

LINUX VOICE

TUTORIAL

BEN EVERARD

KEY EXCHANGE: THE SCIENCE OF SECURITY

Maths and physics – two ways of keeping your data safe. Read on, bold explorer...

WHY DO THIS?

- Understand advanced cryptography techniques.
- Beat GCHQ (and the CIA as well).
- Gives you an excuse to use the laser beams that you're stuck to your sharks' heads.

We're used to thinking of secure communications in terms of encryption. If an attacker can't crack the encryption then they can't get into the data, right? Wrong. The encryption method is only one of the many parts that make up a secure exchange.

One chink in the electronic armour is the key exchange. That is, the process by which the two parties decide on which key to use for the symmetric encryption. They both have to know the key, so this has to travel between them in order to communicate. However, an attacker may be listening in, and the key has to be sent in such a way as to stop them being able to find it out.

The simplest way to agree on a key is to use public key cryptography. In this, one party can simply generate a symmetric key, encrypt it with the other's public key, and send it. Then both parties can communicate using the symmetric key.

It's a very simple method, and it works fine. Anyone who intercepts the message won't be able to read it because they don't have the private key to decrypt the message. The only flaw is the fact that the symmetric keys are used over a long period of time, and if one is compromised once, all previous messages can be decrypted. This is a particular problem when organisations like GCHQ and the NSA are intercepting and storing huge amounts of data.

Stopping this is known as Perfect Forward Secrecy (or PFS), and is possible using the Diffie-Hellman key

exchange algorithm, which doesn't use public keys. There's no long-term data that could be compromised that could be used to decrypt past data wholesale.

Perfect security

The Diffie-Hellman algorithm involves three parts that are combined to make the key. It also needs a method of combining them that is a form of encryption, which has the basic property that $(s + a) + b$ is the same as $(s + b) + a$. The '+' symbol is used here generically to mean any secure form of combination. By secure, we mean that if you know s and $(s+a)$, you can't use that to work out a .

If Alice wants to communicate with Bob, and GCHQ are trying to listen in, Alice starts by sending Bob a random number that we'll call s . This is sent in plain text, so everyone can read it. Then, both Alice and Bob make up their own random numbers. We'll call these a and b respectively. They don't send them, but combine them with s first. Alice then sends $(s + a)$ to Bob, and Bob replies with $(s + b)$.

Now they have these, Alice can calculate $(s + a) + b$, while Bob can calculate $(s + b) + a$. As we've said, these are equal, so they're used as the symmetric key.

GCHQ knows a , $(s + a)$ and $(s + b)$, but has no way of calculating $(s + a) + b$ provided we have picked a suitably secure method for combining the numbers. As with all encryption methods, if a suitably powerful computer could be found, then it would be possible to subject this to a brute-force to find the key. However, this would have to be done separately for each run of Diffie-Hellman rather than just once, making it a far less attractive proposition.

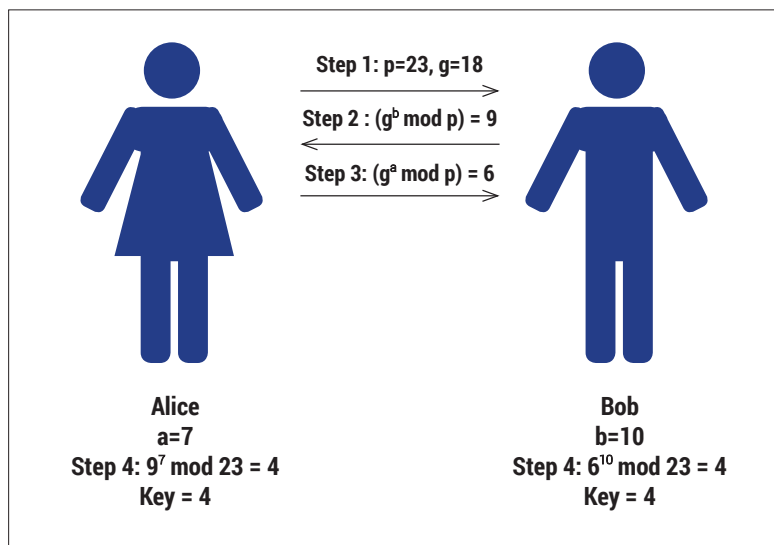
The method that Diffie and Hellman used to combine s , a and b is the fact that for a prime number (p) , and a primitive root (g) , $(g^a \bmod p)^b \bmod p = (g^b \bmod p)^a \bmod p$. Therefore, both p and g are sent in plain text first, then $(s + a)$ was $g^a \bmod p$ and $(s + b)$ was $g^b \bmod p$.

A quantum of security

Diffie-Hellman requires us to have a method of combining the numbers that can't be broken. While there are several options with no known weaknesses, it's possible that a way will be found to decompose the messages $(s + a)$ and $(s + b)$, and this would allow an attacker to break the encryption.

Instead of relying on mathematical properties to allow you to transmit the data securely, you could rely

GCHQ can listen in on steps 1, 2 and 3, but they won't be able to find out the key.



on physical properties. If our key exchange is protected by the laws of physics, we can be far more confident that GCHQ isn't listening in.

This is possible using the Quantum Key Distribution algorithm. In this method, each bit of information is encoded as a single photon sent from Alice to Bob. The data is in the polarisation. If you think of the light wave travelling through space, the wave could be moving up and down, side to side, or at any other orientation. This orientation of the wave is its polarisation.

It is possible to measure the polarisation, but not precisely. Because of quantum indeterminacy, you can only measure it against two perpendicular axes. For example, if you set your axes as vertical and horizontal (0 and 90 degrees), and a photon is polarised at 0 degrees, you'll get a reading as vertical, and likewise for a horizontal photon. However, if a photon has a polarisation of 45 degrees, there's a 50% chance you'll get a reading of vertical and a 50% chance of horizontal.

What's more, by reading the state of the photon, you destroy it.

Using these two properties, Charles Bennet and Gilles Brassard developed a system to send a key so that it can't be intercepted. Again, we'll look at an example where Alice sends a key to Bob and GCHQ tries to listen in.

Alice has a photon transmitter that can send polarised photons in four different orientations: 0 degrees, 45 degrees, 90 degrees and 135 degrees. These are in two groups: 0 and 90 are vertical, while 45 and 135 are diagonal. For each photon, she randomly selects to use either vertical or diagonal. In vertical, 0 degrees represents a binary 0 and 90 degrees represents a binary 1. In diagonal, 45 is 0 and

Public and symmetric key encryption

There are two different types of encryption: public key and symmetric key. In public key encryption, everyone has two keys, one public, one private. The public key is made public, while the private key is known only to that user. When someone wants to send data to the user, they can encrypt it with the public key, and then it's only decryptable with the private key.

In symmetric key encryption (sometimes known as private key

encryption), there is just one key to encrypt and decrypt the message.

Symmetric key encryption is much faster than the public key, and so is used for almost all purposes except authentication. SSH, for example, will use public key encryption to make sure that the server you're communicating with is really who it says it is, and once that's done it will negotiate a symmetric key using one of these key distribution algorithms.

90 is 1. She then sends a series of 0s and 1s with photons and switches between vertical and diagonal at random.

Bob has receiving equipment that he can set up at vertical and diagonal orientations as well. However, if he is set up vertical while Alice is set up diagonal, he will receive the photon incorrectly, and likewise if he is diagonal and Alice is vertical. As Alice sends her stream of ones and zeros, Bob also randomly changes between vertical and diagonal.


Keep GCHQ in the dark

After Alice has sent a long enough string of bits, she sends Bob a list of what orientations she was using for which bits. Bob compares this to how he had his receiving equipment set up. On average, they should have had the same orientation for half of the bits, so Bob replies by saying which bits he was correctly set up for. Both of these messages can go unencrypted since they are

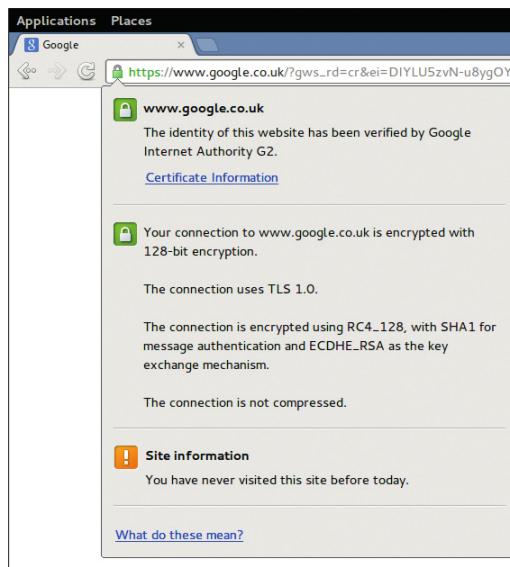
no use to an attacker. Alice and Bob can then use the values of bits that Bob received correctly as the key.

GCHQ can't intercept the photons since they don't know what orientation Alice is as she sends them.

For example, if Alice is vertical, and GCHQ intercept the photon, they have to guess between vertical and diagonal. If they are diagonal, then there's a 50% chance that they will read it incorrectly. In reading in, they destroy the original photon, so they have to create a new one to send to Bob. They have no way of knowing if they read the original one correctly, so they can't be sure either what value it is or what orientation Alice was in. They may get lucky on a few photons, but if they're building up a key of 512 bits, then the errors will quickly mount up.

This might sound fanciful, but there are already some implementations, and April 2014 marks the tenth anniversary of the first bank transfer protected by Quantum Key Distribution (see www.secoqc.net/downloads/pressrelease/Banktransfer_english.pdf for details). 

“Quantum Key Distribution sounds fanciful, but there are already some implementations.”



To see if you're using PFS, look at the technical details of the certificate. If you see ECDHE (Elliptic Curve Diffie Hellman Ephemeral) or DHE (Diffie Hellman Ephemeral), then you have perfect forward security.