CSC 2419: Lattice-based Cryptography **Fall 2025**

Lecture 6: Non-Interactive Zero-Knowledge

Instructor: Akshayaram Srinivasan *Scribe: William He*

Date: 2025-10-20

6.1 Recap

In the previous lecture, we saw Rate-1 FHE, which allows compressing the size of ciphertexts.

Specifically, given an l-bit message $(m_1, \ldots, m_l)$, Rate-1 FHE compresses it into a header of size $\mathsf{poly}(n)$ and a payload of size l. To retrieve the message, we XOR the payload with an l-bit string that can be derived from the header and the secret key.

In this lecture, we will further investigate how to use Rate-1 FHE to construct **Non-Interactive Zero-Knowledge Proofs (NIZKs)**. Before doing so, let's talk about interactive proofs.

6.2 Interactive Proofs

6.2.1 Definition of Interactive Proofs

An interactive proof for a problem involves a computationally limited entity (the verifier) and a computationally powerful entity (the prover). Only the prover can efficiently determine the truthfulness of the statement. The prover wants to convince the verifier of the truth of the statement.

Setting: We will consider a problem involving a language $\mathcal{L}$. Given some statement x, the prover wants to prove to the verifier that $x \in \mathcal{L}$. The interaction works as follows:

1. At the start, the prover and the verifier both see the string x. Either party may send the first message.
2. In subsequent rounds, the prover and the verifier take turns sending messages to each other.
3. In the end, the verifier outputs accept/reject to indicate whether it is convinced that $x \in L$.

A **malicious prover** may try to convince the verifier that $x \in L$ even when $x \notin L$, unlike an **honest prover**.

We now formally define a language that has an interactive proof.

Definition 6.1 *A language $\mathcal{L}$ has an interactive proof if there is a pair of machines (P, V) with prover P (possibly computationally unbounded) and PPT verifier V that satisfies the following two properties:*

- ***Completeness:*** *P is an honest prover such that $\forall x \in L$*

$$\Pr[V \text{ accepts}] = 1$$

- ***Soundness:*** $\forall\ x \notin L, \forall$ *Prover* P^* *(including malicious provers)*

$$\Pr[V \text{ accepts}] \leq \epsilon(n)$$

If there is some (P, V) *that satisfies the above requirements, we call* (P, V) *an* ***interactive proof system*** *for* $\mathcal{L}$*.*

Example 6.2 *Every language* $\mathcal{L} \in$ NP *has an* ***interactive proof****.*

Proof: Consider a language $\mathcal{L} \in$ NP. Denote by $\mathcal{R}(x, \cdot)$ the PPT machine that verifies whether $x \in \mathcal{L}$ given a certificate/witness. By definition,

$$\mathcal{L} = \{x : \exists w \in \{0,1\}^{poly(|x|)} \text{ s.t. } \mathcal{R}(x, w) = 1\}$$

Here is a way of constructing the **interactive proof**

Given input x:

1. P computes and sends a witness w to V. If no witness is found, it can send an arbitrary message.
2. V outputs $\mathcal{R}(x, w)$.

We can easily show **completeness** and **soundness**

- **Completeness:** If $x \in L$, there exists a witness w such that $\mathcal{R}(x, w) = 1$. The honest prover P would send that w, and the verifier would always accept.
- **Soundness:** If $x \notin L$, there is no string w makes $\mathcal{R}(x, w) = 1$. Since there is no witness, no matter what prover send, the verifier would always reject.

■

Remark 6.3 *In the above case, where* $\Pr[V \text{ accepts}] = 0$ *for all* $x \notin L$ *(i.e., zero soundness error), we call this* ***perfect soundness****.*

The above proof is easy for the verifier to check given the witness w. However, very often w is private information that the prover does not want to reveal. We will extend this example to ensure that the verifier can still be convinced while learning nothing about the witness w. This type of interactive proof is called a **Zero-Knowledge Proof**, introduced by Goldwasser, Micali, and Rackoff [GMR85].

6.3 Zero-Knowledge Proofs

We first formally define **Zero-Knowledge Proofs**. The goal of the definition is to ensure that the verifier learns only that $x \in L$ and nothing else. Here are two attempts, but neither yields the desired guarantee.

- **Witness hiding:** The verifier cannot output the witness at the end of the interaction.

  However, this does not prevent the verifier from learning partial information about the witness. This is weaker than Zero-Knowledge.

- **Witness indistinguishability:** When a **malicious verifier** (see Remark 6.4) outputs two witnesses w_0, w_1, the prover uses one of them sampled at random. The verifier cannot guess which witness the prover used when interacting with the prover.

  However, for languages with unique witnesses, witness indistinguishability holds trivially, while the verifier still learns the entire witness.

Now consider the verifier's view in the real world when interacting with the prover. Given the statement x as input, the view is defined as $\text{View}_{V,x} = (x, r, \text{msg}_{p \to v})$, where r is the verifier's internal randomness, and $\text{msg}_{p \to v}$ are all the messages that the prover sends to the verifier.

Remark 6.4 *For an **honest verifier**, the messages it sends are deterministic functions of* $\text{View}_{V,x}$*. By contrast, a **malicious verifier** may follow any arbitrary strategy to send messages.*

Note: *We will only consider honest verifiers for Zero-Knowledge Proofs.*

Also consider an ideal world, where there is a PPT simulator Sim that takes only the statement x as input. The verifier interacts with this simulator in the ideal world. If the verifier's view in the real world is computationally indistinguishable from its view in the ideal world, i.e.,

$$\text{View}_{V,x} \approx_c \text{Sim}_V(x)$$

then the verifier learns no information about the witness from the interaction of the simulator. This yields the formal definition.

Definition 6.5 (Honest Verifier (Computational) Zero-Knowledge) *An Interactive Proof* (P, V) *for language* $L \in \text{NP}$ *is called a **Zero-Knowledge Proof** if* $\exists$ *PPT* Sim *such that* $\forall x \in L, \text{View}_{V,x} \approx_c \text{Sim}_V(x)$

Remark 6.6

- *For **Honest-Verifier Statistical Zero-Knowledge**, the above definition must hold with* $\text{View}_{V,x} \approx_s \text{Sim}_V(x)$*.*
- *For **Honest-Verifier Perfect Zero-Knowledge**, the above definition must hold with* $\text{View}_{V,x}$ *and* $\text{Sim}_V(x)$ *identical in distribution.*

We will show that the Hamiltonian Cycle problem, which determines whether a graph has a Hamiltonian cycle, has a Zero-Knowledge Proof.

Recall that this problem is NP-complete. Moreover, a graph has a Hamiltonian cycle iff there is a cycle that visits all the vertices exactly once. We will construct the Zero-Knowledge Proof by first introducing the commitment scheme.

6.3.1 Commitment Scheme

Idea: A **commitment scheme** is a digital analogue of a locked box. A message is put inside the box. The box cannot be opened unless the committer provides the key.

Setting: We assume there is a committer C and a receiver R. The commitment scheme works as follows:

1. Both C and R have access to a (public) **commitment key** ck.
2. C sends R the **commitment** to a message m, denoted $\text{Com}(m) = c$.

3. C can provide an **opening** op to $\mathrm{Com}(m)$, which R can use to retrieve m.

We also require the commitment scheme to follow two properties:

Hiding: $\forall m, m', \{\mathrm{ck}, \mathrm{Com}(m)\} \approx_c \{\mathrm{ck}, \mathrm{Com}(m')\}$.

Statistically Binding: For any string $c, \nexists\, \mathrm{op}_1, \mathrm{op}_2$ such that, R retrieves m, m' by opening c using $\mathrm{op}_1, \mathrm{op}_2$ respectively, where $m \neq m'$.

Example 6.7 (Commitment scheme for single-bit messages from LWE) *Suppose the committer C wants to send the receiver R the message $b \in \{0,1\}$. We have:*

- $\mathrm{ck} = (A, s^\top A + e^\top)$.
- $\mathrm{Com}(b)$: *Sample* $r \leftarrow \{0,1\}^m$, *output* $(Ar, (s^\top A + e^\top)r + b \cdot \frac{q}{2})$.
- op : *Output* r.

Now we check this is a valid commitment scheme.

Valid opening: *Given $r \in \{0,1\}^m$, the verifier R checks that the first component of* $\mathrm{Com}(b)$ *equals Ar. Then, given* ck, *it computes $(s^\top A + e^\top)r + b \cdot \frac{q}{2} - (s^\top A + e^\top)r = b \cdot \frac{q}{2}$, which retrieves b.*

Hiding: *(As in proving semantic security for Regev's encryption.) By the* ***hardness of LWE****, we have*

$$(A, s^\top A + e^\top, Ar, (s^\top A + e^\top)r + b \cdot \frac{q}{2}) \approx_c (A, u^\top, Ar, u^\top r + b \cdot \frac{q}{2}),$$

for a uniformly random vector u. Then, by the ***Leftover Hash Lemma****, both Ar and $u^\top r$ are statistically close to uniform. Hence*

$$(A, u^\top, Ar, u^\top r + b \cdot \frac{q}{2}) \approx_s (A, u^\top, v_1, v_2 + b \cdot \frac{q}{2}),$$

where v_1, v_2 are uniformly random. This completely hides b, since adding a fixed offset to a uniform value preserves uniformity.

Binding: *Suppose there exist r_0 and r_1 that open the commitment to 0 and 1, respectively, and*

$$(Ar_0, (s^\top A + e^\top)r_0) = \left(Ar_1, (s^\top A + e^\top)r_1 + \frac{q}{2}\right).$$

This implies

$$e^\top r_0 = e^\top r_1 + \frac{q}{2},$$

which is impossible since both $e^\top r_0$ and $e^\top r_1$ are short in norm.

Remark 6.8 *This commitment scheme can be extended to multi-bit LWE by generating commitments to each bit independently.*

Now, we construct a 3-message **Zero-Knowledge Proof** for the Hamiltonian cycle problem using the **commitment scheme**.

Consider the following protocol for a graph $G = (V, E)$:

1. Both the prover P and the verifier V have ck and G that is represented as an adjacency matrix.

2. P computes a witness w, i.e., a Hamiltonian cycle in G.
3. P chooses a random permutation π of the vertices V and computes $H = \pi(G)$, the permuted graph.
4. P sends $\text{Com}(H)$ (the adjacency matrix committed entrywise) and $\text{Com}(\pi)$.
5. V sends a random $b \in \{0, 1\}$.
6. If $b = 0$, then P sends $\text{op}_H, \text{op}_\pi$, the openings for H and π.
   (a) V checks whether $\pi(G) = H$ (by applying π to the adjacency matrix of G and checking whether it obtains the adjacency matrix of H).
7. If $b = 1$, then P sends $\pi(w)$, the opening to the Hamiltonian cycle in H, which opens the edges in the cycle to 1 while leaving the rest unopened.
   (a) V checks whether the opened edges form a Hamiltonian cycle.

Now we check this is indeed a **Zero-Knowledge Proof**.

Completeness: For every Hamiltonian graph G, the honest prover can always find a witness w representing a Hamiltonian cycle. Any permutation $\pi(G) = H$ is still Hamiltonian. Therefore, V always accepts.

Soundness: If G is not Hamiltonian, the prover can ensure either that H is a valid permutation of G or that H is Hamiltonian, but not both. Hence, the prover can cheat for at most one value of b, with probability $\frac{1}{2}$.

To drive the soundness error down to negligible, we can run the protocol n times in parallel for sufficiently large n. In this case, the prover chooses n random permutations of G, the verifier samples n random challenges in $\{0, 1\}$, and the prover answers each. This reduces the soundness error to negligible.

Zero-Knowledge (idea): By definition, we want $\text{Sim}_V(G)$ to simulate $G, b, \text{Com}(H), \text{Com}(\pi)$ and the openings.

Given G, the simulator does the following:

1. Samples $b \in \{0, 1\}$ uniformly at random.
2. If $b = 0$, Sim samples a random permutation π and computes $H = \pi(G)$. It commits to $\text{Com}(H), \text{Com}(\pi)$ and provides the corresponding openings to V.
3. If $b = 1$, Sim generates a random cycle graph H (as an adjacency matrix) and sets π to the identity permutation. It commits to $\text{Com}(H), \text{Com}(\pi)$ and provides openings only for the edges of the cycle.

Again, we run the protocol for the simulator in parallel to obtain negligible soundness error.

When $b = 0$, the distribution of the view in the ideal world from the simulator and the view in the real world are identical (the simulator chooses π the same way as the prover). Hence, the views are perfectly indistinguishable.

When $b = 1$, the verifier sees only the random cycle that is opened. By the hiding property of the commitment scheme, the unopened entries of the adjacency matrix of H and the permutation π can be replaced as generated by the simulator. Hence, $\text{Com}(H), \text{Com}(\pi)$ are computationally indistinguishable in this case.

Remark 6.9 *This 3-message **Zero-Knowledge Proof** is called "Commit, Challenge, Response". The challenge from the verifier must be sent after the prover has committed. Otherwise, the prover could change the commitment.*

This also leads to the following result.

Corollary 6.10 *Every language in* NP *has an honest verifier* ***Zero-Knowledge Proof****.*

Proof: On any input x, both the prover P and the verifier V run the polynomial-time reduction from the NP language to Hamiltonian Cycle to obtain a graph G. Then P and V proceed with the Zero-Knowledge Proof on the graph G.

6.4 Non-interactive Zero-Knowledge Proofs

Often, interaction is costly and we would like to avoid it. We now consider **Non-Interactive Zero-Knowledge Proofs**, which work as follows.

Setting: We have a prover P and a verifier V:

1. The prover and the verifier have a commitment key ck and a common reference string crs.
2. P uses $(\text{ck}, \text{crs}, x, w)$ to generate a proof π for V.
3. V accepts/rejects based on $(x, \text{ck}, \text{crs}, \pi)$.

Goal: Instead of the verifier generating challenges, we want the prover to compute the challenges on its own.

We use the idea first proposed by Fiat and Shamir [FS86] to resolve this.

Idea:

1. The prover first generates the commitment c as the first-round message.
2. It uses a hash function $\text{Hash}(c)$ (modeled as a random oracle that outputs a random challenge for each commitment) to generate challenges $b_1, \ldots, b_n$.
3. The prover provides the openings based on these challenges.

By doing this, the prover emulates the honest verifier, which would send random challenges. We now construct such a hash function.

Recall that if $x \notin \mathcal{L}$, after the prover sends the commitment, there is exactly one value of b for which the prover can fool the verifier. If the prover sends n commitments to the verifier, there is exactly one string $b_1^*, \ldots, b_n^*$ such that, under this string, the prover can fool the verifier for each challenge. Hence, we want the hash function to avoid producing $b_1^*, \ldots, b_n^*$. This type of function is defined in [CGH04] as follows.

Definition 6.11 (Correlation-Intractable Hash Functions) *We say* Hash *is a* ***correlation-intractable hash function*** *if for any unbounded adversary $\mathcal{A}$,*

$$\Pr[\text{Hash}(\text{hk}, c) = (b_1^*, \ldots, b_n^*) : \text{hk} \leftarrow \text{Keygen}(1^n), \mathcal{A}(\text{hk}) = c] \leq \epsilon(n)$$

In our case, $\mathcal{A}$ is the unbounded prover P, and c is the commitment produced in the first round.

We now construct such a hash function using Rate-1 FHE. This construction is inspired by the construction of CI hash function from [BKM20].

Suppose a trusted third party generates ck using the LWE secret s and sets $\text{hk} = (\text{FHEENC}(s), (r_1, \ldots, r_n))$, where $(r_1, \ldots, r_n) \leftarrow \{0,1\}^n$.

The hash function takes hk and the first-round commitments $\text{Com}_1, \ldots, \text{Com}_n$ and does the following:

$\text{Hash}(\text{hk}, \text{Com}_1, \ldots, \text{Com}_n)$:

1. Decrypt Com_i using s under FHE and get (H_i, π_i).
2. Given each H_i, π_i
   - If $\pi_i(G) = H_i$, set $b_i^* = 0$
   - Else, set $b_i^* = 1$
3. By doing above under FHE, we get $\text{FHEENC}(b_1^*, \ldots, b_n^*)$, compressing it gives header h, and payload $(\alpha_1, \ldots, \alpha_n)$
5. Output $(b_1, \ldots, b_n) = (\alpha_1, \ldots, \alpha_n) \oplus (r_1, \ldots, r_n)$

Correlation-Intractability: Previously, we encrypted and compressed $(b_1^*, \ldots, b_n^*)$. By the property of Rate-1 FHE, we can obtain some $(\beta_1, \ldots, \beta_n)$ from the header and the secret s such that $(\alpha_1, \ldots, \alpha_n) \oplus (\beta_1, \ldots, \beta_n) = (b_1^*, \ldots, b_n^*)$. Thus,

$$
\begin{aligned}
&\Pr[(b_1, \ldots, b_n) = (b_1^*, \ldots, b_n^*)] \\
=&\Pr[(\alpha_1, \ldots, \alpha_n) \oplus (r_1, \ldots, r_n) = (b_1^*, \ldots, b_n^*)] \quad \text{[Definition of the hash function output]} \\
=&\Pr[(\alpha_1, \ldots, \alpha_n) \oplus (r_1, \ldots, r_n) = (\alpha_1, \ldots, \alpha_n) \oplus (\beta_1, \ldots, \beta_n)] \quad [(\beta_1, \ldots, \beta_n) \text{ obtained from the header and secret key}] \\
=&\Pr[(r_1, \ldots, r_n) = (\beta_1, \ldots, \beta_n)] \\
\leq&\frac{2^{|\text{header}|}}{2^n} \quad [\text{ as } (r_1, \ldots, r_n) \leftarrow \{0,1\}^n, (\beta_1, \ldots, \beta_n) \text{ depends only on the header}] \\
\leq&\epsilon(n) \quad \text{[This works for large enough } n \text{ where header grows sublinearly in } n]
\end{aligned}
$$

Zero Knowledge (Idea): The simulator would first replace $\text{FHEENC}(s)$ by $\text{FHEENC}(s')$ for some random s'. Then we can use hiding property of the commitment, it samples $b_1, \ldots, b_n$ uniformly, and the commitments. It would then uses Rate-1 FHE to generate the corresponding header and $(\alpha_1, \ldots, \alpha_n)$, which then generates $(r_1, \ldots, r_n)$ such that $(\alpha_1, \ldots, \alpha_n) \oplus (r_1, \ldots, r_n) = (b_1, \ldots, b_n)$. Then, the rest follows similar to the previous simulator.

References

[BKM20] Z. Brakerski, V. Koppula, T. Mour. NIZK from LPN and Trapdoor Hash via Correlation Intractability for Approximable Relations. In *Proceedings of CRYPTO 2020*, pages 738–767. Springer, 2020.

[FS86] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Proceedings of CRYPTO 1986*, pages 186–194, 1986.

[GMR85] S. Goldwasser, S. Micali, C. Rackoff. The Knowledge Complexity of Interactive Proof Systems. In *Proceedings of STOC '85*, pages 291-304, 1985.

[CGH04] R. Canetti, O. Goldreich, S. Halevi. The random oracle methodology, revisited. In JACM, 04.