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-

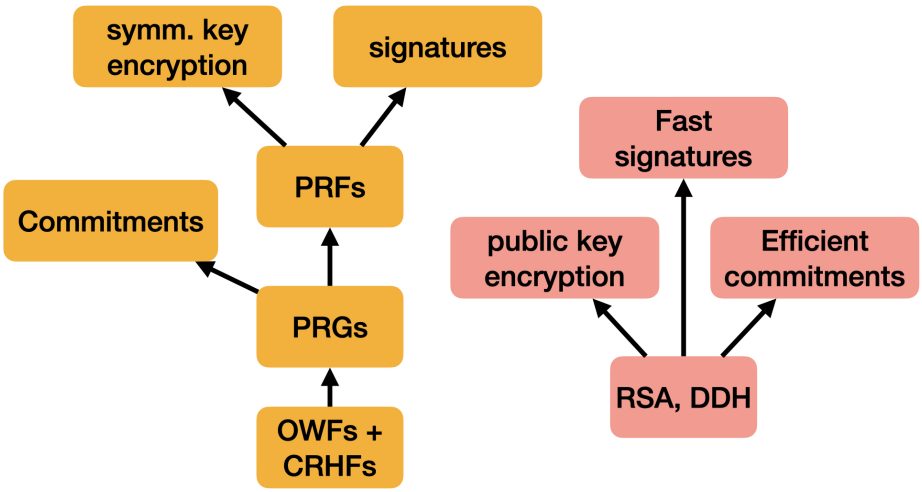


Figure 1: COL759: summarized in one figure. Some cryptographic primitives that can be built from OWFs and CRHFs. In practice, we have very efficient OWFs, PRGs, PRFs and CRHFs. Others such as public key encryption and key agreement rely on computational number-theoretic problems such as RSA or DDH.

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 4) and statistically hiding (no adversary can win the hiding game defined in Figure 5).*

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.

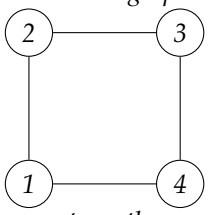
The main issue is that for certain graphs (such as the triangle and empty graphs), the set of graphs isomorphic to such graphs are identical to the graph. Such permutations are called *automorphisms* of the graph.

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.

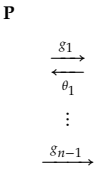
Common Input: 3CNF formula ϕ and integer $t \leq 2^n$. Equivalently, we can think of the common input as a prime $q > 2^n$, a number $t \leq 2^n$ and the canonical arithmetization g_ϕ, a polynomial over $\mathbb{Z}_q$.

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.

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

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 $H = G_{b_i}$, 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 perfect/statistical/computational zero knowledge if for every (possibly malicious) probabilistic polynomial time verifier $\mathbf{V}^*$, there exists a probabilistic polynomial time simulator $\mathbf{S}$ 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.

VK: Fix protocol numbering

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 5.11. □

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 5.11, 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 $\mathbf{w} = (v_1, \ldots, v_n)$

for $k = 1$ *to* n **do**
  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 $H = \pi(G)$, $c_{i,j} = \mathsf{Commit}(A_H[i,j]; r_{i,j})$.
  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`, **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 $w_i = \pi(v_i)$, $\mathsf{op} = \left(\left(w_i\right), \left(r_{w_i, w_{i+1}}\right)\right)$. 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.

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 5.11.

□

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

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). In fact, 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.

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 know that the simulator's transcript, when interacting with a honest verifier, will be accepted by the last-stage verifier. However, the simulator gets to feed randomness to the honest verifier too, and maybe the simulator feeds bad randomness to the honest verifier, in which case we cannot argue that the simulator's interaction with the honest verifier is identical to its interaction with $\mathcal{B}$.

Therefore, we will instead consider the simulator's interaction with a malicious verifier $\mathbf{V}^*$ 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-state verifier; (b) this verifier's interaction with the simulator is also accepted by the last-stage verifier; (c) this verifier's interaction with the simulator is indistinguishable from the simulator's interaction with $\mathcal{B}$.

Putting all these together, we get that if $x \in \mathsf{L}_{\text{yes}}$, then the algorithm $\mathcal{B}$ correctly guesses that $x \in \mathsf{L}_{\text{yes}}$.

Proof. For simplicity, let ℓ denote the length of each message exchanged (recall, we are assuming $\mathbf{V}_4$ is deterministic). Let $F : \mathcal{K} \times \{0,1\}^\ell \to \{0,1\}^\ell$ be a pseudorandom function (towards the end of this proof, we will discuss how to make this proof unconditional).

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

- On input x, $\mathcal{B}$ first chooses randomness $r_{\mathbf{S}}$ for the simulator, and t uniformly random strings $\{r_i\}_{i\in[t]}$. These t strings will be used to respond to the simulator's queries. The i^{th} unique query by $\mathbf{S}$ gets r_i from the verifier.

  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 analyse the two cases, depending on whether $x \in \mathit{Lyes}$ 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 three 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 first and second transcript are identical (using properties of t-wise independent hash). Using the black-box zero-knowledge property, the transcript output by $\mathbf{S}$ is also accepted with probability $1 - \text{negl}(|x|)$. □

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 behaviour 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 $\mathsf{L} \in \mathsf{BPP}$.

Concluding remarks: The above proof used a pseudorandom function. However, we only need a function that is t-wise independent, since the simulator is fixed before the verifier is chosen. The t-query simulator should work for any malicious verifier. As a result, once the simulator is fixed, we can choose an appropriate t-wise independent function instead of the PRF. □

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 *Proofs of Knowledge*

Lecture 8: January 27th, 2023

5.9 *Constant Round Zero-Knowledge Protocols*

Lecture 9: January 31st, 2023

5.10 *Other Results*

- Goldreich and Krawczyk [GK96b] showed that black-box, three-round zero-knowledge proofs (with negligible soundness) for L implies $\mathsf{L} \in \mathsf{BPP}$. This result was extended by Katz [Kat08], who showed that black-box, four-round ZKPs (with negligible soundness) for L implies $\mathsf{L} \in \mathsf{coAM}$. Finally, Goldreich and Kahan [GK96a] showed a five-round, black-box zero-knowledge protocol (with negligible soundness) for any language in NP. This completes the picture as far as black-box simulation is concerned.

In class, we saw that black-box, public-coin zero-knowledge proofs, with constant number of rounds, cannot have negligible soundness (unless $\mathsf{L} \in \mathsf{BPP}$.

5.11 *Missing Proofs*

References

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, USA, 1st edition, 2009.

[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.

[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.

[Blu86] Manuel Blum. How to Prove a Theorem so that No One Else Can Claim It. 01 1986.

[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.

[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.

[Kat08] Jonathan Katz. Which Languages Have 4-Round Zero-Knowledge Proofs? In Ran Canetti, editor, *Theory of Cryptography, Fifth Theory of Cryptography Conference, TCC 2008, New York, USA, March 19-21, 2008*, volume 4948 of *Lecture Notes in Computer Science*, pages 73–88. Springer, 2008.

[Mic18] Daniele Micciancio. On the Hardness of Learning With Errors with Binary Secrets. *Theory of Computing*, 14(13):1–17, 2018.

[Reg09] Oded Regev. On Lattices, Learning with Errors, Random Linear Codes, and Cryptography. *J. ACM*, 56(6), sep 2009.

Appendix: Cryptographic Primitives

Definition 1 (Non-interactive commitment scheme with setup)**.** *A non-interactive commitment scheme with setup consists of two algorithms:* Setup *and* Commit *with the following syntax:*

- $\mathsf{Setup}(1^n)$*: takes as input the security parameter, and outputs a public key* pk*.*
- $\mathsf{Commit}(\mathsf{pk}, \mathsf{msg}; r)$ *: takes as input the public key* pk*, the message* msg *to be committed, randomness* r*, and outputs a commitment* com*.*

A non-interactive commitment scheme with setup must satisfy the following two security properties:

- ***Binding property****: A commitment scheme satisfies the binding property if for any prob. poly. time adversary* $\mathcal{A}$*, there exists a negligible function* $\mu(\cdot)$ *such that for all* n*,* $\Pr[\mathcal{A}$ *wins the binding security game*$] \leq \mu(n)$ *where the binding security game is defined below:*

Binding-Game
– Challenger chooses $\mathsf{pk} \leftarrow \mathsf{Setup}(1^n)$ and sends pk to $\mathcal{A}$.
– Adversary sends a commitment com, together with two (message, opening) pairs (msg_0, r_0) and (msg_1, r_1). The adversary wins if $\mathsf{Commit}(\mathsf{pk}, \mathsf{msg}_0; r_0) = \mathsf{Commit}(\mathsf{pk}, \mathsf{msg}_1; r_1) = \mathsf{com}$.

Figure 4: The Binding Security Game

- ***Hiding property****: A commitment scheme satisfies the hiding property if for any prob. poly. time adversary* $\mathcal{A}$*, there exists a neglibile function* $\mu(\cdot)$ *such that for all* n*,* $\Pr[\mathcal{A}$ *wins the hiding security game*$] \leq 1/2 + \mu(n)$ *where the binding security game is defined below:*

Hiding-Game
– Challenger chooses $\mathsf{pk} \leftarrow \mathsf{Setup}(1^n)$ and sends it to the adversary.
– The adversary sends two messages $\mathsf{msg}_0, \mathsf{msg}_1$.
– Challenger chooses $b \leftarrow \{0,1\}$, computes $\mathsf{com} \leftarrow \mathsf{Commit}(\mathsf{pk}, \mathsf{msg}_b)$ and sends com to $\mathcal{A}$.
– Adversary sends guess b' and wins if $b = b'$.

Figure 5: The Hiding Security Game

$\Diamond$