Statistical Zero Knowledge

CS6230: Topics in Information Security
Lecture 6 (Sep 14, 2021)
Lecture and notes: Prashant Nalini Vasudevan

Today, we will see a different, perhaps more natural, relaxation of Perfect Zero-Knowledge that uses statistical indistinguishability instead of computational indistinguishability. Before we do, however, let us briefly discuss composition of zero-knowledge proofs.

1 Composition of Zero-Knowledge Proofs

As we saw in previous lectures and problem sets, repetition of an interactive proof is a simple way to amplify its completeness and soundness. When any interactive proof is repeated k times, either in sequence or in parallel, with the verifier checking that a majority of these repetitions accept, the completeness and soundness errors go down almost exponentially with k.

With the zero-knowledge property, however, things are more complicated. For instance, it is possible to construct simple zero-knowledge protocols that are such that even repeating them twice in parallel makes them lose the zero-knowledge property. In fact, even the possibility of non-trivial constant-round public-coin zero-knowledge protocols with negligible soundness error was a major open question until some very innovative techniques were developed to construct such protocols (see [Gol13] for a more detailed discussion). We will cover this much later in the class if we have time.

Exercise 1. *Construct two protocols that are each zero-knowledge (computational or otherwise) on their own, but when run together in parallel, are no longer zero-knowledge. That is, a cheating verifier can somehow exploit the honest prover in the execution of one of the protocols to learn something non-trivial from the honest prover of the other protocol.*

This behaviour is critical to understand, as often zero-knowledge protocols are used as sub-protocols in larger cryptographic protocols in order to protect secrets that some party might have. These sub-protocols may be run several times in the course of execution of the larger protocol, and we would like the secrets to still be protected. The design of zero-knowledge protocols that compose well is the subject of research in concurrent zero-knowledge, a topic we will not get into.

When it comes to sequential repetition, however, things are much simpler. The definitions of zero-knowledge we saw earlier were actually weaker versions of the ideal definition, which also allows for what is known as *auxiliary information*. This is an additional arbitrary input that is given to the verifier (and simulator), and represents additional knowledge that the verifier may have from external sources. In the above example of the larger protocol, this may represent anything the verifier may have learnt from the execution of other sub-protocols int he past. It turns out that this stronger definition composes well under sequential repetition, with the zero-knowledge property being preserved, and the completeness and soundness error going down as expected.

This definition for perfect zero-knowledge with auxiliary information is as follows. In the execution of a protocol (P, V), the verifier V is given, in addition to the input x, some arbitrary input z (of size polynomial $|x|$). The verifier's view in such an execution is denoted by $view^P_{V(z)}(x)$. The completeness and soundness requirements for the protocol are the same as before. For zero-knowledge, for any auxiliary input z, we give the simulator z and want it to simulate the verifier's view when it is given z.

Definition 1.1 (PZK with Auxiliary Inputs)**.** An interactive proof (P, V) for a language L is said to be *perfectly zero knowledge with auxiliary inputs* if, for every possible PPT verifier algorithm V^*, there exists a PPT *simulator* algorithm S_{V^*} such that:

- On any input x and auxiliary input z, $S_{V^*}(x, z)$ outputs $\perp$ with probability at most $1/2$
- For any input $x \in L$ and any auxiliary input z, the distribution of $S_{V^*}(x, z)$ conditioned on $S_{V^*}(x, z) \neq \perp$ is identical to $view^P_{V^*(z)}(x)$

Note that allowing the verifier an auxiliary input makes it more powerful than just allowing it to be non-uniform, as this z could even depend on x. Other notions of zero-knowledge are also usually defined with resistance to such auxiliary inputs, but we will continue to ignore this for simplicity.

Exercise 2. *Write down a similar definition for CZK with auxiliary inputs. Show that both of these definitions allow zero-knowledge to be preserved under sequential repetition.*

2 Statistical Zero Knowledge

Statistical Zero Knowledge (SZK) is a different, perhaps more natural, weakening of Perfect Zero Knowledge. Here, we relax the requirement that the simulator's output be the same as the view of the verifier, to requiring that the distributions of the simulator's output and the verifier's view be *statistically close.* In order to quantify "statistically close", we will need a meaningful notion of distance between probability distributions. There are several such measures, but we will be using perhaps the simplest, which is called *total variation distance.* As this is the only notion of distance we will be using, we simply refer to it as "statistical distance".

2.1 Statistical Distance

Given a distribution D over a domain X, we will use the following notation. As before, $x \leftarrow D$ denotes a sample from D. For any $x \in X$, we write $D(x)$ to denote the probability mass of x under D. That is, $D(x) = \Pr_{x' \leftarrow D}[x' = x]$. For any set $S \subseteq X$, $D(S)$ is the probability mass on all elements in the set – that is, $D(S) = \sum_{x \in S} D(x)$. Since we are discussing probability distributions, we will often identify such a set $S \subseteq X$ with the event of a sample from D belonging to S.

Definition 2.1 (Statistical Distance)**.** Given two distributions D_0 and D_1 over the same (discrete and finite) domain X, the statistical distance between them is the largest difference in the probability of any event occurring under the two distributions. That is,

$$\Delta(D_0, D_1) \triangleq \max_{S \subseteq X}(D_0(S) - D_1(S)) = \frac{1}{2}\sum_{x \in X} |D_0(x) - D_1(x)|$$

Exercise 3. *Show that the two expressions in the above definition are indeed equal.*

One may verify that this definition has the following properties that justify it as a reasonable notion of "distance" between distributions.

- For any D_0 and D_1, $\Delta(D_0, D_1)$ is between 0 and 1
- D_0 and D_1 are the same distribution if and only if $\Delta(D_0, D_1) = 0$
- D_0 and D_1 have disjoint supports if and only if $\Delta(D_0, D_1) = 1$

This notion of statistical distance is pervasive in cryptography and complexity theory because it captures the power of any algorithm in telling whether a sample came from one distribution or another. Recall the following definition of the advantage of an algorithm in distinguishing between distributions.

$$Adv_A^{D_0,D_1} = \left|\Pr_{x \leftarrow D_{0,\lambda}}[A(x) = 1] - \Pr_{x \leftarrow D_{0,\lambda}}[A(x) = 1]\right|$$

Almost by definition, we can see the following:

$$\Delta(D_0, D_1) = \max_A Adv_A^{D_0,D_1} \tag{1}$$

where the maximum is over all algorithms A, even computationally unbounded ones. Another simple corollary of the definition is that, for any set $S \subseteq X$, the probability that $x \in S$ when x is sampled from D_0 and when it is sampled from D_1 can only differ by at most as much as the distance between them. In different terms, for any event $E(x)$ that depends arbitrarily on x, we have:

$$\Pr_{x \leftarrow D_0}[E(x)] - \Delta(D_0, D_1) \leq \Pr_{x \leftarrow D_1}[E(x)] \leq \Pr_{x \leftarrow D_0}[E(x)] + \Delta(D_0, D_1) \tag{2}$$

Finally, similar to computational indistinguishability, we define statistical indistinguishability of two ensembles of distributions.

Definition 2.2. Two ensembles of distributions $D_0 = \{D_{0,\lambda}\}$ and $D_1 = \{D_{1,\lambda}\}$ are said to be *statistically indistinguishable* if, for any constant $c \in \mathbb{N}$, there exists a $\lambda_c \in \mathbb{F}$ such that for all $\lambda > \lambda_c$:

$$\Delta(D_{0,\lambda}, D_{1,\lambda}) < \frac{1}{\lambda^c}$$

2.2 Defining SZK

With this notion of statistical indistinguishability in hand, the definition of statistical zero-knowledge may be obtained as a natural analogue of that of computational zero-knowledge, which we saw earlier. Here too the prover and verifier (and simulator) are given as additional input a security parameter 1^λ, which specifies the level of security we require of the zero-knowledge. We define the corresponding distribution ensembles $S(x) = \left\{S(1^\lambda, x)\right\}_{\lambda \in \mathbb{N}}$ and $view_{V^*}^{P}(x) = \left\{view_{V^*(1^\lambda)}^{P(1^\lambda)}(x)\right\}_{\lambda \in \mathbb{N}}$

Definition 2.3. An interactive proof (P, V) for a language L is said to be *statistically zero knowledge (SZK)* if, for every possible PPT verifier algorithm V^*, there exists a PPT *simulator* algorithm S_{V^*} such that for any input $x \in L$, the ensembles $S_{V^*}(x)$ and $view_{V^*}^{P}(x)$ are statistically indistinguishable.

Note that any PZK protocol is also an SZK protocol, and that any SZK protocol is also a CZK protocol. The latter is because two distribution ensembles that are statistically indistinguishable are also computationally indistinguishable (compare (1) and the definition of computational indistinguishability from the previous lecture). Thus, statistical zero-knowledge is a "stronger" property than computational zero-knowledge. In practical terms, SZK protocols are not as useful as CZK protocols since (i) In most applications the security provided by computational zero-knowledge is sufficient, and, (ii) as the requirements on them are weaker, CZK protocols can be obtained for a much larger set of languages than SZK, while also being more efficient in various ways.

Of course, if security is desired against a computationally very powerful adversary, the stronger guarantees provided by statistical zero-knowledge become more relevant. Further, SZK protocols and the languages that have them turn out to have interesting connections to cryptography and complexity theory. This is the primary reason for our (at least, my) interest in them. The class of problems that have SZK protocols – which we will denote by SZK – contains many problems that are central to cryptography, and has some very natural complete problems. In this and the next lecture, we will see some examples of simple SZK protocols, complete problems, closure properties, and some of these connections. The canonical reference for SZK is Salil Vadhan's PhD thesis [Vad99]. Almost everything we will cover is contained there, and some of the proofs we will see are adapted from his.

2.3 Closeness of Samplable Distributions

The first SZK protocol we will see is a generalisation of a PZK protocol we have seen before – the one for Graph Isomorphism. Recall that this was a protocol (P, V) where P wishes to prove that a given pair of graphs (G_0, G_1) are isomorphic, and it worked as follows:

1. If G_0 and G_1 are not isomorphic, P sends $\perp$ to V and terminates.
2. If not, P samples a uniformly random relabelling permutation $R : [n] \to [n]$, and a uniformly random bit b. It sends $H = R(G_b)$ to V.
3. V samples a random bit b' and sends it to P.
4. P finds a uniformly random relabelling R' such that $R'(G_{b'}) = H$, and sends R' to V.
5. V accepts iff $R'(G_{b'}) = H$.

Pay attention here to the distribution of $R(G_0)$, where R is a random relabelling – call it D_0. Similarly define the distribution D_1 of $R(G_1)$. For any graph H, there exists R' such that $R'(G_{b'}) = H$ if and only if H is in the support of D_b. We can now briefly restate the arguments for the various properties of this protocol as follows.

- Completeness: If G_0 and G_1 are isomorphic, D_0 and D_1 are identical, so any H sampled from either of them is contained in the support of the other. Thus, P can always find such an R', irrespective of what b' is.

- Soundness: If G_0 and G_1 are not isomorphic, then D_0 and D_1 are disjoint, so H is in the support of at most one of them. So irrespective of the prover strategy P^*, such an R' does not exist for at least one value of b'.

- Perfect zero-knowledge: The simulator S_{V^*} samples b and R as P does, computes H and $b' \leftarrow V^*(H)$, and if $b = b'$, outputs all of these. If G_0 and G_1 are isomorphic, D_0 and D_1 are identical. So V^* cannot tell whether H was drawn from D_0 or D_1 at all. So b' is independent of b, and is equal to it with probability $1/2$. So S_{V^*} aborts with probability at most $1/2$, and when it doesn't, its output is identical to the actual protocol (this requires argument, which we will leave out here).

In essence, we reduced the graph isomorphism problem to this: given distributions D_0 and D_1 that are either identical or have disjoint supports, tell which is the case. In the above protocol, then, the prover is really proving that D_0 and D_1 are identical. Here, we dealt with the special case where D_0 and D_1 were obtained by randomly relabelling a given graph. We can consider the much more general case where these distributions are sampled by some algorithm that is given as input.

We will model such an algorithm by a boolean circuit. That is, given a circuit $C : \{0,1\}^m \to \{0,1\}^n$, consider the following process: sample a random input $r \leftarrow \{0,1\}^m$, and output $C(x)$. This process defines a distribution over $\{0,1\}^n$ that we will denote by D_C. In the case of graph isomorphism, we can think of the circuit C_{G_0} that has the graph G_0 hardcoded into it, uses its input r to sample a relabelling R, and outputs $R(G_0)$ (and similarly C_{G_1}). The distribution D_0 we defined above, then, was simply the distribution $D_{C_{G_0}}$ (and similarly D_1 and $D_{C_{G_1}}$). The graph isomorphism problem, then, is equivalent to the following promise problem:

$$L_Y = \left\{(G_0, G_1) \mid \Delta(D_{C_{G_0}}, D_{C_{G_1}}) = 0\right\}$$
$$L_N = \left\{(G_0, G_1) \mid \Delta(D_{C_{G_0}}, D_{C_{G_1}}) = 1\right\}$$

With this setup, we can define languages that deal with distributions by dealing with the circuits that sample them in the above manner. We then define the following generalisation of the above problem, which we call the Statistical Closeness (SC) problem:

$$SC_Y = \{(C_0, C_1) \mid \Delta(D_{C_0}, D_{C_1}) = 0\}$$
$$SC_N = \{(C_0, C_1) \mid \Delta(D_{C_0}, D_{C_1}) = 1\}$$

Actually, we need the additional specification in the sets above that C_0 and C_1 have the same output length, but we skip it for brevity. They need not have the same input length, but for simplicity, in the rest of the document, we will assume both take inputs from $\{0,1\}^m$ and output a string in $\{0,1\}^n$. Following along the same lines as the protocol for GI, we can construct a PZK protocol (P, V) for SC that works as follows given input (C_0, C_1):

1. If D_{C_0} and D_{C_1} are not identical, P sends $\perp$ to V and terminates.
2. If not, P samples a uniformly random input $r \leftarrow \{0,1\}^m$, and a uniformly random bit b. It sends $y \leftarrow C_b(r)$ to V.
3. V samples a random bit b' and sends it to P.
4. P finds a uniformly random input r' such that $C_{b'}(r') = y$, and sends r' to V.
5. V accepts iff $C_{b'}(r') = y$.

Note the analogies to the case of GI. The random choice of the relabelling R there corresponds to the random input r here. The circuit C_b, which takes input r, corresponds to the process of relabelling the graph G_b with a relabelling R. The output y of the circuit thus corresponds to the result H of the relabelling. The arguments of completeness, soundness, and perfect zero-knowledge of the above protocol follow almost identically to those of the GI protocol.

Exercise 4. *Write out the arguments for the completeness, soundness, and perfect zero-knowledge of the above protocol for SC.*

2.4 Relaxing SC

It turns out that the above protocol is actually even more powerful that stated above. We will see that most of the arguments for its completeness, soundness and zero-knowledge could still have been made if the distributions of D_{C_0} and D_{C_1}, instead of being identical, were just statistically very close. The only catch is that the protocol will lose the perfectness of zero-knowledge, and only be statistically zero-knowledge. This allows it to work for the natural generalistion of the SC problem that, instead of asking whether two distributions are identical or disjoint, asks whether they are statistically close or disjoint. For any $\beta \in [0,1)$, this is the SC$^\beta$ problem defined below:

$$SC_Y^\beta = \{(C_0, C_1) \mid \Delta(D_{C_0}, D_{C_1}) \leq \beta\}$$
$$SC_N^\beta = \{(C_0, C_1) \mid \Delta(D_{C_0}, D_{C_1}) = 1\}$$

The protocol then needs to be modified slightly, and works as follows:

1. If D_{C_0} and D_{C_1} are not identical, P sends $\perp$ to V and terminates.
2. If not, P samples a uniformly random input $r \leftarrow \{0,1\}^m$, and a uniformly random bit b. It sends $y \leftarrow C_b(r)$ to V.
3. V samples a random bit b' and sends it to P.
4. P finds a uniformly random input r' such that $C_{b'}(r') = y$, and sends r' to V. If no such r' exists, it sends $\perp$ to V instead.
5. V accepts iff $C_{b'}(r') = y$.

The proofs of the properties of the above protocol when applied to this language are still along the lines of those when it is applied for SC, just a little more complicated.

Soundness. The proof of soundness is identical to the case of SC. If D_{C_0} and D_{C_1} have statistical distance 1, this means their supports are disjoint, and that there is no $y \in \{0,1\}^n$ that is in the image of both C_0 and C_1. So, irrespective of the prover's strategy, there is always at least one value of b' such that there is no r' for which $C_{b'}(r') = y$. Thus the verifier can be made to accept with probability at most 1/2.

Completeness. If $\beta = 0$, the argument for completeness would have been the same as in the protocol for GI – any y in the image of C_0 is also in the image of C_1. However, this statement is no longer true. For any β that is larger than 0, there could potentially be y's in the image of C_0 that is not in the image of C_1. But the statement that their output distributions have statistical distance at most β implies that the number of such y's cannot be large.

Claim 2.1. *For any circuits C_0 and C_1 such that $\Delta(D_{C_0}, D_{C_1}) \leq \beta$, the probability that a random $y \leftarrow D_{C_0}$ does not have a pre-image under C_1 is at most β*

Proof of Claim 2.1. This follows from an application of property of statistical distance stated as (2). Note that for a y sampled from D_{C_1}, the probability that it does not have a pre-image under C_1 is 0. Thus, the probability that $y \leftarrow D_{C_0}$ does not have a pre-image under C_1 is at most β.

In other terms, define $E(y)$ to be the event that y does not have a pre-image under C_1. Then, by (2),

$$\Pr_{y \leftarrow D_{C_0}}[E(y)] \leq \Pr_{y \leftarrow D_{C_1}}[E(y)] + \Delta(D_{C_0}, D_{C_1}) \leq 0 + \beta$$

□

In the protocol, the verifier would reject only if $b' \neq b$, and further the y that the prover sampled from D_{C_b} does not have a pre-image under $C_{b'}$. Thus, using Claim 2.1, the probability that the verifier rejects is at most $\beta/2$.

Statistical Zero Knowledge. The simulator works in the same manner as the one for the GI protocol. For any V^*, given input (C_0, C_1), the simulator S_{V^*} works as follows:

1. Sample uniformly random $r \leftarrow \{0,1\}^m$ and $b \leftarrow \{0,1\}$, and compute $y \leftarrow C_b(r)$
2. Sample r_{V^*} uniformly at random from the appropriate domain, and compute $b' \leftarrow V^*((C_0, C_1), y; r_{V^*})$.
3. If $b \neq b'$, output $\perp$ and terminate.
4. if $b = b'$, compute the output of the verifier $out \leftarrow V^*((C_0, C_1), y, b, r; r_{V^*})$, and output (r_{V^*}, y, b, r, out).

We will now need to prove two things: (i) S_{V^*} outputs $\perp$ with probability less than some constant, and, (ii) conditioned on not outputting $\perp$, the distribution of its output is statistically close to the actual view of V^*. Fix some input (C_0, C_1) such that $\Delta(D_{C_0}, D_{C_1}) \leq \beta$.

The probability that S_{V^*} outputs $\perp$ may be written as follows:

$$\begin{aligned}
\Pr[b \neq b'] &= \Pr[b = 0] \cdot \Pr[b' = 1 \mid b = 0] + \Pr[b = 1] \cdot \Pr[b' = 0 \mid b = 1] \\
&= \frac{1}{2} \cdot \Pr[b' = 1 \mid b = 0] + \frac{1}{2} \cdot (1 - \Pr[b' = 1 \mid b = 1]) \\
&= \frac{1}{2} + \frac{\Pr[b' = 1 \mid b = 0] - \Pr[b' = 1 \mid b = 1]}{2} \qquad (3)
\end{aligned}$$

Consider the event $b' = V^*((C_0, C_1), y; r_{V^*}) = 1$ – this is determined completely by y and r_{V^*}, and does not involve b otherwise. By (2), the probability that this happens when y is sampled from D_{C_0} or from D_{C_1} (and r_V^* is sampled in the same way and independently of y in both cases) can differ by at most β. In other words, $\Pr[b' = 1 \mid b = 0]$ and $\Pr[b' = 1 \mid b = 1]$ can differ by at most β, and so by (3), the probability that $b \neq b'$ is at most $1/2 + \beta/2$. This proves the first part.

The second part of the proof is a little complicated. In class, I said it follows in the same way as this part of the proof for the GI protocol. I was wrong. I forgot to account for that fact that the conditioning $b = b'$, together with D_{C_0} and D_{C_1} not being identical leads to some non-trivial skewing of the distribution of the simulator's output. That is, the distribution of r_{V^*} and y in the simulator's output, upon conditioning on $b = b'$, may be different from the distribution of r_{V^*} and y in the actual view of the verifier. Note that, without the conditioning $b = b'$, these two distributions are indeed the same. Conditioning, however, often causes issues like this. This was not a problem in the case of the GI protocol (or the earlier protocol for SC) because there the two distributions D_{C_0} and D_{C_1} were identical, leading to the event $b = b'$ being independent of the value of r_{V^*}. Here, though, that is not the case. To illustrate, let us write out this proof in the case of $\beta = 0$ (that is, the SC problem) in a slightly more roundabout way than necessary.

The first thing to notice is that, for any fixed value of r_{V^*} and y, conditioning on $b = b'$, the rest of the simulated distribution is identical to the corresponding parts of the view of V^*. This is because once r_{V^*} and y are fixed, the value of b' is determined, and the r that V^* sees in the protocol is a random pre-image of y under $C_{b'}$, which is exactly what the simulator also outputs conditioned on $b = b'$. Our task, then, is to show that, even with the conditioning $b = b'$, the distribution of (r_{V^*}, y) output by the simulator is the same as the distribution in the view. To be more precise, let R and Y denote the random variables corresponding to r_{V^*} and y in the actual view of V^*, and R_S and Y_S the respective parts of the simulator's output. For any value of (r_{V^*}, y), denote by $b'_{r_{V^*},y}$ the output of $V^*((C_0, C_1), y; r_{V^*})$. Suppose r_{V^*} is drawn uniformly from $\{0,1\}^\ell$. The probability mass on this pair (r_{V^*}, y) under the actual view of V^* is given by:

$$\begin{aligned}
\Pr[(R, Y) = (r_{V^*}, y)] &= \Pr[R = r_{V^*}] \cdot \Pr[Y = y \mid R = r_{V^*}] \\
&= \Pr[R = r_{V^*}] \cdot \Pr[Y = y] \\
&= \frac{1}{2^\ell} \cdot \frac{D_{C_0}(y) + D_{C_1}(y)}{2} \qquad (4)
\end{aligned}$$

where the $\Pr[Y = y]$ is calculated by noting that in the protocol the bit b is set to 0 or 1 with probability $1/2$, and then y is sampled from D_{C_b}.

In the simulated distribution, the conditioning $b = b'$ means that Y_S is sampled only from $D_{C_{b'}}$, and not $D_{C_{1-b'}}$. The probability mass on (r_{V^*}, y) in the simulated distribution, conditioned on $b = b'$, is as follows:

$$\begin{aligned}
\Pr[(R_S, Y_S) = (r_{V^*}, y) \mid b = b'] &= \Pr[R_S = r_{V^*} \mid b = b'] \cdot \Pr[Y_S = y \mid b = b' \wedge R_S = r_{V^*}] \\
&= \Pr[R_S = r_{V^*}] \cdot \Pr[Y_S = y \mid b = b'] \\
&= \frac{1}{2^\ell} \cdot D_{C_b}(y)
\end{aligned} \qquad (5)$$

where the equalities follow from two facts:

(i) The event $b = b'$ happens with probability $1/2$ independently of the value of Y_S and R_S, as y contains no information about b if D_{C_0} and D_{C_1} are identical. So the distribution of R_S is also independent of the event $b = b'$.

(ii) Given that $b = b'$, Y_S is just the distribution $D_{C'_b}$, and no longer depends on the value of R_S.

Noting that $D_{C_b}(y)$ is the same as $(D_{C_0} + D_{C_1})/2$ for any value of b then tell us that the above two distributions are identical.

If the two distributions D_{C_0} and D_{C_1} had not been the same, however, we would not have been the first fact (i) above. That is, R_S would not necessarily have been independent of the event $b = b'$. It turns out that we can use the fact that D_{C_0} and D_{C_1} are β-close to show that this is approximately true, but it requires more work. This can be done with some general lemmas about the interaction between statistical distance and conditioning, but we leave this out for now. It turns out that the statistical distance between the simulated distribution conditioned on $b = b'$ and the actual view is at most β. Refer to Section 6.2 of Vadhan's thesis [Vad99] for the complete proof.

References

[Gol13] Oded Goldreich. A short tutorial of zero-knowledge. In Manoj Prabhakaran and Amit Sahai, editors, *Secure Multi-Party Computation*, volume 10 of *Cryptology and Information Security Series*, pages 28–60. IOS Press, 2013.

[Vad99] Salil Pravin Vadhan. *A study of statistical zero-knowledge proofs.* PhD thesis, Massachusetts Institute of Technology, 1999.