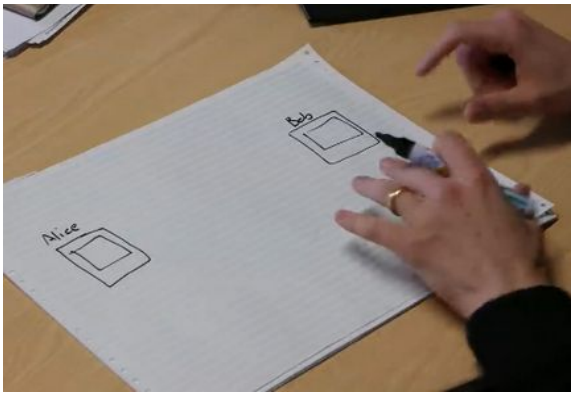
End-to-End encryption (E2EE) [1]

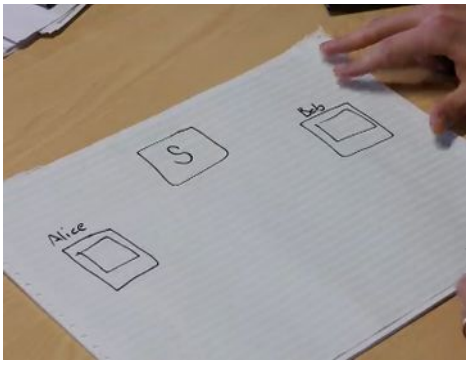
1. So recently, the Home Secretary of the United Kingdom Government was on breakfast news, or something along those lines, and was talking about how criminals are using ***end-to-end encryption*** to essentially evade detection – and that this is unacceptable. Now, in some sense that's very much true: it is unacceptable. Criminal activity is unacceptable, but... what they're suggesting is that we find a way to remove this encryption, or we find a way of only allowing certain parties, like trusted government parties, to have access to it. So, before we declare that as insane, let's look at what that means, and what end-to-end encryption is, and if that's even feasible.



2. Let's imagine that I am using *WhatsApp* or *Facebook Messenger* or some other end-to-end encrypted messenger with you, right? So, you have a phone here, right? It could be a phone, it could be a computer, it's not really important – some device, right, with a screen (this is why I'm not employed to design these things). This is you, but I'm gonna call you *Alice* this time, because we always do that.

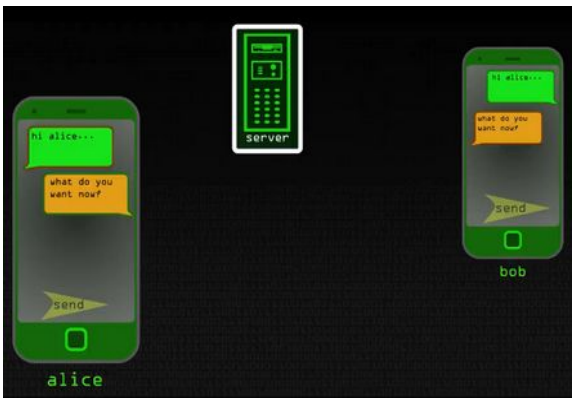
   Brady — Does that make you *Bob*?

   Mike — It does. So we've got ***Alice and Bob*** here having a communication between two phones.



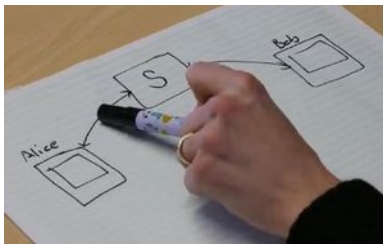
3. There's gonna be some communication mechanism between these two devices, right? It could be SMS or, you know, GSM phone signal, or it could be something like *Wi-Fi* over the Internet. In all of these cases, there's usually gonna be an intermediary server handling this transport. These phones aren't capable of connecting to each other on their own, apart for things like NFC, where you come really close. So, there's going to be some ***server*** in here, which I'm just gonna label *S*, which in the case of *WhatsApp* will be a *WhatsApp* server and obviously it's gotta be a server for whatever product you're using.

   Now, anytime that Bob sends Alice a message, it's going to go via the server, by definition, because that's the thing that relays the message to Alice. It knows how to communicate with Alice, you know, it knows what her phone number is, it has a list of you contacts and things, you know, this is how it works. This could be a phone provider, and there's gonna be, you know, phone antennas and things in this mix, but... it's not important. So, this message here is gonna come in this way from Bob, and it's gonna go to Alice like this.



1 https://www.youtube.com/watch?v=jkV1KEJGKRA

4. The issue is: if we want to encrypt this channel, right, we want certain people ***not to be able to read it***. If I'm sitting on a router somewhere on the Internet, here, we do want me to go "Oh, that's a nice message with your credit card details, and I'll have that", right? So that's what we're trying to avoid here.



Brady — Because that's how email works, right?

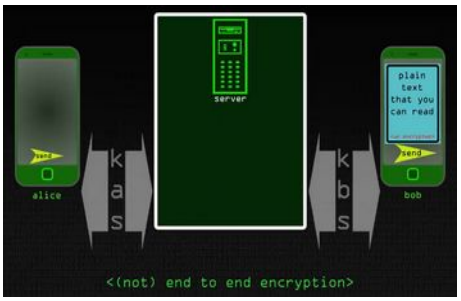
Mike — Yeah!

Brady — You could sit there and…

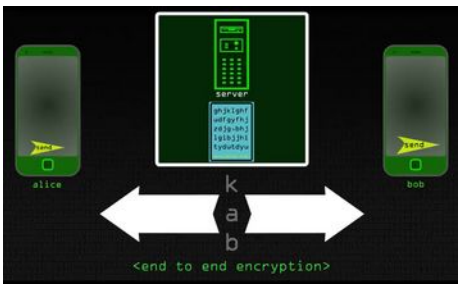
Mike — Absolutely! And people do. ***Encryption of channels*** is nothing new, right? We've seen it for a long, long time, right? These… these techniques – things like public key cryptography and some of these cyphers – has been around for many years. So how do we do this? Well, there's really two options:

5. The first is that Alice could negotiate some ***shared secret key*** with the server. We'll call that key KAS. So, that key there could be used by Alice to talk to the server, and she could send a message encrypted by KAS to the server and say "Please, can you forward this message to Bob?"

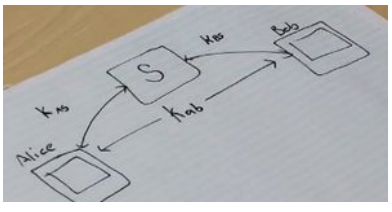
Bob would have another key with the server, KBS, and that's what he uses to communicate. Obviously here, Alice does not know what KBS is, and Bob doesn't know what KAS is. The server decrypts a message, using KAS that it knows, and it re-encrypts it with KBS and forwards it to Bob. Now, this is ***not end-to-end encryption*** because, obviously, it's been decrypted halfway through. In some sense, that's a good thing, right? If I'm a terrorist, or a criminal, and I send a message, this server could perform some kind of rudimentary checks to make sure I wasn't doing anything untoward. But for obvious reasons, a lot of people don't like this idea.



6. What ***end-to-end encryption*** does is replace these two keys with a key that only Alice and Bob know – the idea being that this server is quite happy to relay the packages back and forth, but it doesn't have any idea of what's in them. And this works out very well for this server as well, because when someone says "Can you give us this data?" they can reasonably say "No – not because we don't want to, but because we actually can't!" The process we use for this is something called a ***key exchange***. The obvious problem here is that, at some point, Alice and the server have got to share a key, without an encrypted channel. When she first ever connects, they haven't got this key yet, right? And so, how do we get the key? There's a sort of chicken and egg problem.



7. The solution was proposed by Diffie and Hellman, which is the ***Diffie-Hellman key exchange***, right? We're not gonna go into the details of mathematics of Diffie-Hellman in this video, but I'll simply say that Alice and Bob both have public and private components of this key. They share the public ones, and then they use the private ones in secret to create a shared key that no one else can know. That's essentially how it works.



So it is a way of, even via this server, producing a ***shared key*** KAB that no one else knows. So now, they have this shared communication channel. So when you first connect, you will send some identifiers to the server, you will establish a public and private key pair, and then from then on, anytime you want to connect to anyone new, you'll generate one of these keys. It's called *ephemeral*, which means that basically you generate one almost every message – if not every message for some of these apps.

8. The ***important thing*** is that the server, although they[2] relay these messages, is not involved in this key exchange process, and can't inject itself in the middle, which means that it doesn't know what KAB is, and it can't decrypt the message physically. When the Minister or someone in the media says "what we really want to do is allow some kind of entry for government into the system" you can quite reasonably say "that is impossible!" because you'd have to inject something in the middle of this key exchange, which would completely undo it. So let's think about the different ways we could do it, and discuss whether they're practical.
9. Okay, so the ***first one*** is we could go back to this system here.
    - So, we could have Alice talking to the server in a secure way, using a key exchange.
    - We could have Bob talking to the server in a secure way, through key exchange

    And the advantage would be that if, let's say, a judge ordered a warrant on some of this data, the company would have it on their servers probably, decrypted, and they could send it off. In some sense, I don't absolutely object to that because I don't really have anything to hide, right? – that's the obvious argument. But the problem is that, if this server ever gets hacked, everyone's messages and emails and pictures get dumped out on the Internet, right? We see that happen lots of times. We can't know for sure that this is secure, right? So, in some sense, what we're doing is introducing a ***very big point of failure*** that could be catastrophic, simply so that the very few people who do things illegally could... we could serve a warrant on those people.
10. ***Another one*** alternative that gets sort of suggested is this kind of backdoor. Now, in some sense, this is a backdoor already – this double key mechanism. But when we talk about a backdoor, what we're really talking about is some mathematical property of this key exchange that no one else knows about. That means that we could actually decrypt the messages – [that] is the idea. Again, this is a huge problem. It's a problem because if someone else – a criminal – finds out this flaw, then again all of our photos are dumped on the Internet. And it seems unlikely to me that the majority of people who found this flaw would publicize it straight away, right? They would quite happily sit on it and see what interesting things they could find out. That's a kind of... that's kind of worrying. So again, I have some concerns about that approach.

    Brady — As long as we don't have a backdoor, then there's no way for them to get in there, is it?

    Mike — Erm... Well, so yes and no, right?
11. The ***issue*** is that the messages have to be decrypted somewhere because they have to be presented onto your screen, right? So Alice receives this message, her mobile app receives this message, using KAB decrypts it and then, it's on the screen, right? At this point, someone just steals her phone, runs off and reads her messages or bugs her phone... and reads her messages routinely, has them forwarded on. In this day and age of quite secure end-to-end encryption, the much more likely target of attack is not the encryption itself, it's just the end points! So, I've got your phone here, right?... which you have kindly left the PIN code off for me. And I can just scroll thought your messages and read them all, right? They're not encrypted because that encryption has been removed once it got to this end point.

    Brady — So it's basically automatically decrypted, then.

    Mike — Well, yes. To have a good user experience, it's got to essentially hide all that encryption away and it presents you with a nice set of readable messages. So, in some sense, then your security relies on your PIN code, and the operating system running on your phone or your laptop device, and if those are vulnerable then, you know, really the end-to-end encryption is completely circumvented.
12. [Teaser] This is directly adding content to my normal vision. The problem is the area that it has to add this content is really very narrow, I think it's the [...]

2 Transcriber's note: this proposition should be "it relays [...]" because the subject is the server (singular).