RSA

Hi again. In this video, we're going to be looking at a modern example of public key cryptography. We'll be exploring the RSA algorithm. This is used in modern cryptography in tools such as VPNs. We'll start by having a look at how we generate our public and private key pair. Let's look now at the detail of generating a key using RSA. There have been variations on the RSA algorithm, but we will use the one that was originally published in the paper. The first step is to identify two large and distinct primes, p and q. The next step is relatively straightforward in that we calculate the product p and q. We'll label this n. As we've discussed before, this is very difficult to then prise apart the individual value of p and q from the final product as long as p and q are sufficiently large.

Our next step is to calculate what's referred to as Euler's function. Euler's function is the number of positive integers up to the value of an integer n which are coprime with n. Coprime means that the largest common divisor is 1. When we're working with the product of two primes, as we have here, Euler's function becomes the product of the first prime minus 1 multiplied by the second prime minus 1 which makes the calculation fairly straightforward. Next we need to select an integer e.

e to be less than Euler's function applied to n and it also has to be coprime with this value. e is then selected as the public key, and it's the combination of e with the value n which is public. We then calculate d. d is congruent to e to the power of minus 1 mod the quotient of n. It's not necessary for us to determine how this value is found, and it can be a little bit tricky. Obviously, as with other cryptography algorithms, it's not a good idea to try and implement this yourself, but it is suffice for this module to know that that is how d is calculated.

As you may imagine, d is then the private key in combination with n. Let's have a look at how this works by using a toy example. Toy example is a phrase which indicates that the example is not practical for use, but it does help to demonstrate functionality without having to be overly complex. In our case, we are going to select p as 5 and q as 11. n is then the product of these two values, 5 times 11, and so we have an n value of 55. We then calculate Euler's function of n which in this case is 5 minus 1, which is 4, multiplied by 11 minus 1 which is 10.

So phi of n in this instance is 40. We then need to determine e. e is a integer value which is less than 40 and also has to be coprime to 40. There are multiple values which could work here, but one of the values which does work is e being equal to 7. 7 is a prime. The largest number that divides itself is 7. 7 does not exactly divide 40, and therefore, it leads to our coprime because the only common divisor is 1. Then we need to determine d. Recall that d is congruent to the inverse of e mod 5 n which in our instance is 40. So if you think about what this equation really means, we can determine that there is some value d, which is a positive integer, when multiplied by 7

and divided by 40, gives a result of 1.

This is because e is the multiplicative inverse of d. That is to say that e times d is going to be 1. So we can rephrase this as 7 d is congruent to 1 mod 40. So now we need to find a value d which when multiplied by 7 and divided by 40 gives us a result of 1. It's not necessary to step through trying the different values for d, but note that in this instance, d as 23 gives us the result which works. It's congruent to 1 mod 40. As I say, it's not dreadfully important that you're able to perform the calculation itself or to calculate what d might be. Clearly, as we increase the size of p and q, this becomes more difficult. But the main principle of how this works is what we're trying to achieve here.

And hopefully, that simple example helps you to do that. Having now generated our mathematically linked public and private key pair, we need to see how this is applied to encryption and decryption. Recall that our public key is e, n and our private key is d, n. In order to encrypt, we split our message into blocks. So we can have plaintext block 1, block 2, and so forth for the length of our message. The next step in encryption is to take our block, raise it to the power of e, and calculate the result mod n. For decryption, we take our cipherblock, raise it to the power of d, and calculate the result mod n. And this gets us back our plaintext.

So it may be easier to remember d for decryption, e for encryption, and e is your private key with d being public. Let's go back to the example that we had to see how this could work in practice. We set e as 7 and d as 23. We had n as 55. And let's consider an example which is very simple. We'll take just a single letter and encrypt it. And we'll take that letter as B and we'll use a number to represent that. We'll just say that it's the position in the English alphabet. So we're saying that B is 2. To encrypt this, we take our plaintext block, which is just 2 in this instance, raise it to the power of e and calculate the result mod 55. When we complete this calculation, we have 128 mod 55. And taking off two multiples of 55 from 128, we end up with the result of 18. So this is now our first ciphertext block.

If we wish to decrypt that value, we take 18. We raise it to the power of 23. We then calculate the result mod 55. And when we complete this calculation, the end result is 2 which is what we expected as our original plain block. Of course, as with the other ciphers that we've seen, you need to start considering integrity of your message and making use of modes of operation such as cipher block chaining to further increase the security. Hopefully, this example gives you an idea of how the encryption and decryption works with our public and private key pair.

That's us for this video. We've looked at how RSA generates key pairs, and we've also looked at how this is then applied in encryption and decryption. I hope you've enjoyed the video, and I'll see you next time.