this argument work is to ask for stronger commitments where there is at most one opening for every commitment. Such commitments are called *strictly binding commitments*, which we define below.

Definition 9.4. *Let* $(\mathsf{Commit}, \mathsf{Verify})$ *be a perfectly binding commitment scheme. We say that this scheme satisfies* strict binding *if for any string* com*, there is at most one pair* (m, op) *such that* $\mathsf{Verify}(\mathsf{com}, m, \mathsf{op}) = 1$. ◊

See [AC02] for a construction of strictly binding commitments.

If we use strictly binding commitments in Blum's protocol, then we have an extractor for the first easy case. The extractor is shown in Figure 14 below. If both the measurements on M_3 register produce a valid third message, then the extractor can use them to extract a witness.

Figure 14: Extractor for Easy Case 1



Easy case 2: Let us consider another easy case. It is similar to the first one, except that the prover always succeeds if challenge bit is 1, but succeeds with probability ϵ if the challenge bit is 0. Here, a natural idea would be to swap $\mathbf{U}_0$ and $\mathbf{U}_1$. Does this suffice?

Unfortuantely, it does not. Consider an adversarial prover that has multiple Hamiltonian cycles in the graph. The prover sends a superposition of these cycles. As a result, the measurement after applying $\mathbf{U}_1$ will disturb the state. To prevent this, we need to tweak Blum's protocol : *in the first message, the prover must also commit to the Hamiltonian cycle that it will use!* The verifier will also perform an additional check if $b = 1$. It will also check the openings for the committed cycle, and will compare this against the openings provided for com_2.

General Case

The above cases illustrate a few necessary changes that need to be made to Blum's protocol.

- First, we need to use strictly binding commitments.
- Next, the prover must also commit to the Hamiltonian cycle in the first message.

These two changes ensure the following: given the first message msg_1, for $b \in \{0,1\}$, there exists at most one third message msg_3 such that $\mathbf{V}_2(\mathsf{msg}_1, b, \mathsf{msg}_3) = 1$. Moreover, as in Blum's protocol, an extractor can extract a witness given two different transcripts with the same first message.

What should the general case extractor look like? As a first attempt, we can try the extractor that worked for the easy cases described above (in Figure 14). Suppose our prover succeeds with probability $1/2 + \epsilon$. As in the classical case, we will assume that the first message msg_1 is ϵ-good (this happens with probability $\Omega(\epsilon)$). Corresponding to msg_1, for each $b \in \{0,1\}$, let msg_3^b be the

unique string such that $\mathbf{V}_2(\mathsf{msg}_1, b, \mathsf{msg}_3^b) = 1$, and let $\Delta_{\mathsf{msg}_1,b} = \left|\mathsf{msg}_3^b\right\rangle\!\left\langle\mathsf{msg}_3^b\right| \otimes \mathbf{I}$ be the projection matrix that projects onto an accepting transcript for each $b \in \{0,1\}$. The two-outcome projective measurement $\left(\Delta_{\mathsf{msg}_1,b}, \mathbf{I} - \Delta_{\mathsf{msg}_1,b}\right)$ can be performed using the verifier $\mathbf{V}_2(\mathsf{msg}_1, b, \cdot)$.

Since msg_1 is ϵ-good, we have the following:

$$\left\|\Delta_{\mathsf{msg}_1,0} \cdot \mathbf{U}_0 \cdot \left|\phi_{\mathsf{msg}_1}\right\rangle |\mathbf{0}\rangle\right\|^2 + \left\|\Delta_{\mathsf{msg}_1,1} \cdot \mathbf{U}_1 \cdot \left|\phi_{\mathsf{msg}_1}\right\rangle |\mathbf{0}\rangle\right\|^2 \geq 1 + \epsilon \tag{9.5}$$

Using this, we wish to show a lower bound on the extractor's success probability (conditioned on msg_1 being ϵ-good). The extractor's success probability is

$$p_{\text{Ext}} = \left\|\Delta_{\mathsf{msg}_1,1} \cdot \mathbf{U}_1 \cdot \mathbf{U}_0^\dagger \cdot \Delta_{\mathsf{msg}_1,0} \cdot \mathbf{U}_0 \cdot \left|\phi_{\mathsf{msg}_1}\right\rangle |\mathbf{0}\rangle\right\|^2.$$

Note to reviewer: more explanation needed for the above expression?

Before we prove this formally, let us see what Equation 9.5 gives us, and what we need to prove. Suppose $\Delta_{\mathsf{msg}_1,0}$ and $\Delta_{\mathsf{msg}_1,1}$ are both one-dimensional subspaces (this is not the case here, but let us consider this simplification since this has a neat geometric picture). The two one-dimensional spaces are given by vectors $|v_0\rangle$ and $|v_1\rangle$. We are given a vector $|w\rangle$ (which is $\left|\phi_{\mathsf{msg}_1}\right\rangle |\mathbf{0}\rangle$) such that the angle between $|w\rangle$ and $|v_b\rangle$ is θ_b, and $\cos^2(\theta_0) + \cos^2(\theta_1) = 1 + \epsilon$. Now consider p_{Ext}. This is equal to $\cos^2(\theta_1) \cdot \cos^2(\theta)$, where θ is the angle between $|v_0\rangle$ and $|v_1\rangle$. Note that the angle between these two vectors is at most $\theta_0 + \theta_1$. As a result, it suffices to have a lower bound on $\cos^2(\theta_0) \cdot \cos^2(\theta_0 + \theta_1)$, subject to the restriction $\cos^2(\theta_0) + \cos^2(\theta_1) = 1 + \epsilon$. Suppose we take $\theta_0 = 0$, $\theta_1 = \arccos\left(\sqrt{\epsilon}\right)$. In this case, $\cos^2(\theta_0) \cdot \cos^2(\theta_0 + \theta_1) = \epsilon$. Is this the lowest possible value that $\cos^2(\theta_0) \cdot \cos^2(\theta_0 + \theta_1)$ can take?

Formal analysis: The formal analysis closely follows the proof of Lemma in [CSST11].

Note for scribe: please add intermediate steps here. Check with me for handwritten notes for this calculation if needed.

$$\begin{aligned}
&\Delta_{\mathsf{msg}_1,1} \cdot \mathbf{U}_1 \cdot \mathbf{U}_0^\dagger \cdot \Delta_{\mathsf{msg}_1,0} \cdot \mathbf{U}_0 \cdot \left|\phi_{\mathsf{msg}_1}\right\rangle |\mathbf{0}\rangle \\
&= \sqrt{p_{1,\mathsf{msg}_1}} \left|\mathsf{msg}_3^1\right\rangle |v_1\rangle - \sqrt{1 - p_{0,\mathsf{msg}_1}} \Delta_{\mathsf{msg}_1,1} \cdot \mathbf{U}_1 \cdot \mathbf{U}_0^\dagger \left|w_0^\perp\right\rangle
\end{aligned}$$

The square of length of this vector is at least $\left(\sqrt{p_{1,\mathsf{msg}_1}} - \sqrt{1 - p_{0,\mathsf{msg}_1}}\right)^2$, and this quantity is at least ϵ^2.

Exercise 9.1. *Consider a three-message protocol that satisfies special soundness. The challenge space (for the second message) is $\{0,1,2,3\}$, and the protocol is a proof of knowledge with knowledge error $1/4$. Propose a quantum extractor for this protocol.*

9.2 *Using Computationally Binding Commitments in Blum's Protocol*

Lecture 21: March 31st, 2023

In the last section, we saw how to modify Blum's protocol by using strictly binding commitments to achieve extraction even against malicious quantum provers. Abstractly, let us recall the protocol. The challenge space is $\{0,1\}$. The prover, in the first message, sends a commitment c to $\mathsf{ans}_0, \mathsf{ans}_1$. On receiving the challenge bit b, it produces ans_b, together with the corresponding opening op_b. The verifier $\mathbf{V}_2$ checks if op_b is a valid opening for ans_b (w.r.t. commitment c), and checks if ans_b is a correct answer for challenge b.

We had to use strict commitments — for every c, there is at most one (message, opening) pair that verifies. However, there are a couple of drawbacks wrt using such commitments:

- First, since the commitment scheme is perfectly binding, the commitment should (information-theoretically) contain the witness. In particular, this means that in Blum's protocol, the amount of communication must be at least quadratic in the number of vertices (since we need a commitment to the adjacency matrix). Can we have more succinct commitments by relaxing the protocol's soundness requirement from statistical to computational?
- If we need a protocol with statistical zero knowledge, then the underlying commitments need to be statistically hiding (and therefore, we need to rely on computational binding for such commitments).

In this section, we will prove soundness of Blum's protocol (wrt computationally bounded provers) by using an appropriate notion of computationally binding commitments. If our commitments are also statistically hiding, this will give us a protocol with (post-quantum) computational soundness and statistical zero knowledge.

Information/disturbance tradeoff, and using it for extraction

First, we will present a slightly different perspective for the extraction described in the previous section. This perspective is based on a well-studied principle in quantum information. It goes by various names, but the basic idea is similar: *if a measurement does not disturb a quantum state, then you don't learn any new information from the measurement*.

The above statement is quite informal (how to define 'information learnt from a measurement'?), but it can be made formal via the 'Gentle Measurement Lemma'. First, let us understand the scenario through a few examples. Consider a single qubit $|\psi\rangle$, and suppose you know that $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$. Let $\mathcal{P} = (|\psi\rangle\langle\psi|, \mathbf{I} - |\psi\rangle\langle\psi|)$ be a two-outcome projective measurement applied to $|\psi\rangle$. Clearly, this measurement does not disturb the state. On the other hand, you also know that this measurement will always produce outcome '0', and therefore, you don't learn any new information.

Next, let us consider a different projective measurement $\mathcal{P}' = (|\psi'\rangle\langle\psi'|, \mathbf{I} - |\psi'\rangle\langle\psi'|)$, where $|\langle\psi|\psi'\rangle| = \sqrt{1-\epsilon}$. In this case, the measurement may output '1' with (low) probability ϵ. However, if the output is '0', then the resulting state is $|\psi'\rangle\langle\psi'|$, and this state is not very far from the original state $|\psi\rangle\langle\psi|$. The trace distance between the pre-measurement state and the post-measurement state is bounded

by ϵ. The 'gentle measurement lemma' generalizes this for arbitary state and measurements.

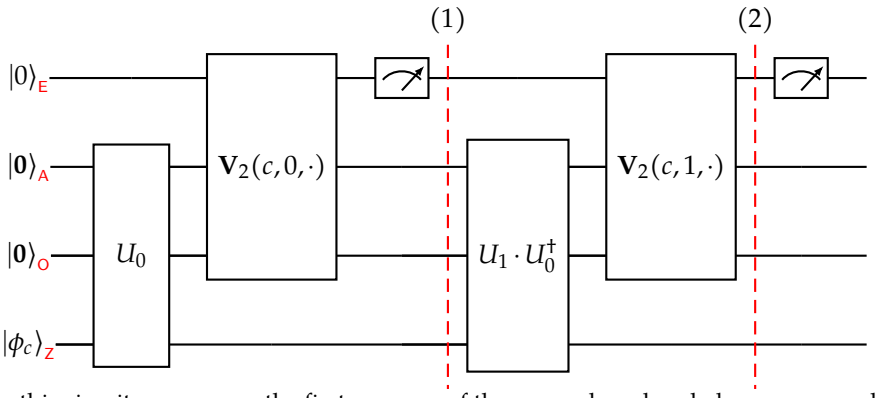
Lemma 9.6. *Let ρ be a mixed state, and $\mathcal{P} = (\mathbf{P}, \mathbf{I} - \mathbf{P})$ any projective measurement such that $\mathsf{Tr}(\mathbf{P} \cdot \rho) = 1 - \epsilon$. Then*

$$\left\| \rho - \rho' \right\|_{\mathrm{tr}} \leq 2\sqrt{\epsilon}$$

where ρ' is the post-measurement state, conditioned on measurement output being 0.

Let us use the gentle mesaurement lemma for analysing the extractor described in previous section. First, consider the quantum circuit described in Figure 15.

Figure 15: Quantum Circuit for Consecutive Verification-measurements



In this circuit, we assume the first message of the prover has already been measured (and the measured value c is used by the verifier $\mathbf{V}_2$). The residual prover state $|\phi_c\rangle$ is in register Z. The prover also takes two other registers A, O which together will contain the third message (register A contains ans_b, and register O contains the corresponding opening). The verifier $\mathbf{V}_2$ takes three inputs: the first message c, a bit $b \in \{0,1\}$ and the prover's third message (which includes an answer and an opening). In the above circuit, we run this verification circuit twice, with $b = 0$ and $b = 1$ respectively. The verification's output is stored in register E.

Suppose c is the first message, and $p_{b,c} = \Pr\left[\mathbf{V}_2(c, b, \mathsf{msg}_3) = 1\right]$. If the first measurement was not present in the above circuit, then we know that the final measurement outputs 1 with probability $p_{1,c}$. Using the gentle measurement lemma, we can show a bound on the probability of both measurements outputting 1. The details are included as an exercise in Assignment 4.

Introducing Collapse-binding Commitments

In the previous section, we argued that if a prover succeeds with probability $p_{b,c}$ when the challenge bit is b, and $p_{0,c} + p_{1,c} \geq 1 + \epsilon$, then performing a '0' verification (that is, running $\mathbf{V}_2(c,0,\cdot)$ and then measuring the verification output) followed by a '1' verification (that is, running $\mathbf{V}_2(c,1,\cdot)$ and then measuring the verification output) will produce a '1' in both verifications with probability $\mathrm{poly}(\epsilon)$. However, for extraction of witness, we need to measure the register A,

which contains the prover's answers ans_0 and ans_1. If we add these measurement to the circuit in Figure 15, then we are no longer guaranteed that both verifications output 1 with probability $\mathrm{poly}(\epsilon)$. Maybe, the moment we measure A the first time, the state is disturbed so much that we can have no guarantees on the second verification outputting 1.

What if we had commitments which would guarantee that measuring register A *does not disturb the state at all?* Note that if the commitment scheme was perfectly binding, then we know that if the first verification output 1, then the measurement of register A will not disturb the state. Can we have similar guarantees with commitment schemes that are not perfectly/statistically binding? This motivates the definition of *collapse-binding commitments*. While these were first studied by Unruh [Unr16] in the context of computational soundness of ZK protocols, it happens to be the 'right' analogue of computational binding in the quantum setting!

Definition 9.7 (Collapse-binding Commitments)**.** *A (keyed) collapse-binding statistically-hiding commitment scheme consists of two algorithms* Commit, Verify *with the usual syntax, and the following two security requirements:*

Collapse-Binding: *any q.p.t. adversary $\mathcal{A}$ has negligible advantage in the following security game:*

1. *Challenger picks a key k and sends to the adversary.*
2. *Adversary sends a commitment value c together with a quantum state $|\psi\rangle_{\mathsf{A},\mathsf{O}}$ over two registers* A, O *such that*

$$|\psi\rangle_{\mathsf{A},\mathsf{O}} = \sum_{\substack{(a,o):\\ \mathsf{Verify}(k,c,a,o)=1}} \alpha_{a,o} |a\rangle_{\mathsf{A}} |o\rangle_{\mathsf{O}} .$$

   The adversary can also maintain some additional registers entangled with A, O*.*
3. *Challenger picks a bit $b \leftarrow \{0,1\}$. If $b = 0$, it measures register* A*, else it does nothing. It sends the resulting state to $\mathcal{A}$.*
4. *$\mathcal{A}$ sends its guess b' and wins if $b = b'$.*

Statistical Hiding: *Same as the classical definition.* ◊

It is not immediately clear why this definition is useful. For instance, suppose we use collapse-binding commitments in some scheme. When the adversary sends a quantum state, how do we ensure/verify that this state is a superposition over all valid (answer, opening) pairs? There is an equivalent security game for collapse-binding which we present below.

Definition 9.8 (Collapse-binding Commitments)**.** *A (keyed) collapse-binding statistically-hiding commitment scheme consists of two algorithms* Commit, Verify *with the usual syntax. We say that it satisfies collapse-binding if any q.p.t. adversary $\mathcal{A}$ has negligible advantage in the following security game:*

1. *Challenger picks a key k and sends to the adversary.*
2. *Adversary sends a commitment c and a quantum state $|\psi\rangle_{\mathsf{A},\mathsf{O}}$ over two registers* A, O *such that*

$$|\psi\rangle_{\mathsf{A},\mathsf{O}} = \sum_{(a,o)} \alpha_{a,o} |a\rangle_{\mathsf{A}} |o\rangle_{\mathsf{O}} .$$

The adversary can also maintain some additional registers entangled with A, O.

3. *Challenger initializes a new register* $|0\rangle_{\mathsf{V}}$*, and computes* $\mathsf{Verify}(c, a, o)$ *in superposition over registers* A, O*, and stores the value in register* V*. It measures the register* V*. If the measurement is* 0*, it sends* $\perp$ *to* $\mathcal{A}$*. Else, it picks a bit* $b \leftarrow \{0,1\}$*. If* $b = 0$*, it measures register* A*, else it does nothing. It sends the resulting state to* $\mathcal{A}$*.*
4. $\mathcal{A}$ *sends its guess* b' *and wins if* $b = b'$*.*

Statistical Hiding: *Same as the classical definition.* ◊

Unruh [Unr16] showed that these two definitions are equivalent. Typically, when we want to show that a construction is collapse-binding, we use Definition 9.7, and when we use collapse-binding commitments in a construction, we use Definition 9.8.

Collapse-binding commitments satisfy the usual computational binding property too. Suppose there exists an adversary that can break the computational binding property. Then it can find two different (message, opening) pairs corresponding to the same committed value c. It can create a uniform superposition of these two pairs and send them to the challenger. Later, when the challenger returns the registers, it can apply an appropriate unitary to detect whether the measurement was performed or not.

This is similar to distinguishing the $|+\rangle\langle+|$ *state and the maximally mixed state* $0.5(|0\rangle\langle 0| + |1\rangle\langle 1|)$*.*

In the classical setting, computationally binding commitments are constructed using collision-resistant hash functions. Therefore, it is natural to ask: what is the 'right' notion of collision-resistance in the quantum setting? Similar to commitments, we can define the notion of *collapsing hash functions*. These were first defined by Unruh [Unr16].

Definition 9.9. *Let* $\mathcal{H} = \left\{H_k : \{0,1\}^n \to \{0,1\}^{n/2}\right\}_{k \in \mathcal{K}}$ *be an efficiently computable compressing keyed function family. We say that* $\mathcal{H}$ *is a collapsing hash function if no q.p.t. adversary* $\mathcal{A}$ *can succeed in the following security game with non-negligible advantage:*

1. *Challenger sends a hash key* k*.*
2. *Adversary sends a string* $h \in \{0,1\}^{n/2}$ *together with a quantum state* $|\psi\rangle_{\mathsf{X}} = \sum_{x:H_k(x)=h} |x\rangle_{\mathsf{X}}$*.*
3. *Challenger picks a bit* b*. If* $b = 0$*, it measures* $|\psi\rangle$*, else it does nothing.*
4. $\mathcal{A}$ *sends a bit* b' *and wins if* $b = b'$*.*

◊

Similar to collapse-binding commitments, we can have an equivalent definition where the adversary just sends a state $|\psi\rangle_{\mathsf{X}}$. The challenger computes the hash on a separate register A, and either measures register X or register A.

Lecture 22: April 1st, 2023

Partial measurement vs no measurement. The above definition of collapsing hash (and also collapse-binding commitments) says that the adversary cannot distinguish between full measurement and no measurement of the register X. Can we use this to conclude that no q.p.t. adversary can distinguish between

a partial measurement and no measurement? More generally, suppose f is some efficient classical function. Consider the projective measurement $\mathcal{P} = \left\{\mathbf{P}_a = \sum_{x:f(x)=a} |x\rangle\langle x|\right\}_a$. Suppose a hash function is collapsing, can we conclude that no adversary can guess whether $\mathcal{P}$ is applied or not? This follows from a hybrid argument.

- **Hybrid** H_0: Challenger performs projective measurement $\mathcal{P}$ on register X.
- **Hybrid** H_1: Challenger performs full measurement on register X.
- **Hybrid** H_2: Challenger performs no measurement.

Since the adversary can distinguish between Hybrid H_0 and Hybrid H_2, it can either distinguish between H_0 and H_1, or can distinguish between H_1 and H_2. In both these cases, it can be used to break collapsing property (note that Definition 9.9 can be extended to allow the adversary to send a mixed state instead of a pure state; this would be needed to use the distinguisher between H_0 and H_1.

From Collapsing Hash to Collapse-Binding Commitments

Classically, we have a few constructions of computational-binding commitments from collision-resistant hash. The following construction was proposed in class;

Construction 9.10 (Comp. Binding Comp. Hiding Commitment Scheme)**.** *This construction uses a perfectly correct semantically secure public-key encryption scheme and a collision-resistant hash function. The commitment key is the hash key k.*

- $\mathsf{Commit}(k, m)$: *The commitment algorithm first samples* $(\mathsf{pk}, \mathsf{sk})$ *using randomness* r_1*. Next, it computes* $\mathsf{ct} = \mathsf{Enc}(m, \mathsf{pk}; r_2)$ *and finally outputs* $H_k(\mathsf{ct})$*. The opening is* (r_1, r_2)*.*

The scheme is comp. binding and comp. hiding. ◊

The following commitment scheme, proposed by Halevi and Micali [HM96], is comp. binding and stat. hiding.

As pointed out by one of the students in class, here the size of the commitment is larger than the size of the message. One of the motivations for using comp. binding commitments is that we can hope for shorter commitments. This can be achieved by computing a hash of the commitment. See Exercise 9.2.

Construction 9.11 (Comp. Binding Stat. Hiding Commitment Scheme)**.** *The construction uses a hash function family* $\mathcal{H} = \{H_k : \{0,1\}^m \to \{0,1\}^n\}$ *where m is sufficiently large (this is used for the statistical hiding part).*

- $\mathsf{Commit}(k, x \in \{0,1\}^n)$: *The commitment algorithm first samples* $\mathbf{u} \leftarrow \{0,1\}^m$*, matrix* $\mathbf{A} \leftarrow \mathbb{Z}_2^{n \times m}$*. It computes* $h = H_k(x)$*,* $\mathbf{b} = \mathbf{A} \cdot \mathbf{u} + \mathbf{x} \bmod 2$*. The commitment is* $(h, \mathbf{A}, \mathbf{b})$ *and the opening is* $\mathbf{u}$*.*

Intuitively, if m is large, then h reveals only n bits of information about $\mathbf{u}$*. As a result, there is sufficient entropy in* $\mathbf{u}$*, even conditioned on h. Finally, using an LHL-style argument, we can conclude that* $(\mathbf{A}, \mathbf{A} \cdot \mathbf{u} + \mathbf{x})$ *information-theoretically hides* $\mathbf{x}$*. See [HM96] for a formal proof for the statistical hiding part.*

Computational binding follows from the collision-resistance of $\mathcal{H}$. If an adversary can find $\mathbf{x}_1, \mathbf{x}_2, \mathbf{u}_1, \mathbf{u}_2$ such that $(H_k(\mathbf{u}_1), \mathbf{A}\cdot\mathbf{u}_1 + \mathbf{x}_1) = (H_k(\mathbf{u}_2), \mathbf{A}\cdot\mathbf{u}_2 + \mathbf{x}_2)$ and $\mathbf{x}_1 \neq \mathbf{x}_2$, then $\mathbf{u}_1 \neq \mathbf{u}_2$, giving us a collision for $\mathcal{H}$.

$\Diamond$

Below, we prove Construction 9.11 is collapse-binding.

Claim 9.12. *Suppose there exists a q.p.t. adversary $\mathcal{A}$ that breaks the collapse-binding property of Construction 9.11. Then there exists a q.p.t. adversary $\mathcal{B}$ that breaks the collapsing property of $\mathcal{H}$.*

Proof. Suppose the adversary sends a commitment $c = (h, \mathbf{A}, \mathbf{b})$ and registers X, U containing the message and the opening respectively. A natural first idea for our reduction is the following: the reduction sends h and U to the collapsing challenger, and then returns X, U to the adversary. However, this does not work.

Suppose the adversary sent a commitment $c = (h, \mathbf{A}, \mathbf{b})$ and state

$$|\psi\rangle_{\mathsf{X},\mathsf{U}} = \sum_{\substack{x,u:\\ h=H_k(u)\\ \mathbf{A}\cdot\mathbf{u}+\mathbf{B}=\mathbf{x}}} \alpha_{x,u} |x\rangle_{\mathsf{X}} |u\rangle_{\mathsf{U}}$$

and the reduction forwards h and register U to the challenger. If the challenger does not measure, then this is identical to what the adversary $\mathcal{A}$ expects when $b = 1$. However, if the challenger measures the register U, then the resulting mixed state is a mixture of

$$\rho_1 = \{p_u, |\mathbf{A}\cdot\mathbf{u} + \mathbf{b}\rangle_{\mathsf{X}} |u\rangle_{\mathsf{U}}\}_u$$

where $p_u \in [0,1]$ is the probability of seeing measurement u. However, this is not what the adversary $\mathcal{A}$ expects when $b = 0$. When $b = 0$, the register X should have been measured, resulting in the following mixed state:

$$\rho_2 = \left\{ p_x, \sum_{\substack{u:H(u)=h\\ \mathbf{A}\cdot\mathbf{u}+\mathbf{b}=\mathbf{x}}} |x\rangle_{\mathsf{X}} |u\rangle_{\mathsf{U}} \right\}_x .$$

Check that this is not the same as ρ_1.

Instead, the reduction must specify an appropriate projective measurement on the register U. Consider the projective measurement

$$\mathcal{P} = \left\{ \mathbf{P}_x = \sum_{\substack{u:H(u)=h,\\ \mathbf{A}\cdot\mathbf{u}+\mathbf{B}=\mathbf{x}}} |u\rangle\langle u| \right\}_x \bigcup \left\{ \mathbf{P}_\perp = \sum_{u:H(u)\neq h} |u\rangle\langle u| \right\}$$

defined using $\mathbf{A}, \mathbf{b}$. Applying this projective measurement on register U is identical to measuring register X. This completes our reduction.

□

Exercise 9.2. *Let $\mathcal{H} = \{H_k\}_k$ be a collapsing hash function, and* Commit *a collapse-binding commitment scheme. Show that the following commitment scheme is also collapse-binding:*

- $\mathsf{Commit}'((k_1,k_2),x;r)$*: Compute* $c = \mathsf{Commit}(k_2,x;r)$ *and output* $H_{k_1}(c)$.

Exercise 9.3. *Show that Construction 9.10 is a collapse-binding commitment scheme, assuming the public key encryption scheme is perfectly correct and $\mathcal{H}$ is collapsing.*

Lecture 23: April 11th, 2023

Construction of Collapsing Hash from Standard Cryptographic Assumptions

Somewhat surprisingly, collapsing hash can be constructed from a number of classical cryptographic primitives. Below, we present one construction, using injective/lossy functions.

Definition 9.13 (Injective/Lossy Functions)**.** *An injective/lossy function family with lossiness parameter $\alpha > 1$ consists of a keyed function $f : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^m$ where $m > n$, together with sampling algorithms* $\mathsf{Setup\text{-}Inj}(1^n)$ *and* $\mathsf{Setup\text{-}Lossy}(1^n)$*. It has the following correctness requirements:*

Injectivity *For every $k \leftarrow \mathsf{Setup\text{-}Inj}(1^n)$, the function $f(k,\cdot)$ is an injective function.*

Lossiness *For every $k \leftarrow \mathsf{Setup\text{-}Lossy}(1^n)$, $|\{f(k,x) : x \in \{0,1\}^n\}| \leq 2^{n/\alpha}$.*

For security, we require that a key sampled using Setup-Inj *is computationally indistinguishable from a key sampled using* Setup-Lossy. ◊

We can relax the correctness requirements to hold for almost all keys in both the modes.

Construction 9.14 (Collapsing Hash from Injective/Lossy Functions)**.** *Let $f : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^m$ be an injective/lossy function with lossiness parameter 5, and let $\mathcal{G} = \{G_k : \{0,1\}^m \to \{0,1\}^{n/2}\}_k$ be a pairwise-independent function family.*
The key for our hash function is sampled as follows: pick $k_1 \leftarrow \mathsf{Setup\text{-}Lossy}(1^n)$, let k_2 be a uniformly random key for $\mathcal{G}$. The hash key is $k = (k_1,k_2)$.

$$H_{(k_1,k_2)}(x) = G_{k_2}\left(f_{k_1}(x)\right).$$

◊

Lemma 9.15. *Construction 9.14 is a collapsing hash function.*

Proof. To build intuition for this proof, the reader is advised to consider the construction without G_{k_2}. This function is not compressing (the function G_{k_2} is

used to make the overall function compressing). However, this simpler variant is collapsing, assuming security of the lossy/injective function.

A key observation: with high probability over the choice of k_2, the function $G_{k_2}(\cdot)$ is injective over the image of $f(k,\cdot)$. This is summarised below, and the proof follows via union bound.

Observation 9.16. *Let S be any subset of $\{0,1\}^m$, of size at most $2^{n/5}$. There exists a negligible function* $\mathrm{negl}(\cdot)$ *such that the following holds:*

$$\mathsf{Pr}_k\left[G_k(x_1) = G_k(x_2) \text{ for some } x_1, x_2 \in S\right] \leq \mathrm{negl}(n).$$

Below, we present a sequence of hybrid experiments, where the first hybrid corresponds to $b = 0$ in the collapsing game, and the last hybrid corresponds to $b = 1$.

- **Hybrid** H_0: The challenger samples $k_1 \leftarrow \mathsf{Setup\text{-}Lossy}(1^n)$, k_2 and sends them to the adversary. The adversary sends h, together with

$$|\psi\rangle_{\mathsf{X}} = \sum_{\substack{x: \\ G_{k_2}(f(k_1,x))=h}} \alpha_x |x\rangle_{\mathsf{X}}.$$

  The challenger measures register X.

- **Hybrid** H_1: This is identical to the previous hybrid, except that the *challenger aborts if G_{k_2} is not injective on the image of $f(k_1,\cdot)$*. Note that this happens with negligible probability (using Observation 9.16). If G_{k_2} is injective over the image of $f(k_2,\cdot)$, then let h' be the unique point in the image space such that $G_{k_2}(h') = h$. We can then express $|\psi\rangle_{\mathsf{X}} = \sum_{x:f(k_1,x)=h'} \alpha_x |x\rangle_{\mathsf{X}}$.

- **Hybrid** H_2: This is similar to the previous hybrid, except that the challenger chooses $k_1 \leftarrow \mathsf{Setup\text{-}Inj}(1^n)$. This is indistinguishable from the previous hybrid, assuming security of lossy/injective function family. Note that since f_{k_1} is injective, $|\psi\rangle$ contains a single state $|x\rangle_{\mathsf{X}}$ in the superposition.

- **Hybrid** H_3: This is similar to the previous hybrid, except that the challenger *does not measure register* X. This is identical to the previous hybrid since $|\psi\rangle_{\mathsf{X}}$ contains a single state $|x\rangle_{\mathsf{X}}$ in the superposition.

- **Hybrid** H_4: In the hybrid, the challenger samples $k_1 \leftarrow \mathsf{Setup\text{-}Lossy}(1^n)$, and does not measure register X. This is the final hybrid.

□

Conclusion: Collapsing hash functions, and collapse-binding commitments appear to be the right definitions for hash functions and commitments in the quantum setting. In Assignment 4, we discuss why collapse-binding implies another natural security definition of commitment schemes, called *sum-binding*. However, if a scheme is only computationally binding (and not collapse-binding), then we cannot show that it is sum-binding.

The soundness/AoK proof of Blum's protocol, using collapse-binding commitments, is discussed in Assignment 4.

We saw one construction for collapsing hash using injective/lossy functions. Another construction from claw-free functions, is discussed in Assignment 4. Finally, in Assignment 5, we will see constructions of these primitives from standard assumptions such as LWE.

An interesting question is whether every collision-resistant hash function is also collapsing. Or, can we show a CRHF that is not collapsing? Such a separation would be very interesting, because it implies *publicly verifiable quantum money*, which is the next topic of discussion in this course.

10 Quantum Goldreich-Levin Theorem

We discussed this topic in Lecture 19, before discussing quantum proof of knowledge in Lectures 20-22.

In this section, we will see yet another classical scenario that requires rewinding the adversary.

10.1 *Constructing a PRG from a one-way permutation*

Lecture 19: March 21st, 2023

Let $f : \{0,1\}^n \to \{0,1\}^n$ be a secure OWP. Can we construct a pseudorandom generator G using f? First, let us consider a few natural candidates.

- Let $G_f(x_1, \ldots, x_n) = (x_1, f(\mathbf{x}))$. This gives us a length-expanding function, but G_f may not be a secure PRG, even if f is a secure one-way permutation. For instance, consider a OWP that simply outputs the first bit (that is, $f(\mathbf{x}) = (x_1, \ldots)$). There exist secure OWPs that satisfy this property. However, if we consider the output of G_f, then the first two bits are always identical, and hence the output is not pseudorandom.

  Such an OWP can be constructed using a secure OWP $f' : \{0,1\}^{n-1} \to \{0,1\}^{n-1}$.

- Let us consider something a bit more complicated. Let $G(\mathbf{x}) = (\mathbf{z}, y_n)$ where $\mathbf{y} = f(\mathbf{x})$, $\mathbf{z} = f(x_1, y_1, \ldots, y_{n-1})$. Even here, we cannot use the one-wayness of f to argue that G is a secure PRG. Suppose f is such that it outputs the first bit of its input in the clear at the end (that is, $f(\mathbf{x}) = (\ldots, x_1)$). Then note that the last two bits of the PRG output are always identical, and hence G is not a secure PRG.

A secure construction follows from *Goldreich-Levin's hardcore bit theorem*, which we state next.

The Goldreich-Levin Theorem also holds for OWFs, but we state it for OWPs here.

Theorem 10.1 (Goldreich-Levin Hardcore Bit Theorem)**.** *Let $f : \{0,1\}^n \to \{0,1\}^n$ be a secure OWP. For any p.p.t. adversary $\mathcal{A}$, there exists a negligible function μ such that for all n,*

$$\mathsf{Pr}_{\mathcal{A},\mathbf{x},\mathbf{r}}\left[\mathcal{A}(f(\mathbf{x}), \mathbf{r}) = \mathbf{x} \cdot \mathbf{r} \bmod 2\right] \leq \frac{1}{2} + \mu(|\mathbf{x}|).$$

In other words, if there exists a p.p.t. adversary $\mathcal{A}$ that can predict $\mathbf{x} \cdot \mathbf{r} \bmod 2$ with non-negligible advantage given $f(\mathbf{x})$ and $\mathbf{r}$, then there exists a p.p.t. algorithm $\mathcal{B}$ that can compute $\mathbf{x}$ given $f(\mathbf{x})$.

Using Goldreich-Levin theorem, we can construct a secure PRG $G : \{0,1\}^{2n} \to \{0,1\}^{2n+1}$ as follows:

$$G(\mathbf{x}, \mathbf{r}) = (f(\mathbf{x}), \mathbf{r}, \mathbf{x} \cdot \mathbf{r} \bmod 2).$$

We will not prove the Goldreich-Levin theorem here, but will instead prove an easier version of the theorem (which will illustrate the challenges in establishing a quantum analogue).

GL Theorem for deterministic adversaries that succeed w.p. $3/4 + 1/\text{poly}$: Suppose there exists a p.p.t. adversary $\mathcal{A}$ such that for all x,

$$\Pr_{\mathbf{r}}\left[\mathcal{A}(f(\mathbf{x}), \mathbf{r}) = \mathbf{x} \cdot \mathbf{r}\right] \geq 3/4 + 1/p(n).$$

We will construct a p.p.t algorithm $\mathcal{B}$ such that for all x,

$$\Pr\left[\mathcal{B}(f(\mathbf{x})) = \mathbf{x}\right] \approx 1.$$

Classical Reduction: From hardcore bit prediction to OWP inversion

for $i = 1$ *to* n **do**
  for $j = 1$ *to* $n \cdot p(n)$ **do**
    $\mathbf{r}_{i,j} \leftarrow \{0,1\}^n$
    $y_{i,j} = \mathcal{A}(f(\mathbf{x}), \mathbf{r}_{i,j}) \oplus \mathcal{A}(f(\mathbf{x}), \mathbf{r}_{i,j} \oplus \mathbf{e}_i)$
  Set $x_i = \text{maj}\left\{y_{i,j}\right\}_j$
Output $\mathbf{x} = (x_1, \ldots, x_n)$.

The following observation is easy to verify. It follows from the success probability of $\mathcal{A}$.

Observation 10.2. *For all i, j,*

$$\Pr_{\mathbf{r}_{i,j}}\left[y_{i,j} = x_i\right] \geq 1/2 + 2/p(n).$$

Since all the $\left\{r_{i,j}\right\}_j$ are chosen independently, the random variables $\left\{y_{i,j}\right\}_j$ are independent and identically distributed. Using *Chernoff bounds*, the probability that the majority of $\left\{y_{i,j}\right\}_j$ is not equal to x_i is negligible in n. And therefore, algorithm $\mathcal{B}$ outputs each of the bits correctly with probability close to 1.

Chernoff bounds: suppose you toss a coin with bias ϵ (which is some non-negligible quantity). After sufficiently many tosses, you expect to see a H in $1/2 + \epsilon$ fraction of the tosses. Chernoff bound states that after $poly(n/epsilon)$ independent tosses, the probability that majority of tosses output T is negligible in n.

Goldreich-Levin theorem proves the hardcore bit property for any adversary that has non-negligible advantage. A formal proof can be found here.

10.2 *Quantum Goldreich-Levin Theorem*

A quantum analogue of the Goldreich-Levin theorem was given by Adcock and Cleve [AC02]. We start with a formal statement of our theorem.

TODO ...

References

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, USA, 1st edition, 2009.

[ABKM22] Gorjan Alagic, Chen Bai, Jonathan Katz, and Christian Majenz. Post-Quantum Security of the Even-Mansour Cipher. In Orr Dunkelman and Stefan Dziembowski, editors, *Advances in Cryptology - EUROCRYPT 2022 - 41st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Trondheim, Norway, May 30 -*

June 3, 2022, Proceedings, Part III, volume 13277 of *Lecture Notes in Computer Science*, pages 458–487. Springer, 2022.

[AC02] Mark Adcock and Richard Cleve. A Quantum Goldreich-Levin Theorem with Cryptographic Applications. In Helmut Alt and Afonso Ferreira, editors, *STACS 2002, 19th Annual Symposium on Theoretical Aspects of Computer Science, Antibes - Juan les Pins, France, March 14-16, 2002, Proceedings*, volume 2285 of *Lecture Notes in Computer Science*, pages 323–334. Springer, 2002.

[AG11] Sanjeev Arora and Rong Ge. New Algorithms for Learning in Presence of Errors. In Luca Aceto, Monika Henzinger, and Jirí Sgall, editors, *Automata, Languages and Programming - 38th International Colloquium, ICALP 2011, Zurich, Switzerland, July 4-8, 2011, Proceedings, Part I*, volume 6755 of *Lecture Notes in Computer Science*, pages 403–415. Springer, 2011.

[AH91] William Aiello and Johan Håstad. Statistical Zero-Knowledge Languages can be Recognized in Two Rounds. *J. Comput. Syst. Sci.*, 42(3):327–345, 1991.

[Ajt96] M. Ajtai. Generating Hard Instances of Lattice Problems (Extended Abstract). In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '96, page 99–108, New York, NY, USA, 1996. Association for Computing Machinery.

[Bab85] László Babai. Trading Group Theory for Randomness. In Robert Sedgewick, editor, *Proceedings of the 17th Annual ACM Symposium on Theory of Computing, May 6-8, 1985, Providence, Rhode Island, USA*, pages 421–429. ACM, 1985.

[Bar01] Boaz Barak. How to Go Beyond the Black-Box Simulation Barrier. In *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*, pages 106–115. IEEE Computer Society, 2001.

[BL02] Boaz Barak and Yehuda Lindell. Strict Polynomial-Time in Simulation and Extraction. STOC '02, page 484–493, New York, NY, USA, 2002. Association for Computing Machinery.

[Blu86] Manuel Blum. How to Prove a Theorem so that No One Else Can Claim It. 01 1986.

[BLV04] Boaz Barak, Yehuda Lindell, and Salil P. Vadhan. Lower Bounds for Non-Black-Box Zero Knowledge. *IACR Cryptol. ePrint Arch.*, page 226, 2004.

[BS23] Dan Boneh and Victor Shoup. *A Graduate Course in Applied Cryptography*. 2023.

[CSST11] Claude Crépeau, Louis Salvail, Jean-Raymond Simard, and Alain Tapp. Two Provers in Isolation. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and*

Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings, volume 7073 of *Lecture Notes in Computer Science*, pages 407–430, 2011.

[For87] Lance Fortnow. The Complexity of Perfect Zero-Knowledge (Extended Abstract). In Alfred V. Aho, editor, *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, pages 204–209. ACM, 1987.

[GK96a] Oded Goldreich and Ariel Kahan. How to Construct Constant-Round Zero-Knowledge Proof Systems for NP. *J. Cryptol.*, 9(3):167–190, 1996.

[GK96b] Oded Goldreich and Hugo Krawczyk. On the Composition of Zero-Knowledge Proof Systems. *SIAM J. Comput.*, 25(1):169–192, 1996.

[GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The Knowledge Complexity of Interactive Proof-Systems (Extended Abstract). In Robert Sedgewick, editor, *Proceedings of the 17th Annual ACM Symposium on Theory of Computing, May 6-8, 1985, Providence, Rhode Island, USA*, pages 291–304. ACM, 1985.

[GMW91] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs That Yield Nothing but Their Validity or All Languages in NP Have Zero-Knowledge Proof Systems. *J. ACM*, 38(3):690–728, jul 1991.

[Gol98] Oded Goldreich. *Modern Cryptography, Probabilistic Proofs and Pseudorandomness*, volume 17 of *Algorithms and Combinatorics*. Springer, 1998.

[Gol01] Oded Goldreich. *The Foundations of Cryptography - Volume 1: Basic Techniques*. Cambridge University Press, 2001.

[GS86] S Goldwasser and M Sipser. Private Coins versus Public Coins in Interactive Proof Systems. In *Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing*, STOC '86, page 59–68, New York, NY, USA, 1986. Association for Computing Machinery.

[HI19] Akinori Hosoyamada and Tetsu Iwata. 4-Round Luby-Rackoff Construction is a qPRP. In Steven D. Galbraith and Shiho Moriai, editors, *Advances in Cryptology - ASIACRYPT 2019 - 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019, Proceedings, Part I*, volume 11921 of *Lecture Notes in Computer Science*, pages 145–174. Springer, 2019.

[HM96] Shai Halevi and Silvio Micali. Practical and Provably-Secure Commitment Schemes from Collision-Free Hashing. In Neal Koblitz, editor, *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings*, volume 1109 of *Lecture Notes in Computer Science*, pages 201–215. Springer, 1996.

[IHM+19] Gembu Ito, Akinori Hosoyamada, Ryutaroh Matsumoto, Yu Sasaki, and Tetsu Iwata. Quantum Chosen-Ciphertext Attacks Against Feistel Ciphers. In Mitsuru Matsui, editor, *Topics in Cryptology - CT-RSA 2019 - The Cryptographers' Track at the RSA Conference 2019, San Francisco, CA, USA, March 4-8, 2019, Proceedings*, volume 11405 of *Lecture Notes in Computer Science*, pages 391–411. Springer, 2019.

[KM10] Hidenori Kuwakado and Masakatu Morii. Quantum distinguisher between the 3-round Feistel cipher and the random permutation. In *2010 IEEE International Symposium on Information Theory*, pages 2682–2685, 2010.

[LFKN92] Carsten Lund, Lance Fortnow, Howard Karloff, and Noam Nisan. Algebraic Methods for Interactive Proof Systems. *J. ACM*, 39(4):859–868, oct 1992.

[Lin13] Yehuda Lindell. A Note on Constant-Round Zero-Knowledge Proofs of Knowledge. *J. Cryptol.*, 26(4):638–654, 2013.

[Mic18] Daniele Micciancio. On the Hardness of Learning With Errors with Binary Secrets. *Theory of Computing*, 14(13):1–17, 2018.

[NC16] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information (10th Anniversary edition)*. Cambridge University Press, 2016.

[Reg09] Oded Regev. On Lattices, Learning with Errors, Random Linear Codes, and Cryptography. *J. ACM*, 56(6), sep 2009.

[Ros04] Alon Rosen. A Note on Constant-Round Zero-Knowledge Proofs for NP. In Moni Naor, editor, *Theory of Cryptography, First Theory of Cryptography Conference, TCC 2004, Cambridge, MA, USA, February 19-21, 2004, Proceedings*, volume 2951 of *Lecture Notes in Computer Science*, pages 191–202. Springer, 2004.

[Sch91] Claus-Peter Schnorr. Efficient Signature Generation by Smart Cards. *J. Cryptol.*, 4(3):161–174, 1991.

[SS17] Thomas Santoli and Christian Schaffner. Using Simon's algorithm to attack symmetric-key cryptographic primitives. *Quantum Inf. Comput.*, 17(1&2):65–78, 2017.

[Unr16] Dominique Unruh. Computationally Binding Quantum Commitments. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 497–527. Springer, 2016.