

[SQUEAKING]

[RUSTLING]

[CLICKING]

Yael Kalai: OK. So should we continue?

Audience: [INAUDIBLE]

Yael Kalai: OK. So where are we? So we did a little detour to zero knowledge, and we even showed a zero-knowledge proof. Now, I know I did it very, very hand-wavy. I actually didn't plan to really teach zero knowledge, even though it's hard not to because it's so beautiful. But what I did want to-- the reason I'm pointing out this-- I mean, we will use it. We will touch zero knowledge some more next time. But this protocol is at least a soundness $1/2$.

Now let's see what happens if we apply Fiat-Shamir to it. And let's say we apply even the random oracle model, Fiat Shamir in the random oracle model. Because remember, we started by asking-- the Fiat-Shamir, is it sound? We don't know. And then we said, you know what? But if instead of actually a hash function, you use random oracle-- like, a completely random function-- then of course it should be sound. Because, intuitively, what does it matter if the prover is talking with the verifier or is talking with the hash function?

So if it was sound when he was talking to the verifier-- the verifier sends him random bits-- it should be sound when you're talking to the hash function because the hash function also sends a random bit-- almost, almost true, but not. Why? Let's look at this example. I'm going to tell you how I can easily cheat in the random oracle model.

So I have a G . It's not in the language. It does not have a Hamiltonian cycle. What I'm going to do-- I'm going to hope for the best. What do I mean, hope for the best? I'm going to guess your b . I'm going to say, you know what? Let me try to guess what the b of the hash function's going to be. I don't know. It's a random function. Let me guess it's 0.

So because it's 0, I know I'm going to be asked to open on the cycle. So I'm going to commit to a random cycle. This is some string, α . I'm going to go to my hash function, give him x and α , my G and α . And I get back a bit b . If it's 0, huh. I open, I'm happy. If it's not 0, I'll try again.

Again, I'm going to choose a random-- I can always choose 0, or I can just choose kind of a random bit. If it's 1, I'm going to-- I can commit to the all 0's. Because anyway, I just need to open 0's. I'll just commit to the all 0's. And then when you open, I give you π , and I'll open the non-edges, seeing it's 0.

The point is I can rerun you. In the actual protocol, yeah, with probability $1/2$, I win. But I can't tell you, OK, let's try. Oh, didn't work, let me try again. You say no, you failed. You don't get to try again. I know that if you try again, again, eventually, you'll win. So you got one try. But with a random oracle, I have the oracle, so I can abuse it. So I can try many, many times. And that's how I break.

OK, so what did we learn? Let's go back to the random oracle model. So first, it's not secure or not sound if the protocol has non-negligible soundness.

If you can cheat with probability non-negligible, you're going to try, try, try until it works. And then you present that as a proof. So that was Leo's question. And the answer is, it's not sound.

OK. So then you say, fine. Is it sound if the protocol has negligible soundness? And the answer is no. OK, so also not sound if protocol has negligible soundness.

OK. That seems a bit sad, but let me give you the counterexample. So let's go back to our beautiful zero-knowledge protocol here. And it has only soundness $1/2$. That was kind of the complaint, right? Fine. Let me make the soundness negligible, even $2^{-\lambda}$ for you.

OK, how? Let's repeat it sequentially λ times. OK, so we have a G . I send you a commitment of, supposedly, a random cycle. You send me a bit, I open accordingly. But then you're like, OK, OK, wait, I'm not convinced. Let's try again. Fine. I send you another commitment. You send me another random bit. I reveal. Then you say, no, no, no, still not happy. Let's do it again. And we do it λ times.

Now if all the time, every single time, I'm accepted, you're pretty convinced. Now, the probability that I can cheat is $1/2$ to the λ because every time with probability $1/2$, you catch me if I did all of them correctly. So it's very negligible. It's exponentially small. Now, I want to argue, I can cheat in the random oracle model. How? Exactly the same thing. How do I cheat? First, do not have $\alpha_1, \beta_1, \gamma_1, \alpha_2, \beta_2, \gamma_2, \alpha_3, \dots$.

Let's start. I'm the cheating prover. I try. I guess β_1 is going to be 0. α_1 , no, fine. I work hard, I get my $\alpha_1, \beta_1, \gamma_1$. Good. Now α_2 -- same thing. I guess β_2 . But I keep the first transcript. I don't toss it away. I keep the first transcript. And then I guess β_2 . If I guess correctly, good. If I didn't, I'm going to redo, da, da, da.

And after a constant number of times, I'm going to get the second transcript-- probably after twice, but whatever. And then I always keep the first-- I don't know-- as many transcripts as I had, and I extend it. So I can convince you. Yeah?

AUDIENCE: What if I do parallel repetition things?

Yael Kalai: Ah, very good, very good. So the question is, what about parallel repetition? OK. Yeah?

AUDIENCE: This gradual thing is fine if you hash the entire previous transcript. Like--

AUDIENCE: No, because you can just--

Yael Kalai: No.

AUDIENCE: No?

AUDIENCE: You can just get a new--

Yael Kalai: Good.

AUDIENCE: --commitment until you're happy.

AUDIENCE: Ah, that's [INAUDIBLE].

Yael Kalai: [LAUGHS] Yeah. So let me emphasize, actually-- it's a good point that you're making. Note, when we apply Fiat-Shamir, I'm applying the hash. And what I said is, oh, I do the same exact attack. I get one protocol. Like, I get one transcript, another, da, da, da, lambda transcript, and I'm done.

Note that every time, I hash everything from the beginning. Like, the hash is an α_1 , β_1 , γ_1 , α_2 , β_2 -- but so what? I'm fine. Let's say I get the first-- let's say three transcripts worked out. So I have α_1 , β_1 , γ_1 , α_2 , β_2 , and α_3 . Now I'm in the fourth.

I guess β_4 . I choose α_4 accordingly. I hash everything from the beginning, like the Fiat-Shamir hash, and I hope to see my β_4 . I see it, I'm happy. I can compute γ_4 , and I continue. I don't see it, I try α_3 again. I don't hash my first two that I already did successfully. I, every time, extend. And the point is, I can always extend in constant overhead kind of thing. Yeah?

Audience: Shouldn't that be hard for this bounded power?

Yael Kalai: So the runtime-- no. OK, so the question-- look, how long does it take-- I'm a bounded, cheating prover. How long does it take me? Every transcript, it takes me-- on average, I do constant number of tries, an average of two tries. So it's actually quite efficient. I can generate my transcript quite fast.

Because how do I generate it? Again, I have G . It may not have. It may have, it may not. It doesn't have a Hamiltonian cycle. I don't even know if it has a Hamiltonian cycle, neither do I care, because I'm never going to use it. What do I do? I guess β_1 randomly.

Very efficiently, I commit. Because if it's 0, I just commit to a random cycle. If it's 1, I commit to all 0's. That's very easy. And then I apply the hash. With probability $1/2$, I got the β_1 that I chose. And then I can very easily compute my answer, γ_1 , which is just a trivial opening. And I only need to repeat it-- like, it's only bad with probability $1/2$. So, overall, I won't need to repeat the same one too many times.

Audience: So that's where the non-soundness comes from, from possibly equal sign [INAUDIBLE]?

Yael Kalai: That is where?

Audience: It's [INAUDIBLE]--

Yael Kalai: That's why this is not sound. Exactly. So I have negligible soundness, but still it's not sound. Exactly. OK, so this is bad. But that's the only time when it's not sound. That's it. Everything else is sound. So that goes to the question about parallel repetition. So now let me actually give you the theorem. Let's see.

So the theorem is that the Fiat-Shamir paradigm is sound-- finally is sound-- in the random oracle model if the underlying protocol is-- first, it has negligible soundness and constant round.

So if it has negligible soundness and it has constant number of rounds, then it's sound. Before, I got negligible soundness by repeating-- getting many, many, many rounds. Yeah?

Audience: In public [INAUDIBLE]--

Yael Kalai: Oh.

Audience: --we run--

Yael Kalai: Yeah. Otherwise, you can't even apply-- if the public coin. Yeah. If it's not public coin, you can't even-- like, completeness, it doesn't hold. So I'm always talking about public coin protocols.

But let me say, this is not a characterization. This is not an if and only if. We do have protocols with many rounds for which the random oracle model does apply-- in particular, GKR. So GKR is many rounds. So can we reduce interaction in GKR? And the answer is, yes, we can, even though it has many, many rounds.

And the reason why we can do it is because it's sound also-- or, more generally, if the underlying protocol has what's called round-by-round soundness. I'll define it later, maybe not today.

And any such protocol has round-by-round soundness. But this is more general. And GKR has round-by-round soundness. So this is kind of in parentheses because we're not going to talk about it today. Maybe we will. We'll see. OK, but let me first prove this.

OK. So why is it-- let me do it here. OK. So let's do the proof. So take a protocol, a public coin protocol with constant number of rounds that has negligible soundness. And I'm going to prove to you that it's sound in the random oracle when I apply Fiat-Shamir on the random oracle model.

OK, So fix any protocol. Let's call it P, V . Let's begin-- let's assume-- just for simplicity of notation. You'll see how it extends-- that it's three messages. So suppose P, V has three messages and negligible soundness. We'll later see how to extend it to more rounds.

So P, V -- let's say there are some x . The question is in the language or not. And they both have-- so before the hash, there's like, α . He gets back a random β . He gets back γ . And he chooses to accept or not. OK. Now suppose there exists a poly-size adversary P^* that cheats. I want to argue that then it means I can cheat in the original, interactive protocol.

So suppose there's P^* and x not in L such that the probability that P^* with H and V with H -- this is kind of the Fiat-Shamir-- it accepts with probability at least, let's say, ϵ .

I'm going to argue that then I can cheat in the original scheme with probability a little less than ϵ but not too small. OK. So if there exists a cheating prover that can cheat in the Fiat-Shamir version, but in the random oracle model, then I can use them to cheat in the original scheme.

OK, so how? So first, note, because we're a negligible probability, this is large. So β is public coin. It's random in-- I don't know-- let's call it k . But k 's super-logarithmic. Because there are only polynomially many options, on one of them, I can cheat. So this is not statistical sound-- sorry. Erase what I said. Let me try again.

So these protocols-- on one of them, I can cheat. Like, for any α , I can always find a β for which I can cheat. Otherwise, if I can't cheat on any β , there's no point in doing interaction to begin with. I'll just give you α and a fixed bit. I'll give you γ , and that's it.

So in these protocols, there's always β s for which I can cheat on. And therefore, because I have negligible soundness, β must come from a super-polynomial universe. If it's only polynomial, then I can play the same game that I did before, kind of hope to succeed.

It's like a-- too bad. In Hebrew, we have a very good saying for this. Like, when you cheat, the hope of not being caught-- there's a word for it.

[LAUGHTER]

But yeah, Americans are too-- they don't cheat, so they don't have words for this.

AUDIENCE: Would be like a con or some--

Yael Kalai: What?

AUDIENCE: Like a con? It would be like--

Yael Kalai: Is that a word? OK.

AUDIENCE: Yeah.

Yael Kalai: Con?

AUDIENCE: As in a confidence man or like--

Yael Kalai: Oh, I see.

AUDIENCE: --you are lying to someone.

Yael Kalai: Exactly.

AUDIENCE: Like--

Yael Kalai: Yeah.

AUDIENCE: Could not be con.

Yael Kalai: Exactly, exactly. Yeah. OK. So suppose there exists--

AUDIENCE: Fake it til you make it.

Yael Kalai: Fake it til you make it, exactly. Totally.

[LAUGHTER]

OK, so suppose there exists a cheating prover. Now, how does he cheat? Let's see what this prover does. This prover gives-- so this prover outputs some α , β , and γ , like a transcript. And this β is hash of x and α . That's how he cheats. That's a valid transcript. That's a transcript that will-- and such that the original verifier accepts.

So in the Fiat-Shamir, the prover, he just outputs this. And the verifier checks that β is indeed H of x and α and that this is a valid transcript. OK. Now, the cheating prover, he sends oracle queries to his hash function. Now, this prover is bounded. He's polynomially bounded.

Let's say he sends-- so suppose that P^* sends q queries to the oracle, to H . He's bounded, so he runs in poly time. So what does he do? He has oracle access. He is trying to cheat for an x .

Now, he can ask his oracle, OK, here's alpha 1. Give me beta 1. Here's alpha 2. Give me beta 2. He says at most polynomially many queries because he's polynomially bounded. So there's some polynomial, q , queries that he sends to h . And let's denote them by alpha 1 up to alpha q .

OK. Now, first thing, note that the probability that he succeeds in generating alpha, beta, gamma so that alpha is not one of the queries is negligible. If he didn't ask, he doesn't know what beta is. The only way he gives beta, which is the right H , is by asking. He has no idea what beta is. And so if he didn't ask, he can guess with probability, like, $1/2$ to the k .

OK. Let me-- maybe I'll go here, continue here. So first thing, note, the probability that alpha, beta, gamma are accepting with x -- so meaning the verifier accepts them-- and alpha is not alpha 1 up to-- it's not one of them-- this probability is negligible.

And this is because k -- yeah, it's something like, $1/2$ to the k .

AUDIENCE: The x over [INAUDIBLE]?

Yael Kalai: What?

AUDIENCE: x alpha--

Yael Kalai: Ah, sorry, alpha q . And, sorry, what did you say?

AUDIENCE: Like, x alpha has to be one of the queries, not just alpha.

Yael Kalai: Oh, no. Sorry. Oh, x alpha, yeah. Right. But alpha needs to be one of the alpha 1's. Sorry, yeah. So, OK, he sends kind of x comma alpha 1 up to x comma alpha k . That's kind of what he sends. I mean, he can send other ones but then--

AUDIENCE: They shouldn't help.

Yael Kalai: Yeah, it shouldn't help him. So, OK, it should be one of the queries he asks, first of all. Because if he didn't ask that, he has no idea what the beta is. So he guesses the beta correctly with probability $1/2$ to the k . That's negligible. OK. So this means that if he succeeds with probability epsilon that's non-negligible almost with this probability, or with probability epsilon minus negligible, he doesn't only succeed, but he succeeds with alpha that he queried.

So now, because I know that he succeeds with alpha that he queried, I'm going to use him to cheat in the interactive protocol. Again, note, I'm really abusing the random oracle model. Because I'm assuming that the verifier, really, the only way he interacts with the oracle is by kind of queries, and I know the queries. And I can use it to cheat.

So let me now construct P^{**} that cheats in the interactive, in the interactive protocol. OK. How do I cheat in the interactive protocol?

OK. So first, I'm going to give the same x -- with x , the same x . So this P^* succeeded with some x in the language. I'll use the same x . Now, first thing I need-- so here I am, P^{**} . I'm ready to go. Here's x , V . I need to start sending alpha. What do I do? Which alpha do I send? Now, what do I know? I know that-- I want to use this guy. I can't cheat. I'm going to use him. He's the cheater. I'm going to use him.

What do I know about him? I know he cheated with α , β , and γ . And one of these α s he used as a query. OK, which one? I don't know. But here's what I'm going to do. I'm going to guess randomly. And what's the saying again? You fake it til you make it.

[LAUGHTER]

I'm going to guess randomly and hope. So what I'm going to do, I'm going to guess-- so what does P^* do? First, I'm going to guess a random i in q . So just the index, which-- so I know he's going to make q queries. I don't know yet which queries. But let's say I have a bound on his runtime. So I have a bound.

So P^* there runs in time at most q , let's say. So he makes at most q queries to the oracle. That, I know. I'm going to choose a random index. Like, suppose we have random query of one of-- the i -th query to his oracle.

And what I'm going to do, I'm going to send here α_i . Now, what is α_i ? I don't know what α_i is. But don't worry. What I'm going to do-- I am going to simulate the oracle for him. So I'm going to pretend I'm the random oracle. OK. I'm going to start by saying, OK, P^* , I'm going to use you. Let's start. You gave me x . Send me α . Generate this. Generate α , β , γ .

Now, what does P^* do when he generates? He's like, OK, let me think. α_1 , I'll give him a random β_1 . α_2 -- or x , α_2 , I'm going to give him a random β_2 . α_3 , random. α_4 , random-- until α_i . Once I get α_i , I'm happy. I'm like, ah, α_i . I'm ready, α_i . I get a random β back. What do I do? I tell him, oh, your answer from the oracle is β .

OK, so I'm simulating P^* 's oracle. And I'm behaving like I'm the oracle sitting on top of P^* . It gives me α_1 . And then, sure, here's a β_1 . That's your answer. α_2 , β_2 . α_3 , β -- α_i for random i . It's β . Now he gives me γ . OK. Now he gives the answer.

And I continue. $\alpha_i + 1$, random. $\alpha_i + 2$, random, random, random, random. OK. At the end, he gives me a transcript. But with probability $1/q$, it's with my α_i . If he use both my α_i , hooray. I'm going to give the γ_i . If it's not my α_i , I'm-- well-- going to try again. I mean, then I fail.

So what is the probability-- so that's my protocol. So if he didn't cheat with a γ_i with the α_i -- if this α is not α_i , I'm going to lose. I'm going to say, sorry, I failed.

What is the probability-- so guess, and then simulate the oracle for P^* randomly, except for α_i -- I simulate via the verifier. For α_i , answer with β . That's all.

And then if α , β , γ is such that α equals α_i , then-- let's call this γ -- then send γ . That's the protocol. That's the cheating prover. That's P^{**} .

Now, what is the probability that P^{**} is accepted? Again, what does P^{**} do? P^* stars accept with probability ϵ . So with probability ϵ , he gives me α , β , γ . But he chooses α . He chooses this after querying the oracle q times with different α s.

Let's say with all of them, he uses the same x . Otherwise, he's stupid. But whatever. It doesn't really matter. We established that, except negligible probability, one of these α is in the i -th query. So what is the probability that I succeed?

AUDIENCE: That probability q [INAUDIBLE].

Yael Kalai: Very nice. Great. So the probability-- I'm out of space. So the probability that P^* is accepted is at least the probability that P^* accepted over q . He's accepted if he chose α correctly. But it's random.

One of them, he cheated with, and I choose it randomly-- so with probability at least $1/q$ -- and minus the negligible, which is minus $1/2$ to the k , which is kind of the probability that he didn't even guess-- the α is kind of-- he just didn't use one of the queries. But that's really small, so it's still pretty big. So there is a loss. There is a loss of q .

But q 's polynomial. And if α is non-negligible, then α over q 's also non-negligible. And that's a contradiction. Any questions? Yeah?

AUDIENCE: So the [INAUDIBLE] proof where we actually use the random oracle model already, where we couldn't just put in another hash function, is like, that top line there?

Yael Kalai: Yes, it's this top line there. But also, the fact that we can say that there exists α 1 up to α q -- the fact that we're able to observe what he knows about the hash is in the random oracle model. So the proofs in the random oracle model, it really means one needs to be careful. Because people want to--

OK. The intuition behind the random oracle model is to say, what do we need from the hash function? We want it to look random. Because the verifier's message are random, we want also to look random. And now, OK, because we look-- and that was kind of the motivation for this random oracle model.

And then it's tempting to say, so now all we need is a hash function that kind of behaves randomly. Let's try to use, I don't know, like, people say-- like pseudorandom functions, which were not defined here. Whatever. OK, but you can. But who knows what the-- how will you prove soundness?

This proof-- unless, really, the prover, what it does is really just runs the function. If the cheating prover, all he does-- he gets the function. He's like, ah, this really looks like crap. I don't know what to do. I'll just feed him inputs, see what output I get. That's the only thing I can do. Then this proof in the random oracle model makes sense. But as long as he can do anything else besides, like, blindly feed an input and an output, I don't have a proof. Yeah?

AUDIENCE: What if you have one with [INAUDIBLE]?

Yael Kalai: Oh, $1/2$ to the k ? Yeah, the only reason I added this negligible fraction here is, there's $1/2$ to the k probability-- which is tiny-- that P^* star is accepted with none of the α s. So maybe he just chose α that was not queried at all. But he managed to choose β randomly, which happens with probability $1/2$ to the k . Because β should be H . And if he didn't ask H , then he doesn't know what β is, and β is random. But there is that probability that he chooses α that's not one of the q .

So here, if he chooses one of the q , I guess the one he chooses correctly with probability $1/q$, so I get ϵ over q . But I guess I should put here-- yeah. But if he chooses randomly, then I'm out of luck. Yeah?

AUDIENCE: How do we know q ?

Yael Kalai: Good. How do I know q ? Very good question. I actually don't know q . And the q even can be a little bit adaptive. So maybe this cheating prover he gets, he sends α_1 . And depending on β_1 that he gets, he decides how many queries he wants to-- and maybe he's very happy with his β_1 , and he's done. And maybe he's very unhappy and sends another.

So the number of q can be adaptive. But since P^* is fixed and I know it's polynomial time-- so let's say the runtime of P^* is λ^3 . I know that he cannot ask more than λ^3 queries, because each query takes one unit time to ask. So I can upper bound the number of queries by the runtime of P^* .

So really, the probability that is accepted-- if I have no idea what q is and in the worst case, it's the runtime, it's ϵ over the runtime of P^* -- which that, I know. Because I'm giving P^* , let's say, as a circuit.

Audience: Wouldn't you have a circuit that doesn't halt or--

Yael Kalai: No. I'm promised that P^* 's polynomial size. I know it halts. So my contradiction assumption is as follows. Suppose there exists a poly size. Poly size-- I don't know-- λ^3 size P^* . Now, I know that P^* asks at most λ^3 queries.

Audience: [INAUDIBLE]

Yael Kalai: I think of the circuit of P^* as being a circuit that has kind of H gates, like random oracle gates. So you can feed it, like, α or x and α . And out comes-- and it's like one gate. It's like one time unit. But he cannot have more than λ^3 of them because that's the size of the circuit.

Audience: OK. So you could necessarily not have loops or--

Yael Kalai: No, it doesn't-- no, no, no. It doesn't have loops. Yeah, no. The runtime is fixed. It can compute at most λ^3 gates. It has λ^3 gates in it.

Audience: Right. I see.

Yael Kalai: So any other questions? OK. This was three rounds. What about more? So what if you have, I don't know, c rounds? What happens then? Any guess in how the probability decays with the number of rounds? Yeah?

Audience: ϵ over G to the $c/2$?

Yael Kalai: Exactly. Yeah, exactly. So it's going to be-- if you have c rounds-- or c messages from the verifier, let's say-- if you have c message from the verifier, it's going to be something like ϵ over q to the c . And to see why-- it's really the same argument. What we need to do is suppose that in each round, P^* sends at most q queries.

So maybe he sends q queries of the form α_1 , like x comma α_1 . And then he sends q queries of the form x alpha β . And then he sends-- sorry-- x alpha, β , γ . And then he sends q queries of the form x , alpha-- so for each partial transcript, he has q queries.

Now, if you have c rounds, you need to guess correctly for each partial transcript the correct i . I'm going to guess that-- so I'm not going to write it down because it's really the same, but let me just say it in words. P^* , what he does is, for every round of the verifier, he guesses the index of the query of-- he gets the index of the query to the hash function, which is going to be used in the proof.

So he's going to guess. And here, we said you guess i . Now we're actually going to guess i_1 up to i_c . OK. For each message of the verifier, we're going to guess an index. And now we again just emulate P^* , exactly as before.

So how do I emulate? I go to P^* . He asks me just α_1 -- like, if he asks me just of the form for the first message-- like α_1 , α_2 , α_3 -- I give him random, random, random, random, random until he asks for α_i , and then I give β_i .

Now, in between, he may ask for bigger. He may ask all of a sudden for x comma α , β , γ -- oh, random, then random. I give everything random. But the first point, I get α , the i -th query for the first-- corresponding to the transcript, the first message. Then I'm like, aha, I got it, α_i . Thank you very much. I put my α_i . I get β . I enter β as the answer from the oracle. And now I'm happy.

Now I start counting. Like, for i_2 , I start counting, how many messages are the form α , β , and some γ ? α , β , γ_1 . α , β , γ_2 . α , β -- until I get to i_2 . All of them, I answer randomly. And any other queries for that I don't know what from-- random.

Once I get to α , β , γ i_2 -- like, the i_2 -- I'm happy. I run here. I put my γ , I get a δ , I answer with δ . Now I start counting, how many queries of the form α , β , γ , δ ? ϵ . Sorry, ϵ 's fixed-- next one. I count until I get i_3 . And the i_3 of them, I give the answer, give-- I just do it for as many rounds as needed.

But for me to be solving, I need to make all these guesses correctly. And each guess is going to be probability q . So that's why the probability decays a lot. Yeah?

AUDIENCE: So does that mean that the number of rounds is not security parameters? You're still secure?

Yael Kalai: Yeah. Good, good, good-- no, it's not. OK. If the number is \log , it's almost secure. But then the probability is q to the \log . q is not 2. So you'll get a little negligible. So it will be negligible, but not too negligible. It's going to be kind quasi-poly, like 1 over quasi poly.

So if you have a \log rounds, if you happen to know that the original interactive protocol has soundness less than quasi-poly, like subexponential-- let's say you have the guarantee that a cheating prover cannot cheat except with 1 over subexponential probability. Then \log rounds will be good.

So the number of rounds, c , this is really your probability of cheating. And as long as your interactive protocol has the guarantee that you cannot cheat with this probability, then you have a proof. Yeah. Yeah?

AUDIENCE: Would you replace the formula q to the power of c with P over c to the power of c so you distribute q queries?

Yael Kalai: Good, good, good. Yeah, so the question was, can we maybe replace this by q over c to the power of c because q is the total number of queries? The thing is-- so, OK, I'll tell you the-- it's not that simple. Because we don't know exactly how many-- so suppose the prover, total, asks q queries.

We don't know how many queries he-- maybe he has all of them on the first message and then just one. And it can be adaptive. It can depend on his answers. Maybe he asks α_1 . He gets β_1 . He's happy. He's not going to ask again. And all the answers are going to be in the second messages. Or maybe he sends α_1 , β_1 . He's so unhappy, all the q goes for the first message.

So how do you choose the i ?

AUDIENCE: Now they choose [INAUDIBLE].

Yael Kalai: How do you choose-- for me, I said, let's choose them randomly.

AUDIENCE: Oh, I see.

Yael Kalai: But you may be able to do better. One can think, can you do better than this? Possibly. Because you're saying, forget about the end, end, end. The end, it's not going to-- most of them, you won't go all the way to the end, so you can probably choose it more from the beginning. Probably, you can improve on this. I didn't do the calculations. Because usually, we don't really care about constants. But you're saying-- oh?

AUDIENCE: [INAUDIBLE] because if you make c equal to q , then it's going to be [INAUDIBLE].

Yael Kalai: That's why it's not true in general. In general, you can't. But you probably can improve it a little bit, you know what I mean? Yeah, maybe you want to choose it in a different distribution, not randomly, eh-- more towards the beginning because most of them are not going to be at the end. But yeah, there are probably optimizations that can be done. Great, great questions. Any other question?

OK. So let's see. So maybe I will do then-- I'm debating if to go to the soundness in the standard model or to do the round by round. You know what? I'll do the round-by-round soundness because I want to--

AUDIENCE: So this says Kilian.

Yael Kalai: Oh, good. Yeah, Thank you. OK, good. So what did we-- thank you. What did we prove? We proved that the random oracle model is sound. The Fiat-Shamir applied to the random oracle is sound if you have constant number of rounds and negligible error. Now, Kilian-Micali protocol has constant number of rounds-- four, to be precise-- and negligible error if you take the P , c , P with negligible error, which you can by-- we have P , c , P 's with negligible error.

You put it in and you have four round, negligible error You apply Fiat-Shamir in the random oracle model. It's sound. So Fiat-Shamir is sound in the random oracle model for Kilian-Micali. That's great. As I said, we don't have-- so now you can ask, do we have a proof of Fiat-Shamir for Kilian-Micali in the standard model? OK, random oracle model, we have. How about standard model?

And this is important because, for a long time, people believed that if we have a proof in the random oracle model, it means there must be some hash function for which this protocol is sound. Namely, the belief was, if you have a proof in the random oracle model, and then if you use it with Fiat-Shamir, something goes wrong-- like, someone breaks it-- we need to blame the hash function. It's that you used a bad hash function for Fiat-Shamir.

OK. So typically, when Fiat-Shamir's used in practice-- it's used, let's say with SHA-256. That's a standard, off-the-shelf hash function that's used. And now if someone ever breaks your protocol, the conclusion was, SHA-256 is not a good hash function. Use another hash function. However, this protocol is an example of a protocol-- it's very natural. It's great. It was invented not as a counterexample for Fiat-Shamir.

And it's secure in the random oracle model. So Fiat-Shamir is secure in the random oracle for this protocol. Yet, there are hash functions contrived-- but there are contrived hash function such that this protocol is insecure no matter which-- it's not secure in the Fiat-Shamir for any explicit hash function that you use. It's a good random oracle model, but it's not secure with any actual function family.

So what we learned from this is, the random oracle model is a nice sanity check, but it's really not a proof. I mean, we knew it's not a proof but it doesn't even guarantee-- it doesn't guarantee that there exists a proof. It's just a minimal sanity check.

Yet, let me tell you, in practice-- actually, that's how people check that things are secure in the random oracle model. If it verifies this check, they use it. And, as far as I know, things never broke because of the random oracle model being insecure. So it's more of a theoretical kind of interesting thing. But actually, in practice it's not an issue. Yeah?

AUDIENCE: Has anyone tried to break the random oracle's [INAUDIBLE]?

Yael Kalai: OK. The question-- is anybody trying to break? The answer is-- OK. Things are broken all the time. Things are hacked and broken all the time. It's very rare that what's broken is actually the crypto. It's not the crypto that's broken. But people do try to break-- for example, one thing that, often, people break-- they use to break is-- our cryptography uses randomness. We use randomness in our crypto.

And in practice, people don't use-- like, we keys. We assume our secret keys are random. But in practice, where do you get this randomness? And often, the implementations are not done well. And somehow, guess what? Things turned out not to be random or not random enough, and then things are attacked. So things are attacked a lot, actually, in practice. There's a lot of breaches, security breaches. But it's very rare that it's in the crypto itself.

Now, why is it not that in the crypto? One would believe that just because the crypto's actually secure and it's implementation that are the problem. And how many people are trying to attack the crypto? It's very hard to say. But my guess is that-- and let me say, there's a lot of people who are working really hard to attack the crypto. But these are done behind sealed doors so-- I remember, though, there's a fantastic guy [INAUDIBLE] a student in our community called-- his name is Joe Killion. And he was a professor, and he moved to NSA.

And I remember once I asked him-- a long time after he was already at NSA, I asked him, so, do you guys know how to factor?

[LAUGHTER]

And he told me-- his answer was, no, the worst part about my job-- I'm not allowed even to joke about these things. So the answer is, we don't know.

AUDIENCE: You don't know if they don't know.

Yael Kalai: Yeah, exactly.

[LAUGHTER]

OK. Any other questions? Yeah?

AUDIENCE: This is just kind of general. But in general, do you think private-- like, the NSA and private cryptography is further than what is publicly known or--

Yael Kalai: The question was, do I have a guess?

AUDIENCE: Yeah.

Yael Kalai: And so I do not know. So let me rephrase. Your question is, do I have a guess as to if the NSA knows more than we do? My guess would be yes, because-- they know everything we know because we don't hide anything. And--

[LAUGHTER]

And they have smart people. So why won't they know a little more? Do they know how to factor? That, I don't know. Is their world hugely different than ours? That, I don't know. But I'm sure they probably have something better than we do.

And we've seen it in history. Like, I think even--

AUDIENCE: RSA.

Yael Kalai: --RSA was invented, but actually, it was privately known before. So we know examples, so probably today too. But I wish I-- it would be nice to know. Yeah?

AUDIENCE: Do people publish negative reports saying, like, we tried to break this random oracle protocol with SHA-256 and [INAUDIBLE]?

Yael Kalai: That's another thing. No. People don't tend to publish their failures. So it's very hard to know-- going back to your question. How much do people-- when we say, look, there's no break. OK, what does that mean? Are there no-- it means we don't know of a break. It doesn't mean there's no break. We don't know of a break. OK. How much did we as a community try to break? It's very hard to know. Because, like exactly-- your question was a very good one. People don't say-- they don't report how many hours they tried to break and fail.

So we don't know. Yeah. Yeah?

AUDIENCE: If you do break something, are you not supposed to publish it and you're supposed to-- until they edit it privately and then wait, like, six months and then-- because it might be dangerous.

Yael Kalai: Oh, oh. You're saying, if you do break, maybe you need to be careful to publish it. Yeah, yeah.

AUDIENCE: Maybe people break something all the time, but no one ever--

Yael Kalai: Never know.

[LAUGHTER]

AUDIENCE: [INAUDIBLE]

Yael Kalai: Yeah. I actually don't-- so that's not the art I kind of do. I don't usually break schemes. I like to construct things. I don't work on trying to break. But I know from folks here that do more security work, people get in trouble for publishing and breaking things.

And I don't know. We talk to them. They always mention the word jail in their conversations. It's like, wow, OK. Try to stay out of jail. Try to stay out of jail. You're like, OK, no. And here, in theory land, it's pretty easy to stay out of jail. They don't run after us. So yeah. But yeah, this can be a dangerous business. Any other questions? OK.

So I'm trying to think if I want to-- maybe I'll end with round-by-round soundness. It's not as interesting, but let me just say a few words on it just so I can check the box. I think it's in--

AUDIENCE: I have a question.

Yael Kalai: Yeah?

AUDIENCE: So what was the verdict on the zero-knowledge proof? You cannot do it in parallel?

Yael Kalai: Good, OK. So the question was-- going back to the zero knowledge. OK, so zero knowledge, if you repeat it sequentially, it still stays zero knowledge. It still stays zero knowledge because-- exactly the same reason as before.

Each time-- I mean, I can simulate. Essentially, I can simulate this transcript exactly the way I can attack it in the random oracle model. So, remember, what is the definition of zero knowledge, which I believe I erased? Look at this. I didn't. OK. So I want to say, for any cheating verifier, you can simulate the transcript. If I repeat it in parallel, I can simulate the transcript. How? I commit. I'm the simulator. I'm going to simulate a transcript for you-- at least, a physical transcript with a commitment as a safe.

I commit. I fake it till I make it. I guess $\beta = 1$. I commit. I hope V^* . And then I compute V^* 's answer on my commitment, hope it's $\beta = 1$. If it's $\beta = 1$, I can open. And I continue. If it's not $\beta = 1$, I try again. And that's how I simulate the protocol exactly as I cheat in the random oracle model.

What happens when things are sequential-- when things are parallel, sorry-- when things are parallel? Now, in the parallel world, you get $\alpha = 1$ up to $\alpha = \lambda$, so λ parallel partition of the commitments.

So you commit all the matrix, all the commitments. $\alpha = 1$, you commit, you commit. You do it in parallel. You get all the $\beta = 1$ up to $\beta = \lambda$ -- tells you if to open with respect to 0 or 1. And then you need to do all the openings, $\gamma = 1$ up to $\gamma = \lambda$. Ah.

AUDIENCE: The thing that helped you simulate--

Yael Kalai: Wow.

AUDIENCE: --also helped you break the model.

Yael Kalai: Exactly. The thing that helps me simulate is exactly the break of the random oracle model. Exactly, exactly. Now, what happens if I do it in parallel? If I do it in parallel, it's not-- OK. If V^* was honest, was completely honest, then yes, I'll just simulate each one separately. Because I can do the same thing. I just choose each one randomly, simulate each one randomly, simulate.

The thing is, V^* may be malicious. Here's a really problematic V^* . Suppose V^* is such that I give him all the commitment. So it's digital. So I told you in a safe, but actually, there's cryptography going on in there. Let me just gloss over the crypto. The safe is going to be digital, so there's just bits.

And now suppose V^* chooses β and some function of α . He applies some-- he's a cheating verifier. He wants to learn. His goal is to learn. And how does he learn? He gets all these things. He applies some f , some function f , to all of them together. Think of it like a random function-- I don't know-- and gets β_1 up to β_λ . And that's what he sends me. That's the cheating prover.

Now, how do I simulate this? I can guess β_1 and try to get this. I can't guess all the β s at the same time. The thing is, I can try to guess β_1 , give a commitment here, hope for the best. But the problem is, if I do α_2 , the β_1 may change because the β s are a function of all the α s together. It's really not clear how to simulate this.

OK. So the question holds. Is this zero knowledge? And this was an open problem for a very, very long time, over 20 years. We had no idea. People really tried. This we know because people did publish some partial results. And everybody worked on it for a very long time. We shared our frustration.

And then in 2019-- actually, here at MIT-- and be within in our neighborhood-- a bunch of people-- I'm going to tell you about it-- actually proved that this is not zero knowledge, not parallel zero knowledge. And the interesting thing-- the way they proved that it's not zero knowledge is actually by proving that the Fiat-Shamir is sound for this protocol, which is very weird.

[LAUGHTER]

So what they proved is that this protocol-- this is a three-round protocol. You can apply Fiat-Shamir to it. What they proved is if you-- so there was a question. Is Fiat-Shamir sound in the standard model? Is it sound? Is it sound? Nobody knew. They gave kind of a breakthrough result showing it's sound in the random oracle model. So that's what I'm going to show you next time. Next time, we're going to prove that this protocol is actually sound if you apply Fiat-Shamir to it.

The interesting thing is, the fact that it's sound if you apply Fiat-Shamir shows-- so we're going to show this is sound. So of course it's sound. If you just do parallel partition, you'll get soundness. But it's also sound with-- there exists a hash function-- so there exists a hash function such that even this hash, under a standard cryptographic assumption known as learning with error-- we'll talk about later-- almost learning with error with circular security. Never mind.

There's a standard cryptographic assumption so that under that assumption, there exists a hash family such that this protocol is Fiat-Shamir sound. So it's sounds even after you apply the Fiat-Shamir paradigm to it. Yeah?

AUDIENCE: So you're talking about the Hamiltonian path in NP. Is that specific protocol?

Yael Kalai: Yeah.

AUDIENCE: And not just any zero knowledge?

Yael Kalai: No, not any, not any. There may be zero knowledge protocols that you can repeat in parallel and zero knowledge-- yeah, sorry. Let me clarify. I'm not saying that any zero-knowledge protocol, if you repeat it in parallel, it's not zero knowledge anymore. I'm not saying that.

I'm saying that this specific protocol for graph Hamiltonian, if you repeat this protocol in parallel, it's not going to be zero knowledge. And the reason it's not going to be zero knowledge, which-- this should be really like, what? You should not understand what I'm saying. Or at least, it takes time to absorb. Or at least, to me, it took time to absorb.

The reason it's not zero knowledge is because, actually, this protocol is secure if you apply the Fiat-Shamir-- it's sound if you apply the Fiat-Shamir paradigm to it. And this is what we'll see next time. So this is weird. Yeah?

Audience: Is this because it's not simulatable? Like, you can generate it in that order?

Yael Kalai: Exactly, exactly. So why is it not zero knowledge? So what does it mean to be zero knowledge. What's the definition of zero knowledge? Zero knowledge says, for any V^* , you can simulate. But now I'll be the V^* that you cannot simulate. My V^* star-- I'm going to be a Fiat-Shamir V^* . I'm going to compute my beta. I'm V^* . I want to learn something. My goal is to learn something.

So I don't want to be able to simulate you. Then I didn't learn anything. My goal's just to learn. So how do I learn? I'm going to choose my betas as a Fiat-Shamir. I'm going to use the Fiat-Shamir hash. So you told me it's sound. There's some hash family that it's sound. Great. I'm going to choose a random hash key from this family. And my beta is going to be hash of alpha.

Now you're saying you can simulate me. For any x , you can simulate to me. If you can simulate me for any x , then you should be able also to simulate me for x in the language and for x outside the language. Because you don't know which is in and which is out. You're poly bounded. So I want to argue there is no simulator. Let me actually say this slowly. So I'm not going to do this round-by-round soundness today. It's not that interesting. But it will come maybe to haunt us later.

But let me-- it's nicer to do. And we'll just end with that. We have a couple minutes left. We'll just end with that. OK. So, again, next time, we'll see that parallel repetition of this specific protocol with a specific commitment-- with a specific digital safe-- is sound when you apply the Fiat-Shamir to it.

Now here's the claim. Fiat-Shamir soundness implies that it's not zero knowledge. And let me argue that. Let me argue that. So here's the proof.

Suppose it's zero knowledge. OK, so suppose it's wrong. Suppose I'm going to get a contradiction. So suppose there exists a simulator, a PPT. There is an efficient simulator such that-- I'm just copying the definition. Suppose it's zero knowledge. It means that there is an efficient simulator such that for any poly size V^* , for any cheating verifier, and for every x in the language, you can simulate.

OK. So for every V^* , for every x , Sim-- what does Sim do? Sim-- an input x , outputs alpha, beta, gamma that are accepted with respect to x . Because the prover's accepted. He's honest. So the prover is always accepted. You're talking with an honest prover that has a witness.

So he should also accept. They're indistinguishable. Real and ideal are-- the real view should be-- this is from the simulated view. The real view outputs something that's accepted, so the simulator should output something that's accepted. OK.

So first, I want to argue, this implies that also for x not in L -- or for random, I should say-- or there exists, at least-- such that Sim and x also outputs α , β , γ that are accepted. So this is accepted.

Let me write-- I'll write it like-- and so this should also be accepted. Why? Why? So I want to argue if, for every x in the language, this efficient simulator with oracle access to an efficient verifier outputs an expected transcript, there must be at least one x not in the language so that he outputs something that's accepting.

Because he can't distinguish in the language-- it's hard problem, and he's efficient. If all the time that it's in the language, he outputs accepting, and all the time not in the language, then the simulator broke your language. He broke NP. OK, so that can't be. So there exists x not in the language for which he manages to generate a transcript that breaks the Fiat-Shamir.

Because what does V^* do? V^* , this means that this is hash of x in α . That's my V^* . My V^* applies the Fiat-Shamir. OK. The V^* that I'm interested in is the Fiat-Shamir V^* . We assumed that the Fiat-Shamir is sound with respect to some H .

This V^* is going to use that H . He's going to choose a random hash key. And that's how he chooses the β . Or a random hash key is fixed. That's it. That's the hash key. It's fixed. Now Sim has to use that hash key. Because otherwise, you can distinguish. So he broke the Fiat-Shamir. That means he cannot simulate. There's no simulator. A simulator will break the Fiat-Shamir.

So it's an interesting-- It was a weird state of the world. Because people believed, in some sense, that parallel zero-knowledge should hold. Why not, eh? You can't learn. What can you learn? Try to learn, eh. But people also learned that Fiat-Shamir is true. But these two are not-- you can't believe both. And it wasn't clear which one is the right thing. And so now we know-- at least, for this protocol-- Fiat-Shamir is true, and it's not zero knowledge. Yeah?

AUDIENCE: So actually, there is something stronger, like that any non-interactive proof cannot be zero knowledge. Is that basically with--

Yael Kalai: Any interactive proof which Fiat-Shamir holds cannot be zero knowledge. Exactly. Yeah.

AUDIENCE: OK. Because yeah, if you use a specific instantiation, like hash then it's basically calling him hash [INAUDIBLE].

Yael Kalai: Exactly, exactly.

AUDIENCE: There's no interaction that breaks zero knowledge.

Yael Kalai: Yeah. Exactly.

AUDIENCE: That's what the proof I see.

Yael Kalai: Yeah, yeah, yeah. OK, we're out of time.