Cryptography Mini Lecture

Newcomers to number theory are sometimes heard to say, "Prime numbers and number tricks may be interesting to some, but what practical use do they have." In other words, I occasional hear students say, "Why would anyone care about this?"

My first general personal response is that an ability to think in a logical, abstract and structured way is a valuable skill in life. Mathematics, in particular number theory, allows us to explore how far we can push ourselves to think through and solve problems. Certainly, integers are at the heart of many mathematical problems and a complete and full understanding of their properties is not wasted. The understanding of these numbers and generalizations from their properties extends to all areas of mathematics from the very pure to the very applied. But this response is too philosophical for some people.

My response to people that want *practical* value (I guess that means something that can turn a profit) in everything they learn is that if you need a profit in order to learn something, then let me give you a quarter and you can be on your way or you can stay and listen, but I hope you find what I'm offering is more satisfying than money. Money and profit is great, but having the skill to logically think through, structure and solve problems is more valuable to me. Certainly, if money was my only goal, I would have taken other job offers and I wouldn't be a lecturer at UW. Number theory is the one of the oldest of mathematical disciplines and many problems have been passed down through generations. Some eventually get solved and some still have yet to be conquered. It is these problems that give the ultimate test of the limits of your true ingenuity against the great minds of centuries past. (Alright Dr. Loveless, you still are being too philosophical, get to the nitty gritty.)

If these are not good enough reasons to know a bit about number theory, then here is one 'concrete' reason (among many others I could choose from) that each of you that use technology MUST be interested in. Cryptography is the study of how to send a message from a sender, Alice, to a receiver, Bob, in such a way that an eavesdropper, Eve, cannot read the message. In other words, cryptography is the study of how Alice can 'jumble up' (encode) a message in such a way that Bob can decode it, but Eve cannot.

In the modern era of the internet, this is a real daily issue. Imagine the following problem: I want to buy a song from iTunes (actually I'm too much of an old man, because I've never done this, but perhaps you have, if not iTunes, make it amazon.com, or ebay, or any other website). I enter my credit card for the purchase and click submit. My credit card number then travels out of my computer over my internet cables, through a router, to my ISP server, back out to the iTunes server and to the iTunes processing computer, then to the bank to verify the number. Thus, my credit card number passes through several servers and other computers. What is to stop a person running one of those other computers from viewing my credit card and using it for their next purchase. Well, my credit card number must have been encrypted. But how could it have been encrypted, I have never personally met with the internet security director of iTunes to discuss how we would secretly send credit cards. So how was this done? Our computer must have encoded our credit card number in some general and publicly known algorithmic way that only iTunes could decode it.

If this were 40 years ago, then there would be NO method to do this. Cryptography is an old science with methods that date back centuries, but all the old methods required that sender and receiver had agreed upon a secret key for a method they both knew. In the late 1970's, mathematicians and computer scientist devised methods that could be exploited where the encryption method could be public to everyone, but the only person with the secret key was the one who made the encryption key (the key iTunes made). This was called public key encryption and is largely based on prime numbers properties.

RSA Cryptography

RSA Cryptography was published in a paper by Rivest Shamir, and Adleman in 1978 (one of my advisors in undergrad had to get permission from the government to read this mathematical paper because they had classified it). This is one of the most widely used public key encryption methods. There are other public key encryption methods and a large number of them exploit number theory, in particular prime numbers, in some way. The RSA method is as follows:

1. PRIVATELY:
   (a) The computer randomly finds two 'large' prime numbers, p and q.
   The primes numbers are more than 100 digits long, but there are good, fast algorithms for finding primes numbers of this size. (My dissertation was related to some of these methods)
   (b) We calculate $n = pq$.
   (Multiplication can be done quickly even for very big numbers).
   (c) We randomly pick a number e that is relatively prime to $(p-1)(q-1)$.
   (d) We solve $ed \equiv 1 \pmod{(p-1)(q-1)}$ for d. (The Euclidean algorithm does this quickly).
2. PUBLICLY: We publish the numbers n and e on the internet for everyone to see.
3. ENCODING: To encode message m, we calculate $c \equiv m^e \pmod{n}$. (Using successive squaring).
4. DECODING: To decode message c, we calculate $m \equiv c^d \pmod{n}$. (Using successive squaring).

Note that only the original person knows d, so the original person is the only one that can decode. To find d, you have to know $(p-1)(q-1)$, so you need to know p and q. The numbers n and e are public and you, the creator of your particular code numbers, are the only one that knows p and q, but it is computational 'impossible' to find p and q if n is 200 digits or longer. This is why this method is so secure because it is so difficult to factor large integers even with computers.

The fact that the decoding works is a consequence of Fermat's little theorem (that you proved in problem 6.37b). Remember that Fermat's little theorem states that if p is a prime and a is an integer, then $a^p \equiv a \pmod{p}$. If in addition, $\gcd(a,p) = 1$, then we can cancel and a to get $a^{(p-1)} \equiv 1 \pmod{p}$. Here is a quick proof that $m \equiv c^d \pmod{n}$.

Proof: Since $ed \equiv 1 \pmod{(p-1)(q-1)}$, by definition, $(p-1)(q-1)$ divides $ed-1$. Thus, there exists an integer k such that $ed - 1 = k(p-1)(q-1)$, so $ed = k(p-1)(q-1)+1$. We know from the definition of the encoding that $c \equiv m^e \pmod{n}$. Hence, by the replacement theorem, $c^d \equiv (m^e)^d \equiv m^{ed} \pmod{n}$. Since $ed = k(p-1)(q-1)+1$, we have $m^{ed} = m^{k(p-1)(q-1)+1}$ which we will rewrite in two equivalent ways as $(m^{(p-1)})^{k(q-1)}m^1$ and $(m^{(q-1)})^{k(p-1)}m^1$. By Fermat's little theorem, $m^{ed} \equiv (m^{(p-1)})^{k(q-1)}m^1 \equiv (1)^{k(q-1)}m \equiv m \pmod{p}$ and $m^{ed} \equiv (m^{(q-1)})^{k(p-1)}m^1 \equiv (1)^{k(p-1)}m \equiv m \pmod{q}$. Hence, p divides $m^{ed} - m$ and q divides $m^{ed} - m$ and since $\gcd(p,q) = 1$, we have $n = pq$ divides $m^{ed} - m$. Thus, $c^d \equiv m^{ed} \equiv m \pmod{n}$. $\square$

Small Nonsecure Example. Here is a small example that completely illustrates all the steps of the RSA algorithm, as you go through it imagine that in practice the primes are much, much larger, but the computation steps are still fast (except for it becomes impossible for the eavesdropper to factor n to find p and q).

1. PRIVATELY:
   (a) $p = 7$ and $q = 13$
   (b) $n = 7 \cdot 13 = 91$
   (c) $e = 5$
   (d) To solve $5d \equiv 1 \pmod{(7-1)(13-1)}$ for d.
   We want $(7-1)(13-1) = 72$ to divide $5d-1$. In other words we want to solve $5d - 1 = 72m$ for integers d and m which is the same as $5d + 72(-m) = 1$. We can solve this with the Euclidean algorithm.
   - Euclidean Algorithm:
   $$\begin{aligned} 72 &= 14 \cdot 5 + 2 \\ 5 &= 2 \cdot 2 + 1 \\ 2 &= 2 \cdot 1 + 0 \end{aligned}$$
   - To find one solution to $5d + 72k = 1$ (by Back Substitution):
   $$\begin{aligned} 1 &= 5 - (2)2 \\ 1 &= 5 - 2(72 - 14 * 5) \\ 1 &= 5 - 2 \cdot 72 - 28 * 5 \\ 1 &= 5(29) + 72(-2) \end{aligned}$$
   Thus, one solution is $d = 29$ and $k = -2$.
   - Thus, $d = 29$. (A computer can be programmed to do this very quickly even for extremely large numbers).
2. PUBLICLY: $n = 91$ and $e = 5$
3. ENCODING: To encrypt our mini-credit card number: $m = 53$, we calculate $c \equiv 53^5 \equiv 79 \pmod{91}$. Send the encrypted message $c = 79$.
4. DECODING: To decode message c, we calculate $m \equiv 79^{29} \equiv 53 \pmod{91}$.

   In this case, this wouldn't be secure because it wouldn't take long for someone to figure out that $91 = 7 \cdot 13$. Once the eavesdropper knows that $p = 7$ and $q = 13$, then they can do all the same things you did in private to find d and decode.

Now imagine making the initial primes were bigger.

For example $p = 906845800845204585085213995793$ and $q = 455907616696362673680389793 79$ (these are 30 digit long primes that mathematica gave me at random, mathematica can almost instantaneously find random primes this large). Multiplying these numbers together gives

$$n = 413437907774441573910870021210012365731915494864621197525 47.$$

So n has 60 digits. Certainly, if I only gave you n you would have a hard time finding p and q with pencil and paper. Even with a computer and a basic checking algorithm it would be difficult. If you starting testing sequential for divisors you would have a lot of numbers to test 2, 3, 4, 5, etc. If every check took 1 billionth of a second and you had to check all the numbers from $k = 1$

to $k = n$ that would do n billionth of a second calculations and n times 1 billionth of second is still 413437907774441573910870021210012365731915494864 62.119752547 seconds (this amount of time is many billions of billions of billions times the age of the universe). So if we had to check sequential that would be impossible. There are some factoring methods that are faster than sequentially checking, but they still take a while. The basic factoring method on Mathematica on my computer took about 4 minutes to factor this 60 digit number, so it doesn't take the age of universe to factor a 60 digit number. The n I gave above would not be secure, my computer would be able to break the code (*i.e.* factor n) in about 4 minutes. A 60 digit number is roughly the limited of what can be factored 'quickly' with known methods, as the numbers get larger factoring starts to slow down a lot.

So then I picked two 50 digit primes at random (again mathematica did this very fast). And I multiplied these primes and got

$$n = 130088460220590899340559900197761055602844002772842723 1$$

$$061861218804456963251078522429798303489928103$$

I asked mathematica to factor this number. I let it run for 24 hours and it never finished (it would would have taken days/weeks if it could keep going, in actuality it would have run out of memory space or crashed before it ever could have finished). With this being said, the number above is still not secure. With a slightly faster method and a faster computer, the factorization could be given in less than a day. However, as soon as we get to n being say 200 digits, then it becomes nearly impossible for current computers to factor in our lifetime. If this isn't secure enough, we can have n be say 500 digits. Even if n has a million digits it doesn't take that much more computational time to created (only an extra couple seconds to create), but the amount of time that it takes to break the code (factor) grows exponentially. To factor an integer n that has even a thousand digits would certainly take many times the age of our universe to finish factoring.

It is this property that large keys are quick to generate, but factoring of large numbers is very computational difficult, that makes RSA encryption so secure. The security is somewhat enhanced by the fact that the factoring problem has been studied for hundreds of years and no one has come up with a fast factoring method (and there is strong evidence that no such algorithm exists, although some think that a quantum computer, if one can be created that can be appropriately programmed, will be able to factor quickly, but even then code makers have designed methods to encode using quantum computers if this ever happens).