Foundations of Cryptography, Fall 2025
Problem Set 4
Due Monday, November 24

Total Number of Points: 100

Collaboration Policy: Collaboration is permitted and encouraged in small groups of at most three students. You are free to collaborate in discussing answers, but you must write up solutions on your own, and must specify in your submission the names of any collaborators. Do not copy any text from your collaborators; the writeup must be entirely your work. Do not write down solutions on a board and copy it verbatim; again, the writeup must be entirely your own words and your own work and should demonstrate clear understanding of the solution. **Solutions should be typeset in LaTeX.** You may make use of material referenced on the course website, provided that you clearly acknowledge all sources/tools used. Of course, scavenging for solutions from prior years is forbidden.

On the Use of LLMs: You may use AI however you wish to deepen your understanding of the lecture material. Upload the notes, talk to your AI about them, ask for more explanation or examples; it's all fine. **You may not use AI/LLMs in any way to work on your homework.** You may not upload assignments, ask for hints, ask how certain concepts from the lectures might be applied to specific homework problems, or upload your assignments to check for correctness or clarity or anything else. You may not include any AI-generated content whatsoever in your homework submissions. If it becomes clear that you have used an AI tool when working on your homework (either directly by making edits or to ask for hints/solutions), we may mark your grade down to reflect that.

1 Perfectly Binding Hash Functions: An Oxymoron? (20 points)

A hash function $\mathsf{H} : \{0,1\}^n \to \{0,1\}^m$ compresses its n-bit input into an m-bit output for some $m < n$. A perfectly binding function $\mathsf{H} : \{0,1\}^n \to \{0,1\}^m$ is injective, i.e. given $\mathsf{H}(x)$, x is uniquely determined. Could one have a perfectly binding hash function? Clearly not! Nevertheless, in this problem, we will construct hash functions that achieve a variant of perfect binding.

Definition 1. *A* bit-wise perfectly binding *hash function is a pair of polynomial-time algorithms* $(\mathsf{Gen}, \mathsf{H})$ *where*

- $\mathsf{hk} \leftarrow \mathsf{Gen}(1^\lambda, i)$ *takes as input an integer* $i \in [n(\lambda)]$ *and outputs a public hashing key* hk*;*
- $y = \mathsf{H}(\mathsf{hk}, x)$ *is a deterministic algorithm that takes an* $n(\lambda)$*-bit input* x *and produces an* $m(\lambda)$*-bit output, where* $m(\lambda) < n(\lambda)$ *for all sufficiently large* λ*.*

We additionally require the following two properties:

1. ***Index Hiding.*** *For every pair of indices* $i_0, i_1 \in [n(\lambda)]$*, the hash keys* $\mathsf{hk}_0 \leftarrow \mathsf{Gen}(1^\lambda, i_0)$ *and* $\mathsf{hk}_1 \leftarrow \mathsf{Gen}(1^\lambda, i_1)$ *are computationally indistinguishable.*
2. ***Bit-wise Perfectly Binding.*** *For any* $i \in [n(\lambda)]$*,* $\mathsf{hk} \leftarrow \mathsf{Gen}(1^\lambda, i)$*, and* $x, x' \in \{0,1\}^{n(\lambda)}$*, if* $\mathsf{H}(\mathsf{hk}, x) = \mathsf{H}(\mathsf{hk}, x')$*, then* $x[i] = x'[i]$*.*

1. (10 points) Recall the Quadratic Residuosity Assumption from the third problem set:

   Assumption 1 (Quadratic Residuosity Assumption for Blum Integers)**.** *For any poly-size algorithm* $\mathcal{A}$*, its advantage in solving the Quadratic Residuosity Problem is negligible in the security parameter* λ*. That is, for random* $p \neq q$ *of* λ *bits each where* $p, q \equiv 3 \bmod 4$*, and random* $x \in \mathbb{Z}_N^*$ *with* $\left(\frac{x}{N}\right) = 1$*,*

   $$\Pr[\mathcal{A}(N, x) = 1 \text{ if } x \text{ is a residue, and } 0 \text{ otherwise}] \leq \frac{1}{2} + \mathrm{negl}(\lambda).$$

   Construct a bit-wise perfectly binding hash function, assuming the hardness of the quadratic residuosity problem. Prove that your construction is correct.

2. (10 points) Construct a bit-wise perfect binding hash function, assuming the existence of a fully homomorphic encryption (FHE) scheme $(\mathsf{Gen}_{\mathsf{FHE}}, \mathsf{Enc}_{\mathsf{FHE}}, \mathsf{Dec}_{\mathsf{FHE}}, \mathsf{Eval}_{\mathsf{FHE}})$ with perfect correctness. Prove that your construction is correct.

   You may also assume that $(\mathsf{Gen}_{\mathsf{FHE}}, \mathsf{Enc}_{\mathsf{FHE}}, \mathsf{Dec}_{\mathsf{FHE}}, \mathsf{Eval}_{\mathsf{FHE}})$ is *truly compact*. More precisely, this means there exists some polynomial $p(\lambda)$ such that the following statement holds: for all polynomial-size circuits C which output a single bit, all inputs $\mu_1, \ldots, \mu_n \in \{0,1\}$, and all keys $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}_{\mathsf{FHE}}(1^\lambda)$, $|\mathsf{Eval}_{\mathsf{FHE}}(\mathsf{pk}, C, \{\mathsf{Enc}(\mathsf{pk}, \mu_i)\}_{i=1}^n)| \leq p(\lambda)$.

2 Statistically Hiding Commitments (15 points)

In lecture, we have seen a construction of computationally hiding and statistically binding commitments, which can be used to construct computational zero-knowledge proofs. In this problem, we consider how to construct statistically hiding and computationally binding commitments, which can be used to construct statistical zero-knowledge arguments.

We will work with the discrete logarithm assumption over a family of groups $\{G_\lambda\}_{\lambda\in\mathbb{N}}$, where each G_λ is a group of prime order q_λ:

Assumption 2 (DLOG for $\{G_\lambda\}_{\lambda\in\mathbb{N}}$)**.** *Let g_λ be a uniformly random generator of G_λ.*[1] *For every poly-size adversary $\mathcal{A}$, there exists a negligible function μ such that for all $\lambda \in \mathbb{N}$,*

$$\Pr_{x\leftarrow\mathbb{Z}_q^*, g}[A(1^\lambda, g, g^x) \equiv x \bmod q] \leq \mu(\lambda),$$

where $q = q_\lambda$ and $g = g_\lambda$.

DLOG-Based Commitment Scheme

- $\mathsf{Gen}(1^\lambda)$: Sample uniformly random generators $g = g_\lambda$ and $h = h_\lambda$ from G_λ. Output $\mathsf{pp} = (g, h)$.
- $\mathsf{Com}(\mathsf{pp}, m)$: Parse $\mathsf{pp} = (g, h)$. Sample random $r \leftarrow \mathbb{Z}_q$ and output $\mathsf{com} = g^m \cdot h^r$. The opening is r.
- $\mathsf{Ver}(\mathsf{pp}, \mathsf{com}, m, r)$: Parse $\mathsf{pp} = (g, h)$. Output 1 iff $g^m \cdot h^r = \mathsf{com}$.

It is not hard to see that the commitment scheme has perfect correctness: that is, if the committer and receiver behave as prescribed, the receiver will always accept the committer's opening. We now take a look at two other properties of this commitment scheme.

1. (5 points) Show that the commitment scheme is *perfectly hiding*: for all $\lambda \in \mathbb{N}$ and any two messages $m_0, m_1 \in \mathbb{Z}_{q_\lambda}$,
$$(\mathsf{pp}, \mathsf{Com}(\mathsf{pp}, m_0)) \equiv (\mathsf{pp}, \mathsf{Com}(\mathsf{pp}, m_1)),$$
where $\mathsf{pp} \leftarrow \mathsf{Gen}(1^\lambda)$.

2. (10 points) Assuming the discrete logarithm is hard in $\{G_\lambda\}_{\lambda\in\mathbb{N}}$, show that the commitment scheme is *computationally binding*: for all poly-size adversaries $\mathcal{A}$, there exists a negligible function μ such that for all $\lambda \in \mathbb{N}$,
$$\Pr_{\mathsf{pp}\leftarrow\mathsf{Gen}(1^\lambda)}\left[\begin{array}{l} m_0 \not\equiv m_1 \bmod q_\lambda \ \wedge \\ \mathsf{Ver}(\mathsf{pp}, \mathsf{com}, m_0, r_0) = 1 \ \wedge \ : (\mathsf{com}, m_0, r_0, m_1, r_1) \leftarrow \mathcal{A}(1^\lambda, \mathsf{pp}) \\ \mathsf{Ver}(\mathsf{pp}, \mathsf{com}, m_1, r_1) = 1 \end{array}\right] \leq \mu(\lambda).$$

[1] Note that since G_λ is of prime order (and hence cyclic), a uniformly random generator is simply a uniformly random (non-identity) element.

3 Does Zero Knowledge Compose in Parallel? (15 points)

We have seen in class that assuming one-way functions exist, there is a three message zero knowledge proof system for an NP-complete problem (graph 3-colorability) with soundness $\frac{1}{\text{poly}(n)}$, and that the soundness can be amplified by repeating this proof system many times in sequence to obtain a zero knowledge proof system with negligible soundness. Unfortunately, this results in a very long protocol. Wouldn't it be nice if we could instead repeat the three message protocol many times *in parallel*? Then maybe we could get negligible soundness with only three messages!

Unfortunately, the problem is that the resulting proof system (which is actually sound!) may not be zero knowledge! In this problem, we will see how repeating an interactive proof can ruin zero knowledge. Consider the following zero knowledge proof Π for the discrete logarithm:

- Public input: a string $h = g^x$ (along with a group $\mathbb{G}$ and generator g); prover input: the discrete logarithm x.
- First message: P chooses a uniformly random r and sends $R = g^r$ to V.
- Second message: V chooses a uniformly random bit b.
- Third message: P sends $z = r + bx$ to V. V accepts if $g^z = R \cdot h^b$.

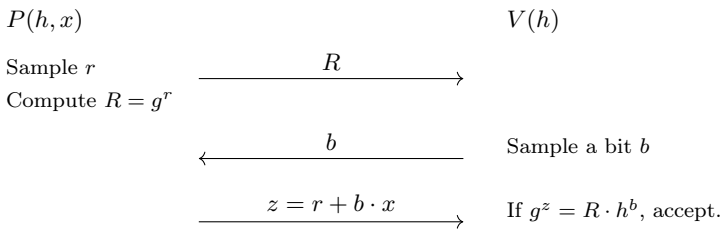

Figure 2: The Zero Knowledge Proof Π for Discrete Log.

The Counterexample We will now consider the following strange modification $\tilde{\Pi}$ of Π (see Figure 3):

- Public input: a string $h = g^x$ (along with a group $\mathbb{G}$ and generator g); prover input: the discrete logarithm x.
- First message: V makes a uniformly random guess x^* and checks if $g^{x^*} = h$. If this happens, V sets a variable mode to 1, and otherwise V sets mode to 0. V then sends mode to P.
- If $\mathsf{mode} = 0$, P and V proceed to execute Π λ times in sequence. If $\mathsf{mode} = 1$, V instead proves to P (λ times) that V knows x, which is equal to x^*! That is, V and P execute Π (λ times) with their roles reversed. If V convinces P that V knows x, P then sends x to V in the clear.

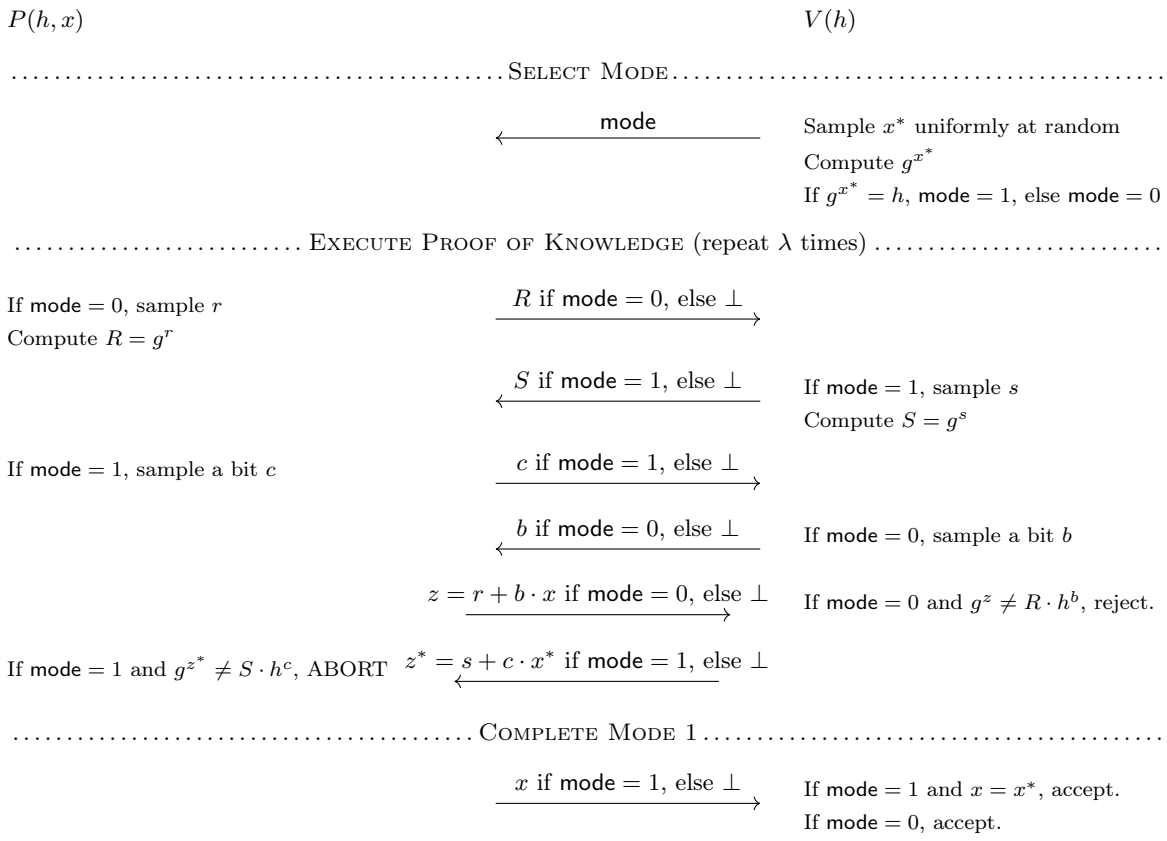

Figure 3: The Protocol $\tilde{\Pi}$.

One can show that $\tilde{\Pi}$ is still a zero knowledge proof for discrete logarithm (we will not ask you to do this). **Instead, prove that if the discrete logarithm problem is hard, then $\tilde{\Pi}$ executed two times in parallel is *not* zero knowledge! In particular, show how a cheating verifier can learn the discrete logarithm x with certainty.**

4 But Really, Does Zero Knowledge Compose in Parallel? (25 points)

We saw in the previous problem that zero knowledge is not always preserved when a protocol is executed many times in parallel. However, the example seems contrived, and we would like to say something about parallel composition anyway! To do so, we will introduce a new (weaker) notion of security against malicious verifiers, which we call witness indistinguishability (WI).

Definition 2. *A proof system Π for a language $L \in \mathsf{NP}$ is* witness indistinguishable *if for every (efficient) malicious verifier V^*, every instance $x \in L$, every pair of witnesses w_1, w_2 for x, and every auxiliary input z,*

$$\text{view}_{V^*}\langle P(w_1), V^*(x,z)\rangle \approx_c \text{view}_{V^*}\langle P(w_2), V^*(x,z)\rangle.$$

In other words, V^* cannot tell which witness an honest prover P is using in an execution of the protocol, even given arbitrary auxiliary information about (x, w_1, w_2) (possibly including both w_1 and w_2).

As a motivating example, consider the following scenario. Suppose that there is some organization whose membership is a known list of public keys $\mathsf{pk}_1, \ldots, \mathsf{pk}_n$. Each member i of the organization has secret key sk_i. Now, suppose that a member of the organization wants to prove to an outside party (say, a journalist) that they belong to the organization *without revealing* which member they are. Witness indistinguishable proofs allow you to do this: you are trying to prove a statement with n possible witnesses (the secret keys), and you do not want to reveal which witness you have.

1. (10 points) **Zero Knowledge Proofs are Witness Indistinguishable**. Prove that every (auxiliary input) zero knowledge proof system Π (for a language $L \in \mathsf{NP}$) is also witness indistinguishable.

2. (15 points) **Composition of WI Proofs**. From part 1, we see that the three-message protocol for 3-colorability from lecture is witness indistinguishable. However, it still has poor soundness. Prove that (three-message) WI proof systems actually compose under parallel composition![2] That is, if Π is a three-message WI proof system for some language L with soundness μ and completeness 1, prove that Π^t (a t-wise parallel repetition of Π) is a WI proof system for L with soundness μ^t and completeness 1.

   Remark 1. *You may assume that the honest prover is efficient (i.e. runs in polynomial time) given any NP witness. This is similar to the honest prover efficiency requirement for zero-knowledge proofs for NP.*

[2] In fact, composition holds for WI proof systems with an arbitrary polynomial number of rounds, but we will restrict to three-message proofs for simplicity.

5 The Power of Homomorphism (25 points)

In this problem, we will see that homomorphism is quite a powerful tool—often, homomorphic properties already present enough structure to imply public-key encryption!

5.1 PKE from re-randomizable encryption (10 points)

Recall the definition of re-randomizable encryption, which is a property that most homomorphic encryption schemes tend to have:

Definition 3 (Re-randomizable secret-key encryption)**.** *A randomized secret-key encryption scheme* $\mathsf{SKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *over message space* $\mathcal{M} = \{0,1\}$ *is* re-randomizable *if there exists a PPT algorithm* ReRand *such that for every key* $k \leftarrow \mathsf{Gen}(1^\lambda)$*, message* $m \in \mathcal{M}$*, and encryption* $c \leftarrow \mathsf{Enc}(k, m)$*,*

$$\mathsf{ReRand}(c) \overset{c}{\approx} \mathsf{Enc}(k, m),$$ [3]

and $\mathsf{Dec}(k, \mathsf{ReRand}(c)) = \mathsf{Dec}(k, c)$ *for all ciphertexts* c*.*

Suppose there exists a (CPA-secure) re-randomizable secret key-encryption scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{ReRand})$. Show that there exists a (CPA-secure) public-key encryption scheme $(\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}')$.

5.2 PKE from additive homomorphism (15 points)

Let $\mathsf{SKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Add})$ be a *secret-key* encryption scheme for single-bit messages that satisfies the standard compactness and (CPA) security guarantees, and supports homomorphic addition:

- **Additive homomorphism mod 2.** Add is a deterministic algorithm such that for any ciphertexts $\{c_i\}_{i \in [n]}$,
$$\mathsf{Dec}(k, \mathsf{Add}(c_1, \ldots, c_n)) = \oplus_{i=1}^{n} \mathsf{Dec}(k, c_i).$$

Now, define a public-key encryption scheme $\mathsf{PKE} = (\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}')$ as follows:

- $\mathsf{Gen}'(1^\lambda)$: Let $\ell = \ell(\lambda)$ be a polynomial to be specified later. Sample $k \leftarrow \mathsf{Gen}(1^\lambda)$. For each $i \in [\ell]$, sample fresh
$$X_i \leftarrow \mathsf{Enc}(k, 0), \qquad Y_i \leftarrow \mathsf{Enc}(k, 1).$$
Output $\mathsf{pk} = \big((X_1, \ldots, X_\ell), (Y_1, \ldots, Y_\ell)\big)$ and $\mathsf{sk} = k$.
- $\mathsf{Enc}'(\mathsf{pk}, m)$: Parse $\mathsf{pk} = \big((X_1, \ldots, X_\ell), (Y_1, \ldots, Y_\ell)\big)$. Sample a uniformly random subset $S \subseteq [\ell]$ conditioned on $|S| \equiv m \bmod 2$. Form the ℓ-tuple Z by setting $Z_i := Y_i$ if $i \in S$ and $Z_i := X_i$ if $i \notin S$. Output
$$c := \mathsf{Add}(Z_1, \ldots, Z_\ell).$$
- $\mathsf{Dec}'(\mathsf{sk}, c)$: Parse $\mathsf{sk} = k$ and output $\mathsf{Dec}(k, c)$.

Suppose the ciphertexts $c \leftarrow \mathsf{Enc}(k, m), k \leftarrow \mathsf{Gen}(1^\lambda)$ satisfy $|c| \le n(\lambda)$ for some polynomial n. Specify the polynomial ℓ such that PKE is CPA-secure, and prove correctness and security. You may take the following information-theoretic statement for granted.

Theorem 1. *Let* $X_1, \ldots, X_\ell$ *and* $Y_1, \ldots, Y_\ell$ *be i.i.d. over a finite set. Sample a uniformly random bit* $\sigma \in \{0,1\}$ *and let* $S \subseteq [\ell]$ *be uniformly random subject to* $|S| \bmod 2 \equiv \sigma$*. Define* $Z_i := X_i$ *for* $i \notin S$ *and* $Z_i := Y_i$ *for* $i \in S$*. Let* $X := (X_1, \ldots, X_\ell)$*,* $Y := (Y_1, \ldots, Y_\ell)$*, and* $Z := (Z_1, \ldots, Z_\ell)$*. For any (possibly unbounded) adversary that sees* X*,* Y*, and any* n*-bit (possibly randomized) function* $g(Z)$ *of* Z*, its advantage in predicting* $\sigma \equiv |S| \bmod 2$ *is at most* $2^{-\ell/5+n+1}$*.* [4]

[3] The ciphertext on the right-hand side is a *fresh* encryption of m with the key k which is not necessarily equal to c.

[4] For this theorem to make sense, $g(Z)$ cannot depend on X, Y, and Z together (or else it can simply leak σ!). Thus, the function g is restricted to depend only on the bits in Z (and not X, Y, and Z simultaneously). Of course, g can depend on the *distribution* of X and Y.