CIS 511: Theory of Computation **Apr 13, 2017**

Lecture 23: More Cryptography and Random Number Generation

Professor Sampath Kannan *Zach Schutzman*

NB: *These notes are from CIS511 at Penn. The course followed Michael Sipser's Introduction to the Theory of Computation (3ed) text.*

Cryptography Wrap-Up

The existence of trapdoor functions is a stronger claim than the existence of one-way functions.

To make a trapdoor function, we need a generator $G(x)$ which is a probabilistic function which takes as input the length of the string and produces an index i into a family of functions and t, the trapdoor information ('secret'). Sometimes i is called the **public key** and t the *private key.*

For the forward direction, we have a function $c = f(m, i)$, where m is the message in plaintext, i is as above, and c is the ciphertext. We have f easy to compute in poly-time (by definition). We also want that for all i and any probabilistic poly-time algorithm E, $\Pr(f(E(f(w, i), i)) = f(w, i))$ is negligible. That is, the probability that a hacker can find any of the preimages of c under f is arbitrarily small.

For the reverse direction, we say there exists a poly-time algorithm h such that $h(f(w, i), t) = w$. That is, knowing t allows easy decryption.

If trapdoor functions exist, then secure public-key encryption is possible.

One function believed to be a trapdoor function generates two n-bit primes, p, q (we expect n trials to generate each). We set the public key to be $i = pq$ and $t = (p, q)$. Our $f(m, i) = m^2 \mod i$ and $h(m^2, (p, q))$ finds the square root of $m^2 \mod p$ and q. We showed last time that that finding square roots modulo composite numbers is hard, and it turns out that finding square roots modulo prime numbers is easy. Knowing $m^2 \mod p$ and $m^2 \mod q$ gives easy access to $m^2 \mod pq$, but if we don't know either of these, we do not have easy access to this quantity. Therefore, if factoring integers is hard, this f is a trapdoor function.

The **RSA Cryptosystem**(Rivest, Shamir, Adleman) is a more popular cryptosystem. Bob, the recipient of the message, picks two primes p, q and computes $N = pq$. He then chooses an encryption exponent $e \in [0, N-1]$ and computes a decryption exponent d such that for any message $m \in Z_N^*$, $(m^e)^d = m \, (mod \, N)$. Bob publishes (N, e). There are $p + q - 1$ values in $[0, N-1]$ not relatively prime to N, so a random number has probability $\frac{p+q+1}{pq}$ of not being in Z_N^*, which is very small. When Alice wants to send a message to Bob, she computes $m^e \mod N$ which Bob can decrypt.

How do we choose e? We use the Euler totient function $\phi(N) = (p-1)(q-1) = |Z_N^*|$. $\phi(p) = p - 1$ for prime p. If p and q are relatively prime, then $\phi(pq) = \phi(p)\phi(q)$, by the Chinese Remainder Theorem. From group theory (by Lagrange's Theorem), for any $a \in Z_N^*$, $a^{\phi(N)} mod \, N = 1$.

If Bob can pick e, d such that $ed = 1 \mod \phi(N)$, then $m^{ed} = m^{k\phi(N)+1} = m$. Therefore, e should be relatively prime to $\phi(N)$. To pick d, Bob finds d which satisfies $de + k\phi(N) = 1$.

Bob can do all of his computations and decryption in polynomial time, and Alice can do the encryption in polynomial time as well. If we assume factoring is hard, then an eavesdropper without access to p and q cannot decrypt the message.

RSA doesn't work if the messages are too short (drawn from a small space). An eavesdropper can brute-force it and/or we run into trouble with the exponent not causing the values to wrap around N.

Suppose our eavesdropper Eve sees the ciphertext and knows that the message being transmitted is one of m_1 or m_2, which she has access to in plaintext.

Definition 23.1 *A public-key encryption scheme is called* ***semantically secure*** *if Eve has only a negligible advantage over random guessing at determining whether the true message is m_1 or m_2.*

Semantic security is obviously impossible for any deterministic encryption scheme. One randomized scheme is to add random padding to the message before encrypting it. It is conjectured that RSA with random padding is semantically secure.

Pseudorandom Number Generation

Definition 23.2 *A* ***pseudorandom number generator*** *(PRNG) is a deterministic function which takes as input a string of a short length (m) and produces a string of a much larger length (n) such that the output string looks random. That is, the distribution of the output strings is indistinguishable from a draw from a uniform distribution over strings of length n.*

In actuality, the PRNG can't truly mimic the uniform distribution. There are 2^n possible uniform strings, but the PRNG has output space of size 2^m.

Definition 23.3 *A PRNG is* ***pseudorandom*** *if for any probabilistic poly-time algorithm A which outputs 0 or 1, the probability that A outputs 1 given input from the PRNG minus the probability that A outputs 1 given input drawn from the uniform distribution is negligible. That is, there is no such A which can meaningfully differentiate between draws from the PRNG and a true uniform distribution.*

Claim 23.4 *If one-way functions exist, then PRNGs exist.*

PRNG from Rabin: *Start with a seed $x \in \{0,1\}^m \bmod N$. Take $x^2 \bmod N$ and output the most significant bit, then repeat. The proof that this works is difficult.*

Definition 23.5 *A generator is* ***next-bit unpredictable*** *if no algorithm can do negligibly better than random guessing for the ith bit of a string given the first $i-1$.*

Claim 23.6 *A generator is a PRNG is it is next-bit unpredictable.*