CPSC 467: Cryptography and Computer Security

Michael J. Fischer

Lecture 14
October 26, 2015

Primitive Roots

Using the ElGamal cryptosystem

To use the ElGamal cryptosystem, we must be able to generate random pairs (p, g), where p is a large prime, and g is a primitive root of p.

We now look at primitive roots and how to find them.

Primitive root

We say g is a *primitive root* of n if g generates all of $\mathbf{Z}_n^*$, that is, $\mathbf{Z}_n^* = \{g, g^2, g^3, \ldots, g^{\phi(n)}\}$.

By definition, this holds if and only if $\text{ord}(g) = \phi(n)$.

Not every integer n has primitive roots.

By Gauss's theorem, the numbers having primitive roots are $1, 2, 4, p^k, 2p^k$, where p is an odd prime and $k \geq 1$.

In particular, every prime has primitive roots.

Number of primitive roots

Theorem

The number of primitive roots of a prime p is $\phi(\phi(p))$.

Gauss's theorem shows that p has at least one primitive root. The following lemma show that there are at least $\phi(\phi(p))$ primitive roots. We omit the proof that there are no more than that number.

Lemma (powers of primitive roots)

If g is a primitive root of p and $x \in \mathbf{Z}^*_{\phi(p)}$, then g^x is also a primitive root of p.

Proof of lemma

We need to argue that every element h in $\mathbf{Z}_p^*$ can be expressed as $h = (g^x)^y$ for some y.

- Since g is a primitive root, we know that $h \equiv g^\ell \pmod{p}$ for some ℓ.
- We wish to find y such that $g^{xy} \equiv g^\ell \pmod{p}$.
- By Euler's theorem, this is possible if the congruence equation $xy \equiv \ell \pmod{\phi(p)}$ has a solution y.
- We know that a solution exists iff $\gcd(x, \phi(p)) \mid \ell$.
- But this is the case since $x \in \mathbf{Z}_{\phi(p)}^*$, so $\gcd(x, \phi(p)) = 1$.

Primitive root example

Let $p = 19$, so $\phi(p) = 18$ and $\phi(\phi(p)) = \phi(2) \cdot \phi(9) = 6$.

Consider $g = 2$ and $g = 5$. The subgroups S_g of $\mathbf{Z}_p$ generated by each g is given by the table:

k	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
2^k	2	4	8	16	13	7	14	9	18	17	15	11	3	6	12	5	10	1
5^k	5	6	11	17	9	7	16	4	1	5	6	11	17	9	7	16	4	1

We see that 2 is a primitive root since $S_2 = \mathbf{Z}_p^*$ but 5 is not.

Now let's look at $\mathbf{Z}_{\phi(p)}^* = \mathbf{Z}_{18}^* = \{1, 5, 7, 11, 13, 17\}$.

The complete set of primitive roots of p (in $\mathbf{Z}_p$) is then

$$\{2, 2^5, 2^7, 2^{11}, 2^{13}, 2^{17}\} = \{2, 13, 14, 15, 3, 10\}.$$

Lucas test

Theorem (Lucas test)

g is a primitive root of a prime p if and only if

$$g^{(p-1)/q} \not\equiv 1 \pmod{p}$$

for all $q > 1$ such that $q \mid (p-1)$.

Proof of correctness for Lucas test

Suppose the Lucas test fails for some $q > 1$, $q \,|\, (p-1)$. That means $g^{(p-1)/q} \equiv 1 \pmod{p}$. It follows that

$$\text{ord}(g) \leq \frac{p-1}{q} < p - 1 = \phi(p),$$

so g is not a primitive root of p. Why?

Conversely, if g is not a primitive root of p, then $\text{ord}(g) < p-1$, or equivalently, $(p-1)/\text{ord}(g) > 1$. Hence, the test will fail for $q = (p-1)/\text{ord}(g)$ since then

$$g^{(p-1)/q} = g^{\text{ord}(g)} \equiv 1 \pmod{p}.$$

Problems with the Lucas test

A drawback to the Lucas test is that one must try all the divisors of $p - 1$, and there can be many.

Moreover, to find the divisors efficiently implies the ability to factor. Thus, it does not lead to an efficient algorithm for finding a primitive root of an arbitrary prime p.

However, there are some special cases which we can handle.

Special form primes

Let p and q be odd primes such that $p = 2q + 1$.

Then, $p - 1 = 2q$, so $p - 1$ is easily factored and the Lucas test easily employed.

There are lots of examples of such pairs, e.g., $q = 41$ and $p = 83$.

Number of primitive roots of special form primes

Recall $p = 2q + 1$. We just saw that the number of primitive roots of p is

$$\phi(\phi(p)) = \phi(p-1) = \phi(2)\phi(q) = q - 1.$$

Hence, the density of primitive roots in $\mathbf{Z}_p^*$ is

$$(q-1)/(p-1) = (q-1)/2q \approx 1/2.$$

This makes it easy to find primitive roots of p probabilistically — choose a random element $a \in \mathbf{Z}_p^*$ and apply the Lucas test to it.

Density of special form primes

How many special form primes are there?

We defer the question of the density of primes q such that $2q + 1$ is also prime but remark that we can relax the requirements a bit.

Relaxed requirements on special form primes

Here's another way of generating a prime pair (p, q).

Let q be a prime. Generate numbers $u = 2, 4, 6, \ldots$ until we find u for which $p = uq + 1$ is prime.

[Why do we skip odd u?]

Then $p - 1 = uq$ for small u.

u can be factored by exhaustive search. At that point, we can apply the Lucas test as before to find primitive roots.

How many u must be tried?

By the prime number theorem, approximately one out of every $\ln(q)$ numbers around the size of q will be prime.

While that applies to randomly chosen numbers, not to the numbers in this particular sequence, there is at least some hope that the density of primes will be similar.

If so, we can expect that $u/2$ will be about $\ln(q)$, so u is easily factored for cryptographic-sized primes q.

Functions That Look Random

Random variables revisited

Recall the definitions from lecture 5.

A *discrete probability distribution* over sample space Ω is a function $p\colon \Omega \to [0 \ldots 1]$ such that $\sum_{\omega \in \Omega} p(\omega) = 1$.

A *discrete random variable* is a function $X\colon \Omega \to \mathcal{X}$, where $\mathcal{X}$ is a countable set.

Informally, X is like a variable that assumes different values in $\mathcal{X}$ at different times. Alternatively, a random variable is the process of sampling from $\mathcal{X}$, where $\Pr[\mathrm{X} = \mathrm{x}] = \sum_{\omega:\mathrm{X}(\omega)=\mathrm{x}} \mathrm{p}(\omega)$. Here, p is an associated probability distribution.

An *experiment* consists of making a random selection ω from space Ω, thereby determining the "value" of X.

Random Functions

A *random function* extends the notion of a random variable by allowing additional parameters $x_1, \ldots, x_k$. We'll assume a single parameter $x \in \mathcal{X}$ in the following discussion.

Formally, a random function F takes two arguments, x and ω, and produces a value in its range Y. We write $F(x, \omega) \in Y$.

As with a random variable, ω is chosen by sampling Ω according to an underlying probability distribution p. Thus, an experiment consists of randomly selecting ω from Ω, which then determines the value $y = F(x, \omega)$.

Random functions (continued)

Depending on when the experiment is performed, we can view a random function in two different ways.

1. If we fix x but defer the experiment, then we may write $F(x) = Y_x$, where Y_x is the random variable $Y_x(\omega) = F(x, \omega)$.
2. If we perform the experiment first by randomly choosing ω, then we are in effect choosing a function f_ω from the set of functions

$$\mathcal{F} = \{f_\omega \mid f_\omega(x) = F(x, \omega)\}.$$

Here, f_ω is an ordinary function mapping $\mathcal{X}$ to $\mathcal{Y}$.

Uniform random functions

A *uniformly distributed random function* F from $\mathcal{X}$ to $\mathcal{Y}$ is a random function F where every function f in $\mathcal{X} \to \mathcal{Y}$ is equiprobable.

Equivalently, for each x, $F(x)$ is a uniformly distributed random variable over $\mathcal{Y}$, and the random variables $\{F(x) \mid x \in \mathcal{X}\}$ are mutually independent.

In particular, for any x and randomly chosen ω, each $y \in \mathcal{Y}$ is equilikely to be the value of $F(x, \omega)$, independent of the values of F for other $x' \neq x$.

Example

Suppose random variable $F(k)$ is a biased coin with probability of "heads" equal to $1/k$. Then $F(3)$ is the distribution on coin flips that results in "heads" with probability $1/3$ and "tails" with probability $2/3$. Successive evaluations of $f(3)$ result in a sequence of coin outcomes where, on average, $1/3$ of the outcomes are "heads" and $2/3$ are "tails". $F(4)$ is similar, except fewer "heads" and more "tails" are expected.

Contrast this with a particular random instantiation f of $\mathcal{F}$. Now, $f(3)$ might be either "heads" or "tails", but whatever $f(3)$ is, it is always the same every time $f(3)$ is evaluated. Considering all possible instantations of f, $1/3$ of them will have $f(3) =$ "heads" and $2/3$ of them will have $f(3) =$ "tails".

Cryptographic Hash Functions

Cryptographic use of random functions

Let $\mathcal{M}$ be a message space and $\mathcal{H}$ a hash value space, and assume $|\mathcal{M}| \gg |\mathcal{H}|$. A random function h chosen uniformly from $\mathcal{M} \to \mathcal{H}$ gives a way to protect the integrity of messages.

Suppose Bob knows $h(m)$ for Alice's message m, and Bob receives m' from Alice. If $h(m') = h(m)$, then with very high probability, $m' = m$, and Bob can be assured of the integrity of m'.

One problem with this approach is that we have no succinct way of describing random functions, so there is no way for Bob to compute $h(m')$. The other problem is that h should be chosen anew for each message. Otherwise, there is a small chance being stuck with a bad h (for example a constant function) forever and ever.

Message digest functions

A *message digest* (also called a *cryptographic hash* or *fingerprint*) function is a fixed (non-random) function that is designed to "look like" a random function.

The goal is to preserve the integrity-checking property of random functions: If Bob knows $h(m)$ and he receives m', then if $h(m') = h(m)$, he can reasonably assume that $m' = m$.

We now try to formalize what we require of a message digest function in order to have this property.

We also show that message digest functions do not necessarily "look random", so one should not assume such functions share other properties with random functions.

Formal definition of message digest functions

Let $\mathcal{M}$ be a message space and $\mathcal{H}$ a hash value space, and assume $|\mathcal{M}| \gg |\mathcal{H}|$.

A *message digest* (or *cryptographic one-way hash* or *fingerprint*) function h maps $\mathcal{M} \to \mathcal{H}$.

A *collision* is a pair of messages m_1, m_2 such that $h(m_1) = h(m_2)$, and we say that m_1 and m_2 *collide*.

Because $|\mathcal{M}| \gg |\mathcal{H}|$, h is very far from being one-to-one, and there are many colliding pairs. Nevertheless, it should be hard for an adversary to find collisions.

Collision-avoidance properties

We consider three increasingly strong versions of what it means to be hard to find collisions:

- One-way: Given $y \in \mathcal{H}$, it is hard to find $m \in \mathcal{M}$ such that $h(m) = y$.
- Weakly collision-free: Given $m \in \mathcal{M}$, it is hard to find $m' \in \mathcal{M}$ such that $m' \neq m$ and $h(m') = h(m)$.
- Strongly collision-free: It is hard to find colliding pairs (m, m').

These definitions are rather vague, for they ignore issues of what we mean by "hard" and "find".

What does "hard" mean?

Intuitively, "hard" means that Mallory cannot carry out the computation in a feasible amount of time on a realistic computer.

What does "find" mean?

The term "find" may mean

- "always produces a correct answer", or
- "produces a correct answer with high probability", or
- "produces a correct answer on a significant number of possible inputs with non-negligible probability".

The latter notion of "find" says that Mallory every now and then can break the system. For any given application, there is a maximum acceptable rate of error, and we must be sure that our cryptographic system meets that requirement.

One-way function

What does it mean for h to be one-way?

It means that no probabilistic polynomial time algorithm $A_h(y)$ produces a message m such that $h(m) = y$ with non-negligible success probability.

(Such an m is called a *pre-image* of y under h.)

This is only required for random y chosen according to a particular hash value distribution. There might be particular values of y on which A_h does succeed with high probability.

Hash value distribution

The hash value distribution we have in mind is the one induced by h applied to the assumed distribution on the message space $\mathcal{M}$.

Thus, the probability of y is the probability that a message m chosen according to the assumed message distribution satisfies $h(m) = y$.

This means that h can be considered one-way even though algorithms might exist that succeed on low-probability subsets of $\mathcal{H}$.