Lecture 19: Commitment Shchemes and Non-Interactive Zero-Knowledge (NIZK)

Notes by Yael Kalai

MIT - 6.5620
Lecture 19 (November 17, 2025)

> ***Warning:*** This document is a rough draft, so it may contain bugs. Please feel free to email me with corrections.

Recap

In the last class, we constructed the zero-knowledge proof for the language 3COL, and hence for all of NP, assuming the existence of physical safes.

Outline

- Show how to implement physical safes using the notion of commitment schemes. We will see definitions and constructions.
- Define non-interactive zero-knowledge.
- Construct NIZK for NP in the Random Oracle Model, or assuming the soundness of the Fiat-Shamir paradigm.

Commitment Schemes

Definition 1. A commitment scheme corresponding to a message space $\mathcal{M}$ consists of a pair of algorithms $(\mathsf{Gen}, \mathsf{Com})$:

- Gen is a PPT algorithm that takes as input the security parameter 1^λ and outputs public parameters, denoted by $\mathsf{pp} \in \{0,1\}^{\mathrm{poly}(\lambda)}$.
- Com is a polynomial-time computable function that takes an input public parameters pp, a message $m \in \mathcal{M}$ and randomness $r \xleftarrow{R} \{0,1\}^\lambda$ and outputs a commitment $\mathsf{Com}(\mathsf{pp}, m, r)$.

One can think of both Gen and Com as randomized algorithms. We chose to explicitly include the randomness of Com, and hence think of it as being deterministic since when the commitment is opened the randomness is revealed

The following two properties are required to hold:

- **Hiding:** For every $m_0, m_1 \in \mathcal{M}$,

$$(\mathsf{pp}, \mathsf{Com}(\mathsf{pp}, m_0, r_0)) \approx (\mathsf{pp}, \mathsf{Com}(\mathsf{pp}, m_1, r_1))$$

for $\mathsf{pp} \leftarrow \mathsf{Gen}(1^\lambda)$ and $r_0, r_1 \xleftarrow{R} \{0,1\}^\lambda$.

- **Binding:** For every (even all-powerful) adversary $\mathcal{A}$:

$$\Pr_{\mathsf{pp} \leftarrow \mathsf{Gen}(1^\lambda)}[A(\mathsf{pp}) = (m_0, r_0, m_1, r_1) \text{ s.t. } m_0 \neq m_1 \ \wedge \ \mathsf{Com}(\mathsf{pp}, m_0, r_0) = \mathsf{Com}(\mathsf{pp}, m_0, r_0)] = \mathrm{negl}(\lambda).$$

Remark. If the hiding property holds *for every* pp, then we can replace each physical safe with a commitment to the relevant color, w.r.t. pp randomly chosen and sent by the verifier. In this case, the ZK protocol above becomes a computational ZK protocol, due to the computational hiding guarantee. It still has same completeness and soundness guarantees, up to a negligible loss in soundness due to the fact that there is a negligible probability that pp is sampled in a way that does not bind the cheating prover.

If the hiding property holds only for a random pp (as opposed to every pp), then we cannot let a (possibly malicious) verifier sample it. Nevertheless, jumping ahead, a hiding guarantee for random pp is useful for NIZK where there is a trusted common referenced string which may contain pp.

Remark. In the above definition, the hiding requirement is *computational* and the binding is *statistical*. One can define a commitment scheme to be *statistically hiding* and only *computationally binding* (as you will see in the HW). If we replace our opaque boxes with such a commitment, that is statistically hiding for every pp, then we obtain statistical ZK (due to the statistical hiding property of the commitment scheme), but get soundness only against computationally bounded cheating provers since an all-powerful cheating prover can break the binding of the commitment scheme. We will talk more about such computational soundness guarantee in the upcoming lectures.

Constructing a Commitment Scheme

It is known how to construct a commitment scheme from the minimal assumption that one-way functions exist. We will see two constructions: The first assumes the existence of one-way functions. The second assumes the existence of injective one-way functions, but does not need any public parameters!

Construction from OWFs: Let $G : \{0,1\}^* \to \{0,1\}^*$ be a pseudorandom generator (PRG) that takes inputs of length λ bit string and outputs a length 3λ bit string. Recall that we know how to construct such a PRG from any OWF [6].

We saw a construction from any OWP.

- $\mathsf{Gen}(1^\lambda)$ outputs a uniformly random string $u \leftarrow \{0,1\}^{3\lambda}$.

- $\mathsf{Com}(u, b, r) = G(r) \oplus b \cdot u$.

Hiding (for every $\mathsf{pp} = u$) follows from the assumption that $G(r)$ is indistinguishable from uniform in $\{0,1\}^{3\lambda}$, which in turn implies that for every $u \in \{0,1\}^{3\lambda}$,

$$G(r) \oplus u \approx G(r).$$

Binding follows from the fact that

$$\Pr_{u \leftarrow \{0,1\}^{3\lambda}}[\exists r, r' \in \{0,1\}^{\lambda} : G(r) \oplus u = G(r')] =$$
$$\Pr_{u \leftarrow \{0,1\}^{3\lambda}}[\exists r, r' \in \{0,1\}^{\lambda} : G(r) \oplus G(r') = u] \leq$$
$$\frac{2^{2\lambda}}{2^{3\lambda}} = 2^{-\lambda}.$$

Construction from injective OWF without pp*:* We will see a simple construction from injective one-way functions. This construction does not need pp.

Fix an injective one-way function $f : \{0,1\}^* \to \{0,1\}^*$. Let P be a (randomized) hardcore predicate corresponding to f. Define

$$\mathsf{Com}(b, (r, s)) = (f(r), s, P(r, s) \oplus b)$$

Note that this scheme has no public parameters. Computational hiding follows from the fact that P is a hardcore predicate of f, which by definition implies that

$$(f(r), s, P(r, s)) \approx (f(r), s, U)$$

where $U \leftarrow \{0,1\}$ is a uniformly random bit.

Binding follows from the fact that f is injective.

Non-Interactive ZK (NIZK)

Definition 2. A NIZK for a language $\mathcal{L} \in \mathsf{NP}$ consists of three (non-interactive) PPT algorithms (Gen, P, V):

1. Gen**:** Takes as input the security parameter 1^λ and outputs crs.
2. **Prover** P**:** Takes as input crs and an instance x and a corresponding witness w, and outputs a proof π.
3. **Verifier** V**:** Takes as input crs an instance x and a proof π and outputs a bit 0 or 1, indicating whether it rejects or accepts the proof.

The following three properties are required to hold:

- **Completeness:** There exists a negligible function μ such that for every $\lambda \in \mathbb{N}$ and every $(x, w) \in \mathcal{R}_{\mathcal{L}}$,

$$\Pr[(V(\mathsf{crs}, x, P(\mathsf{crs}, x, w)) = 1] \geq 1 - \mu(\lambda)$$

  where the probability is over $\mathsf{crs} \leftarrow \mathsf{Gen}(1^\lambda)$ and over the randomness of the prover P.

- **(Adaptive) Soundness:** For every poly-size (cheating) prover P^* there exists a negligible function μ such that for every $\lambda \in \mathbb{N}$,

$$\Pr[P^*(\mathsf{crs}) = (x, \pi) : \ x \notin \mathcal{L} \ \wedge \ V(\mathsf{crs}, x, \pi) = 1] \leq \mu(\lambda)$$

  where the probability is over $\mathsf{crs} \leftarrow \mathsf{Gen}(1^\lambda)$.

- **(Non-Adaptive) Zero-knowledge:** For every PPT V^* there exists a PPT simulator S, such that for every polynomial time instance generator algorithm $\mathcal{I}$, that on input 1^λ outputs $(x, w) \in \mathcal{R}_{\mathcal{L}}$,

$$(\mathsf{crs}, x, P(\mathsf{crs}, x, w)) \approx S(1^\lambda, x)$$

  where $\mathsf{crs} \leftarrow \mathsf{Gen}(1^\lambda)$ and $(x, w) = \mathcal{I}(1^\lambda)$.

Remark. One can strengthen the above zero-knowledge condition to be multi-theorem ZK which asserts that there exists a simulator that simulates multiple proofs (as opposed to only one). Indeed, this would be a more meaningful definition. One can also strengthen the definition by considering an adaptive version, where the instance generator algorithm $\mathcal{I}$ takes as input crs and may generate the instances (x, w) as a function of crs. Indeed, this adaptive multi-instance ZK version is the desired one.

NIZK *Construction in the Random Oracle Model*

The Random Oracle Model (ROM) assumes that the parties have oracle access to a hash function H that is truly random. Note that black-box access to a pseudorandom function is indistinguishable from a ROM.

The NIZK construction in the ROM is the following: Parallel repeat the ZK construction for 3COL, denoted by (P, V). This is no longer ZK but it is honest-verifier ZK (HVZK), since parallel repetition preserves HVZK. Next convert this three message protocol into a non-interactive one by replacing the message sent by the verifier with a Random Oracle (RO) H. Specifically, the NIZK scheme, denoted by $(\mathsf{Gen}, P_{\mathsf{NIZK}}, V_{\mathsf{NIZK}})$ is defined as follows:

- $\mathsf{Gen}(1^\lambda)$ generates a RO H and pp for the underlying commitment scheme.
- $P_{\mathsf{NIZK}}(\mathsf{crs}, x, w)$ repeats the ZK protocol in parallel $k = \mathrm{poly}(\lambda, |G|)$ times as follows:
  1. Generate $a = (a_1, \ldots, a_k)$, where each a_i is generated by independently running P would compute its first message.
  2. Compute $e = H(a)$, where $e = (e_1, \ldots, e_k) \in \{0,1\}^k$.[1]
  3. Compute the corresponding answers $z = (z_1, \ldots, z_k)$, where each z_i is generated using the underlying prover P to complete the i'th execution.
  4. Output $\pi = (a, e, z)$.
- $V_{\mathsf{NIZK}}(\mathsf{crs}, x, \pi)$ does the following:
  1. Parse $\pi = (a, e, z)$.
  2. Check that $e = H(a)$. If this does not hold then output 0.
  3. Parse $a = (a_1, \ldots, a_k)$, $e = (e_1, \ldots, e_k)$ and $z = (z_1, \ldots, z_k)$.
  4. Output 1 if and only if for every $i \in [k]$

$$V(x, a_i, e_i, z_i) = 1.$$

[1] We assume for simplicity, and without loss of generality, that H outputs elements in $\{0,1\}^k$. This is WLOG since if H outputs only one bit, then we can run H on each input (a, i), for $i \in [k]$.

Completeness follows immediately from the completeness of the underlying ZK proof. Soundness follows from the fact that parallel repetition in proof systems reduces soundness to negligible, and from the fact that H is a random oracle, and hence "communicating with H is similar to communicating with V." ZK follows from the fact that one can program the RO. Specifically, first use the simulator for an *honest* verifier, to generate a simulated transcript (a, e, z). Then program the RO H to one such that $H(a) = e$. Note that since we can program H on many inputs, we get ZK for many instances.

Replacing ROM with an Explicit Hash Function

In general, we do not know how to prove the security of the NIZK when instantiated with an explicit (real-world) hash function. Once H is replaced with an explicit hash function, both soundness and ZK are in danger!

Zero knowledge: We know how to prove ZK for a *single statement* by using a special "programmable" hash function that can be programmed so that $H(a) = e$. For example given any keyed hash function H with key k, one can define a new hash function with key (k, u) where $u \leftarrow \{0,1\}^\lambda$, and define

$$H_{k,u}(x) = H_k(x) \oplus u.$$

In this case one can simulate the view by generating (a, e, z) using the simulator corresponding to the honest verifier, and then setting choosing u so that $H_{k,u}(a) = e$, which can be done easily by setting

$$u = H_k(a) \oplus e.$$

This works assuming a does not depend on u, which indeed holds if x is chosen non-adaptively. There are 3-message ZK proofs for NP where the first message a does not depend on the instance x at all, in which case x can be chosen adaptively (for example, the ZK proof for Graph Hamiltonicity due to [3]). Thus, with this hash function, we get adaptive ZK single-instance for NP. One can then use a beautiful idea, due to Feige, Lapidot and Shamir [3] to get to many-instance ZK. The basic idea to get many instance ZK, is by making u really large and using a "fresh" part of u for each NIZK proof; and then shrinking u via a PRG.

Soundness: Whether soundness is preserved is a great question that is still poorly understood. In general, the idea of eliminating interaction from public-coin[2] interactive proofs by replacing the verifier with an explicit hash function dates back to a beautiful paper by Fiat and Shamir [4].[3] Since then, there has been a lot of works on trying to understand whether this paradigm is sound; namely, does there always exist an explicit hash function that if we replace the verifier with this hash function we are guaranteed to have soundness. We know that for many message (non-constant round) protocols, even if we replace the verifier with a RO, we may lose soundness. This is the case for sequential repetition of our ZK protocol for 3COL, since a cheating prover can guess the hash value simulate the proof, and if the guess was wrong try again.

Even for constant round protocols with negligible soundness we have counter examples [1, 5], showing that the Fiat-Shamir paradigm is not secure, if the underlying protocol is only *computationally sound*. For statistically sound protocols, we believe that there should be a sound way to instantiate the Fiat-Shamir paradigm (for constant round protocols with negligible soundness), but we still do not know how to prove this from standard assumptions.

In 2019, there was a breakthrough result [2, 7] that showed that there exists a 3-message ZK proof (the one for graph Hamiltonicity from [3]), that if repeated in parallel and converted into a NIZK as above, then there does exist an explicit hash function that makes it sound and zero-knowledge.

[2] A public-coin protocol is one where the verifier simply sends uniformly random (unstructured) bits.

[3] They considered the specific application of eliminating rounds from interactive identification schemes to obtain signature schemes, but by now we refer to the Fiat-Shamir paradigm as a general paradigm for eliminating interaction from public-coin protocols.

References

[1] Boaz Barak. How to go beyond the black-box simulation barrier. In *42nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 106–115. IEEE Computer Society, 2001.

[2] Ran Canetti, Alex Lombardi, and Daniel Wichs. Fiat-shamir: From practice to theory, part ii (NIZK and correlation intractability from circular-secure FHE). Technical Report 2018/1248, Cryptology ePrint Archive, 2018.

[3] Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple non-interactive zero knowledge proofs based on a single random string. In *31st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 308–317. IEEE, 1990.

[4] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, 1986.

[5] Shafi Goldwasser and Yael Tauman Kalai. On the (in)security of the fiat-shamir paradigm. In *44th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 102–113. IEEE Computer Society, 2003.

[6] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.

[7] Chris Peikert and Sina Shiehian. Noninteractive zero knowledge for NP from (plain) learning with errors. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019*, volume 11692 of *Lecture Notes in Computer Science*, pages 89–114. Springer, 2019.