PROFESSOR: So let's get ready. If good, you should receive some handouts. So the TAs are walking around, so you should slowly get those. This is the second lecture on number theory, and we're going to cover for a lot of stuff. And actually, we're going to start with encryption, which is an application of number theory. And we'll take that as a theme throughout the whole lecture. And so, in this way, you can see how useful number theory is.

Now, encryption-- yeah, what is it? So let's first talk about it a little bit. Maybe some of you have heard about it. Cryptology in general is the art of hiding information. And encryption is a very useful tool. I'll only give a very high level overview. I mean, if you really want to know more about this, you should do a class in crypto, or practical security. So what's encryption?

The idea is usually that, beforehand, we're going to share a whole bunch of keys. So keys are exchanged between a receiver and a sender. And the whole idea is that if I want to transmit, like, a message to one of you, well, there could be someone in the middle who wants to intercept my message, and wants to find out what I'm saying. So I do not like this. And in order to avoid it, I will use encryption, which is some kind of algorithm that actually transforms my message m-- my plain message-- by using some kind of algorithm, E. Some encryption algorithm-- it can be a function as well-- that uses the keys in order to transform this into an encryption, which we call m prime here.

So the plain text is, in the clear, all the information that I want to convey to one of you, for example. And m prime is somehow a complete-- well, a mixture of bits, out of which I'm not able to distill any information about the plain text. So encryption to a very special kind of thing. We're not going to talk about the precise definitions here, but we just take the ID into this lecture.

Now, decryption is transforming the cypher text back into the plain text. So we will start with the encryption, m prime, and we have some kind of decryption algorithm. And again, we can make use of the keys we exchanged. And we transform it back into the plain message. So only

if I know the keys, I can actually transform back the encrypted version into the plain version. So in encryption schemes we like both these algorithms to be is really efficient. And the security of such a scheme-- well, the first kind of intuition that we can get is well, if I'm a man in the middle somewhere, intercepting an encrypted message-- m prime-- I should not be able to get any information about m if I have no knowledge about those keys.

So this is the example that we're going to take throughout this whole lecture. So let's start with a first possible scheme. Turing, who lived around 1936-- he was 24 years old-- and he lived to about-- actually, I think he was about 54 when he died. In any case, Turing was the one who first originally proposed to use number theory in cryptography. And before he joined the British army, before the Second World War, he actually proposed a scheme, but it got never published. So here in this class, we are going to try to think about what he could have thought about.

So we will have a first scheme, which we will call Turing's code, version number one. And the whole idea is that we're going to translate a message first of all into a prime number, because we want to use numbers. We're here in number theory class, so we want to use some tricks with numbers in order to encrypt it. So let's do this. So for example, let's take the word victory. We can map this into an integer. For example, we could say, well, m is 22, where I map V to-- because I know V is the 22nd letter in the alphabet, I just start with 22. I is the ninth letter in the alphabet, so I append 09. C is the third letter in the alphabet, I append 3-- 03. I continue like this and in end, over here I've mapped Y to the 25th letter. Because it's the 25th letter of alphabet, I write 25.

It turns out, I can just add a couple of digits more, if I specially compute those. In this case, I could add 13, and then it changes into a prime number. Now we are not going to talk about why this is, and how this can be done, and so on. But it turns out that the prime numbers are densely distributed over the integers, and it's really possible to, just with a few extra digits-- by selecting them in a smart way-- to actually create a prime number. And also verify that it is a prime number very efficiently. So it's very easy to compute-- to translate such a word into a prime number.

So this is how it all starts. And just like in an encryption scheme, beforehand we are going to exchange a key. So we exchange the secret prime in this example, which we call k. And the encryption is very simple. We are just going to multiply m with k. Now, you may wonder why this is such a fantastic idea. But let's just bear with me. So m is this first prime number, and we

multiply it by a second prime number, and that's going to be our encryption.

Now how do we decrypt? That seems to be pretty straightforward, right? How do we do it? We start off with m prime. I know we have exchanged this secret prime k, so if I receive from you this message-- this encrypted message-- well, I know the key, k. So I just divide it by k. Well, which is m k divided k, and I get m. So that's pretty straightforward. Now, as it turns out actually-- and I'll write it up here, because we need it later-- that it's not so trivial to actually just give an m prime to figure out what m is, or k. m prime is a product of two very large prime numbers, and that turns out to be a really hard problem. Up to now, nobody has really been able to get a really efficient algorithm to solve that.

So we may think this is secure. So let me write it down It's hard to factor a product of two large primes. You will actually need this also, when we come to the final encryption scheme-- RSA that we will discuss, which is widely used. But something is wrong here, though. This seems to be too simple, right? So what can we do if we have like, say, suppose I intercept two encrypted messages. What can I do? So suppose I have a first message, m prime 1, which is the product of a first plain message times the key, k. And I have a second message that is encrypted by using the same key, which is m 2 times k. Does anybody have an idea what I could do here?

AUDIENCE: Find the GCD and that would give you the key.

PROFESSOR: Yeah, you could find the GCD of m 1 prime and m 2 prime. I've intercepted those two. Now, m 1 is a prime number. k is prime number, and 2 is a prime number. k is a prime number here, also, right? So they're all relatively prime towards one another. The GCD of these two-- well, the greatest common divisor is k. So just by calculating the GCD of the two encrypted messages, I'll be able to figure out what k is-- the key. Well, if I know the key, then I can do the decryption of any cypher text, any encryption of a message that I can intercept. So this is not secure.

So how can we change this? Can we create a different kind of encryption scheme? Let's do something much more difficult, and then we will get into modular arithmetic and things like that. So let's do this. So Turing's code, version number two-- we try to do something much more complicated than just multiplying by prime. So let's do the following. So beforehand, we're going to exchange not only a secret prime k, but we will also exchange a public prime. So we exchange-- by public we mean that anybody can see this prime. It's common knowledge. A public prime p, and also a secret prime k. Let's see whether this would work.

We have encryption. Well, we're going to start out exactly the same way. First of all, I should tell you how a message is represented. The message is going to be represented as a number, m, in the range from 0, 1, all the way to p minus 1. And we will compute the encryption as follows. m is going to be the remainder of m times k, after dividing out as many multiples of p as possible. So notice we do kind of the same thing. We just multiply by k, but now we just take the remainder after taking out as many multiples of p.

Well, let's see whether we can do the decryption. It seems to be, like, a next level of complexity. So maybe that'll help us here, right? So how will we do decryption? Well, somehow we would like to divide by k. But we cannot really do that, right? This does not make any sense. So the decryption-- we have no idea at this point how to do this. And now we can get into modular arithmetic, because it turns out that we can sort of divide by k. There exists what we call multiplicative inverse of k modulo p. And I will explain all those terminologies to you. And then we will be able to take m prime, and transform it back to m. So we will be able to get a lot of this machinery going.

So let's find out how this works. So first of all, last time we saw that we defined a and b to be relatively prime. So let me repeat that. So a and b are relatively prime if and only if-- that's how we define it-- if the GCD of a and b is equal to 1. And in last lecture and in the recitation, you got a different proof, I think. And we proved that, actually, the GCD of a and b is equal to the smallest positive linear combination of a and b. So that means that one, in particular, is a linear combination of a and b. So there exists integers-- s and t-- such that s times a, plus t times b equals 1. It turns out that it can also go the other way around, because if I can write one as a linear combination of a b, well I cannot get much lower than that, right? So that's really the smallest possible that I can achieve. So the GCD must be equal to 1. So this is a property that we will be using.

And from this property, we can already figure out an interesting property. So suppose we have a linear combination that looks like this. Then you can imagine that, sort of, that s times a is equal to 1, plus or minus some linear multiple of b. So it's sort of-- s times a is sort of equal to 1, you can say, up to a multiple of b. So you can see a sort of as an inverse of s. Because s times a is equal to one, sort of.

So this is what we're going to use-- this kind of feeling. And in order to do that, we are going to define congruency. So that's the first definition for this lecture. We say x is congruent to y modulo n, if-- which we denote as follows-- x with three bars, y in between brackets, mod n.

And we say that this is the case if n divides the difference between x and y. So let's have a look at some examples.

So let's take 31, and I would like to show to you that this is congruent to 16 modulo, between brackets, 5. Why is this? Well, I take the difference between 31 and 16, which is 15, and I know that 15 is 3 times 5, so 5 divides this difference. And then by definition, we can write it like this. And we say-- that's the definition-- 31 is congruent to 16 modulo 5. Another example is-- no, we will stick with this example. It's pretty clear. So once we have defined this, we can continue and talk about this inverse that I was talking about. So we like to, sort of, explain in this encryption scheme how we can divide by k. Actually, we would like to multiply by an inverse of k. And we're going to use this framework.

So we'll have a new definition that talks about the multiplicative inverse. So it's a new concept. And you'll give a couple of examples. So the multiplicative inverse of x modulo n is a number, which we denote by x and then minus 1 on top of here. It's a number in the interval 0, 1, all the way up to n minus 1. Such that x times x inverse-- so x times its multiplicative inverse-- is congruent to 1 modulo n. So this is the definition for a multiplicative inverse.

So let's have some examples. So let's do that over here. For example, we have that 2 times 3-- which equals 6-- is equal to 1 modulo 5. Why is this? Well, 6 minus 1 is divisible by 5, so I know is congruent to 1 modulo 5. So what does this mean? Well, we can say that two is actually equal to the multiplicative inverse of 3 modulo 5. We can also say that write 3 is the multiplicative inverse of 2 modulo 5. Let's have another example, just to make it a little bit more clear. We know that 5 times 5 equals 25, and this is congruent to 1 modulo 6, because 25 is 1 plus 4 times 6. So now we see something funny happening, because 5 is actually equal to its own multiplicative inverse modulo 6. So are there any questions about these concepts? Because these are really basic for the whole lecture, and this is what you really need to understand if you do all the problem sets as well. Are there any questions?

So now we can actually start talking about this second version of the Turing code that we invented. Let's have a look at this remainder. So let's write it out again. The remainder of m times k, after dividing out as many multiples of p. Well, we know that this is congruent to m times k modulo p. So why is this? Well, we just apply the definition over here. We take the difference between those two. So here we have the remainder of m times k, after dividing out as many multiples of p. And if you subtract that from m times k, well then we have something that is a multiple of p, because that's what we divided out.

So the difference is a multiple of p, so that means that p divides the difference. And that's the definition of saying that this remainder is congruent m k modulo p. So this is kind of interesting, because now we can rewrite over there-- well, not really rewrite, but we can use this to analyze the encryption over here. So m prime is equal to this remainder. And we get a beautifully equation. We see that the encryption is congruent to the plain message, times the key modulo p.

So how do I do decryption? I can use the multiplicative inverse of k, right? So then I can divide k out. So let's write this out as well. So suppose I have a multiplicative inverse-- k to the power minus 1-- that is congruent to 1 modulo p. Why do I do this? Why am I writing this out? Because you will see that it is not always possible to have a multiplicative inverse. That's going to be really a big problem. And that's where all these other functions come in-- the Euler totient function and Euler's theorem, and so on.

So it's not really always the case. We'll give an example in a moment. But suppose that we have a multiplicative inverse-- modulo p-- well, then I'm able to easily compute. I take m prime, I multiply it with k inverse. Well, I'm substituting for m prime, m times k-- times k inverse. I still have this left. Now I can say that this is equal to 1 modulo p. So this is equal, or is congruent, to m modulo p. So now we see how we can do decryption. We simply use the multiplicative inverse of k. And as in the first Turing code, we are able to somehow divide out-- so where did I have it-- to divide out k. I did it very differently-- I have a very different mathematical structure-- but the idea is, essentially, the same. I have a multiplicative inverse of k, and if I multiply this with the encryption, I will get the plain message, modulo p.

So, am I finished now is the question. Well, I know that m is in the range of 0, 1, all the way up to p minus 1. So this means that I can also rewrite m as-- and I use a similar trick as what I did over here-- I can rewrite m as they remainder of m prime times k inverse, after dividing out as many copies of p as possible. So what we did here is to first prove that the difference between those two is a multiple of p. That's essentially what congruent modulo p means-- that's the definition. So now that I know that the difference is a multiple of p, and if I know that m is actually in this range up to p minus 1, I can use what we learned last lecture-- and what the book was talking about-- that the definition of the remainder of m prime times k inverse-- this thing-- after finding out as many copies of p, is exactly this plain message m.

OK, so now we the decryption. So this is decryption. And that sounds great. So now, of course,

we are wondering, well, if we can do this, can we also attack this scheme, right? Can we do something bad with it? Well, it can be used to break this code, but in a slightly more complicated way.

So let's see where I put that. Right, so what we are going to use, now-- so when we talk about security, and so on, you can think of all kinds of ways to break an encryption scheme. So we started out with, in the first version, well, what if I just know the encryption? And then, well, I cannot know anything about the plain text, simply because I know that it's very hard to factor a product of two large primes. Then I said, but suppose I know a little bit more as an adversary. Suppose I have a plain message together with an encrypted message. Well, then I could do this GCD trick and figure out and break the scheme. And now we're going to do something similar. And we say, well, suppose that I do not know two encryptions, but I know a plain message. And in corresponding, encrypted message. So if I know such a pair, which is-- I mean, in practice, such type of information will be leaked. Then in this case, I can break it.

So we call this the known the plain text attack. And it assumes that we know-- as an adversary, I know a message-- a plain message, m-- and also an encryption of this message, m prime. And m prime, according to this scheme, is the remainder of m times k, after the dividing out as many multiples of p as possible. Now, we saw-- now, suppose I know these two. I'm going to show you how to break it. So let's have a look.

The encryption, m prime, is congruent to m times k modulo p. We just proved it over here. And what do we know? We know that p is a public prime. I know p. Since it's a prime, I know that the GCD of m and p equals 1. So these two are relatively prime. So now if they are relatively prime-- and that's what we wrote up here-- we know that there exists this linear combination of, in this case, m and p that is equal to 1. And this way we can figure out how to compute the inverse-- the multiplicative inverse of m.

So that's what we're going to do. So we can compute and the multiplicative inverse, such that m times m inverse is congruent to 1 modulo p. So now we can do the next step. So what could I do next? So let's see, if I have such an inverse, then I can take my encrypted message that I have as an attacker, which is m prime. I know m, and since p was public, I can compute this multiplicative inverse. So I could just compute this product, and I know that this is actually equal to, well, k times m times m inverse. This is again equal to-- congruent to 1 modulo p. So this is equal to k modulo p.

So now, all of a sudden, I know k modulo p, because I know those two. So if I know k, I can compute its multiplicative inverse. So I can compute k inverse modulo p. And if I know k inverse, again, I can use it to decrypt any other encrypted message that I receive. So for all future encrypted messages, I do not need anymore the plain messages. I've already used-- just by using one plain message encryption pair, I have been able to compute the secret key, and the whole scheme is broken.

So security is kind of an interesting science. We can always think of more tricks to break schemes, or other assumptions, that we haven't thought about before. And so now, let's talk about-- so what did we do here? We have been talking about encryption-- these two schemes. We talked about modular arithmetic. We talked about congruence, and also the multiplicative inverse. And we showed how to use that to break Turing's code, version number two.

So we need something much more fundamental in order to create a scheme that is really secure. And that's what we're going to do now. We're going to start off with Euler's totient function, and improve a related theorem. And with that we will be able to actually explain RSA algorithm, which is a famous algorithm invented here at MIT in 1977, by Rivest, Shamir, and Adleman. And they actually also got the Turing Award for this a few years ago. And it's widely used in practice. But we will be able to actually explain this algorithm with just this fundamental piece of number theory. So that's really exciting. So let's do this.

So we're going to first define Euler's totient function. And it's related to this multiplicative inverse. This function is denoted phi of n, and it denote the number of the integers in 1, 2, 3-- all the way up to n minus 1, that are relatively prime to n. So this just drops out of the air, you may think. But this is a fundamental quantity, and Euler's theorem is what we will try to prove next. But let's give a couple of examples about this, just to see how it works.

So for example, let's take n to be equal to 12. What would be the value of Euler's totient function, evaluated for 12? So if you take 12, we have the following numbers to consider. 1, 2, 3, 4, 5, 6, 7, 8, 9, 10,11, and 12. So now, let's have a look. You want to count the integers that are relatively prime to 12. Well, 1 is relatively prime to 12. Why is that? Because to GCD of 12 and 1 is equal to 1, and that's the definition of being relatively prime. So this is a good number. The GCD of 12 and 2, well both are divisible by 2. So they're not relatively prime, because the GCD is at least two. In this case, equal to 2. 3 also divides 12. 4 divides 12. 5 is relatively prime with respect to 12. 6 actually is dividing 12 as well. 7 is relatively prime, and 11 over here as well. And all the others have a greatest common divisor that's larger than 1.

So here we see that the Euler's totient function evaluated in 12 is equal to 4. There are one, two, three, four integers that are relatively prime. OK, so let's have a different one. Say, n equals 15. It's little bit different, because here you may think this is kind of coincidental that 1, 5, 7, and 11 are actually, also, primes. But for 15 it looks a little bit different. So let's write out all the numbers.

Again, let's have a look. Well, one is the greatest common divisor equal to 1, with respect to 15. 2 is also relatively prime, with respect to 15. 3 is dividing 15. 4-- yeah, 4 and 15 have greatest common divisor equal to 1. So that's relatively prime. 5 is not. 6 is not. 7 is relatively prime. 8-- again, this is only divisible by power of 2, and this has no power of 2 in it. 9, 10 are-- this one is divisible by 5, divisible by 3. 11 is, again, relatively prime. And then we have 13 is relatively prime as well. And 14 as well, because this is 2 times 7, and this is 3 times 5. So how many do we see now? 1, 2, 3, 4, 5, 6, 7, 8. So the Euler's totient function evaluated in 15 is actually equal to 8.

Now, it turns out that this function has really nice properties, and you can easily calculate-- if you know the decomposition of n into its primes, and powers of primes-- you can easily compute this number, phi of n. Now, you will be able to find this in the book. You should read it. And also, problem set talks about this. But for now, let's just talk about this is an abstract notion, because that's all that we will use in this lecture. But you will have a few exercises to talk about computing this kind of stuff.

OK, so let's talk about Euler's totient function and Euler's theorem. So let me use this blackboard. Now, this is really an exciting theorem. And it's a little bit hard to prove. So why we'll need a little bit more time to do this. Euler's theorem says the following. If the GCD of n and k is equal to 1, then k to the power [INAUDIBLE] the Euler's totient function evaluated in n is actually congruent to 1 modulo n. So this is what we're going to prove here.

So why is this so interesting? Well, we'll talk about an application which is a direct consequence of this theorem, which we'll call Fermat's little theorem. And that, in turn, we will use to explain the RSA algorithm, and show how the decryption works, and so on. So how can we prove this? It will start with a first lemma, and then we're going to do a few tricks-- mathematical tricks, you will see it. So the first lemma is that if I know that the GCD-- which is what we are assuming, the statement of the theorum-- if I know that the GCD of n and k equals 1, then I know that and a times k-- if a times k is congruent to b times k modulo n, then

this applies that a is congruent to b modulo n. All this seems to be kind of a straightforward lemma. I will only talk about its proof.

So how do we do it? Well, first of all, I know that the GCD of n and k equals 1. So that means that I can create a multiplicative inverse, because I know that there's such a linear combination that will ends up to 1. So I have the multiplicative inverse. I multiply both sides with this, and then I will end up to, a is congruent to b modulo n. And actually, you can use some of the facts on your sheet to prove this. And in the problem set, you probably have seen that there are a number of problems related to this. So you will recognize this, and prove a few of these things yourself.

OK, so this is-- so let me see. So let me see what we have done here. Actually, I noticed that I've missed one statement that I would like to explicitly mention. I mean, I've used it a few times. Let me do that first. Which is that, we know that if the GCD of n and k equals 1, then-- this is if, and only if, the case-- if k has a multiplicative inverse-- I've not yet explicitly stated it-- and we can easily see this-- let me just give a quick proof to show how this works. Well, if the GCD is equal to 1, then we use the statement up there. So this is if, and only if, there exists a linear combination. So an s and a t, such that n times s plus k times t equals 1. Well, then I also know that there exists a t such that, actually, the difference between 1 and k times t is divisible by n.

So n divides the difference of k times t minus 1. So why is that? Well, if I look at the difference between k times t and 1, the difference is n times s. And n times s is divisible by n. So now, by the definition of congruence, I just apply the definition over here. We have written it out here. We can say that k times t is congruent to 1 modulo n. And this is the definition of the multiplication inverse. So we have essentially shown that if the greatest common divisors in any case equal to 1, then it has a multiplicative inverse.

OK, we have been using this property over here because we assume that the greatest common divisor is equal to 1. So we now know that there exists a multiplicative inverse of k. We use that one to multiply away, essentially, the k out of this equation, and get a is congruent to b modulo n.

Also note that we use the property over here. We said that we wanted to compute, what was it again? Over here-- we started off with the GCD of m and p to be equal to 1, and I know p. And now I can compute the multiplicative inverse of m. And I should have said why it exists. And it

exists because of that lemma that I just mentioned up here.

OK, so now let's go back to Euler's theorem. This first lemma we are going to use to prove a second lemma, and that second lemma we can finally use to prove the theorem. All right, so this lemma I will put on a separate board, because it contains quite a number of steps to prove. So the lemma states that, if we suppose that the GCD of n and k equals 1-- so it's the same assumptions as before-- if it lets k1 all the way up to k r to be those integers in the range 1, 2, 3, and so on, to n minus 1, that are relatively prime-- so these denote the integers relatively prime to n-- then we can prove a very interesting property.

Now, notice by the way, that r in here is equal to this value of the Euler's totient function evaluated in n. Because this counts the total number of numbers that are relatively prime with respect to n. So now we can prove something really spectacular. We can show that the set that contains all of these remainders-- the remainder of k 1 times k, after dividing out this many multiples of n as possible, all the way to the remainder off k r times k, after dividing out as many multiples of n as possible, this set is actually equal to the set k 1 up to k r. So this is what we're going to prove. And we'll do it in two steps.

We first show that this set has exactly r numbers. So the cardinality of that set is equal to r. So that will be our first step in the proof. And over here we will show that every remainder is actually relatively prime to n, so it must be part of this set. So we will show that this is a subset of this set in a second part of the proof. And combining those two, we are able to prove equality. Why's that? Well I have r distinct elements in this set, I have r distinct elements in this set, this one is a subset of this. So that can only happen if they are equal.

So this is the method for the proof. I should-- so we'll start with the first part. And the way to do that is to see whether it is possible that any two remainders in that set, can they be equal to one another? We will show that that's not possible. So if it's not possible, then all these remainders must be different. And we have exactly r of those. So let's do this.

So the proof for 1 is as follows. Let's assume that we he have to remainders. Say, k i times k, and a remainder k j times k, after dividing out as many multiples of n. And supposed that they are equal to one another. We're going to show that this can only happen if k i is equal to k j. And if you can see that, well then we know that all these different remainders are actually different from one another. And if they're all different from one another, then we must have exactly r. Because we have k 1 up to k r in here.

OK let's see where we can do this. Well, if you know that these two remainders are equal to one another, we can look at them with respect to these definitions over here. And we can show that k i times k is actually congruent to k j times k modulo n. So why is this? Well, these two remainders are the same. And k i times k is equal to this remainder, plus a multiple of n. This k j times k is equal to this remainder plus, a multiple of n. So the difference between those two is also a multiple of n. And that's the definition of congruence.

So now we can use our first lemma, which is stated over here. We know that we assumed in Euler's theorem that the GCD of n and k is equal to 1. If a times k is congruent to b times k modulo n, then we know that's a is congruent to b modulo n. So let's apply it over here, and take for a k i, and for b we can take k j. So now we see that key k i is congruent to k j modulo n. And from this we will conclude-- and that takes an extra step-- that k i is actually equal to k j. So how can we do this?

Well, we know that k i and k j are both in the range from 1 all the way up to n minus 1. So if I look at the difference between those two-- so by definition of congruence I know that n divides k i minus k j. I know that this one is in the range from 0 up to n minus 1. This one is in the range of 0 up to n minus 1. The only way how a difference of two numbers in this range can be divisible by n, is if this thing is equal to zero. And that means that k i equals k j.

So now we are done with the first part, because we have shown that if I take any two remainders over here it must be that they can only be equal to one another if, actually, the k i is equal to the k j. So actually we're looking at the same remainder. So they remainders in this set are all different, and there are exactly r of those.

So now we go to the second part of the proof. And notice that we are-- so far we've only been proving the second lemma, and we still need to go to Euler's theorem as well. So it still takes a few steps. So how do we do the second part? Well, we saw in last lecture that we were explaining Euclid's algorithm. And we used, essentially, this property. We said that the greatest common divisor between n and the remainder of say, k i times k and n, is actually equal to the greatest common divisor of n and k i times k.

So why is this, again? Well, the remaining is actually equal to k i times k, minus a multiple of n, right? So the greatest common divisor is therefore-- between n and this-- is the same as the greatest common divisor between n and k i times k. So you should have to look at last lecture. And now we are pretty much done. Why is this? Because we have assumed that the greatest

common divisor between n and k is equal to 1. And k i, in the statement of the lemma, is relatively prime to n. And that means, according to the definition over there, that the greatest common divisor between k i and n is also equal to 1.

So we know that both these greatest common divisors are equal to 1. So that means that there is no common divisor between n and these, except 1 of course. So what does this say? Well, this means that this remainder, according to our definition, is relatively prime to n, because this greatest common divisor is equal to 1. So if it is an integer relatively prime to n, then it must be one of those k i's, k j's in this set that is stated in the lemma. So this shows that it must be part of this set over here. So if proven, the fact [? is ?] that the set of all the remainders is a subset of the set k 1 of the k r. So now we're done.

So now that we have shown this particular lemma, we can continue and prove Euler's theorem. And I'll probably need to wipe out some of this. So let's use this lemma to prove this theorem. So this is really a neat trick. So the proof of Euler's theorem is as follows. We're going to take the product of all those k i's over there, and see where we can find a nice relationship. So we take k 1 times k 2, all the way times k r. And we know, because those two sets are actually the same, that this is equal to the remainder-- the first remainder-- k 1 times k, after dividing out as many multiples of n. And we go all the way up to the final one, the remainder of k r times k, dividing out as many multiples of n.

So now we can see that-- well, we've already shown that each of those remainders is congruent to, in this case, this one is congruent to k 1 times k modulo n. And this one is congruent to k r times k modulo n. So let's write it out. So it's k 1 times k. And then we have k 2 times k. And finally we have k r times k modulo n. So let's regroup those. We see k 1, k 2, all the way up to k r reappearing. And we have a k here, a k here, and we have that r times-- so we have times k to the power r modulo n. So now we are able to, again, use this particular lemma over here.

So what do we do? Well, we know that k 1 is relatively prime to n. And k 2 is as well, all the way up to k r. So this whole product is also relatively prime, with respect to n. That means that the greatest common divisor of this whole product with n is equal to 1. So that means that I can divide out this whole product. We have-- so let's do this. We have 1 times this product. We take this for a, and we take this for b. And then we can divide this whole thing out according to this particular lemma, by using the multiplicative inverse of that product. So now we see that the 1 is equal to k to the power r modulo n. And remember, in our theorem, r over here in this

lemma-- r is actually equal to the Euler's totient function in n. So now this equation proves the whole theorem.

OK, so now I'm going to talk about RSA, which is the last part here. So maybe you all would like to have a little break of a couple of minutes, just to relax a bit. And then-- and shake hands with your neighbors, and jump up in the air if you'd like to.

All right let's start with the RSA algorithm. So we have done everything up to this point. And we are actually-- we have done these two over here. We still have to talk about Fermat's little theorem. But then we can go for RSA, and it uses this consequence of Euler's theorem.

So Fermat's little theorem is actually talking about what happens if n is a prime number. It says, well, suppose p is a prime. And if you have k in the range 1, 2, all the way up to p minus 1, then we can conclude that k to the power of p minus 1 is congruent to 1 modulo n. And we can directly prove this by using Euler's theorem. So how do we do this?

Well, we know that p is prime. So the numbers 1, 2, all the way up to p minus 1, are actually relatively prime to p. So why is that? Well, p is prime. So the greatest common divisor between any of those with p is equal to 1. That's the definition of relatively prime. And we know that we can now apply Euler's theorem over here, and see that k to the power phi of p is, of course, congruent to 1 modulo p. That's Euler's theorem. But now, since we know that these are the exact ones that are relatively prime, we can explicitly compute the Euler's totient function of p. Because there are p minus 1 numbers that are relatively prime to p. So that's the definition over here. So the number of integers in the range 1 up to p minus 1-- they're all relatively prime.

So we know that 5 p is equal to p minus 1. So now we have shown that k to the power of p minus 1 is congruent to 1 modulo k. This is kind of interesting because we can use this theorem also to compute the multiplicative inverse of k, in this particular case. So how do we do this? We just take k, and we look at what happens if you multiply k with k to the power of p minus 2. Well, this is equal to k to the power p minus 1, which is congruent to 1 modulo p according to Fermat's theorem.

So now, when we look at the definition of multiplicative inverse over here, we see that k to the power p minus 2 is actually the multiplicative inverse of k. So k inverse is actually equal to k to the power p minus 2 modulo k. All right, so this theorem we're going to use now in the description of RSA.

As I said, it was only decades later, after Turing, Rivest, Shamir, and Adleman were the first to really show how number theory could be applied so successfully in cryptography. And they essentially showed the first public key encryption scheme in which a sender and receiver do not necessarily have to exchange a secret key. That's not necessary. So they had a public key method. And that's still used today, and it's a great inventions.

So how does the RSA work? Again, we have to talk with an encryption scheme about what happens beforehand. So beforehand, we need to generate this public key and a secret key. So the idea is that the receiver creates a public key, and also a secret key. He will publish the public key, and he will keep the secret key for himself. And now anybody can use the public key to encrypt a message. The encrypted message is sent to the receiver, and he's going to use his secret key to get back to the plain message.

So how is he going to do this? Well in the first step, the idea is to generate two distinct primes. Turns out that this can be done in a very efficient way. There are lots of primes among the integers. So you just sample. And it turns out that you can test primality with a pretty high probability, very efficiently. And recently, actually, there has been a deterministic algorithm that is polynomial. And a number of bits of the primes that can actually tell you whether you have a prime or not. So you can do this.

Two, we're going to create the product of these two distinct primes. And that's where our assumption is going to help us. That it is hard to factor a product of two large primes. That's going to be the underlying hardness assumption of the RSA encryption scheme. So let n be this. Three, we are going to select an integer e, such that the greatest common divisor of e with the product p minus 1 times q minus 1 is actually equal to 1. And the public-- and once we have created this, the public key is going to be a pair that consists of e itself together with n. So that's the public key.

And the secret key is going to be computed as follows, in step four. We are going to compute. d, such that d times e is congruent to 1 modulo that product of p minus 1 times q minus 1. Can we do this? Yeah, because the greatest common divisor between e and that product is equal to 1. And we have shown over here that, therefore, it has a multiplicative inverse. And first of all, we know that the solution d exists, and we can also efficiently compute it.

So the secret key is going to be the pair that consists of d and also n. So how does it work?

The sender knows the public key, e and n, and uses those to encrypt a message. I will explain it in a moment. And then the receiver knows the secret key, d and n, and then is able to decrypt. OK, so let's see how encryption works. And then we will have to do a lot of mathematics to get the decryption going.

So m prime, which is computed-- so the encrypted plain text is computed as the remainder of m to the power e-- which is part of the public key-- and then dividing out as many multiples of n as possible. It turns out-- and we are going to prove this-- that decryption works as follows. We can compute m by using m prime. So we receive m prime. What do we do? We're going to take m prime, raise it to the power d-- which is part of the secret key-- and then dividing out as many multiples of n.

Now, why would that work? Why would this work? So let's prove this step over here. Well, it turns out that they can apply Fermat's theorem, and the idea is as follows. So let's have a look. We know that m prime is equal to the remainder of m raised to the power e, which is congruent to m to the power e, modulo n. We've seen this now a number of times, right?

So what does this imply? This implies that m prime to the power d is actually equal-- is congruent to m to the power e, to the power d. So I just raised this side to the power d, and I raised this side to the power d. And I still-- and I know, now, that m prime to the power d is equal to m to the power e d-- congruent to m to the power e d modulo n. So now we know that there exists an integer r. We are going to use the fact that we have that e and p minus 1 times q minus 1 as a greatest common divisor over 1. So we know that e times d is actually equal to 1, plus r times p minus 1, times q minus 1. Actually, what I use here is the fact-- is this over here.

By definition of congruency, I know that the difference between those two-- d times e and 1-- is divisible by this product. So I know that there exists an integer such that e times d equals 1, plus a multiple of that product. That's how it works. So we know that m prime to the power d-- we already saw that it is equal to m to the power e d, which is congruent to-- well, we just replace e d by 1 plus r times this multiple. So we have m to the power 1, times m to the power-- this part-- r p minus 1, q minus 1.

So now we're finally going get to Fermat's theorem. We know that n is the products of p and q, and I'm not actually sure I do this here. So let's apply Fermat's theorem and see how we can use this. So if m is not congruent to 0 modulo p, well then we can apply Fermat's theorem.

Where is it? It's over here. We can only apply this if k is in the range from 1 to p minus 1 is not equal to zero. So if m is not equal to 0-- not congruent to 0 modulo p-- then we can apply the theorem and state that m to the power p minus 1 is congruent to 1 modulo p.

And in the same way, we can do this for q. So if this is not true-- modulo q-- then m to the power q minus 1 is congruent to 1 modulo q. So here we have used Fermat's theorem twice. Now we can apply what we have learned before, which is what we wrote down over here. And prime to the power d is congruent to this modulo n. Now n is p times q, so let's have a look at what that means. It means that since n is a product of p and q, we can also look at this congruent modulo p. So in particular, we know that m prime to the power d is congruent to m times m r p minus 1 times q minus 1 modulo p. Why is that?

Well, we know that n divides this difference by the definition of congruency. n is equal to p times q. So if n is dividing this difference, also p is dividing this difference. So that's why we can write it m prime to the power d is congruent to this thing, modulo p. And of course, we can repeat this for q. So let me write this out as well. So there it is. So now we can there use what we have figured out over here.

So we know that if m is not equal to 0, then this thing over here cancels out. Because m to the power p minus 1 is congruent to 1. So we have that m prime to the power d is congruent to m modulo p, if m is not congruent to 0 modulo p, right? Because if m is not congruent to zero, we have this particular equation. We plug it in over here, this all cancels, and we just are left with m.

Now, if m is equal-- is congruent-- to 0 modulo p, then we can see that it is equal to 0. So it's equal to 0. So this actually holds for any case. Now we can do the same for q-- the same argument-- and show that this must hold. So now we know that p divides the difference of m prime d minus m. q is another prime that divides this difference. And the only way that's possible is if the product of p and q is dividing this particular number-- this difference. And we have two different primes dividing the same number, so the product must [? divided. ?] So p times q divides m prime d minus m. Oh, but p times q is equal to n. So now we're almost done, because now we can state-- by the definition of congruency-- that m prime to the power d is congruent to m modulo n.

Now, since m is a message that's in the range of-- so I did not write it down here-- so m is a message which is in the range of 0 all the way up to n minus 1. We know, and we have seen it

before with Turing's code, that we can rewrite this and say that m equals the remainder of m prime to the power d, after multiplying out as many multiples of n as possible.

So here you go. So this is the decryption rule, and it works. We have shown that this equation truly holds. So RSA has really withstood the test of time. It's already out there for many decades, and it's still widely used. I wanted to talk a little bit about this, but there seemed not to be enough time.

But I'd just like to mention that only 2009 Craig Gentry proved a beautiful theorem, and was able to evaluate Boolean circuits. Or, say, certain types of programs under encryption. So you can sort of add and multiply cypher text encryptions together, and it is as if you multiply them at the plain text level. That was a fantastic-- that was an enormous open problem. And he solved it. And only a few months earlier, in 2010, in joint work with Craig and some other colleagues at IBM, we showed it with very simple arithmetic that just uses modulo p and modulo 2 kind of things. We could show a construction this has such a property-- such an encryption scheme of the integers.

So there's still a lot of stuff going on in this area. And really, we use this type of very basic stuff. The problem in cryptography is to show that it is secure. So you have to show that breaking the scheme needs to be reduced to some really hard problem. And that is always the really difficult part of such type of research. OK, well, have lots of fun with recitation.