MAT 302: LECTURE SUMMARY

For RSA to work, it is necessary for Bob to determine two enormous (100-digit) primes, so large that Oscar would not be able to guess them. How to do this?

One approach is to use a prime-producing algorithm, several of which are known. Perhaps the most famous of these are Matiyasevich's polynomials, which have the curious property that their positive outputs coincide with $\mathbb{P}$, the set of all prime numbers. More precisely, there exists a polynomial $P : \mathbb{Z}^M \to \mathbb{Z}$ such that

$$\{P(\mathbf{X}) \ : \ P(\mathbf{X}) > 0 \text{ and } \mathbf{X} \in \mathbb{Z}^M\} = \mathbb{P}.$$

A concrete, particularly simple (!) example of such a polynomial was constructed by Jones, Sato, Wada, and Wiens in their paper *Diophantine Representation of the Set of Prime Numbers*, Amer. Math. Monthly, vol. 83, 1976, pp. 449-464; it is of degree 20 and in 26 variables:

$$\begin{aligned}
&(k+2)\times\\
&\Big(1-[wz+h+j-q]^2-[(gk+2g+k+1)(h+j)+h-z]^2-\\
&[2n+p+q+z-e]^2-[16(k+1)^3(k+2)(n+1)^2+1-f^2]^2-\\
&[e^3(e+2)(a+1)^2+1-o^2]^2-[(a^2-1)y^2+1-x^2]^2-\\
&[16r^2y^4(a^2-1)+1-u^2]^2-\Big[\Big(\big(a+u^2(u^2-a)\big)^2-1\Big)(n+4dy)^2+1-(x+cu)^2\Big]^2-\\
&[n+l+v-y]^2-[(a^2-1)l^2+1-m^2]^2-[ai+k+1-l-i]^2-\\
&[p+l(a-n-1)+b(2an+2a-n^2-2n-2)-m]^2-\\
&[q+y(a-p-1)+s(2ap+2a-p^2-2p-2)-x]^2-\\
&[z+pl(a-p)+t(2ap-p^2-1)-pm]^2\Big)
\end{aligned}$$

This is a potentially great method of producing large primes, but it is not obvious how to pick values of the variables so that the output is positive and of the desired size.

Another well-known algorithm for generating primes is J. H. Conway's prime-producing machine, which works as follows. Consider the following ordered list of fractions:

$$\begin{array}{cccccccccccccc}
\frac{17}{91} & \frac{78}{85} & \frac{19}{51} & \frac{23}{38} & \frac{29}{33} & \frac{77}{29} & \frac{95}{23} & \frac{77}{19} & \frac{1}{17} & \frac{11}{13} & \frac{13}{11} & \frac{15}{14} & \frac{15}{2} & 55 \\[1em]
A & B & C & D & E & F & G & H & I & J & K & L & M & N
\end{array}$$

We define a sequence s_i by setting $s_0 = 2$, and letting $s_{n+1} = Xs_n$ where X is the first entry in the above sequence such that $Xs_n \in \mathbb{Z}$. Thus the sequence begins as follows:

$$2 \xrightarrow{M} 15 \xrightarrow{N} 825 \xrightarrow{E} 725 \xrightarrow{F} 1925 \longrightarrow \cdots$$

Date: March 17th, 2011.

After a bunch of steps, the algorithm produces 2^2. After a bunch more, it produces 2^3. Then 2^5. Then 2^7, 2^{11}, 2^{13}, etc. In fact, it can be proved that it produces all numbers of the form 2^p, p a prime, in order of increasing p; moreover, no other powers of two are produced. (For proofs of these statements, see the lovely article by R. Guy called *Conway's prime producing machine*, in Math. Mag. **56** (1983), no. 1, pp. 26-33.) Although this 'machine' successfully generates all primes, it is an impractical approach to generating large primes. First of all, it takes longer and longer to produce the next prime; in fact, the process is slower than using the classical sieve of Eratosthenes. Moreover, it is plagued by the same problem the sieve would have: one cannot immediately generate a large prime without first generating all the smaller ones. This makes it unusable for the task of generating 100-digit primes.

Yet another approach, rather less well-known, is a prime generation method discovered by J. M. Gandhi in 1971. Suppose you know the first n primes are $p_1, p_2, \ldots, p_n$. Gandhi proved that the unique integer X satisfying

$$1 < 2^X \left(\sum_{d \mid p_1 p_2 \cdots p_n} \frac{\mu(d)}{2^d - 1} - \frac{1}{2} \right) < 2$$

is p_{n+1}, the next largest prime after p_n. (For a particularly nice proof of this, see Golomb, *A direct interpretation of Gandhi's formula*, Amer. Math. Monthly **81** (1974), pp. 752-754.) Unfortunately, this result is difficult to apply, since it is not clear how to solve the inequality in an efficient way. Moreover, as in Conway's machine, to generate a prime this way one must know all the previous primes.

It remains an important open problem to directly generate large primes. In lieu of this, cryptographers developed an ingenious solution to the problem – rather than generating a large prime, generate a large random odd number (which is easy to do) and then test whether or not it's prime. If it is, you're done! If not, generate another large random odd number and repeat. As you saw in problem 4.4(d), the Prime Number Theorem predicts that this process will hit upon a prime after a few hundred attempts. Thus, the problem of quickly generating a random large prime is reduced to quickly determining whether a given large odd number is prime.

The obvious approach to testing whether a given integer n is prime is to check whether it has any nontrivial divisors. Actually, the obvious approach is extremely inefficient, since one must test divisibility by $O(\sqrt{n})$ integers, an impossible task when n is large. Thus, checking whether n satisfies the *definition* of being prime is impractical.

A different approach to primality testing is to find a property satisfied by primes, and test whether n satisfies this property. If one is lucky, the property might be significantly easier to test than trying to factor n. Here is one example of such a property (which you will prove in problem 5.6):

Theorem 1. $(x+1)^n \equiv x^n + 1 \pmod{n}$ *if and only if n is prime.*

The congruence should be interpreted as congruence of the two functions, i.e. for all inputs x, $(x+1)^n$ and $x^n + 1$ are congruent modulo n. Equivalently, this says that all the coefficients of the two functions are congruent modulo n. For example:

$$(x+1)^2 = x^2 + 2x + 1 \equiv x^2 + 1 \pmod 2$$
$$(x+1)^3 = x^3 + 3x^2 + 3x + 1 \equiv x^3 + 1 \pmod 3$$
$$(x+1)^4 = x^4 + 4x^3 + 6x^2 + 4x + 1 \equiv x^4 + 2x^2 + 1 \pmod 4$$

We thus have an alternative characterization of the primes (alternative to the definition). Unfortunately, this particular characterization is unusable as a primality test for large n, because expanding $(x+1)^n$ requires calculating $O(n)$ coefficients.

How else can we characterize the primes? In other words, can we find a property which the primes have, but which composites do *not*? Playing around a little bit, you might hit upon the following property:

Proposition 2. *$p \nmid (p-1)!$ for all primes p.*

Does this property fail for composites? Well, suppose n is composite. Then we can write $n = ab$, where a and b lie strictly between 1 and n. Most of the time, we will be able to choose $a \neq b$, in which case we see immediately that $n \mid (n-1)!$. If we cannot choose a and b to be distinct, then we must have $n = p^2$ for some prime p (why is this?). One expects that $p^2 > 2p$, in which case both p and $2p$ are in the product $(n-1)!$, whence $n \mid (n-1)!$. This leaves only the case that $n = p^2$ and $p^2 \leq 2p$, in which case $n = 4$. We have proved:

Theorem 3. *$n \nmid (n-1)!$ if and only if $n = 4$ or n is prime.*

This is promising, since we now only have to check divisibility by one number (namely, by n). One immediate difficulty is that when n is large, $(n-1)!$ is far too enormous to calculate precisely. Just as was the case with exponentiation in RSA, we don't actually *need* to compute $(n-1)!$ to check divisibility by n: we need only to compute the value of $(n-1)! \pmod n$. This resolves the problem of dealing with enormous numbers (we reduce modulo n at each stage of the calculation of $(n-1)!$), but fails to address another issue: the number calculations required to evaluate $(n-1)!$, modulo n or otherwise. Recall that in the case of exponentiation, there was a shortcut: we needed to calculate relatively few exponents to be able to construct any exponent we liked (this was the Square-and-Multiply algorithm). Unfortunately, no analogous shortcuts for calculating factorials seem to be known.

A totally different property the primes enjoy is Fermat's Little Theorem:

Theorem 4. *For any prime p and any $a \in \mathbb{Z}_p^\times$, $a^{p-1} \equiv 1 \pmod p$.*

Again, the key question for primality testing is whether this property distinguishes primes from composites. In a certain sense, the answer is yes:

Proposition 5. *$a^{n-1} \equiv 1 \pmod n$ for all $a < n$ if and only if n is prime.*

You will prove this in the Assignment 5. This characterization isn't ideal for primality testing, since it requires checking the condition for $O(n)$ values of a, which is impractical for large values of n. However, this characterization has a decided advantage over the ones discussed previously –

it can be partially verified, giving a *probabilistic* primality test. In other words, rather than testing all $a < n$, one might simply test a bunch of randomly selected values of a. If Fermat's relation holds for all of the a tested, it is likely (but not guaranteed) that n is prime. By contrast, if the criterion fails for even a single value of a, then n *must* be composite. The more a are tested, the greater the likelihood that the test gives the correct output. This primality test is called *Fermat's test.*

As mentioned above, Fermat's test is only probabilistic, depending on how many values of a are used. One might ask about the fewest number of a which must be checked to insure that the Fermat test gives a correct output. Unfortunately, this smallest number can be rather large; it turns out that there exist infinitely many composite numbers n such that $a^{n-1} \equiv 1 \pmod{n}$ for all $a \in \mathbb{Z}_n^\times$. Such n are called Carmichael numbers; the smallest example is 561. That there are infinitely many such numbers was proved in 1994 by W. R. Alford, A. Granville, and C. Pomerance in *There are infinitely many Carmichael numbers*, Annals of Mathematics 139: pp. 703-722. Fortunately, Carmichael numbers are quite rare, so the Fermat test is fairly reliable. Most importantly, it's extremely fast!

Over the past few decades there have been many ingenious ideas of how to improve Fermat's test (i.e. to increase the likelihood that Fermat's test correctly identifies a prime). One such improvement is called the *Miller-Rabin* test, which will describe below, after discussing a bit of background.

Earlier in the course we studied congruences of the form $x^e \equiv y \pmod{N}$. Recall that the method we developed for solving such a congruence depends on inverting $e \pmod{\varphi(N)}$, which in turn depends on e being coprime to $\varphi(N)$. Since $\varphi(N)$ is even for all $N \neq 2$, we cannot employ our attack when the exponent e is even. For example, consider the congruence

$$(\dagger) \qquad x^2 \equiv 1 \pmod{N}.$$

Certainly, $x \equiv \pm 1 \pmod{N}$ are solutions, but are they the only ones? A bit of experimentation shows that there can be others. For example, for $N = 8$, (†) has the four solutions $x \equiv 1, 3, 5, 7 \pmod{8}$. When N is prime, however, this congruence is easy to solve:

Proposition 6. *Let p be prime. The only solutions to the congruence $x^2 \equiv 1 \pmod{p}$ are $x \equiv \pm 1 \pmod{p}$.*

Proof. If $x^2 \equiv 1 \pmod{p}$, then $p \mid x^2 - 1 = (x+1)(x-1)$. From a previous problem set, this implies that either $p \mid x+1$ or $p \mid x-1$; these correspond precisely to the statement that $x \equiv \pm 1 \pmod{p}$. □

With this result in hand, we're ready to discuss the Miller-Rabin primality test. Our task is to determine whether a given (large, odd) integer n is prime. We begin by applying Fermat's test: for a bunch of a, we check whether or not $a^{n-1} \equiv 1 \pmod{n}$. If the congruence fails to hold for even a single choice of a, we know that n must be composite. But what if for all the a we try, the congruence holds? Fermat's test is inconclusive, so we must apply a more precise test. Note that if n were prime, then by Proposition 6 we would have

$$(\ddagger) \qquad a^{(n-1)/2} \equiv \pm 1 \pmod{n}.$$

This provides a back-up test to Fermat: even if some composite slips through Fermat's test, if it fails (‡) then it cannot be prime. And if n satisfies both Fermat's test and (‡) for all a tested, then it's very likely to be prime. Still, the conclusion that n is prime will not always be correct.

What I've just described is the simplest case of Miller-Rabin. If $\frac{n-1}{2}$ is even and it turns out that

$$a^{(n-1)/2} \equiv 1 \pmod{n}$$

then we can apply Proposition 6 once again, and conclude the if n were prime,

$$a^{(n-1)/4} \equiv \pm 1 \pmod{n}$$

would have to hold. In general, we could continue in this way until either the exponent is no longer even, or until the power of a on the left hand side is congruent to $-1 \pmod{n}$. This leads to the following formal statement of the Miller-Rabin primality test. Please make sure that you understand why this is equivalent to the description above.

Theorem 7 (Miller-Rabin primality test)**.** *Given a large odd number n, write*

$$n - 1 = 2^e m$$

where m is odd and $e \geq 1$. If we can find an integer $a \leq n-1$ such that

- $a^m \not\equiv 1 \pmod{n}$*, and*
- $a^{m2^k} \not\equiv -1 \pmod{n}$ *whenever* $0 \leq k \leq e-1$,

then n is composite. Otherwise, n is probably prime.

Both Fermat's test and the stronger Miller-Rabin test can easily turned into efficient (i.e. polynomial-time) algorithms; however, both suffer from being merely probabilistic. Despite this, Miller-Rabin is in wide use, because the probability of a false positive is extremely low, and it runs extremely quickly.

In the early 2000s, a professor and two undergraduates made an important breakthrough:

Theorem 8 (Agrawal, Kayal, Saxena)**.** *Primality testing can be accomplished in polynomial time.*

Moreover, in their paper (*Primes is in P*, Annals of Mathematics, 2004) they give an explicit algorithm which *deterministically* decides whether a given input n is prime or composite in time $O\Big((\log n)^{7.5}\Big)$. The idea is based on Theorem 1 above, which asserts that $(x+1)^n \equiv x^n + 1 \pmod{n}$ if and only if n is prime. First, from that theorem we deduce the following more general statement:

Corollary 9. *For any $a \in \mathbb{Z}_n^\times$, we have $(x+a)^n \equiv x^n + a \pmod{n}$ if and only if n is prime.*

Proof. We shall prove that $(x+1)^n \equiv x^n + 1 \pmod{n}$ if and only if $(x+a)^n \equiv x^n + a \pmod{n}$.

($\Longrightarrow$)

Note that $(x+2)^n = \Big((x+1)+1\Big)^n \equiv (x+1)^n + 1 \equiv x^n + 2 \pmod{n}$. Similarly, suppose $(x+j)^n \equiv x^n + j \pmod{n}$ for all $j < a$. Then

$$(x+a)^n = \Big((x+a-1)+1\Big)^n \equiv (x+a-1)^n + 1 \equiv x^n + a \pmod{n}.$$

Thus, by induction, if $(x+1)^n \equiv x^n + 1 \pmod{n}$ then $(x+a)^n \equiv x^n + a \pmod{n}$ for all $a \in \mathbb{Z}$.

($\Longleftarrow$)

Suppose $a \in \mathbb{Z}_n^\times$, and $(x+a)^n \equiv x^n + a \pmod{n}$. Then

$$(x+2a)^n \equiv (x+a)^n + a \equiv x^n + 2a \pmod{n}$$

and similarly, $(x+ka)^n \equiv x^n + ka \pmod{n}$ for any positive integer k. Since $a \in \mathbb{Z}_n^\times$, there is an integer b such that $ab \equiv 1 \pmod{n}$, whence

$$(x+1)^n \equiv (x+ab)^n \equiv x^n + ab \equiv x^n + 1 \pmod{n}.$$

$\square$

This can now be made into a probabilistic primality test: plug in a bunch of random values of $a \in \mathbb{Z}_n^\times$ and $x \in \mathbb{Z}_n$ and see whether the congruence $(x+a)^n \equiv x^n + a \pmod{n}$ is satisfied. If it fails to hold for even one pair (a, x) then n must be composite; otherwise, if a bunch of random examples all satisfy the congruence, then n is likely prime.

But Agrawal-Kayal-Saxena's goal is to create a deterministic primality test, not a probabilistic one. Recall that the main problem with using Corollary 9 as a primality test is that it takes too long to expand $(x+a)^n$ when n is large. Agrawal, Kayal, and Saxena had the insight that one can reduce $(x+a)^n$ not only modulo n, but also modulo a *polynomial*, thus getting a polynomial of smaller degree. (We will write $f(x) \equiv g(x) \pmod{h(x)}$ to mean $h(x) \mid f(x) - g(x)$.) Thus, we examine conditions under which

$$(x+a)^n \equiv x^n + a \pmod{n, x^r - 1}$$

where r is a positive integers which will be determined. Although in general this type of reduction kills the iff statement of Corollary 9, A-K-S proved that for very carefully selected values of r one has that $n = p^k$ if and only if $(x+a)^n \equiv x^n + a \pmod{n, x^r - 1}$ for every $a \leq \sqrt{r}\log n$. It turns out that one can produce an appropriate r without especial difficulty, which will be small enough that we won't have to check too many values of a, yet big enough so that $(x+a)^n$ reduced modulo $x^r - 1$ has substantially smaller degree, and is thus easier to work with.

All this gives a polynomial-time algorithm for testing whether n is a prime power. Finally, a bit of thought shows that it's easy to distinguish between primes and prime powers in polynomial time. This leads to a polynomial time, deterministic primality test, now known as the AKS test. Unfortunately, we won't have time to go more in-depth into the work than this; however, the original paper is very readable, and is widely available online. Check it out!