Network Security

Professor Gaurav S. Kasbekar
Department of Electrical Engineering
Indian Institute of Technology, Bombay
Week - 02
Lecture - 14
Principle of Cryptography: Part 4

In the previous few lectures, we discussed symmetric key cryptography. We now start our discussion of public key cryptography. So, the motivation for using public key cryptography is the following. Recall that if a sender, Alice, and a receiver, Bob, use symmetric key cryptography, then they must first agree on a common secret K_S, which is a symmetric key used for encryption and decryption. For example, in the case of Caesar cipher, the symmetric key is the shift value used for encryption and decryption.

And in the case of a block cipher, the symmetric key is the one-to-one mapping used to map the plaintext to the ciphertext. So, before a sender and receiver use symmetric key cryptography, they must agree on a common secret K_S. So, the question is, how can the sender and receiver agree on a common secret K_S? So, in some contexts, this can be easily done. For example, suppose a bank and a customer want to securely exchange messages over a network, such as the internet; then the bank can first send the secret key K_S to the customer by post.

This is possible because the sender and receiver have access to a communication channel other than the insecure channel over which they will be communicating. So, this is called an out-of-band channel. So, when an out-of-band channel is available, it is easy to agree on a secret symmetric key. But suppose that the only way that Alice and Bob can possibly communicate is over a network, and the network has the property that any message that is sent over the network can be sniffed by intruders. In this case, is it possible for Alice and Bob to communicate securely?

The answer is not very obvious. How can Alice and Bob agree on a secret symmetric key K_S when all the messages they exchange can be sniffed by intruders? In the 1970s, it was shown by Diffie and Hellman that the answer to this question is yes, it is indeed possible for Alice and Bob to communicate securely. So, this was a major breakthrough in cryptography. The reason is that for a long time, that is, from the time of the Caesar cipher to Diffie and Hellman's discovery, encrypted communication required that the sender and receiver share a common secret K_S in advance, and this was done manually, for example, via the dispatch of trusted couriers carrying locked suitcases, wherein the key was stored.

So, sharing the secret key K_S in advance was time-consuming and expensive. So, after the invention of public key cryptography in the 1970s by Diffie and Hellman, the sharing of a secret symmetric key K_S became much easier. Suppose Alice and Bob do not have a shared secret key in advance. Alice wants to encrypt and send a message m to Bob. Bob has two keys; one is a public key. Let's denote it by K_B^+, that is available to everyone; for example, it may be included in a database similar to a telephone directory, and the other key that Bob has is a private key, K_B^-, that is only known to Bob. So, care must be taken to ensure that the private key K_B^+ does not leak and it is not known to others. The private key K_B^- must be known only to the receiver Bob. The public key K_B^+, in contrast, can be known to everyone.

In fact, it is required for others to be able to send encrypted messages to Bob. This shows the screen used for encryption using public key cryptography. Suppose Alice wants to send an encrypted message to Bob. Alice first fetches Bob's public key, K_B^+, possibly from a database, such as a telephone directory. So, that's shown here; Alice fetches the public key of Bob, K_B^+.

And then, Alice computes the encrypted message $K_B^+(m)$, where m is the plaintext message, and $K_B^+(m)$ denotes the plaintext message encrypted using Bob's public key. So, to compute this message $K_B^+(m)$, Alice uses the public key K_B^+ and a known encryption algorithm, possibly a standardized one, and then Alice sends the message to Bob. So, these are the steps: First to send the message from Alice to Bob, first Alice has to obtain Bob's public key and then encrypt the plaintext message m using Bob's public key; $K_B^+(m)$ is the ciphertext obtained, and then Alice sends $K_B^+(m)$ to Bob. Now, Bob receives $K_B^+(m)$ and then uses his private key to compute $K_B^-\left(K_B^+(m)\right) = m$. This computation is done using a known decryption algorithm. So, notice that the public key and the private key are related because if we take a plaintext message m and then first encrypt it using the public key K_B^+ and the encryption algorithm, and then take the result and decrypt it using the decryption algorithm and the decryption key that is the private key.

In that case, we get back the original message m. So, K_B^+ and K_B^- are related. But the selection of K_B^+ and K_B^- must be done carefully, as we'll see. We study algorithms for selecting the public key K_B^+ and the private key K_B^- and the encryption and decryption algorithms, such that this property is indeed true. First, encrypting using the public key and then decrypting using the private key yields the original message m. So, note that K_B^+ is known to everyone. It's stored in a database, such as a telephone directory.

And K_B^+ and K_B^- must be such that it is impossible or computationally infeasible to find K_B^- using K_B^+. So, the catch is that K_B^+ and K_B^- are related. They are related. It is apparent that they are related from this relation here. So, $K_B^-\left(K_B^+(m)\right) = m$. So, clearly, K_B^+ and K_B^- are related.

- We will study algorithms for selecting K_B^+ and K_B^- and encryption and decryption algorithms such that $K_B^-(K_B^+(m)) = m$

But still, using the knowledge of K_B^+, it must be computationally infeasible to find K_B^-. So, we require a clever algorithm to select the public key K_B^+ and the private key K_B^-. We study the RSA algorithm. It is a widely used public-key cryptography algorithm. It is named after its inventors, Ron Rivest, Adi Shamir, and Leonard Adleman.

So, hence, the abbreviation RSA after the inventors. So, the basic idea in RSA is the following. p and q are two large prime numbers. They are known only to the receiver, Bob. His private key is found using these prime numbers p and q. So, Bob knows p, and Bob knows q. So, Bob individually knows p and q, whereas $n = pq$, that is, the product of p and q is known to everyone.

It is a part of the public key. The security of RSA is based on the fact that there are no known efficient algorithms for factoring a number n into its prime factors p and q. So, the most efficient way known is to essentially try out all the numbers less than $\sqrt{n}$. So, clearly, since $n = pq$, one of the factors must be less than equal to $\sqrt{n}$. So, clearly, it is enough to try out numbers up to $\sqrt{n}$. So, once, we get one factor, then we get the other factor automatically. But the problem is that for factoring n, there is no known efficient algorithm. The only way known is brute force, that is, trying out all the numbers less than equal to $\sqrt{n}$. So, we don't know any algorithms that are much more efficient than brute force.

So, if n is large enough, then it is computationally infeasible to find the private key using the public key. For example, n might require 1024 bits to represent. So, in this case, it's an extremely large number. In this case, it's computationally infeasible to factor it out into its prime factors p and q. So, recall modulo$-n$ arithmetic, which we reviewed in the mathematical background for cryptography. RSA makes extensive use of modulo$-n$ arithmetic.

Recall that $x \bmod n$ denotes the remainder when x is divided by n. For example, $19 \bmod 5 = 4$. So, we have these properties, which we discussed earlier. $(a \bmod n + b \bmod n) \bmod n = (a + b) \bmod n$, and we have similar properties for subtraction and multiplication. We'll use these properties in the sequel. Suppose Alice wants to send a message m, which is a bit string, to Bob.

1) $(a \bmod n + b \bmod n) \bmod n = (a + b) \bmod n$
2) $(a \bmod n - b \bmod n) \bmod n = (a - b) \bmod n$
3) $((a \bmod n) \times (b \bmod n)) \bmod n = (ab) \bmod n$

Now, any bit string can be equivalently represented by a positive integer. For example, the bit string 1001 can be represented by 9. Notice that the binary representation of 9 is 1001. So, in RSA, we represent the original plaintext message by some positive integer. So, the components of RSA are the following.

One component is the choice of the public key and the private key, and the other component is the encryption and decryption algorithms. So, we'll discuss each of these components in detail. First, we'll discuss how to choose the public key and the private key. Then, we'll discuss what the encryption and decryption algorithms are. So, here is the algorithm for generating the public and private keys.

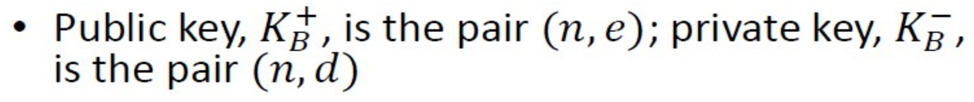
We choose two large prime numbers, p and q. The trade-off is that the larger the values of p and q, the more difficult it is to break RSA. But, on the other hand, encryption and decryption times start increasing with p and q. So, the larger the values of p and q, the more difficult it is to break RSA, but the encryption and decryption times are larger. So, p and q must be selected, taking into account this trade-off. Now, let $n = pq$, and $z = (p-1)(q-1)$.

- Let $n = pq$ and $z = (p-1)(q-1)$

For example, if $p = 5$, and $q = 7$, notice that p and q are prime numbers. In this case, $n = pq = 35$, and $z = (p-1)(q-1) = 4 \times 6 = 24$. So, these are very small numbers. In practice, p and q are obviously much larger. Next, we choose a number $e < n$, such that e and z are relatively prime.

Later, we'll see the reason for the fact that e and z have to be relatively prime. In this example, $e = 5$. So, notice that 5 and 24 are relatively prime. Next, we choose a number d, such that $(ed) mod\ z = 1$. Later, we will show that a number d always exists when e and z are relatively prime.

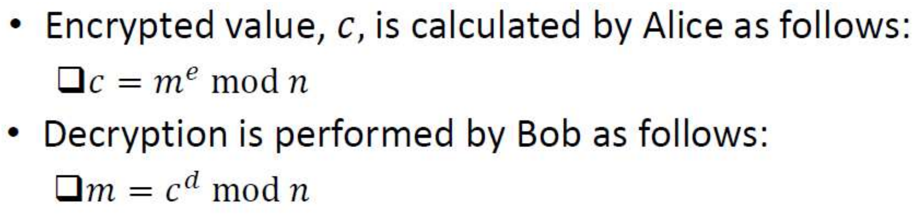
A number d, such that $(ed) mod\ z = 1$, always exists when e and z are relatively prime. So, in this example, $d = 29$. So, notice that $5 \times 29 = 145$, and $145\ mod\ 24 = 1$. So, d is such that $(ed) mod\ z = 1$. Now, the public key K_B^+ is the pair (n, e), and the private key K_B^- is the pair (n, d). So, this is the algorithm for selecting the public and private key.



- Public key, K_B^+, is the pair (n, e); private key, K_B^-, is the pair (n, d)

So, the public key is the pair (n, e), and the private key K_B^- is the pair (n, d). Now, let's discuss how to perform encryption and decryption. Suppose Alice wants to send a message m, which is a positive integer, to Bob. So, $m < n$. If the original plaintext message is large, then we can always break it into chunks such that each chunk satisfies the property that $m < n$. Now, the encrypted value c is calculated by Alice as follows. $c = m^e\ mod\ n$. c is sent to Bob.

Now, at the other end, Bob performs decryption as follows. Bob computes $c^d\ mod\ n$. And the claim is that $c^d\ mod\ n = m$, which is the original plaintext message. So, notice that for encryption, Alice used the public key of Bob, that is, (n, e), and for decryption, Bob used Bob's private key, that is, (n, d). So, these are the steps for encryption and decryption. This is an example which illustrates the operation of RSA. Originally, there are four plaintext messages.



- Encrypted value, c, is calculated by Alice as follows:
  - ❑ $c = m^e \bmod n$
- Decryption is performed by Bob as follows:
  - ❑ $m = c^d \bmod n$

These are 12, 15, 22, and 5. These are encrypted and sent by Alice. So, m^e is this. This column provides the value of m^e, where $e = 5$. And then, this column provides $m^e\ mod\ n$. So, these are the ciphertext messages corresponding to these plaintext messages.

Now, these ciphertext messages are sent over the channel by Alice to Bob. Now, Bob takes these ciphertext messages and performs $c^d\ mod\ n$. So, c^d is shown in this column here. And then, $c^d\ mod\ n$ is shown in this column. So, notice that Bob recovers the same plaintext messages that were sent by Alice. So, performing encryption first and then decryption recovers the original plaintext.

Another key observation is that even for this small example, where $n = 35, e = 5$, and $d = 29$, even in this example with small values of p and q, notice that these numbers are extremely large. c^d becomes large even for this simple example. So, for this reason, RSA is computationally quite expensive. So, public key encryption schemes are, in general, more computationally expensive than symmetric key encryption schemes.

Now, recall that the encrypted value $c = m^e\ mod\ n$. The value obtained after decryption is performed by Bob is $m' = c^d\ mod\ n$. Now, let's substitute for c from this equation into this equation. So, we get $m' = (m^e\ mod\ n)^d\ mod\ n$. By the properties of modular arithmetic that we discussed earlier, this is the same as $m^{ed}\ mod\ n$. Now, the claim is that $m^{ed}\ mod\ n = m$. So, if we prove this claim, then that will prove that the value after decryption is the same as the original plaintext. We have to prove that $m^{ed}\ mod\ n = m$. So, we now study some basic results from number theory, and we'll then later use these results to prove this result.

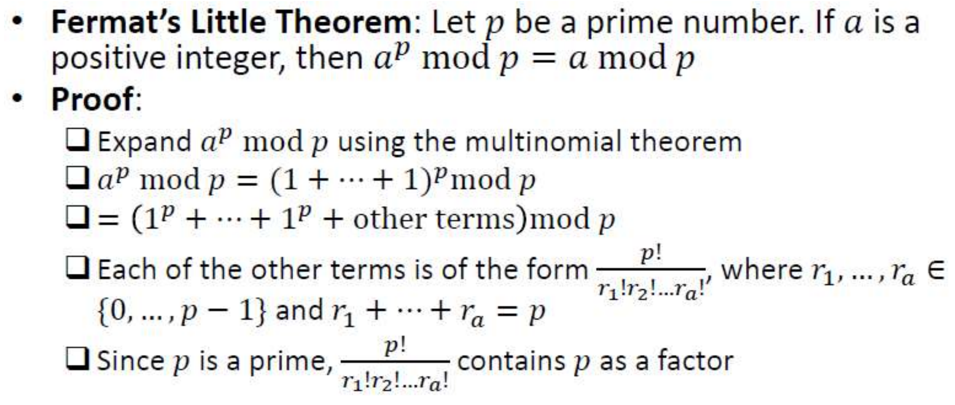
- Value obtained after decryption is performed by Bob is $m' = c^d \bmod n$
- $m' = (m^e \bmod n)^d \bmod n$
- So $m' = m^{ed} \bmod n$
- **Theorem**: $m^{ed} \bmod n = m$

So, we require the following results from number theory. One is Fermat's Little Theorem. Let p be a prime number. If a is a positive integer, then the claim is that $a^p\ mod\ p = a\ mod\ p$. Here's an example. Let $p = 5$ and $a = 8$.

Then, it can be checked that $8^5 = 32768$. So, $a^p\ mod\ p = 3$. And that is the same as $a\ mod\ p$, which is also 3. So, $8\ mod\ 5 = 3$. So, this theorem is verified for this example.

Let's prove this theorem in general. First, we expand $a^p\ mod\ p$, using the multinomial theorem. So, we write $a = (1 + \cdots + 1)$. There are 8 terms in this sum. So, we write $a = (1 + \cdots + 1)$, and then use the multinomial theorem to expand $a^p\ mod\ p$. So, $a^p\ mod\ p =$

$(1 + \cdots + 1)^p \ mod\ p$. Now, recall the multinomial theorem, which is a generalization of the binomial theorem.



- **Fermat's Little Theorem**: Let p be a prime number. If a is a positive integer, then $a^p \bmod p = a \bmod p$
- **Proof**:
  - ❑ Expand $a^p \bmod p$ using the multinomial theorem
  - ❑ $a^p \bmod p = (1 + \cdots + 1)^p \bmod p$
  - ❑ $= (1^p + \cdots + 1^p + \text{other terms}) \bmod p$
  - ❑ Each of the other terms is of the form $\frac{p!}{r_1!r_2!\ldots r_a!}$, where $r_1, \ldots, r_a \in \{0, \ldots, p-1\}$ and $r_1 + \cdots + r_a = p$
  - ❑ Since p is a prime, $\frac{p!}{r_1!r_2!\ldots r_a!}$ contains p as a factor

It allows us to write an expansion for this quantity. So, if we expand this using the multinomial theorem, we get 8 terms, which are all 1^p, that is the same as 1, and $(1^p + \cdots + 1^p + other\ terms)\ mod\ p$. Now, each of these other terms is of this form: $\frac{p!}{r_1!r_2!\ldots r_a!}$, where $r_1, \ldots, r_a \in \{0, \ldots, p-1\}$ and $r_1 + \cdots + r_a = p$.

This is the multinomial coefficient. Notice that it's a generalization of the binomial coefficient, which is $\binom{n}{k}$.

So, if you expand this using the multinomial theorem, we get such an expansion.

Now, since p is a prime number, the factor p in the numerator here does not cancel with any of the terms in the denominator. So, there is a factor p present in this multinomial coefficient. So, each of these other terms has a factor p in it. So, when we take $mod\ p$, all these other terms vanish.

After taking $(other\ terms)\ mod\ p$, all the $(other\ terms)\ mod\ p = 0$.

So, hence, the result is just the first a terms survive, and the result is $a\ mod\ p$. So, that concludes the proof of this theorem that $a^p\ mod\ p = a\ mod\ p$. Another result from number theory that we require is Bezout's identity. This is also known as the GCD theorem. We discussed this result earlier. Recall that if a and b are integers with GCD d, then there exist integers x and y such that $ax + by = d$.

For example, if $a = 5$ and $b = 8$, then such numbers x and y are $x = -3$ and $y = 2$. So, we can check that $ax + by = \text{gcd}(5,8) = 1$. So, we recall these results from the previous slide. Now, a corollary of these results is that if p is a prime number, and if a is a positive integer that is not divisible by p, then the claim is that $a^{p-1}\ mod\ p = 1$. So, we'll require this corollary to prove the correctness of RSA.

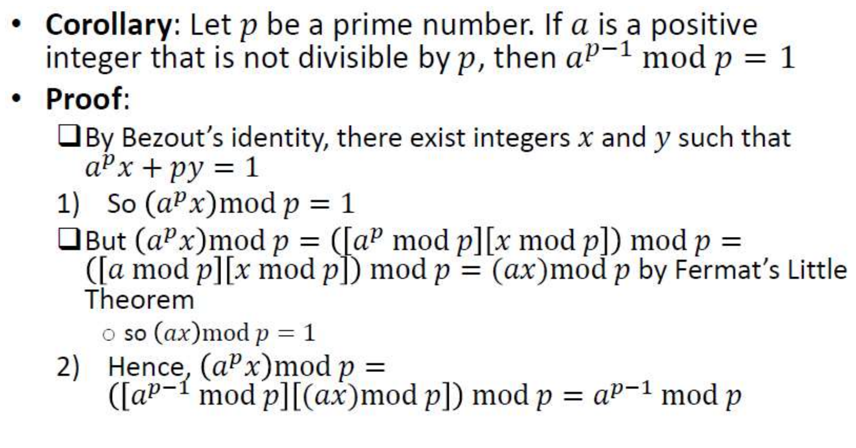
Let's prove this corollary. By Bezout's identity, there exists integers x and y, such that $a^p x + py = \gcd(a^p, p) = 1$. The GCD is 1 because p is a prime number. So, hence, clearly, a^p and p are relatively prime. So, there exist such integers x and y. Now, we are interested in $a^{p-1}\ mod\ p$. So, let's try taking mod p throughout this equation.

So, taking $mod\ p$ throughout this equation, we see that $py\ mod\ p = 0$. So, hence, we get from here $(a^p x)\ mod\ p = 1$.

But $(a^p x) mod\ p = ([a^p\ mod\ p][x\ mod\ p])\ mod\ p$ (by the properties of modulo arithmetic that we discussed earlier. But by Fermat's Little Theorem, $a^p\ mod\ p = a\ mod\ p$. So, hence, we get $([a\ mod\ p][x\ mod\ p])\ mod\ p$. But that is the same as $(ax) mod\ p$ by Fermat's. So, we have used Fermat's Little Theorem in this step while going from here to here. And hence, we get that $(a^p x)\ mod\ p = (ax)\ mod\ p$. So, in this step, we have shown that $(a^p x)\ mod\ p = 1$, so hence, from this step, it follows that $(ax)\ mod\ p = 1$.

Now, consider again $(a^p x)\ mod\ p$.

It can be written as $([a^{p-1}\ mod\ p][(ax)\ mod\ p])\ mod\ p$. So, we've already shown that $(ax)\ mod\ p = 1$. So, hence, this LHS becomes equal to $a^{p-1}\ mod\ p$. So, the result follows from 1) and 2). So, in 1), we have shown that $(a^p x)\ mod\ p = 1$. So, hence, the LHS is 1 in this equation.



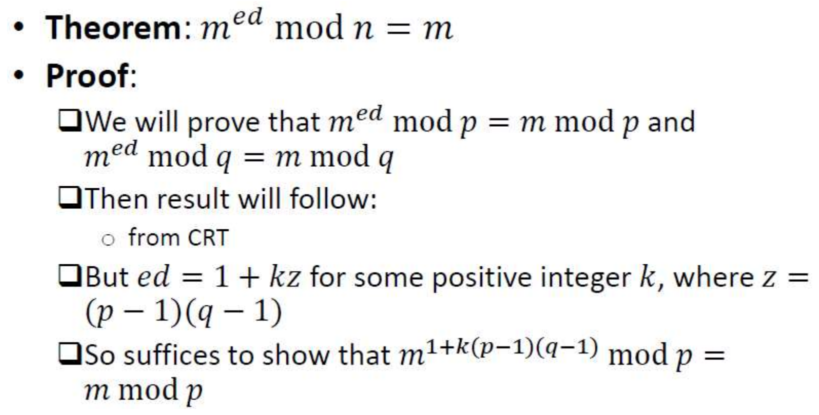
- **Corollary**: Let p be a prime number. If a is a positive integer that is not divisible by p, then $a^{p-1} \bmod p = 1$
- **Proof**:
  - ❑ By Bezout's identity, there exist integers x and y such that $a^p x + py = 1$
  1) So $(a^p x) \bmod p = 1$
  - ❑ But $(a^p x) \bmod p = ([a^p \bmod p][x \bmod p]) \bmod p = ([a \bmod p][x \bmod p]) \bmod p = (ax) \bmod p$ by Fermat's Little Theorem
    - ○ so $(ax) \bmod p = 1$
  2) Hence, $(a^p x) \bmod p = ([a^{p-1} \bmod p][(ax) \bmod p]) \bmod p = a^{p-1} \bmod p$

So, hence, $a^{p-1}\ mod\ p = 1$, which is what we wanted to show. Now, again, recall the Chinese Remainder Theorem, which is another result that we discussed in the mathematical background for cryptography. If p and q are relatively prime, then this set of equations, $x\ mod\ p = a$ and $x\ mod\ q = b$, has a unique solution for x modulo pq. So, we discussed the general form of the Chinese Remainder Theorem, where we have multiple factors. We have k factors, say $p_1, p_2, \ldots, p_k$.

Here, we only require the special case of two factors, p and q. So, for the case of two factors, this is the form of the Chinese Remainder Theorem. Now, we return to the proof of correctness of RSA. Recall this theorem: $m^{ed}\ mod\ n = m$. So, we will prove that $m^{ed}\ mod\ p = m\ mod\ p$, and we will also prove that $m^{ed}\ mod\ q = m\ mod\ q$. So, once we prove one of them, the other follows by symmetry. Then, the claim is that this result will follow once we have proved these properties.

This result follows from the Chinese Remainder Theorem. The reason is that p and q are relatively prime. Since they are prime, they are clearly relatively prime. And $m^{ed}\ mod\ p = m\ mod\ p = a$; let us call that a. And $m^{ed}\ mod\ q = m\ mod\ q = b$. Let us call that b. So, this is a system of equations of the form $x\ mod\ p = a$ and $x\ mod\ q = b$. So, this equation is of the form $x\ mod\ p = a$ and $x\ mod\ q = b$.

But this is exactly the form of equations that appear in the Chinese Remainder Theorem. So, we know that this system of equations has a unique solution modulo$-n$, that is, modulo pq. But clearly, trivially, m is itself a solution of this system of equations because $m\ mod\ p = a$, which is $m\ mod\ p$, and $m\ mod\ q = b$, which is $m\ mod\ q$. So, m is a solution of these equations. But once we have proved that $m^{ed}\ mod\ p = a$ and $m^{ed}\ mod\ q = b$, then it will follow that $m^{ed}\ mod\ n$ is the same as m. So, $m^{ed}\ mod\ n = m$. That will follow from these properties. So, this result holds by the Chinese Remainder Theorem.



- **Theorem**: $m^{ed} \bmod n = m$
- **Proof**:
  - ❑We will prove that $m^{ed} \bmod p = m \bmod p$ and $m^{ed} \bmod q = m \bmod q$
  - ❑Then result will follow:
    - from CRT
  - ❑But $ed = 1 + kz$ for some positive integer k, where $z = (p-1)(q-1)$
  - ❑So suffices to show that $m^{1+k(p-1)(q-1)} \bmod p = m \bmod p$

Now, we recall that $ed = 1 + kz$ for some positive integer k, where $z = (p-1)(q-1)$. This is true because $(ed)\ mod\ z = 1$. So, we can now substitute for ed from here. So, this becomes $m^{1+k(p-1)(q-1)}\ mod\ p$. And we have to show that this is equal to $m\ mod\ p$. Now, consider this LHS. So, this is equal to $\left([m\ mod\ p]\left[m^{k(p-1)(q-1)}\ mod\ p\right]\right) mod\ p$. But by the above corollary to Fermat's Little Theorem and Bezout's Identity, we know that $m^{p-1}\ mod\ p = 1$.

So, hence, this LHS is the same as $m\ mod\ p$. So, hence, this is the same as $m\ mod\ p$, which is what we had to prove. And then the result follows from the Chinese Remainder Theorem. So, this proves the correctness of RSA, that the value obtained after encrypting the plaintext and then decrypting the ciphertext is indeed the original plaintext. Now, we discuss the reason why e and z have to be relatively prime. Recall that $z = (p-1)(q-1)$. and $e < n$, is chosen such that e and z are relatively prime.

So, the reason is that if e and z are relatively prime, then there exists a positive integer d such that $(ed)\ mod\ z = 1$. And we require this positive integer d for decryption. So, the proof is the following. By Bezout's identity, there exist integers x and y such that $ex + zy = \gcd(e, z) = 1$. So, taking $mod\ z$ throughout this equation, we get $(ex)\ mod\ z = 1$.

But since x may be negative, let $d = x + kz$, where k is a sufficiently large positive integer. It's sufficiently large so that d becomes positive. So, now $d\ mod\ z = x\ mod\ z$. So, further, if we substitute d into this equation, then we get that $(ed)\ mod\ z = (ex)\ mod\ z + 0$. So, we get that $(ed)\ mod\ z = 1$. So, hence, there exists a number d such that $(ed)\ mod\ z = 1$.

That's what we were shown. Now, d can be efficiently computed using the extended Euclidean algorithm, which we discussed during the mathematical background of cryptography. So, recall that for given integers a and b, the extended Euclidean algorithm finds the GCD of a and b as well as integers x and y, such that $ax + by = \gcd(a, b)$. Now, why is RSA secure? Recall that the public key is the pair (n, e), and the private key is the pair (n, d). So, n and e are publicly known. So, if an intruder could factor n into p and q, then the intruder knows p and q. Then, from p and q, the intruder can calculate $z = (p-1)(q-1)$.

And once z is known, then the intruder could find out d using the extended Euclidean algorithm. But the problem is that there is no efficient algorithm known for factoring a number n into its prime factors p and q. So, if such an efficient algorithm were found, then RSA would be broken. But on the other hand, it has not been proven that no efficient algorithm exists. So, since there is no efficient algorithm known for factoring a number n into its prime factors p and q, the security of RSA rests on this fact. So, even though n is a part of the public key, it is computationally infeasible to find out p and q from n. And hence, it's computationally infeasible to get the private key.

Also, no efficient algorithm is known for finding d directly using (n, e). So, for these reasons, RSA is secure. Now, recall that if the plaintext is m, then the ciphertext is $c = m^e \; mod \; n$. If m and/or e are small, then it is possible that $m^e \; mod \; n < n$. So, hence, $m^e \; mod \; n = m^e$. So, the intruder could decrypt the message by just finding the eth root of the ciphertext. So, by taking just the eth root of m^e, the intruder can get back the plaintext m. So, breaking the cipher is quite easy. So, such problems can be avoided by padding each message m with some bits before encryption.

So, m is padded with some bits so that it becomes large enough so that m^e becomes greater than n. So, by just merely taking the eth root, the intruder cannot get back the plaintext message m. Now, we showed that $K_B^-\left(K_B^+(m)\right) = m$; that is, by first applying the public key to a plaintext message m and then applying the private key, we get back the original message m. And a simple exercise is to show that if we reverse the order of operations, that is, first we apply the private key K_B^- to m and then apply the public key to the result, $K_B^+\left(K_B^-(m)\right)$. This is the same as m. So, in this case, we first start with the message, say, c, and then we find out $c^d \; mod \; n = x$. Let's call this, say, x. And the claim is that $x^e \; mod \; n = c$. So, that's what this says. So, if we first use the private key and then we use the public key, then we get back the original plaintext message m. So, later we'll use this fact to design digital signatures.

Now, encryption and decryption using DES or other symmetric key ciphers is at least 100 times faster in software and between 1,000 and 10,000 times faster in hardware than RSA. So, this is the general trend that symmetric key algorithms are much more computationally efficient than public key algorithms. So, now, suppose a large file needs to be transferred from Alice to Bob. Alice and Bob do not have a shared key available in advance. So, one way to securely send the file is to encrypt the entire message using RSA.

But this is very slow because, as this shows, encryption and decryption using RSA is very slow. So, can we somehow combine DES and RSA to send the file securely and faster? So, this can be done as follows. Alice first selects a string K_S randomly and then encrypts it with RSA to Bob. So, Alice selects a random key K_S and then encrypts it with RSA, and then sends it to Bob.

Then, the large file is sent after encryption using DES with K_S as the shared key. So, once K_S has been securely sent to Bob using RSA, now Alice and Bob have agreed upon the secret symmetric key K_S. Subsequently, they can use a symmetric key algorithm like DES to send the file itself using symmetric key encryption. So, in this way, we can combine public key encryption with symmetric key encryption to encrypt the file and send it very fast. So, we just mentioned DES and RSA as examples, but we can combine any public key algorithm and any symmetric key algorithm using this procedure, and using the combination, we can send a file faster.

This concludes our discussion of RSA. Thank you.