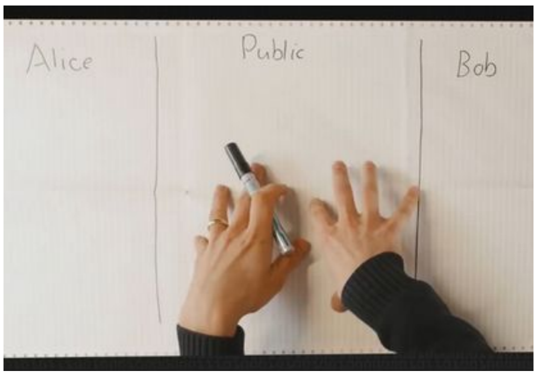
Secret key exchange (Diffie-Hellman) [1]

1. We talked a bit about ***end-to-end encryption*** already, and a lot of the assumption is that we have some kind of symmetric key that we can use to talk privately; so you and me have some kind of secret key, and we use that to talk securely. Diffie-Hellman is how we get that secret key.
2. ***Diffie-Hellman*** was first published in 1976 and has become pretty much a staple for any kind of cryptography at all, right? Whenever we use cryptography, we usually need to have a symmetric key and to get that, we often have to perform some kind of Diffie-Hellman. It's so prevalent that your phone is probably doing it right now, right?
   - When you logged on to the browser to watch this video, you performed a Diffie-Hellman key exchange.
   - When you open up your phone and it connects to any server, it'll almost certainly perform a Diffie-Hellman key exchange – if not now, then in the next few minutes, right?

   It's… its unbelievably important and unbelievably common.
3. The problem with Diffie-Hellman is it's quite ***mathematically complex***; it depends on your level of mathematics. So what I've thought we do is, I thought we cover the mathematics in the extra bits, and then we'd look at a kind of work example of what actually happens as an overview for those people who're just not interested in the mathematics because they don't need to implement it and they don't really… really mind. So, we'll do both and in that way, hopefully, there's something for everyone – we shall see.

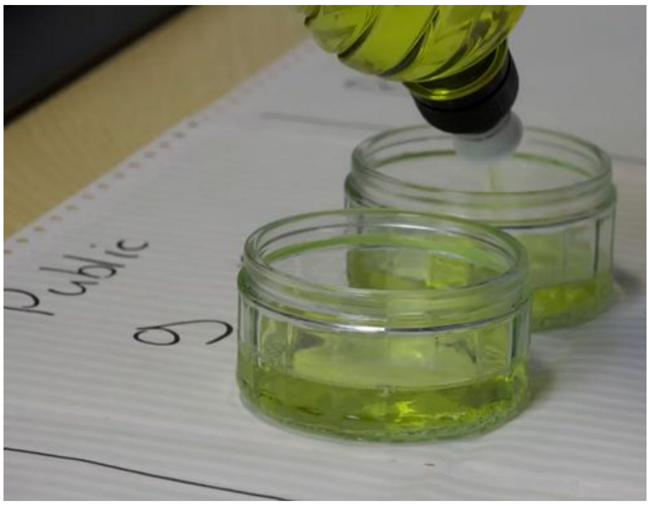
   So, perhaps Diffie-Hellman key exchange is slightly ***misnamed***, in the sense that what we don't actually do is exchange a key – because then it would be out in the public and we'd see it. What we actually do is exchange some public variables and we combine them with some private variables we've kept hidden, so that we can both create the same key, right? – so we're actually creating a key together, in some sense.



4. So, as always, we'll go back to ***Alice and Bob*** for this, so let's have Alice over here, and Bob over here. I'm gonna sort of spread this out a bit, because we're gonna be putting these in… and I don't wanna run out of space. So Alice and Bob are here, these are their own machines, and this is a kind of public area. So anything that Alice and Bob send to each other or agree on in public is gonna be in this area. So as an attacker, if we want to break this key exchange, if we want to find out what the secret key is, we have to use these variables to do it, right? – and that, hopefully, will explain why that's difficult to do.
5. Okay, so, the first thing is, right at the beginning of a Diffie-Hellman key exchange, Alice and Bob have to agree to some ***mathematical parameters*** that they're going to use. This is a value g – or generator – and a big prime number n, right? Now, for this example, I'm gonna use colour mixing to try and explain this (I'm gonna write the letters in as well): n won't have a colour, for the sake of this analogy; g does, right?

1 https://www.youtube.com/watch?v=NmM9HA2MQGI

So ***g*** is gonna be… let's go with yellow, right? Now, I'm gonna sort of squirt this in and hope that it doesn't go everywhere (in fact, we kinda need two copies of *g*, really). So let's just sort of fill it up here… up to about... I wanna get them the same… so far so good, so far it's not all over my desk... alright, close enough, right? Erm… well, it's kind of yellowy, yeah! Often, they're shared at the very beginning of the handshake, sometimes they're just embedded in the standard, or everyone always uses the same one; it depends on the situation. It can take a little time and an extra message to send these things across, so sometimes having them stashed ahead of time is a good idea. So we got *g*, right? this is *g*.



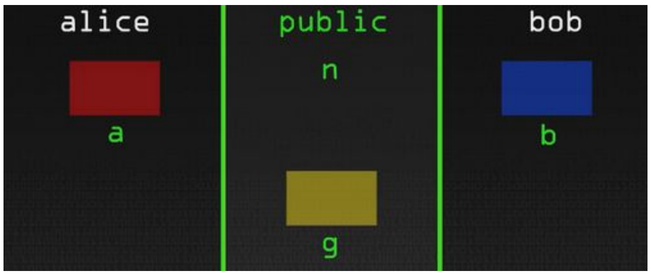
6. Now, Alice, to begin with, needs to calculate a ***private key***, right? – or a private variable. I'm gonna choose red for Alice. Here we go… I probably could have used more food colour than this kind of pale red… Is that red? Yeah, close enough! What do you think?

   Brady — It's rose-coloured.

   Mike — Now Bob is gonna do the same thing, he's gonna have a private value which is going to be blue. Now, I haven't chosen very interesting colours: that's simply because there aren't that many colours available in the shops for food colouring… and I didn't go to that much effort…There we go… I suppose that's blue. Now, these two colours are in their private area. This is ***a*** [red] and this is ***b*** [blue], so I'm gonna label these: this is little *a*, this is little *b*.

   Now, the important thing is that these are ***never shared*** with anyone:

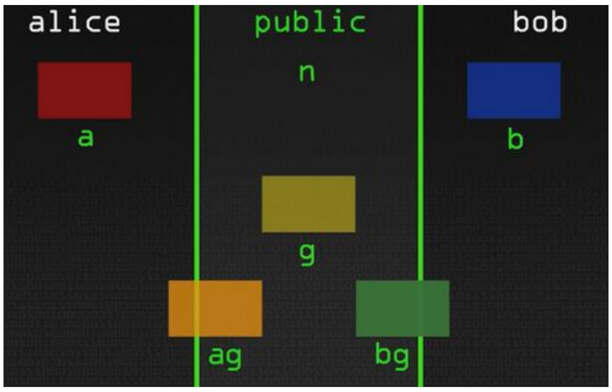
   - Alice doesn't share this with the public;
   - Alice doesn't share this with Bob.



7. Now, the first thing that happens is that we need to ***combine the private key with the generator*** [g] to perform… to produce a public key, right? Now the point is that, once we combined them we can't unmix it, right? That's why people like to use this colour analogy: once we pour two colours together, it's difficult to know what colours went in, right? Because, yes, OK so if I pour red into yellow it maybe makes[2] orange, but it could be that it was a bit more yellow and a bit less red, or, you know, it's difficult to know, right?

2 Transcriber's note: the letter 's' is not pronounced in the video.

8. So there's kind of orange for Alice and Bob's gonna take his blue – we kinda need them to be the same level really – and it does kind of make green. This is a bit orangy – let's not critique me too much. So, yeah, they're very different to the originals and the important thing is that we don't know what went into here, right? We know *g*, but we don't know *a* and we can't find out. So this is actually – this here – this public key here is "*ag*", in some sense: it's got an a in it, it's got *a* g in it, right? This one has got a *b* in it and it's got a *g* in it, and we can't extract the *a*'s, we cannot reverse this process.



9. Now, they then are gonna ***exchange these public variables***, but keep the private ones back. So, we're gonna sort of draw an arrow over here, and an arrow over here, and they're gonna switch them like this, right? So they get sent out in clear text – these are now in the public area because they've been sent in plain text, everyone's seen them.

   So now, as an attacker, I know *bg* (or Bob's public part of this key), *ag* (Alice's public component) and *g* and *n*, right? I don't know anything else; I don't know what *a* and *b* are.

10. Now, this is the ***final part of Diffie-Hellman***, right? It's not actually very long, you can do all this in just three messages:

   - Alice is going to take the public component that Bob sent her, and add her private key.
   - And Bob is gonna take Alice's public component, and add his private key.

   So we're going to get, in essence, a mixture of *a* and *b* and *g*, right? That's the idea. So let's do that now.

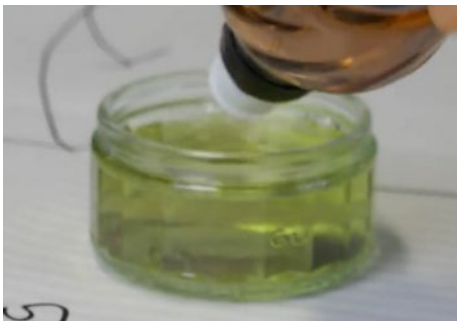
   Brady — So is that in the private domain?

   Mike — Uhm, yes, this will be done privately because these are never... these are never exchanged. So these go into the private domain now – I mean I could make a copy of them, let's not. So Alice is gonna add her red in... so let's go, let's just add some red... up to about there... doesn't really work because the red is really faint... and then Bob adds in his blue... which is gonna be like... that... and hopefully, this is where it all doesn't work or does work: these two values are kind of the same. I mean they're not; they're pretty close! That's a little bit darker, perhaps because the blue is a little bit stronger.

   Brady — Considering you've done that without actually measuring anything.

   Mike — Yeah. I mean obviously...

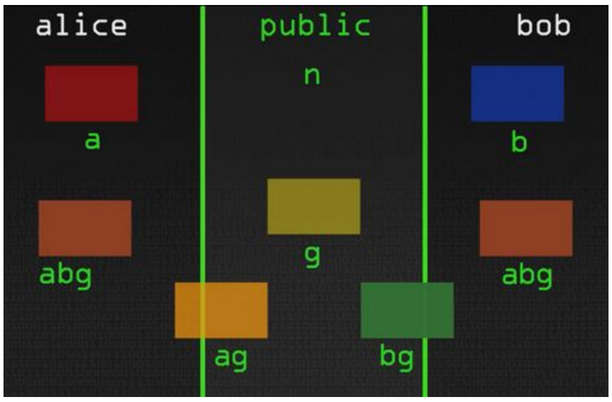
   Brady — That's pretty good!

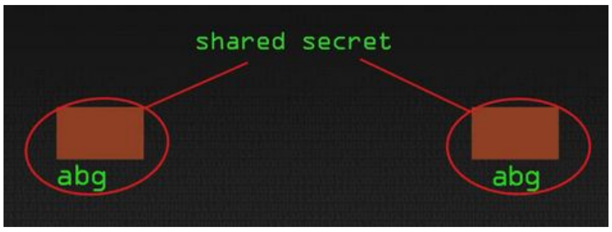


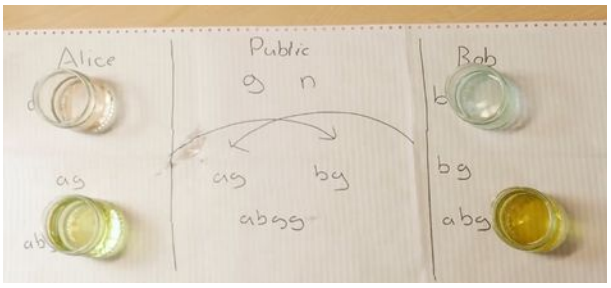
Mike — ... you would do this normally with mathematical, err, mathematics... mathematical functions that are much more precise than my random squirting of... of liquids. Now...

Brady — So those two are now in... in... the private...

Mike — These? Yeah, this is private. So Alice has taken Bobs *bg* and added her *a*, so that gets "*abg*"; and Bob takes Alices *ag* and gets "*abg*" by putting his *b* in there. Right. Now the order doesn't matter. Remember, just like mixing colours, the mathematics is such that if we add in *b* first to *g*, and then we add in *a*, is the same as adding *a* first. So these two values are exactly the same.







11. If you wanted to try and re-create this ***as an attacker***, you can't do it. Because you have *ag* and *bg* and *g*. And, so you could mix these two together, and you'd get "*abgg*" in some sense. Mathematically, this is a little bit... tenuous, but we'll talk about that in the extra bits. The point is, nothing in this public area can be combined in any way to get this value or this value, which are the same. The only way to do that is to find out what *a* and *b* are, and the only way to do that is to split up one of these two public components, which is very, very difficult to do, right? And that's what's so cool about Diffie-Hellman: in a few messages, we've sent some public... some numbers around, and we've used our private numbers to get a shared secret that noone else can know.

    Ehm, now you'd generally do this at the beginning of every conversation, and you would use this number combined with perhaps some session variables or something like this to... derive secret keys to use in things like **AES**, right? So this is actually just gonna be a number which we will then turn... hash to turn into an AES key or something like that.

12. [Teaser] Now, the mathematics behind Diffie-Hellman is, usually, modular arithmetic. Recall that we have our public numbers *g* and *n*: *g* is often very small – it's usually a small prime number; *n* is often very big and needs to be big for the security of this to work (*n* is often 2,000 bits long, or 4,000 bits is more common now) [...]