Randomness, Pseudorandomness and Derandomization Sept 15th, 2010

Lecture Notes 2: Cryptography and Pseudorandomness

Professor: Avi Wigderson (Institute for Advanced Study) *Scribe: Dai Tri Man Lê*

In this talk, we will continue the discussion on pseudorandomness, but in the context of cryptography. In fact, the idea of pseudorandomness was proposed in the context of cryptography.

1 The ambitions of modern cryptography

The goal of cryptography before 1970s mainly concerned with secret communications between two parties, Alice and Bob, in the present of adversaries who want to learn the secret messages. It is based on a very strong assumption that Alice and Bob share some information that no one else has.

Since 1970s, the study of cryptography no longer assumes that Alice and Bob share any private information, but we still want them to be able to communicate privately. One might wonder if it is even possible for Alice and Bob to "speak" a secret language without meeting each other before? In fact, this is required when we want to shop online safely with our credit cards! Secret communication in such a setting is among the main goals of modern cryptography.

The theory of *public-key encryption* is what is really behind the e-commerce security. It was started in the '70s with remarkable papers by Diffie-Hellman, Merkle, Rivest-Shamir-Adleman, Rabin, where the key conceptual idea is to develop cryptography theory based on computational complexity, which leads to notions like *one-way functions* and *trapdoor functions*. However, this is still not yet the mathematics of cryptography. It was due to a paper by Goldwasser-Micali in 1981 and contributions by Blum-Micali and Yao that define key formal definitions, proof techniques, and introduces the important concepts: *computational indistinguishability* and *pseudorandomness*, which form the basic mathematics of cryptography.

However, the goals of modern cryptography are more than just secret communications. We want to deal with many situations where there are requirements for both privacy and resilience. For examples, proving a person's identity, money transfer, public bids, poker game on the phone, etc. We want to do all of these digitally without trusted parties. It is striking that all of these ideas were developed before the emergence of the Internet!

2 The assumptions of modern cryptography

Modern cryptography is built based on two main computational complexity assumptions. The first assumption is that every agent participating in the communication is computational bounded (say, to polytime). Thus, only tasks with efficient algorithms can be performed in this computational limited setting. The second assumption is that factoring integers is hard. It follows from these two assumptions that it is easy for each agent to multiply two prime numbers p and q, but no party can factorize products $p \cdot q$ efficiently without knowing p or q in advance. Thus, we can intuitively think of the pair (p, q) as in an open "envelope" and the product $p \cdot q$ as (p, q) in a sealed "envelope". So the two assumptions provides a form of *digital envelope* in modern cryptography.

We will elaborate a bit more on the idea of digital envelope, also formally called *commitment scheme*. Note that multiplying two integers is just one example of so called *one-way functions*. A

one-way function is a function that is easy to compute on every input, but hard to invert given the image of a random input. So we can construct commitment schemes out of any one-way function. Thus, the second assumption can be more appropriately stated as: there exists a one-way function.

Blum showed in 1981 that digital envelopes can be seen as commitments as demonstrated in the following example. Assume Alice and Bob no longer trust each other, but they want to play a bet over the phone by letting Alice choose a side of a coin, and then Bob tosses the coin. But how does Alice know if Bob is honest about the result of the coin toss? One possible way is the following. First, Alice makes her commitment (head or tail) in advance, puts it inside a digital envelope and then sends it to Bob, and thus Alice can no longer change the content inside of the envelope and Bob can't see what is inside the envelope. After Bob tosses the coin and reports the result to Alice, Alice will open the envelope to show Bob the content hidden inside.

We will now discuss the mathematical construction of a digital envelope more formally. Note that the definitions are simplified and not complete for the sake of simplicity. Assume that we have an injective function $f : \{0,1\}^n \to \{0,1\}^n$, which is one-way, i.e., f can be computed in polytime, and for every "efficient" algorithm A,

$$\mathsf{Prob}_{x\in\{0,1\}^n}[A(x) = f^{-1}(x)] \leq \epsilon,$$

for some $\epsilon = 1/poly(n)$. Then by an important theorem in cryptography due to Goldreich and Levin, there is a function $b : \{0,1\}^n \to \{0,1\}$, called a *hardcore predicate*, such that for every "efficient" algorithm A,

$$\mathsf{Prob}_{x\in\{0,1\}^n}[A(f(x)) = b(x)] \leq 1/2 + \epsilon,$$

for some $\epsilon = 1/poly(n)$. Now, we assume that $\sigma \in \{0,1\}$ is the bit that Alice wants to hide, then we can pick a random $r \in \{0,1\}^n$, and define $C(\sigma, r) := (f(r), b(r) \oplus \sigma)$. We claim that $C(\sigma, r)$ is a valid commitment scheme. First, it follows from a property of hardcore predicates that $f(r)$ fully defines $b(r)$, and so $C(0,r) \neq C(1,r)$. Second, we need to show that the bit σ is computationally hidden. Suppose that some "efficient" algorithm can distinguish $C(0,r)$ from $C(1,r)$, then it can also distinguish $(f(r), b(r)$ from $(f(r), \bar{b}(r))$. Thus it is possible to guess $b(r)$ efficiently given $f(r)$. This contradicts that $b(r)$ is a hardcore predicate.

Another way to look at it is that assuming r is taken from a uniform distribution, no efficient algorithm can distinguish (within some error ϵ) the output distributions D_1 and D_2 of $C(0,r)$ and $C(1,r)$ respectively. In other words, for every "efficient" algorithm A,

$$\left|\mathsf{Prob}_{r\in\{0,1\}^n}[A(C(0,r)) = 1] - \mathsf{Prob}_{r\in\{0,1\}^n}[A(C(1,r)) = 1]\right| \leq \epsilon.$$

We can think of computational distinguishability as a form of distance:

$$d_{\mathsf{comp}}(D_1, D_2) = \max_{\text{all efficient } A} |A(D_1) - A(D_2)|,$$

where $A(D)$ denotes $\mathsf{Prob}_{x\sim D}[A(x) = 1]$. Note that this is different the usual statistical distance between D_1 and D_1:

$$d_{\mathsf{stat}}(D_1, D_2) = \max_{\text{all possible } A} |A(D_1) - A(D_2)|,$$

where we allow tests A with unlimited resources.

Recall that in the first lecture, we actually use computational distinguishability when defining pseudorandom distributions. A distribution D is (computationally) pseudorandom if it cannot be efficiently distinguished from a uniform distribution D, i.e., the computation distance $d_{\mathsf{comp}}(D, U)$ is small.

3 Zero-knowledge proofs and an example on map coloring

Recall the motivation is that the prover wants to have his or her claim of proving a statement S verified without allowing the verifier to learn enough of the proof to publish it first. Goldwasser, Micali, and Rackoff in 1985 were the first to introduced the notion of zero-knowledge proof, and give the first zero-knowledge proof for a concrete problem, that of deciding quadratic nonresidues modulo m. Goldreich, Micali and Wigderson later showed "the universality property" of zero-knowledge proofs, i.e., assuming a one-way function exists then for anything you can prove, we can also give a zero-knowledge proof for it. We will demonstrate the basic intuition of zero-knowledge proofs though the problem of 3-colouring of planar graphs.

As mentioned in the first lecture, the question of determining whether a given graph G with n edges is 3-colourable is NP-complete (and thus just as hard as theorem proving). Thus, it suffices to consider the problem of verifying a very specific type of claim: that a given graph G (which is known to both the verifier and prover) admits a 3-colouring. Suppose that the prover has in fact managed to 3-colour the graph G using three colours red, green, and blue, and wants to convince the verifier of this fact, without revealing to the verifier how to colour this graph, or learn anything other than the fact that G is 3-colourable.

This can be done probabilistically by observing that we can permute the colours around and convert any given 3-colouring into one of 3! = 6 colourings. We can use this observation to create the following verification algorithm:

1. The prover picks a 3-colouring of the graph using red, green, and blue, and permutes the colours randomly. The prover then commits to this colouring; if this were done physically, the prover could place the colour of each vertex in an envelope attached to that vertex. To do it digitally, one needs a cryptographic commitment scheme, which can be set up provided that one-way function exists.
2. The verifier then picks two adjacent vertices of the graph arbitrarily, and asks the prover to reveal the colours of the two selected vertices. If the two colours agree (or if a colour other than red, green, or blue appears), then the verifier rejects.
3. Otherwise, return to Step 1 and start over. If the algorithm does not reject in (say) n^2 iterations of this process, then the verifier accepts.

Note that if the prover does not in fact know of any 3-colouring of G, then the verifier will be able to detect this fact with probability at least $1/n$ in any round of the above procedure, since by picking an edge at random one has at least a $1/n$ probability of picking an edge for which the provers colouring fails to be a 3-colouring. So the algorithm is probabilistically sound. It is also obviously complete. So why is it zero knowledge? Because if the prover randomly shuffles the colours at each round, independently of all previous rounds, then for any given round, the two colours that the verifier gets to see are distributed completely at random among all six ordered pairs of distinct colours. This is a distribution of query responses that the verifier could have generated himself, and so this algorithm is not providing any new information that the verifier did not already have.

(End of the second talk.)

Acknowledgment Some parts of the lecture notes follow an exposition by Terry Tao on a similar talk given by Avi Wigderson in UCLA.