HTTPS explained with carrier pigeons

Korean translation
Portuguese translation
Spanish translation
Mongolian translation
Persian translation
Vietnamese translation
Turkish translation

Cryptography can be a hard subject to understand. It's full of mathematical proofs. But unless you are actually developing cryptographic systems, much of that complexity is not necessary to understand what is going on at a high level.

If you opened this article hoping to create the next HTTPS protocol, I'm sorry to say that pigeons won't be enough. Otherwise, brew some coffee and enjoy the article.

Alice, Bob and … pigeons?

Any activity you do on the Internet (reading this article, buying stuff on Amazon, uploading cat pictures) comes down to sending and receiving messages to and from a server.

This can be a bit abstract so let's imagine that those messages were delivered by **carrier pigeons**. I know that this may seem very arbitrary, but trust me HTTPS works the same way, albeit a lot faster.

Also instead of talking about servers, clients and hackers, we will talk about Alice, Bob and Mallory. If this isn't your first time trying to understand cryptographic concepts you will recognize those names, because they are widely used in technical literature.

A first naive communication

If Alice wants to send a message to Bob, she attaches the message on the carrier pigeon's leg and sends it to Bob. Bob receives the message, reads it and it's all is good.

But what if Mallory intercepted Alice's pigeon in flight and changed the message? Bob would have no way of knowing that the message that was sent by Alice was modified in transit.

This is how **HTTP** works. Pretty scary right? I wouldn't send my bank credentials over HTTP and neither should you.

A secret code

Now what if Alice and Bob are very crafty. They agree that they will write their messages using a secret code. They will shift each letter by 3 positions in the alphabet. For example D → A, E → B, F → C. The plain text message "secret message" would be "pbzobq jbppxdb".

Now if Mallory intercepts the pigeon she won't be able to change the message into something meaningful nor understand what it says, because she doesn't know the code. But Bob can simply apply the code in reverse and decrypt the message where A → D, B → E, C → F. The cipher text "pbzobq jbppxdb" would be decrypted back to "secret message".

Success!

This is called **symmetric key cryptography**, because if you know how to encrypt a message you also know how to decrypt it.

The code I described above is commonly known as the **Caesar cipher**. In real life, we use fancier and more complex codes, but the main idea is the same.

How do we decide the key?

Symmetric key cryptography is very secure if no one apart from the sender and receiver know what key was used. In the Caesar cipher, the **key is an offset** of how many letters we shift each letter by. In our example we used an offset of 3, but could have also used 4 or 12.

The issue is that if Alice and Bob don't meet before starting to send messages with the pigeon, they would have no way to establish a key securely. If they send the key in the message itself, Mallory would intercept the message and discover the key. This would allow Mallory to then read or change the message as she wishes before and after Alice and Bob start to encrypt their messages.

This is the typical example of a **Man in the Middle Attack** and the only way to avoid it is to change the encryption system all together.

Pigeons carrying boxes

So Alice and Bob come up with an even better system. When Bob wants to send Alice a message she will follow the procedure below:

- Bob sends a pigeon to Alice without any message.
- Alice sends the pigeon back carrying a box with an open lock, but keeping the key.
- Bob puts the message in the box, closes the locks and sends the box to Alice.

- Alice receives the box, opens it with the key and reads the message.

This way Mallory can't change the message by intercepting the pigeon, because she doesn't have the key. The same process is followed when Alice wants to send Bob a message.

Alice and Bob just used what is commonly known as **asymmetric key cryptography**. It's called asymmetric, because even if you can encrypt a message (lock the box) you can't decrypt it (open a closed box). In technical speech the box is known as the **public key** and the key to open it is known as the **private key**.

How do I trust the box?

If you paid attention you may have noticed that we still have a problem. When Bob receives that open box how can he be sure that it came from Alice and that Mallory didn't intercept the pigeon and changed the box with one she has the key to?

Alice decides that she will sign the box, this way when Bob receives the box he checks the signature and knows that it was Alice who sent the box.

Some of you may be thinking, how would Bob identify Alice's signature in the first place? Good question. Alice and Bob had this problem too, so they decided that, instead of Alice signing the box, Ted will sign the box.

Who is Ted? Ted is a very famous, well known and trustworthy guy. Ted gave his signature to everyone and everybody trusts that he will only sign boxes for legitimate people.

Ted will only sign an Alice box if he's sure that the one asking for the signature is Alice. So Mallory cannot get an Alice box signed by Ted on behalf of her as Bob will know that the box is a fraud because Ted only signs boxes for people after verifying their identity.

Ted in technical terms is commonly referred to as a **Certification Authority** and the browser you are reading this article with comes packaged with the signatures of various Certification Authorities.

So when you connect to a website for the first time you trust its box because you trust Ted and Ted tells you that the box is legitimate.

Boxes are heavy

Alice and Bob now have a reliable system to communicate, but they realize that pigeons carrying boxes are slower than the ones carrying only the message.

They decide that they will use the box method (asymmetric cryptography) only to choose a key to encrypt the message using symmetric cryptography with (remember the Caesar cipher?).

This way they get the best of both worlds. The reliability of asymmetric cryptography and the efficiency of symmetric cryptography.

In the real world there aren't slow pigeons, but nonetheless encrypting messages using asymmetric cryptography is slower than using symmetric cryptography, so we only use it to exchange the encryption keys.

Now you know how **HTTPS** works and your coffee should also be ready. Go drink it you deserved it 😉