



## Cryptography: Crash Course Computer Science #33

Crash Course: Computer Science

<https://youtube.com/watch?v=jhXCTbFnK8o>

<https://nerdfighteria.info/v/jhXCTbFnK8o>

Hi, I'm Carrie Anne, welcome to Crash Course Computer Science.

Over the past two episodes, we've talked a lot about computer security, but the fact is, there is no such thing as a perfectly 100% secure computer system.

There will always be bugs and security experts know that. So system architects employ a strategy called Defense in Depth, which uses many layers of varying security mechanisms to frustrate attackers.

It's a bit like how castles are designed. First you've got to dodge the archers, then cross the moat, scale the walls, avoid the hot oil, get over the round parts, and defeat the guards, before you get to the throne room.

But in this case, we're talking about one of the most common forms of computer security: Cryptography.

[Intro Music]

The word 'cryptography' comes from the roots 'crypto-' and '-graphy', roughly translating to 'secret writing'. In order to make information secret, you use a cipher, an algorithm that converts plain text into ciphertext, which is gibberish unless you have a key that lets you undo the cipher.

The process of making texts secret is called encryption and the reverse process is called decryption. Ciphers have been used long before computers showed up. Julius Caesar used what's now called a Caesar cipher to encrypt private correspondents.

He would shift the letters in a message forward by three places. So A became D and the name Brutus became this. To decipher the message, recipients had to know both the algorithm and the number to shift by, which acted as the key.

The Caesar cipher is one example of a larger class of techniques called substitution ciphers. These replace every letter in a message with something else according to a translation.

A big drawback of basic substitution ciphers is that letter frequencies are preserved. For example, E is the most common letter in English, so if your cipher translates E to an X, then X will show up the most frequently in the Cipher text.

A skilled cryptanalyst can work backwards from these kinds of statistics to figure out the message. It was the breaking of a substitution cipher that led to the execution of Mary, Queen of Scots in 1587 for plotting to kill Queen Elizabeth.

Another fundamental class of techniques are permutation ciphers. Let's look at a simple example called a Columnar Transposition Cipher. Here we take a message and fill the numbers into a grid. In this case, we've chosen 5 by 5.

To encrypt our message, we read out the characters in a different order. Let's say from the bottom left working upwards, working one column at a time. The new letter ordering, what's called a permutation, is the encrypted message.

The ordering direction as well as the 5 by 5 grid size work as the key. Like before, if the cipher and key are known, a recipient can reverse the process to reveal the original message.

By the 1900s, cryptography was mechanized in the form of encryption machines. The most famous was the German Enigma - used by the Nazis to encrypt their wartime communications.

As we discussed back in episode 15, the Enigma was a typewriter-like machine with a keyboard and lamp-board, both showing the full alphabet. Above that, there was a series of configurable rotors that were the key to the Enigma's encryption capability.

First, let's look at just one rotor. One side had electrical contacts for all 26 letters. These connected to the other side of the rotor that used cross-crossing wires that swapped one letter for another.

If H went in, K might come out the other side. If K went in, F might come out, and so on. The letter swapping behavior should sound familiar. It's a substitution cipher!

But the Enigma was more sophisticated because it used three or more rotors in a row, each feeding into the next. Rotors could also be rotated to one of 26 possible starting positions and they could be inserted in different orders, providing a lot of different substitution mappings.

Following the rotors was a special circuit called a reflector. Instead of passing the signal on to another rotor, it connected every pin to another and sent the electrical signal back through the rotors.

Finally, there was a plug board at the front of the machine that allowed letters coming from the keyboard to be optionally swapped, adding another level of complexity. With our simplified circuit, let's encrypt a letter on this example Enigma configuration.

If we press the H key, electricity flows through the plug board, then the rotors, hits the reflector, comes back through the rotors and plug board, and illuminates the letter L on the lamp-board. So H is encrypted to L.

Note that the circuit can flow both ways. So if we type the letter L, H would light up. In other words, it's the same process for encrypting and decrypting. You just have to make sure the sending and receiving machines have the same initial configuration.

If you look carefully at this circuit, you'll notice it's impossible for a letter to be encrypted as itself, which turned out to be a fatal cryptographic weakness.

Finally, to prevent the Enigma from being a simple substitution cipher, every single time a letter was entered, the rotors advanced by one spot. Sort of like an odometer in a car. So if you entered the text AAA, it might come out as BDK, where the substitution mapping changed with every key press.

The Enigma was a tough cookie to crack for sure, but as we discussed in episode 15, Alan Turing and his colleagues at Bletchley Park were able to break Enigma codes and largely automate the process.

But with the advent of computers, cryptography moved from hardware into software. One of the earliest software ciphers to become widely spread was the Data Encryption Standard, developed by IBM and the NSA in 1977.

DES, as it was known, originally used binary keys that were 56 bits long, which means that there are 2 to the 56th or about 72 quadrillion different keys.

Back in 1977, that meant that nobody, except perhaps the NSA, had enough computing power to brute force all possible keys. But, by 1999, a quarter million dollar computer could try every possible DES key in just two days, rendering this cipher insecure.

So in 2001, the Advanced Encryption Standard, AES, was finalized and published. AES is designed to use much bigger keys. 128, 192,



## Cryptography: Crash Course Computer Science #33

Crash Course: Computer Science

<https://youtube.com/watch?v=jhXCTbFnK8o>

<https://nerdfighteria.info/v/jhXCTbFnK8o>

or 256 bits in size, making brute force attacks much, much harder.

For 128 bit key, you'd need trillions of years to try every combination, even if you used every single computer on the planet today. So you'd better get started!

AES chops data up into 16-byte blocks, and then applies a series of substitutions and permutations, based on the key value, plus some other operations to obscure the message, and this process is repeated ten or more times for each block.

You might be wondering, why only ten rounds? Or why only 128 bit keys and not 10,000 bit keys? Well, it's a performance trade-off.

It took hours to encrypt and send an email or minutes to connect to a secure website, people wouldn't use it. AES balances performance and security to provide practical cryptography.

Today, AES is used everywhere. From encrypting files on iPhones and transmitting data over Wifi with wpa2, to accessing websites using https.

So far, the cryptographic techniques we've discussed rely on keys that are known by both sender and recipient. The sender encrypts a message using a key and the recipient decrypts it using the same key.

In the old days, keys would be shared by voice or physically. For example, the Germans distributed code books with daily settings for their Enigma machines. But this strategy could never work in the internet era. Imagine having to crack open a code-book to connect to YouTube.

What's needed is a way for a server to send a secret over the public internet to a user, wishing to connect securely. It seems like that wouldn't be secure, because if the key is sent in the open and intercepted by a hacker, couldn't they use that to decrypt all communication between the two?

The solution is key exchange, an algorithm that allows two computers to agree on a key without ever sending one. We can do this with one-way functions, mathematical operations that are very easy to do in one direction, but hard to reverse.

To show you how one-way functions work, let's use paint colors as an analogy. It's easy to mix paint colors together, but it's not so easy to figure out the constituent colors that we used to make a mixed-paint color. You'd have to test a lot of possibilities to figure it out.

In this metaphor, our secret key is a unique shade of paint. First, there's a public paint color that everyone can see. Then John and I each pick a secret paint color. To exchange keys, I mix my secret paint color with the public paint color, then I send that mixed color to John by any means. Mail, carrier pigeon, whatever.

John does the same, mixing his secret paint color with the public color, then sending that to me. When I receive John's color, I simply add my private color to create a blend of all three paints. John does the same with my mixed color and voila! We both end up with the same paint color.

We can use this as a shared secret, even though we never sent each other our individual secret colors. A snooping outside observer would know partial information, but they'd find it very difficult to find out our shared secret color.

Of course, sending and mixing paint colors isn't going well for transmitting computer data, but luckily, mathematical one-way

functions are perfect. And this is what Diffie-Hellman Key Exchange uses.

In Diffie-Hellman, the one-way function is modular exponentiation. This means taking one number at the base to the power of the other number, the exponent, and taking the remainder when divided by a third number, the modulus.

So, for example, if we wanted to calculate  $3^5 \bmod 31$ , we would calculate  $3^5$ , which is 243, and then take the remainder when divided by 31, which is 26.

The hard part is figuring out the exponent, given only the result and the base. If I tell you I raised 3 to some secret number modular 31 and got 7 as the remainder, you'd have to test a lot of exponents to know which one I picked.

If we make these numbers big, say hundreds of digits long, then finding the secret exponent is nearly impossible.

Now let's talk about how Diffie-Hellman uses modular exponentiation to calculate a shared key. First, there's a set of public values: the base and the modulus that, like our public paint color, everyone gets to know, even the bad guys.

To send a message securely to John, I would pick a secret exponent, X, then I'd calculate  $B^X \bmod M$ . I send this big number over to John. John does the same, picking a secret exponent Y and sending me  $B^Y \bmod M$ .

To create a shared secret key, I take what John sent me and take it to the power of X, my secret exponent. This is mathematically equivalent to  $B^{(XY)} \bmod M$ .

John does the same, taking what I sent to him to the power of Y. And we both end up with the exact same number. It's a secret shared key, even though we never sent each other our secret number. We can use this big number as a shared key for encrypted communication, using something like AES for encryption.

Diffie-Hellman Key Exchange is one method for establishing a shared key. These keys that can be used by both sender and receiver to encrypt and decrypt messages are called symmetric keys because the key is the same on both sides.

The Caesar Cipher, Enigma, and AES are all symmetric encryption.

There's also asymmetric encryption, where there are two different keys. Most often one that's public and another that's private. So people can encrypt a message using a public key, but only the recipient with their private key can decrypt.

In other words, knowing the public key only lets you encrypt, but not decrypt. It's asymmetric!

So think about boxes with padlocks that you can open with a key. To receive a secure message, I can give a sender a box and a padlock. They put their message in it and lock it shut. Now, they can send that box back to me and only I can open it with my private key.

After locking the box, neither the sender, nor anyone else who finds the box, can open it without brute force. In the same way, a digital public key can encrypt something that can only be decrypted with a private key.

The reverse is possible too. Encrypting something with a private key that can be decrypted with a public key. This is used for signing, where a server encrypts data using their private key.



## Cryptography: Crash Course Computer Science #33

Crash Course: Computer Science

<https://youtube.com/watch?v=jhXCTbFnK8o>

<https://nerdfighteria.info/v/jhXCTbFnK8o>

---

Anyone can decrypt it using the server's public key.

This acts like an unforgeable signature, as only the owner using their private key can encrypt it. It proves that you're getting data from the right server or person and not an imposter.

The most popular asymmetric encryption technique used today is RSA, named after its inventors: Rivest, Shamir, and Adleman.

So now you know all the key parts of modern cryptography: symmetric encryption, key exchange, and public key cryptography. When you connect to a secure website like your bank, that little padlock icon means that your computer is using public key cryptography to verify the server, key exchange to establish a secret temporary key, and symmetric encryption to protect all the back and forth communication from prying eyes.

Whether you're buying something online, sending emails to BFFs, or just browsing cat videos, cryptography keeps all that safe, private, and secure. Thanks cryptography!

[Outro Music]

Crash Course Computer Science is produced in association with PBS Digital Studios. At their channel, you can check out a playlist of shows like Eons, PhysicsGirl, and It's Okay to Be Smart.

This episode was filmed at the Chad and Stacey Emigholtz Studio in Minneapolis. And it was made with the help of all these nice people and our wonderful graphics team: Thought Cafe.

Thanks for the random access memories, I'll see you next time!