Last Lecture

- We looked at NIZK in the Random Oracle (RO) model.
- A negative aspect of the constructions from that lecture were the loss of deniability and non-transferability properties.

This Lecture

- We cover how to (partially) regain deniability as well as non-transferability for NIZKs in the Non-Programmable Random Oracle (NPRO) model using a new primitive called \(\Omega\)-protocols. However these benefits come at the cost of losing non-interaction giving us a 1-round ZK argument in the NPRO model.
- Next, we begin the section on NIZK in the Common Reference String (CRS) model.

1 NIZK in the Non-Programmable Random Oracle Model

Non-Transferability  In the constructions from the previous lecture the simulator \(S\) for the NIZK (used to show the zero knowledge property) crucially relied it’s ability to program the Random Oracle (RO) in order to create a convincing but fake proof. However this leads to the problem that the NIZK proofs are now transferable. Indeed any player with access to the RO can act as a verifier (not just the intended recipient of the proof). So if an honestly prover \(P\) supplies a verifier \(V\) with a proof \(\pi\) for statement \(y\), this now permits \(V\) to convince any other player (with access to to the same RO) that \(y \in L\). Yet this seems to contradict the intuitive meaning of “Zero Knowledge”; namely that \(V\) should learn nothing from \(\pi\) beyond the fact that \(y \in L\).

Non-Programmable Random Oracle (NPRO) Model  To address this shortcoming we look turn to the NPRO model which is a restriction of the full RO model where the simulator can no longer choose the responses to RO queries made by the verifier. Instead the only added power we give \(S\) is to let it see all query/response pairs made by \(V\) to the RO during the verification process. The remainder of this section develops the construction of a 2-round ZK argument in the NPRO model which regains the property of non-transferability. (In fact we will even briefly give some intuition showing that the construction enjoys a weak form of deniability.)

On the bright side we note that the Fiat-Shamir transform preserves both WI and Witness Hiding (WH) even in the NPRO as \(V^*\) is less powerful and can, in particular, still...
not choose challenges. However, on the down side, the Proof of Knowledge (POK) property of last lectures NIZK construction is lost. If, after rewinding \( P^* \), it is not possible to reprogram the RO so that it gives a different challenge (i.e. a different response to the same query “\( a||m \)”) then it is not possible to obtain an alternative transcript of the underlying 3-round public-coin protocol. Thus, the special-soundness extractor can no longer be used to extract a witness.

1.1 Extractable Commitments

Before we can delve into resolving this issue we make a brief digression to introduce a useful yet simple primitive called “extractable commitments”. Informally, these are (hiding and binding) commitments with the added property that for any adversary \( A \), given only a commitment \( \gamma \leftarrow A \) together with the RO queries \( Q \) made by a \( A \), it is as (almost) as easy to extract a valid decommitment as it is for \( A \) to do so.

If \( \delta \) is a valid decommitment of \( \gamma \) then let \( \text{open}(\gamma, \delta) = m \) where \( m \) is the message committed to by \( \gamma \). Otherwise let \( \text{open}(\gamma, \delta) = \bot \). We formalize the notion of extractability for commitments as follows:

**Definition 1 (Extractable Commitment)** We call a commitment scheme \( C \) in the NPRO model extractable if there exists an algorithm \( E' \in \text{PPT} \) such that for all adversaries \( A \in \text{PPT} \) the following is a negligible quantity in security parameter \( \lambda \):

\[
\Pr[(\gamma, \text{state}) \leftarrow A ; \delta' \leftarrow E'(\gamma, Q) ; \delta \leftarrow A(\text{state}) : \text{open}(\gamma, \delta) = 1 \land \text{open}(\gamma, \delta') = 0]
\]

where \( Q \) is the list of RO queries made by \( A \).

A subtle point concerns pathological adversaries \( A' \) which could output a valid \( \delta \) but instead choose not to. In this case we would still like \( E' \) to succeed yet it is not immediate from the above definition since we only consider the case when \( \text{open}(\gamma, \delta) = 1 \). However for any such adversary there is another related adversary \( \bar{A}' \) which behaves just as \( A' \) when producing \( \gamma \) but instead chooses to output a valid \( \delta \). Thus the list \( Q \) is the same for both adversaries as is the commitment \( \gamma \). In other words the view of \( E' \) is identical in both cases, yet in the second case we require it to extract with overwhelming probability. Therefore it will also extract in the first case when playing against the pathological \( A' \).

We now describe a very simple construction in the NPRO model. We require only that the random oracle \( H \) have a sufficiently large input and output spaces to make it collision resistant (such as \( H : \{0,1\}^{2\lambda} \rightarrow \{0,1\}^{\lambda} \) for example).

**Commit**\((m)\): Select fresh random string \( r \leftarrow \{0,1\}^{\lambda} \). Output commitment \( \gamma := H(m;r) \) and decommitment \( \delta := (m,r) \).

**Extract**\((\gamma, Q)\): Find a pair \((m;r) \in Q\) such that \( H(m;r) = \gamma \) and output it. If none exists output \( \bot \).

Clearly the scheme is hiding since the commitment \( \gamma \) is a random value and the only means of finding the message is to first guess \( r \) and then call \( H \). But with overwhelming probability the guess for \( r \) will be wrong. On the other hand the scheme is binding since
breaking this property would imply that the malicious decommitter managed to find a collision for \(H\). Finally for we note that the probability that \(\text{open}(\gamma, \delta) = 1\) but \(A\) did not query \(H\) on \(m||r\) is negligible due to the size of the output space of \(H\). Thus we can assume that this value was queried and so \(E'\) will find it in \(Q\) and output. We leave it to the reader to formalize this proof.

### 1.2 Ω-Protocols

Now that we have such commitments we can use them to modify standard Σ-protocols for a relation \(R\) in the NPRO model to obtain a strengthening called Ω-protocols. These are Σ-protocols with the extra property of straight-line extractability; that is there is an efficient method of extracting a witness from a prover without rewinding the prover (nor programing the RO) that can extract a witness if the honest verifier would accept the proof.

We now formalize this intuition. Let \(\pi \leftarrow \langle P^*(y) \leftrightarrow V(y) \rangle\) be the transcript produced by selecting fresh random tapes and having \(P^*\) and \(V\) interact with each other on input statement \(y\). Further, the predicate \(\text{Accept}(\pi)\) is true if and only if \(V\) accepted during the conversation that produced transcript \(\pi\). Then an extractor for protocol \(\langle P, V \rangle\) is an efficient algorithm \(E \in \text{PPT}\) which is given input a transcript and the RO queries made by \(P^*\) while producing the transcript. At the end of the execution \(E\) produces arbitrary string as output (which is presumably a witness to \(y\)).

**Definition 2 (Ω-Protocol)** Let \(\Pi\) be a Σ-protocol with security parameter \(\lambda\). We call \(\Pi\) an Ω-protocol if there exists an efficient extractor \(E \in \text{PPT}\) such that for any prover \(P^*\) and statement \(y\) we have that:

\[
\Pr[\pi \leftarrow \langle P^*(y) \leftrightarrow V(y) \rangle ; \ x \leftarrow E(\pi, Q) \ ; \ \text{Accept}(\pi) \land (x, y) \notin R] = \text{negl}(\lambda)
\]

Were \(Q\) is the list of RO queries made by \(P^*\), “negl” is a negligible function and the probability is taken over the coin tosses of \(P^*, V\) and \(E\).

**Construction 1:** Let \(H\) be a NPRO. Let \(\Pi\) be a Σ-protocol for relation \(R\) in the NPRO model with messages \((a, c, z)\) and binary challenge space \(c \in \{0, 1\}\). Finally let \((\text{Com, Dec, } E')\) be an extractable commitment scheme also in the NPRO model. We use these primitives to construct an Ω-protocol for statement \(y\) with \((x, y) \in R\) as follows:

\(\mathcal{P}(x) \rightarrow V(y)\):

1. Compute the first flow and both possible final flows of \(k\) independent executions of \(\Pi\) for statement \(y\). In symbols: compute \(\{a_i, z_{i,0}, z_{i,1}\}_{i \in [k]}\) where both \((a, 0, z_{i,0})\) and \((a, 1, z_{i,1})\) are accepting transcripts.
2. Commit to all final messages. In symbols for \(i \in [k]\) and \(b \in \{0, 1\}\) compute \(\gamma_{i,b} = \text{Com}(z_{i,b})\).
3. Send \(\{a_i, \gamma_{i,0}, \gamma_{i,1}\}_{i \in [k]}\) to \(V\).

\(\mathcal{P}(x) \leftarrow V(y)\):

1. Select \(k\) random bits \(c_1, \cdots, c_k\) and send them to \(\mathcal{P}\).

\(\mathcal{P}(x) \rightarrow V(y)\):
1. Compute the decommitments for the challenged third flow messages. In symbols:
   for \( i \in [k] \) compute \( \delta_{i,c_i} = Dec(\gamma_{i,c_i}) \).
2. Send \( \{\delta_{i,c_i}\}_{i \in [k]} \) to \( \mathcal{V} \).

**Claim 3** Construction 1 is a \( \Sigma \)-protocol.

**Proof:** To justify this claim we check that the construction enjoys both special soundness and has an HVZK simulator. For special soundness we note that if a prover \( \mathcal{P}^* \) is able to respond to two conversations with with different challenges \( \vec{c} \neq \vec{c}' \) then there is an \( i \in [n] \) for which both \( (a_i, 0, z_0) \) and \( (a_i, 1, z_1) \) are accepting conversations. Thus the special soundness extraction algorithm of the underlying \( \Sigma \)-protocol can be used to extract a witness to the statement.

The HVZK simulator for construction 1 gets a vector \( \vec{c} \) as input. It runs the HVZK simulator of the underlying \( \Sigma \)-protocol \( k \) times with fresh random tape and input challenge \( c_i \) to obtain conversations \( (a_i, z_{c_i}) \). Finally it sets all \( z_{1-c_i} = 0 \) and commits to these values. Intuitively since only the commitments \( \gamma_{i,c_i} \) will be opened, no commitment to 0 is ever revealed and so by the hiding property of the commitment scheme the resulting transcript produced by the HVZK simulator is indistinguishable from a transcript produced via an interaction with the honest prover \( \mathcal{P} \). \( \square \)

**Lemma 4** Construction 1 is an \( \Omega \)-protocol

**Proof:** In light of claim 3 it remains only to describe the algorithm \( \mathcal{E} \in \text{PPT} \) which extracts a witness \( x \) directly from an (accepting) transcript \( \pi \) and the RO query list \( Q \). To do this \( \mathcal{E} \) first parses out the flow 1 message from \( \pi \). Next it uses the extraction property of the commitment scheme to obtain all pairs \( (z_{i,0}, z_{i,1}) \). For each such pair the special soundness simulator is run on input \( (a_i, z_{i,0}, z_{i,1}) \) and if any of these triples produces a witness \( x \) to \( y \) then output it.

To see that \( \mathcal{E} \) indeed satisfies the definition of an \( \Omega \)-protocol we note that if extraction fails then this means it failed for all \( i \). In other words, for no copy of the underlying \( \Sigma \) protocol could \( \mathcal{P}^* \) respond to both possible challenges (without breaking the binding property of the commitment scheme at least). Thus with probability at most \( (1/2 + \text{negl}(\lambda))^k \) could \( \mathcal{P}^* \) succeed in convincing \( \mathcal{V} \) to accept. Thus by setting \( k \geq \lambda \) we obtain the result. \( \square \)

**Removing Interaction:** We would like remove the need for interaction for the \( \Omega \)-protocol so we apply the Fiat-Shamir transform. This gives us a straight-line extractable non-interactive protocol. In particular we get a witness indistinguishable proof of knowledge (WI-PoK) which will prove useful later on in our quest to restore at least some deniability to our NIZK protocols. In fact the construction preserves the underlying witness hiding (WH) property of the \( \Sigma \)-protocol. Thus if we use a \( \Sigma \)-protocol for an SPR function (which we know to be WH) we get a WH-PoK \( \Omega \)-protocol. As an alternative we could use a \( \Sigma \)-protocol that uses the “OR” trick to get WH (described in lectures 4 and 5).

One might be tempted to think that we have already obtained our goal since applying the Fiat-Shamir transform to a \( \Sigma \)-protocol (i.e. also an \( \Omega \)-protocol) already results in a NIZK. However, for such a protocol, the proof of zero knowledge (in contrast to the WI-PoK proof) now requires programmability of the Fiat-Shamir RO \( H' \) again. This is because
the ZK simulator \( S \) uses the underlying HVZK simulator to produce a transcript \((a, c, z)\). In particular \( a \) and \( z \) are determined after \( c \). So by programming \( H' \), \( S \) must ensure that \( c \) is will be used by the verifier when checking it’s non-interactive proof. In other words \( S \) must program the \( H'(a||y) = c \).

1.3 Feige-Shamir

Nevertheless we can still use the above non-interactive protocol in the NPRO model to at least partly achieve our goal of constructing deniable NIZK proofs. Do do this we turn to the Feige-Shamir paradigm. This is a general method of constructing non-transferable zero knowledge proofs of knowledge (ZK-PoK).

Construction 2: Let \( R \) be an NP relation which is equipped with an efficient sampling algorithm \( \text{Sample} \) which outputs a hard instance \( y \) together with a witness \( x \). Let \( \Omega(y) \) be a witness hiding proof of knowledge (WH-PoK) for \( R \) with statement \( y \) and let \( \Omega'(y, y') \) be a witness-indistinguishable proof of knowledge (WI-PoK) for the statement \((y \in R) \lor (y' \in R)\). Given these primitives, the Feige-Shamir construction is in the plain model. That is it requires no additional setup nor random oracle.

\[
\begin{align*}
\mathcal{P}(x) \leftarrow & \mathcal{V}(y): \\
1. & \text{Sample } (x', y') \leftarrow \text{Sample} \\
2. & \text{Run } \Omega(y') \text{ acting as the prover (using witness } x') \text{ and with } \mathcal{P} \text{ acting as verifier.}
\end{align*}
\]

\[
\begin{align*}
\mathcal{P}(x) \rightarrow & \mathcal{V}(y): \\
1. & \text{Run } \Omega'(y, y') \text{ acting as the prover (using witness } x) \text{ and with } \mathcal{V} \text{ acting as verifier.}
\end{align*}
\]

We argue that under the assumptions on \( \Omega, \Omega' \) and \( \text{Sample} \), construction 2 is a ZK-PoK.

Completeness: This property follows directly from the completeness of \( \Omega' \) and \( \Omega \).

Zero Knowledge: The simulator for construction 2 first uses the (PoK) extractor for \( \Omega \) to extract \( x' \). Then it uses \( x' \) as a witness to run \( \Omega' \) honestly. By the WI property of \( \Omega' \) the resulting transcript is indistinguishable from an honest execution with witness \( x \).

Proof of Knowledge: By assumption \( \Omega' \) is a PoK. Thus if a prover \( \mathcal{P}^* \) can produce an accepting \( \Omega' \) then we can extract a witness; namely \( x \) or \( x' \). However extracting \( x' \) would contradict the WH property of \( \Omega \) (and the assumption that instance \((y', x')\) is hard). Therefor by the PoK property of \( \Omega' \) it must be that we extract \( x \) such that \((x, y) \in R\).

Although in general the two flows may well be interactive, we will use the non-interactive protocol from the previous subsection for \( \Omega' \). As an added benefit we get straight-line extractability for both \( \Omega \) and \( \Omega' \). Further, since we do not require ZK for these building blocks, but instead only the WI-PoK properties we can instantiate construction 2 in the NPRO model (rather then the full RO model). Summing everything up we have shown how to construct a straight-line extractable and straight-line zero knowledge 2-flow protocol in the NPRO model. Jumping ahead, this is our first example of a Universally Composable ZK protocol.
On Deniability: Returning to our main motivation for this lecture we ask if our instantiation of construction 2 is “deniable” (at least in some intuitive sense). Indeed at least non-transferability of the proof seems to have been restored since \( V \) knows the witness \( x' \) by construction and so no other party has reason to believe that \( \Omega' \) actually implies that \( x \in R \) rather then \( x' \in R \). Yet \( V \) is of course convinced since it knows it did not reveal \( x' \) to \( P^* \). Never the less the following informal argument demonstrates that there remain efficient judges \( J \) which can verify that the interaction took place. (I.e. the protocol is not deniable with respect to such a \( J \).)

\( J \) colludes with \( V \) in order to verify that \( P \) actually ran the protocol. To do this it computes the first flow of construction 2 on behalf of \( V \) sending only the resulting message to \( V \) which forwards it on to \( P \). Note that \( J \) did not reveal the value of \( x' \) to \( V \) nor \( P \). Thus when \( P \) responds with a valid second flow for some statement \( y \), \( J \) is now convinced that \( P \) took part in the conversation. In particular the only way to compute a valid second flow to it’s first flow was for someone to have used the witness \( x \) to \( y \) and only \( P \).

Not all is lost though. Indeed it can be shown that for non-colluding judges our instantiation in the NPRO model is deniable. Informally this can be understood to mean that for any non-colluding judge \( J \) any view it might obtain of an interaction with \( P(x) \) and \( V(y) \) can essentially be simulated to \( J \) without even knowing \( x \) (or any other secret \( P \) might use such as it’s secret key).

On the down side though, it turns out that it’s impossible to have a 2-flow deniable ZK proof against general (colluding) judges if it’s not possible to program and extract the RO. Later we will show instead how this can be done in the PKI model.

2 NIZK in the CRS Model

We now leave the RO model behind us and move on to look at NIZK in the it’s most common instantiation, namely in the common reference string (CRS) model.

Recall that in the previous lecture we gave to two brief definitions for NIZK in the CRS model. We now generalize these definitions and expand on several important variations there of. We give two constructions, one efficient number-theoretic construction and another based on general assumptions. We begin with the weakest type of zero knowledge protocol and slowly refine the security notion, giving examples along the way.

Let \( P \) and \( V \) be a pair of probabilistic interactive turing machines. For NP relation \( R \) and pair \( (x, y) \in R \) we denote by \( \langle P(x) \leftrightarrow V(y) \rangle \) the random variable describing the output of \( V \) (“accept” or “reject”) when run with input statement \( y \) against prover \( P \) with private input witness \( x \).

Definition 5 (Interactive Proof System) Let \( P \) and \( V \) be a pair of ITMs. We call \( \Pi = \langle P, V \rangle \) and interactive proof system for NP relation \( R \) (IP for short) if there exists a negligible function “negl” in security parameter \( \lambda \) such that:

Completeness: The honest prover convinces the verifier. In symbols:

\[
\forall(x, y) \in R \quad \Pr[(P(x) \leftrightarrow V(y)) = \text{accept}] \geq 1 - \text{negl}
\]
Soundness: The verifier never accepts a false statement. In symbols: \( \forall P^* \in \text{ITM} \)

\[
\Pr \left[ (y^*, aux) \leftarrow P^*(1^\lambda); b \leftarrow \langle P^*(aux) \leftrightarrow V(y^*) \rangle; (b = \text{accept}) \land (y^* \notin R) \right] \leq \text{negl}
\]

Efficient Honest Provers: Although IP are meaningful even with inefficient honest provers for our purposes we usually aim for and consider only proof systems with efficient \( P \) which, in particular, make critical use of their private input \( x \).

Proof vs. Argument: In the above definition, we ask that soundness holds with respect to unbounded provers. In this case we speak of an interactive proof. An interesting restriction to consider is if soundness holds only with respect to bounded provers \( P^* \in \text{PPT} \). We refer to such protocols as interactive arguments.

2.1 Plain Zero Knowledge

We now turn to the common reference string (CRS) model and introduce an additional property for an IP called zero knowledge. As discussed in previous lectures, in the CRS model all parties are given a public input (the CRS) sampled from a particular public distribution by a trusted third party using a Setup algorithm specified by the protocol designer. The simulator is then either able to sample a “fake” CRS it’s self or, more generally, it is given some trapdoor information concerning the CRS which it uses to simulate proofs. Here we define the weaker form which we sometimes refer to as “plain” ZK or just ZK for short.

More specifically let \( \Pi = \langle P, V \rangle \) be an interactive proof system for NP relation \( R \). Then we equip \( \Pi \) with two efficient probabilistic algorithms Setup and FakeSetup as follows. Both take input the security parameter \( \lambda \) (in unary). Setup(\( 1^\lambda \)) samples a new CRS and we denote it’s output via the random variable \( \sigma \). On the other hand FakeSetup(\( y^*, 1^\lambda \)) takes in additional input a statement \( y^* \) and samples a fake CRS together with a trapdoor and we denote it’s pair of outputs via the random variables \( (\tilde{\sigma}, \tilde{t}) \).

Finally for any verifier \( V^* \), input pair \( (x, y) \) and CRS \( \sigma \) we write View\( _{V^*}(x, y, \sigma) \) to denote random variable describing \( V^* \)’s entire view of the interaction between \( P(x) \) and \( V^*(y) \) with CRS \( \sigma \). The variable is sampled by running the machines with fresh independent random tapes.

Definition 6 (“Plain” Zero Knowledge in the CRS Model) Let \( \Pi = \langle P, V \rangle \) be an interactive proof system for NP relation \( R \) equipped with two efficient probabilistic algorithms Setup and FakeSetup as above.

Then we call \( \Pi \) a zero knowledge protocol (for NP relation \( R \)) in the CRS model (or “plain ZK” for short) if there exists an efficient (simulator) \( S \in \text{PPT} \) with oracle access verifier \( V^* \) such that \( \forall (x, y) \in R \) and all \( V^* \in \text{ITM} \) the following 2 pairs of random variables are indistinguishable:

\[
(\sigma, \text{View}_{V^*}(x, y, \sigma)) \approx (\tilde{\sigma}, S^{V^*}(y, \tilde{t}))
\]
Computational ZK: A meaningful relaxation of Definition 6 is when the two distributions in the defining equation are only computationally indistinguishable (rather than statistically). We refer to such protocols as “computational” ZK. Note that only trivial relations (i.e. those in \( \text{BPP} \)) can have both soundness and ZK hold statistically at the same time.

Uniform Reference String Model: A slight restriction on the CRS model requires that the reference string be a uniformly random string. In this case we speak of the “Uniform Random String” (URS) model. While this assumption makes real world instantiation of the model significantly easier it turns out that it does not significantly restrict the set of relations for which we can construct NIZK protocols as demonstrated by our second construction below. Indeed all examples in this lecture can easily be modified to work in the URS model. However for reasons of exposition we present them in the CRS model.

The Non-interactive Case: As stated above, when it comes to interaction our definitions are actually more general then we need. In particular we will really only be concerned with the special case of 1-message ZK protocols in the CRS model which we refer to as non-interactive ZK (NIZK) protocols. In this case \( \mathcal{V}^* \) disappears and the equation of Definition 6 can be written as:

\[
(\sigma, \mathcal{P}(x, \sigma)) \approx (\tilde{\sigma}, \mathcal{S}(y, t))
\]

2.2 Example: NIZK Proof

We give a quick example of a computational NIZK proof that a pair of integers \((n, e) = y\) is a valid RSA key i.e. that \(\gcd(e, \phi(n)) = 1\). The witness is the pair of prime factors \((p, q) = x\) of \(n\). The CRS is \(\sigma = (r_1, \ldots, r_k)\), a \(k\)-tuple of uniform random elements from \(r_i \leftarrow \mathbb{Z}_n^*\). The NIZK proof is \(\mathcal{P}(x, y, \sigma) \to \pi := \{z_i = (r_i)^e \mod n\}_{i \in [k]}\) i.e. the \(e\)-th roots of all \(r_i\) in \(\mathbb{Z}_n^*\). Finally the verifier accepts a proof \(\pi = \{z_i^*\}\) if and only if \(r_i = (z_i)^e \mod n\) for all \(i \in [k]\).

Completeness is easy to verify. To see that this is sound note that if \(\gcd(e, \phi(n)) \neq 1\) then \(\{z^e \mod n \mid z \in \mathbb{Z}_n^*\}\) \(\leq \frac{\phi(n)}{2}\) and thus the probability that (even an unbounded) \(\mathcal{P}^*\) can extract \(k\) roots for an invalid \((n, e)\) is at most \(\left(\frac{2}{\phi(n)}\right)^k \leq \left(\frac{1}{2}\right)^k\).

For zero knowledge consider the \text{FakeSetup} algorithm which takes input \((n, e)\), samples \(z_i \leftarrow \mathbb{Z}_n^*\) for \(i \in [k]\) and outputs fake CRS \(\tilde{\sigma} := \{r_i = (z_i)^e\}_{i \in [k]}\) and trapdoor \(\tilde{t} = \{z_i^*\}_{i \in [k]}\). Of course the simulator can now simply output \(\tilde{t}\) as it’s fake proof and this will pass verification. However the fake CRS is only computationally indistinguishable from the real CRS making this computational ZK.

2.3 Oblivious CRS

Another strengthening of Definition 6 concerns how the fake CRS is chosen. In particular if the algorithm \text{FakeSetup} does not use input \(y^*\) then we speak of a ZK protocol with “oblivious CRS” sampling. Although this is a serious restriction\(^1\) we will show a very

\(^1\)Note that for example, the previous RSA based NIZK was \textit{not} oblivious CRS.
general construction satisfying this stronger definition.

Intuitively oblivious CRS sampling will play an important role in later lectures when we discuss security in the face of composition. Indeed, when the fake CRS does not depend on which $y^*$ is to be simulated, this opens the door to reusing the CRS for other simulations and even real proofs. A bit more precisely oblivious CRS sampling will prove essential when showing the property of “adaptive” soundness where the adversary $P^*$ is permitted to choose the false statement $y^*$ depending on the CRS.

From this point on we only consider protocols with oblivious CRS sampling unless otherwise specified.

2.4 Same-String ZK

We introduce another strengthening of Definition 6 which requires the same CRS to be used in both real and simulated proofs. This can potentially allow protocols to use both real and simulated proofs in the real world (rather then only permitting simulated proofs in the ideal world of the security proof). This time we equip the IP $\Pi$ with only one additional algorithm SSetup (“Same-string Setup”) which takes input $1^\lambda$ and samples a fresh CRS and trapdoor. We denote it’s output with the random variables $(\sigma, t)$.

**Definition 7 (Same-String ZK)** Let $\Pi$ and SSetup be as above. We call $\Pi$ a same-string zero knowledge proof in the CRS model if there exists an efficient turing machine (simulator) $S$ with oracle access to verifier $V^*$ such that $\forall (x, y) \in R$ and all $V^* \in \text{PPT}$ the following 2 pairs of random variables are indistinguishable:

$$(\sigma, \text{View}_{V^*}(x, y, \sigma)) \approx (\sigma, S_{V^*}(y, t))$$

In the special case of NIZK we can simplify the equation to $(\sigma, P(x, \sigma)) \approx (\sigma, S(y, t))$.

**NIZK Proofs:** In contrast it is easy to show that if there exist one-way functions (OWF) then there are languages in NP for which no same-string NIZK proof exists. Indeed if $\exists$ OWF then there exists 2 indistinguishable random variables $y \approx y'$ which range over some NP language. Now because $S$ and SSetup are both efficient they can’t be used to distinguish $y$ from $y'$ and so $S(y, t) \approx S(y', t)$. Further by the same-string ZK definition it must be that both of these are random variables over accepting proofs. But this gives an unbounded malicious prover $P^*$ a strategy for breaking soundness. Given the CRS $\sigma$ it simply finds the corresponding trapdoor $t$ and runs $S$ to produce a convincing proof.

**NIZK Arguments:** It turns out that that a relation has a (plain) NIZK argument if and only if it has a same-string NIZK argument. In particular one can always set $\sigma := \tilde{\sigma}$ because a bounded cheating prover can not use this fact effectively (with out the trapdoor).

**Corollary 8** If $\Pi$ is a plain NIZK (with efficient prover) for relation $R$. Then $\Pi$ together with algorithm SSetup := FakeSetup is a same-string NIZK argument for $R$.

---

2These can, for example, be derived from the hardcore predicates associated with the OWF via the Goldreich-Levin theorem.
2.5 Single vs. Multi-Theorem

As hinted already in the subsection on oblivious CRS sampling, another important variant on ZK is to consider the multi-theorem setting. Currently we only look at the joint distribution of the CRS and a single proof. However, motivated by the real world constraint that common randomness may be hard to obtain, (not to mention efficiency concerns) we would eventually like to reuse a single CRS for proving multiple theorems. To this end we will need ZK to hold with respect to the joint distribution of multiple proofs. We will return to this point in much greater detail in subsequent lectures.

3 NIZK From General Assumptions

So far we have seen an example of (plain) non-oblivious NIZK in the CRS model based on a specific number theoretic assumption (i.e. that factoring is hard). We now turn to same-string oblivious NIZK in the CRS model and give a construction from general assumptions for the NP-complete language of $n$-vertex graphs containing a Hamiltonian Cycle (HC). Recall that for an (undirected) graph $G$ a Hamiltonian cycle is a path traversing each node in $G$ exactly once.

We begin with an unbounded prover $P$ for which we construct plain NIZK proof. Later we will fix this by showing an almost equivalent but efficient prover giving us same-string NIZK argument. In both cases we consider the relation with statement a graph $G$ and witness $C \subseteq G$ an HC in $G$.

Idea: To give some intuition behind the protocol we recall the alternative Σ-protocol for graph HC from the end of lecture 7. The crucial observation of made there was that the original statement can be modified into an equivalent statement concerning the compliments of the graph $G$ and the cycle graph $C$ with the result that the first two flows of the alternative Σ-protocol are completely independent of the statement being proven. What’s more the only purpose of a challenge bit $c = 0$ is to verify that the first flow was generated honestly.

To make a NIZK (at least for unbounded provers) out of this Σ-protocol we simply push the first flow to the CRS. Now by assumption we are guaranteed that is honestly generated and so the only challenge of interest is $c = 1$. Thus no interaction is required of the verifier and it remains only to show how the prover can compute the final flow for this challenge and for a given CRS. (Of course the real trick will be to show how a bounded prover can do this.) But first we look at the conceptually simpler case of an unbounded prover.

3.1 NIZK with Unbounded Prover

The following proof is repeated $t = n^{2/3} \lambda$ times in parallel. That is the CRS is broken up into $t$ blocks and the following is repeated in parallel for each block.

For the unbounded prover the CRS $\sigma$ is interpreted as a $n^6$ perfectly binding commitments\(^3\) of the entries of an $n^3 \times n^3$ boolean matrix $H$ where each entry is 1 with probability $n^{-5}$. (This can be achieved by viewing a block of $5 \log_2(n)$ of bits as a 1 if only if all entries are 1.) We call $H$ good if it contains an $n \times n$ adjacency matrix $C_H := \{e_{i,j} \in \{0,1\}\}_{i,j \in [n]}$

\(^3\)Note that a simple example of such a commitment scheme is a public key encryption scheme.
of an $n$-vertex Hamiltonian cycle$^4$. The proof proceeds in one of two ways depending on whether $H$ is good or not.

**Bad $H$:** $\mathcal{P}$ computes and reveals all decommitments to the entries of $H$. The proof is accepted if and only if $H$ is bad.

**Good $H$:**

1. $\mathcal{P}$ samples a random vertex-permutation $\delta$ mapping $C$ to $C_G$.
2. $\mathcal{P}$ computes the set $d$ of decommitments to the non-edges of $\delta(G)$ in $C_H$. In symbols it computes: $d = \{\text{Dec}(e_{i,j})\}_{(i,j) \notin \text{Edges}(\delta(G))}$.
3. $\mathcal{P}$ outputs the proof $\pi = (\delta, d)$ which is accepted if and only if all non-edges in $\delta(G)$ are correctly committed and their entry in $H$ is 0.

For the proof of security we require the following combinatoric lemma. For a proof see Fact 4.10.8 of *Foundations of Cryptography: Volume I* by Oded Goldreich.

**Lemma 9** $\Pr[H \text{ is good}] = \Omega(n^{-3/2})$

**Claim 10** The above construction has statistical soundness. I.e. it is an interactive proof.

**Proof:** Completeness is easy to verify. For soundness we first argue that an unbounded cheating prover $\mathcal{P}^*$ running a single iteration (block) of the proof is caught with non-negligible probability. Since the $t$ blocks of the CRS are independent of each other so is the probability that $\mathcal{P}^*$ is caught in each one. Thus for large enough values of $t$ the probability of not being caught is negligibly close to 0.

Consider $i$-th iteration and suppose $H$ is good for this block. Given an accepting proof $\pi$ it is easy for $\mathcal{V}$ to compute $\delta(G)$. Thus it easy to verify that all non-edges of $\delta(G)$ were opened. So it must be that $\delta^{-1}(C_n) \in G$ meaning that $G$ contains an HC. Since the commitment scheme is perfectly binding this property holds with probability 1. Thus by Lemma 9 a cheating prover is caught with probability at least $\Omega(n^{-3/2})$ in this iteration.

Note that the probability of being caught depended only on if $H$ was good or not. Since the values of $H$ are independent for each iteration so are the probabilities of being caught. Thus for $t = n^{3/2}\lambda$ the overall probability of not being caught is at most $(1 - n^{-3/2})^t \leq (1/2)^\lambda$. \hfill \qed

**Claim 11** The above construction is plain computational ZK with oblivious CRS sampling.

**Proof:** We describe the proof for a single iteration. The generalization to all $t$ iterations is straightforward.

Consider the FakeSetup algorithm which first samples a CRS $\sigma$ honestly. If the resulting $H$ is bad it outputs $\tilde{\sigma} := \sigma$ and trapdoor all corresponding decommitments. Otherwise (when $H$ is good) it replaces the commitments the entries of to $C_n$ with commitments to

$^4$More specifically $C_n$ has exactly one 1 in each row and each column and it defines a cycle over all $n$ vertices.
0. It outputs the modified commitments to $H$ as $\tilde{\sigma}$ and the corresponding decommitments as the trapdoor.

To simulate a proof an efficient simulator behaves just as the honest prover when $H$ is bad. When $H$ is good it selects a random permutation $\delta$ and decommits to all non-edges of $\delta(G)$. This proof is always accepted because all entries of the sub-matrix of $H$ are 0. So the only difference for the verifier are the (un-opened) entries of $C_n$ in the commitment of $H$. Thus the output of the simulator is computationally indistinguishable from a real proof and CRS by the hiding property of the commitment scheme.

Note also that the output of $\text{FakeSetup}$ did not depend on the particular instance being simulated. Thus this NIZK is enjoys oblivious CRS sampling.

### 3.2 Efficient Prover

As promised we will now show how to obtain essentially the same properties of our NIZK protocol but with an efficient prover. For this we will use what is called a a family of verifiable hard permutations $F$. On a high level we use these to interpret a random string as a commitments to an adjacency matrix. By allowing the honest prover to select the particular permutation (and it’s trapdoor) in $F$ with which to interpret the CRS it can also compute the corresponding decommitments thereby side-stepping the computationally expensive step of the previous protocol. It remains only to make the CRS long enough that one can guarantee with overwhelming probability that for any choice of a permutation in $F$ at least one commitment to $C_n$ will be derived from the CRS. Thus we can then boot strap from the results of the previous protocol with the unbounded prover.

**Verifiable Hardcore Permutations:** We begin by defining the notion of a family $F$ of verifiable hardcore permutations. This is a set of function pairs $(f, h)$ where $f$ is a one-way permutation and $h$ is the associated hardcore predicate for $f$. The family is equipped with a triple of efficient algorithms ($\text{Gen}, \text{Sample}, \text{Invert}$). Algorithm $\text{Gen}(1^\lambda)$ samples a fresh public/secret key pair $(pk, sk)$ where $F$ is indexed by $pk$. (Abusing notation somewhat we will also write $f_{sk}$ to denote the permutation with the corresponding $pk$ as index.)

The algorithm $\text{Sample}$ is used to sample input/output pairs for a given $pk$. In symbols: $\text{Sample}(pk) \rightarrow (\alpha, \beta)$ such that $f_{pk}(\alpha) = \beta$. Finally algorithm $\text{Invert}$ uses the secret key to invert an image. More precisely $\forall sk$ output by Sample and all $\beta$ in the range of $f_{sk}$ we have $\text{Invert}(sk, \beta) = \alpha$ such that $f_{sk}(\alpha) = \beta$. Jumping we will view the CRS as a sequence of outputs for some $f_{pk}$ and the committed values will be the corresponding hardcore bits.

Given any $f \in F$ the Goldreich-Levin theorem shows us how to efficiently construct a “hardcore predicate $h$ for OWF $f$”. That is for any $pk$ output by $\text{Sample}$ and corresponding hardcore predicate $h_{pk}$ we have that $(f_{pk}(\alpha), h_{pk}(\alpha)) \approx (f_{pk}(\alpha), U_{\{0,1\}})$ where $U_{\{0,1\}}$ denotes the uniform distribution over $\{0,1\}$. Taken in our context one can view $f_{pk}(\alpha)$ as the commitment to the bit $h_{pk}(\alpha)$ with decommitment is $\alpha$.

A more efficient example is based on the RSA assumption. The public/secret key pair is a random RSA key: $pk = (n, e)$ and $sk = d$ which can be sampled efficiently by some algorithm $\text{Gen}$. The algorithm $\text{Sample}$ selects a random “message” $\alpha$ and “encrypts” it under the public key to obtain $\beta$. Algorithm $\text{Invert}$ uses the secret key to “decrypt” the input. Finally a hardcore predicate for RSA is the least significant bit of the message.
We now describe the algorithm $\mathcal{P}$.

1. Sample $(pk, sk) \leftarrow \text{Gen}(1^\lambda)$ and repeat the following steps $t = n^{3/2}(|pk| + \lambda)$ times.

2. Parses the CRS $\sigma = \beta_1, \ldots, \beta_N$ such that each $\beta_i$ is in the range of $f_{pk}$ where $N = 5n^6 \log_2(n)$.

3. For all $i \in [N]$ it compute $\alpha_i = f_{sk}(\beta_i)$ and $b_i = h_{pk}(\alpha_i)$. Divide the $b_i$ into blocks of size $5 \log_2(n)$ and for $j \in [n^6]$ set bit $r_j = 1$ if and only if all $b_i = 1$ in that block.$^5$

4. Finally let matrix $H$ have entries $r_j$ for $j \in [n^6]$. $\mathcal{P}$ proceeds as the (unbounded) prover in the previous protocol using $\alpha_i$’s as decommitments and adding $pk$ to $\pi$.

**Claim 12** The above protocol is interactive argument.

**Proof:** We first note that for all $pk$ in the range of $\text{Gen}$ it holds that $\Pr[H \text{ is good}] = \Omega(n^{-3/2})$ by Lemma 9 for each of the $t$ iterations. Thus a union bound over all outputs of $\text{Gen}$ implies that for any fixed $(pk, sk)$ in the range of $\text{Gen}$ the probability that $H$ is bad in all iterations negligible in $\lambda$.

Further, because $f$ is a OWP there is only one way to decommit the entries of $H$. Therefore by an analogous argument to the unbounded prover case, any $\mathcal{P}^*$ is caught with all but negligible probability.

**Claim 13** The above protocol is same-string computational NIZK with oblivious CRS sampling.

**Proof:** We describe algorithm $\text{FakeSetup}$.

1. Run the $\text{Setup}$ $t$ times to produce $\sigma'_i$ for $i \in [t]$ and sample $(pk, sk) \leftarrow \text{Gen}(1^\lambda)$.

2. For $i \in [t]$ if $H_i$ is bad (with respect to $pk$) then set $\tilde{\sigma}_i = \sigma'_i$.

3. For $i \in [t]$ if $H_i$ is good (with respect to $pk$) then replace the commitments to the HC in $H_i$ with commitments to a 0 graph. More precisely re-sample the appropriate $\beta_i$ conditioned on $b_i = 0$. Call the resulting CRS as $\tilde{\sigma}_i$.

4. Output fake CRS $\tilde{\sigma} = \{\tilde{\sigma}_i\}_{i \in [t]}$ and trapdoor $sk$.

The simulator $S$ can now use $sk$ to decommit the non-edges of $\delta(G)$ of the 0-graphs just as the simulator in the previous NIZK did. The only thing that remains to note is that the distribution of the fake CRS is indistinguishable from a real CRS even given a proof. This follows from the fact that $h_{pk}$ is a hardcore predicate for $f_{pk}$ and so the skewed distribution of the bits $b_i$ can not be detected by a bounded distinguisher.

Note that $\text{FakeSetup}$ did not make use of the instance to be simulated making a ZK protocol with oblivious CRS sampling. However the fake CRS is only computationally indistinguishable from the real one so the protocol is only computational ZK. On the other hand since we are considering only bounded provers the protocol must be same-string NIZK (see paragraph above “NIZK Arguments”).

$^5$In other words divide $\sigma$ up into $n^6$ blocks representing bit-wise commitments to bits that are 1 with probability $n^{-5}$ just like the CRS in the previous protocol.