6.897: Advanced Topics in Cryptography Feb 5, 2004

Lecture 3,4: Universal Composability

Lecturer: Ran Canetti *Scribed by: Yael Kalai and abhi shelat*

1 Introduction

Our goal in these two lectures is to prove the Composition Theorem that was presented at the end of the previous lecture, and to show how it can be applied it to the zero-knowledge functionality. As an example of the latter, we present the Blum Hamiltonicity protocol and discuss some issues related to parallel composition.

2 Composition Theorem

Let P be a protocol that securely realizes a function f. Let Q be a protocol in the f-hybrid model. Recall that the protocol Q^P is the composition protocol in which each call to f is replaced by an invocation of P, and P's outputs are treated as the outputs of f.

Theorem 1 (The Composition Theorem): *Protocol Q^P "emulates" protocol Q. That is, for any B-limited adversary A, there is a B-limited adversary H such that for any any Z we have:*

$$EXEC^f_{Q,H,Z} \sim EXEC_{Q^p,A,Z}$$

We refer the reader to the previous lecture notes for notations and definitions.

3 Proof of the Composition Theorem

Let A be any B-limited adversary that interacts with protocol Q^P in the real-life model. Our goal is to construct a B-limited adversary H that interacts with protocol Q in the f-hybrid model, such that no environment Z can tell the difference between the two interactions. We will construct H via the following three steps.

1. From A we construct an adversary A_P that interacts only with protocol P.
2. From the security of P, there is an adversary S_P in the ideal process for evaluation of f, such that for every environment Z, $IDEAL^f_{S_P,Z} \sim EXEC_{P,A_P,Z}$.
3. We use A and S_P to construct an adversary H and show that for every environment Z, $EXEC^f_{Q,H,Z} \sim EXEC_{Q^P,A,Z}$.

3.1 Adversary A_P

We will use A_P only when interacting with an environment Z, that gives A_P an input which consists of the internal state of A at the beginning of the round where protocol Q^P calls P. Thus, we will define A_P's behavior only with respect to such inputs. (If the input is of any other format then A_P halts.)

A_P, on input an internal state of A at the beginning of the round where protocol Q^P calls P, will run A from this state, while interacting with P. At the end of the run, A_P will output the current state of A.

From the security of P, we know that there exists an adversary S_P such that for every environment Z, $IDEAL^f_{S_P,Z} \sim EXEC_{P,A_P,Z}$.

3.2 Adversary H

Now we will use A and S_P to define an adversary H, which interacts with Q in the f-hybrid model.

1. Until the round where all parties in Q call f, run A. (Indeed, up to this point the protocols Q and Q^P are identical.)

2. At the point where Q calls f, run S_P with the current state of A as input. When S_P generates f-inputs for the corrupted parties, forward these inputs to f (Recall that S_P is an adversary in the ideal process for f). Finally, forward to S_P the outputs that the ideal process f gives to the corrupted parties.

3. Once S_P generates an output, which consists of an internal state of A (follows form the definition of A_P and from the fact that P securely realizes f), continue running A from this state.

4. Halt when A halts, and output whatever A outputs.

This Completes the definition of H.

3.3 Analysis of H

Our goal is to prove that for every environment Z, $EXEC^f_{Q,H,Z} \sim EXEC_{Q^P,A,Z}$. We will prove this by contradiction, as follows. Assume that there exists an environment Z that, on input z, distinguishes (with non-negligible probability) between a run of H with Q in the f-hybrid model and a run of A with Q^P in the plain real-life model. We will construct an environment Z^P that, on input z, distinguishes with the same probability between a run of S^P with the ideal process for f and a run of A_P with P. This will contradict the assumption that P securely realizes f.

The environment Z_P, on input z, will operate as follows:

1. Run Z on input z, and orchestrate for Z an interaction of parties running Q^P with adversary A, until the round where P is called.

2. At the round where P is called, start interacting with the external system, by giving the external good parties the inputs that the simulated good parties would give to P, and by giving the external adversary the current state of A.

3. Upon receiving the external outputs from the parties and the adversary, continue the simulated interaction between A and the parties running Q^P – The good parties use their outputs from the external system as the outputs of P, and A runs from the state given in the output of the external adversary.

4. When the internal outputs are generated, hand them to Z, and output whatever Z outputs.

Notice that if the "external system" that Z_P interacts with is an ideal process for f with adversary S_P then the simulated Z sees *exactly* the interaction with H and Q in the f-hybrid model. If the "external system" that Z_P interacts with is an execution of P with adversary A_P, then the simulated Z sees exactly an interaction with A and Q^P in the plain real-life model. Thus, Z_P distinguishes with the same probability that Z distinguishes.

4 Implication of the Composition Theorem

The main implication of Composition Theorem is that using the our notion of security *composes*. In other words, it enables us to design and analyze protocols in a modular way:

1. Partition a given task T to simpler sub-tasks $T_1, \ldots, T_k$.

2. Construct protocols for realizing $T_1, \ldots, T_k$.

3. Construct a protocol for T assuming ideal access to $T_1, \ldots, T_k$.

4. Use the composition theorem to obtain a protocol for T from scratch.

Thus far, we formally defined what it means for a protocol to securely realize a function f, and we proved the Composition Theorem, which shows that this definition has a nice composition property. We would like to demonstrate the usefulness of such a definition. As a first step, we will use this heave machinery to prove that zero-knowledge protocols compose sequentially, a result which is well known and relatively easy to prove directly.

In order to prove that zero-knowledge protocols compose sequentially, we will first define zero-knowledge in a "secure function evaluation" style, and show that it is essentially equivalent to the standard definition.

5 Zero-Knowledge Proofs [GMR89]

We begin by defining the notion of computational indistinguishability, which will be used in our definition of zero-knowledge.

Definition 1 *Distribution ensembles D and D' are computationally indistinguishable (denoted by $D \approx D'$) if for any polynomial time distinguisher A, for all c, d, and for all large enough k, and a with $|a| < k^d$, $Pr_{a,x \leftarrow D_k}[A(1^k, a, x) = 1] - Prob_{a,x \leftarrow D'_k}[A(1^k, a, x) = 1] < 1/k^c$.*

We next present a formulation of zero-knowledge protocols. Let $R(x, w)$ be a polynomial time relation.

Definition 2 *A zero-knowledge protocol for R is a protocol for two parties P and V, where P has (x, w) as input, V has no inputs, and the following holds.*

- Completeness: *for every pair* (x, w), $Pr[(P(x, w), V) = (x, R(x, w))] \sim 1$, *where* $(P(x, w), V)$ *denotes the output of* V *after interacting with* $P(x, w)$.
- Soundness: *For every possibly cheating prover* P^* *and for every input* z, $Pr[(P(z), V) = (x, 1) \wedge (\forall w\ R(x, w) = 0)] \sim 0$.
- Zero-knowledge: *For every possibly cheating verifier* V^* *there exists a simulator* S *such that for every* x, w, z, $S(x, R(x, w), z) \approx V^*_{P(x,w)}(z)$, *where* $V^*_{P(x,w)}(z)$ *denotes the output of* $V^*(z)$ *after interacting with* $P(x, w)$.

Remarks: This definition is syntactically different from the standard definition:

1. In the standard definition V gets x as input, whereas here V gets no input and instead it outputs P's input.
2. The standard definition of zero-knowledge doesn't make any completeness or secrecy requirement when $R(x, w) = 0$. In this definition it is required that both the completeness property and the zero-knowledge property will hold also in the case where $R(x, w) = 0$.

Note that these are only a syntactic differences and it is easy to translate a protocol from one definition to the other. In what follows, we consider an additional property, which is an enhanced form of the standard proof-of-knowledge property. The proof-of-knowledge property was extensively studied in the literature and was addressed in several different formulations. Here we will give a new formulation which is in the spirit of the "enhanced proof-of-knowledge" property of Lindell [Lin03].

- *Soundness (II) or Extractablity*: For any cheating prover P^* there exists a "knowledge extractor" E, such that for any z, we have $E(z) = (t, x^*, w^*)$, where $(t, (x^*, R(x^*, w^*))) \approx AT(P^*(z), V))$, and AT is the the sequence of exchanged messages in the execution of $(P^*(z), V))$ with the output of V.

The extractability property assures that in each accepting interaction, where $R(x^*, w^*) = 1$, it is possible to explicitly extract the witness w^* from the prover's input. That is, the extracted witness is the"appropriate witness" for this interaction.

5.1 ZKPoK as SFE

Let $R(x, w)$ be a binary relation. Consider the 2-party function:

$$F^R_{ZK}((x, w), -, -) = (-, (x, R(x, w)), (x, R(x, w)))$$

To interpret this function, note that both the Verifier and the Adversary have no inputs and both receive a theorem, x, and the verity of the theorem as represented by the binary relation $R(\cdot, \cdot)$.

Theorem 2 *A two-party protocol securely realizes* F^R_{ZK} *(with respect to non-adaptive adversaries) if and only if it is a zero-knowledge proof for* R *with the extractability property.*

Proof: The proof is given as a homework exercise. □

6 Blum's Graph Hamiltonicity

As a concrete example, we will work through the Blum Graph Hamiltonicity protocol presented in [Blu86]. Recall that a Hamiltonian cycle, H of a graph G is a path which visits each node of G exactly once. This problem is NP-complete.

The first tool used in this protocol is a *Commitment scheme*. A commitment scheme is a two-party, two-step process. In the first phase, known as the Commitment phase, the commiter C gives a value x to the reciever in an envelope. In the Decommitment phase, C opens the envelope to reveal the value for V. We would like to guarantee that at the end of the first phase, the verifier V has no knowledge of the value of x. After the Commitment phase, we would like to guarantee that there is only a single fixed value x to which C can successfully decommit. More formally, a tuple (C, D, V_c, V_d) of interactive Turing Machines is a $b$it commitment scheme if

1. **Completeness:** For all $b \in \{0, 1\}$

$$\Pr[(C(b), V_c) \to (s_c, s_v), (D(s_c), V_d(s_v)) \to (\cdot, b)] \sim 1$$

   Intuitively, the bit to which one commits is the same bit which is revealted via the decommitment protocol. The notation $(C(b), V_c) \to (s_c, s_v)$ represents the fact that the protocol defined by C, V creates outputs s_v, s_c for the Committing and Verifying ITMs respectively.

2. **Secrecy:** For all ppt machines V_c^*, we have that

$$(C(0), V_c^*)_{V^*} \sim (C(1), V_c^*)_{V^*}$$

   Commitments to one and zero are indistinguishable. Recall, that that the output of the interaction between $(C(\cdot), V_c^*)$ is a pair (s_c, s_v), and $(C(\cdot, V_c^*)_{V^*}$ represents V^*'s component of the output. That is, s_v from the left protocol is indistinguishable from s_v on the right.

3. **Binding:** For all C^*, D^*,

$$\Pr[(C^*, V_c) \to (s_c, s_v), (D^*(s_c, 0), V_d(s_v)) \to (\cdot, 0), (D^*(s_c, 1), V_d(s_v)) \to (\cdot, 1)] \sim 0$$

   The adjectives on C^*, D^* determine whether binding is perfect or computational.

The basic Blum protocol is based on the relation, $HC(G, H) = 1$ if H is a Hamiltonian cycle in the graph G. The common input to Blum's protocol is a k-node graph, G. The Prover knows a secret witness, H, which is a Hamiltonian cycle in G.

The protocol is as follows:

1. $P(r, G, H)$: If $HC(G, H) = 0$ then send the value (G, reject) to V. Otherwise, choose a random permutation, $\sigma \in S_k$ on k elements and send $(G, C(\sigma(G)), C(\sigma))$ to V. In this case, $C(\sigma(G))$ represents a bit-by-bit commitment to the adjacency matrix of the permuated graph, $\sigma(G)$.

$$P \xrightarrow{(G, C(\sigma(G)), C(\sigma))} V$$

2. V: If the input is (G, reject), then output $(G, 0)$. Otherwise, select a random bit b uniformly and send it to P.

$$P \xleftarrow{b} V$$

3. P: If $b = 0$ then decommit all of the commitments sent in the first message. This case checks whether the messages sent in the first round are well formed.

   If $b = 1$, exhibit a Hamiltonian path in $\sigma(G)$ by decommiting to the k edges in $\sigma(G)$ which form a Hamiltonian path.

$$P \xrightarrow{\text{decommit to } C(\sigma(G)), C(\sigma) \text{ or } C(\sigma(H))} V$$

4. V: If all of the commitment openings are accepted, and if $b = 1$ and the opened commitments form a Hamiltonian cycle, then accept by outputting $(G, 1)$. Otherwise, output $(G, 0)$.

We want to show that this protocol realizes the functionality F_{zk}^{HC}. However, the soundness error on this protocol is $1/2$ since a cheating Prover might try to guess what V will ask for and manipulate his messages accordingly. In order to achieve full soundness, and moreover, a proof of knowledge, we have to repeat the protocol several times. The hope is that repetition will preserve zero-knowledge and simultaneously diminish the soundness error.

Naturally, sequential repetition of this protocol works. Although this can be proven directly, we shall use the composition theorem. The three steps of the proof are to (1) Define a ZK with a weak proof-of-knowledge property and show that the protocol above satisfies it (2) Show how to realize F_{zk} in the F_{wzk}-hybrid model (3) and finally use the composition theorem to deduce security of the composed protocol.

6.1 ZP and Weak PoK

We take the first step in relaxing the notion of F_{zk} by allowing a soundness error of $1/2$. We formally define this weak zero-knowledge functionality F_{wzk} as follows:

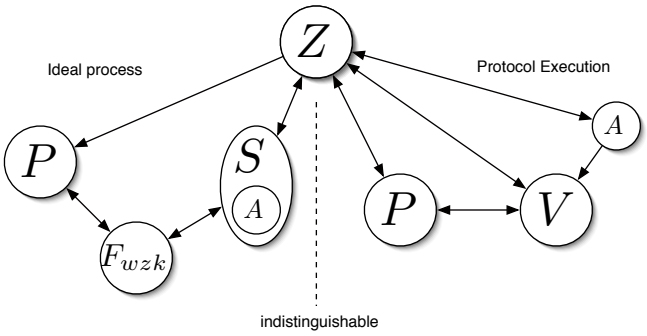
$$F_{wzk}^{R}((x,w),\cdot,c) = \begin{cases} (\cdot,(x,1),(x,1)) & \text{if } R(x,w)=1 \\ (\cdot,(x,0),(x,0)) & \text{else if } c = \text{ no cheat} \\ (\cdot,(x,b),(x,b)) \text{ where } b \leftarrow \{0,1\} & \text{if } c = \text{ cheat} \end{cases}$$

In this case, the third input corresponds to whether there is cheating on the Prover's side. If c indicates cheating, then the transcript passes one half of the time, and fails the other half.

Claim 1 *The basic Blum protocol securely realizes F_{wzk}^{H}.*

Proof Sketch: Let A be a B-limited adversary that interacts with the protocol. We need to construct an ideal-process adversary S such that no environment can distinguish between an execution in the ideal world with S or the real-world with A. Since there are only two parties in the protocol, there are only 4 types of B-limited adversaries. We consider each case below:

1. A controls the verifier (Zero-knowledge):



In this case, S receives the input z' from Z and runs $A(z')$. If the value from F_{wzk} is $(G, 0)$ then give (G, reject) to A. Otherwise, if the value from F_{wzk} is $(G, 1)$ then do:

(a) Choose a random bit b' and a random permutation σ.

(b) If $b' = 0$ then hand A the pair $C(\sigma(G)), C(\sigma)$.

(c) If $b' = 1$ then hand A the pair $C(Q), C(\sigma)$ where Q is a randomly chosen Hamiltonian cycle on k nodes.

(d) If A's response is the same as b' then open the commitments. Else, repeat this up to k times.

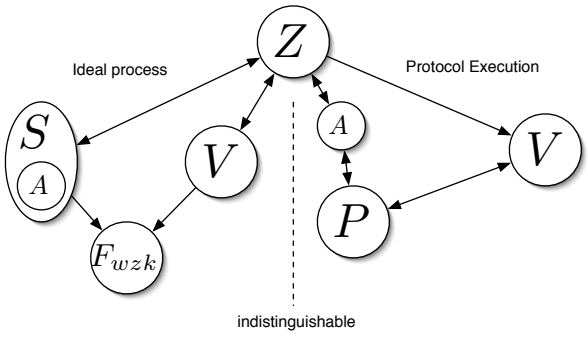
We must show that no PPT environment Z can distinguish between $IDEAL^{F_{wzk}}_{P,S,Z}$ and $EXEC_{P,V,A,Z}$. The reduction is based on the secrecy of the commitment scheme. In particular, an environment Z which can distinguish the two distributions can be used to construct a Z' which can distinguish commitments $(C(0), V)_V$ from commitments $(C(1), V)_V$.

Note that when $b' = 0$, the distribution of the IDEAL view is exactly the same as the distribution of the EXEC view conditioned on the Verifier's message being 0. However, when $b' = 1$, the Simulator commits to the adjacency matrix of a random k-cycle, whereas in the real execution, the Prover commits to the actual graph.

Therefore, any advantage that the environment demonstrates must come from the case when $b' = 1$. Let p_0 represent the probability that the environment Z outputs a 1 when the Simulator has $b' = 1$ and p_{k^2} represent the probability that Z outputs a 1 during a real execution with the adversary controlling the Verifier and $b = 1$. By assumption, $p_1 - p_{k^2} \geq 1/k^c$ for some constant $c > 0$ and security parameter k.

We invoke a hybrid argument to construct a distinguisher for $(C(0), V)_V$ and $(C(1), V)_V$. Consider the distribution over strings induced by $C(Q), C(\sigma)$ and $C(G), C(\sigma)$ respectively. Note that the Hamming distance between the adjacency matrix of G and the adjacency matrix of a randomly chosen k-cycle is at most k^2. Additionally, p_0 as defined above corresponds to the probability that Z outputs a 1 on the first distribution, and p_{k^2} represents the same probability on the latter one. As we incrementally replace the commitments in the randomly selected k-cycle with the same commitments from the graph, there will be one hybrid distribution for the environment Z in which the difference between the probabilities is at least $1/k^{c+2}$. The distinguisher Z' works by introducing the unknown commitment at this step, running the environment Z and echoing its output.

2. In the second case, A controls the prover (weak extraction). In this case, S gets the input z' from Z and immediately runs $A(z')$. S obtains the first message from A which will be of the form $(G, C(G'), C(\sigma))$. S then sends the bit $b = 0$ to A and obtains the opening of G'_0 and σ_0.



Then S rewinds A on the same random input and sends it $b = 1$, obtaining G'_1. If all of the decommitments succeed, then S constructs the Hamiltonian cycle and then gives (G, H) to F_{wzk} as the Prover and sends to the environment the output of A from either the first or second runs chosen at random. If only all of the decomittments for one value of b succeed then the S gives $(G, \cdot)$ to the oracle for F_{wzk} as the prover and sends the message "cheat" on the direct line. After receiving the output (G, b') from the oracle, if $b' = 0$ then S returns the output value from A during the faulty run. Otherwise, S returns the output from A on the successful run.

If in both cases, some of the decommitments fail, then S sends $(G, \cdot)$ to F_{wzk} as the prover, sends "no cheat" on the direct line, and hands Z the output from A on either of the runs chosen at random.

The key point here is that if A answers both challenges correctly, then Z expects V to always accept. If only one challenge is answered, then Z expects V to accept half the time, and if no challenge is answered correctly, then Z expects V to reject. Furthermore, Z expects to see the matching output from A.

S guarentees the same view for Z as in the execution except in the case when some commitment decommits to different values in the two runs of the A. This implies breaking the binding property of the commitment scheme.

3. A controls neither party. In this case, S obtains the output from F_{wzk}, generates a corresponding transcript of the protocol and gives it to A. Then S outputs whatever A outputs.

4. A controls both parties. S runs A and echoes its output.

The indistinguishability follows from the security of the commitment schemes.
□

6.2 From F_{wzk}^R to F_{zk}^R

In this subsection, we present the natural protocol for realizing F_{zk}^R in the F_{wzk}^R-hybrid model. The protocol is sequential repetition.

1. $P(x, w)$: Run k copies of F_{wzk}^R, sequentially. Send (x, w) to each copy.

2. V: Run k copies of F_{wzk}^R sequentially. Let (x_i, b_i) be the output from the ith copy. If all of the (x_i, b_i) values are the same, then output (x_1, b_1).

3. Else output nothing, or some default value $(x_0, 1)$ for some x_0 for which a witness w is known.

Let A be an adversary that interacts with the protocol in the F_{wzk}^R-hybrid model. We can construct an ideal-process adversary (simulator) S that interacts with F_{zk}^R and fools all environments. Again, there are four cases to analyze:

1. If A controls the verifier (zero-knowledge), all A sees is the value (x, b) k times. To simulate this, S simply takes the value (x, b) from the trusted party and outputs it k times to A.

2. If A controls the Prover, A should give k pairs (x_i, w_i) to the TP for F_{wzk}^R. The simulator S runs A and obtains the k pairs. If all of the x_i are identical and all of the w_i are valid witnesses or all are invalid witnesses, then S gives (x_i, w_i) to the TP for F_{zk}^R. Otherwise, S gives a default value (x_0, w_0) to the TP.

3. A controls neither party. As before, S receives output from the Trusted Party, forms it into a transcript and sends it to A. S echoes A's output.

4. A controls both parties. S echoes A's output.

We can also see that the views from Z are a perfect simulation. Note that all of the computational indistinguishability issues have been pushed into the realization of F_{wzk}^R and the composition theorem. The same kind of analysis works for other ZK protocols that have weak extraction. For example, consider the graph coloring protocol with soundness $1/n$. Hence, there is no need to reprove sequential composition for all of these protocols. You only need to prove that the single iteration securely realize F_{wzk}^R.

One might consider parallel repetition of the Blum Hamiltonicity protocol in which the verifier only accepts if all of the copies accept. For this, we cannot use the Composition theorem. Although intuition leads us to believe that parallel repetition is still secure, the standard proof fails because the old simulation now requires exponential time. In particular, for dishonest verifiers, the probability of guessing all k bits of the verifier's challenge is 2^{-k}. Therefore, the rewinding step of the simulation might require exponential time. Although it is unknown whether parallel repetition of this protocol is zero-knowledge (it is known, however, that it cannot be proven to be ZK using traditional black-box simulation unless $NP = BPP$), there are examples of ZK protocols which are no longer ZK when two copies are run in parallel. See Goldreich and Krawczyk [GK96b] and Feige and Shamir [FS92].

One fix to this problem of parallel repetition due to Goldreich and Kahan [GK96a] is to have the Verifier commit to his bits before the Prover sends her first message.

1. Prover sends a key to a perfectly hiding commitment scheme C'.

2. Verifier sends commitments to bits $C'(b_1), \ldots, C'(b_k)$.

3. Prover sends $C(\sigma_i(G), C(\sigma_i)$ for $0 \leq i \leq k$.

4. Verifier decomits to bits: $D'(b_1), \ldots, D'(b_k)$.

5. Prover responds as before using b_i.

6. Verifier accepts if all i responses are valid.

Although this protocol solves the problem with simulation, it introduces a problem with soundness. If P and V use the same commitment scheme, C, and C is malleable in the sense that given $C(b)$ there is an efficient way to generate $C(1-b)$, then a dishonest prover P^* can cheat. After receiving $c_i = C(b_i)$, P^* can generate commitments as follows : if c_i opens to a one, then generate a commitments c'_i which open to a Hamiltonian cycle and otherwise generate a commitment that opens to a permutation of G. Certainly the binding property of the commitment before does not preclude this. We need extra properties on the commitment scheme to work around this problem.

Another solution to the problem was presented by Brassard, Crepeau and Yung [BCY91]. In their scheme, they use "equivocable commitments" such that one you have a secret key, you can open a commitment in either way.

A recent scheme due to Alon Rosen [Ros04] offers a novel way around to achieve constant-round zero-knowledge with a much simpler analysis.

References

[Blu86] M. Blum. How to prove a theorem so no one else can claim it. In *Proceedings of the International Congress of Mathematicians*, pages 444–451, 1986. Berkeley, California.

[BCY91] Gilles Brassard, Claude Crepeau, and Moti Yung. Constant-round perfect zero-knowledge computationally convincing protocols. *Theoretical Computer Science*, 84(1):23–52, 1991.

[FS92] Uriel Feige and Adi Shamir. Multi-oracle interactive protocols with constant space verifiers. *J. Comput. Syst. Sci.*, 44(2):259–271, 1992.

[GK96a] Oded Goldreich and Ariel Kahan. How to construct constant-round zero-knowledge proof systems for NP. *J. Cryptology*, 9(3):167–190, 1996.

[GK96b] Oded Goldreich and Hugo Krawczyk. On the composition of Zero-Knowledge Proof systems. *SIAM Journal on Computing*, 25(1):169–192, 1996.

[GMR89] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof-systems. *Siam Journal of Computing*, 18:186–208, 1989.

[Lin03] Y. Lindell. General composition and universal composability in secure multi-party computation. In *44th Foundations of Computer Science*, pages 394–403, 2003.

[Ros04] Alon Rosen. A note on constant-round zero-knowledge proofs for NP. In *Theory of Cryptography Conference*, pages 191–202, Feburary 2004.