Multi-channel Key Agreement using Encrypted Public Key Exchange (Transcript of Discussion)

Bruce Christianson

University of Hertfordshire

The context for this work is the ubiquitous computing market, where everybody is surrounded by a cloud of little devices that all talk to each other to accomplish various things, and the world we're in is one of talking to strangers.

OK, so Alice and Bob have bumped into each other in a crowded room, they perhaps never met before, they don't know really who the other is, there's perhaps some social protocol that's taken place, for example, Alice has given Bob a business card, or something like that, and on the basis of that, they want their PDAs, let's say, to exchange some information, which means that they need their PDAs to establish a shared key. Now, quite often we say, well of course a global PKI isn't available, and we can't rely on it being there when we want to do things, but in this case it's important to note that a PKI doesn't actually solve the problem that you want to solve anyway. Strong identity based authentication isn't on the menu here, OK, Alice and Bob don't need to know who they are in real life, it might be quite important that they don't discover who the other one really is, and certainly there are lots of applications where they might want plausible deniability, Alice might want to be able to say, well at the time that I did this I had no way of knowing that it was Bob that I was doing it with. So the requirement is, Alice and Bob are each holding a PDA, A and B are the PDAs, Alice and Bob are the people, and Alice is saying to her PDA, I want you to establish a secure session with that PDA there, the one being held by that gentleman there, OK, and I don't want to go into a whole lot of business about who the person holding it is, or what the PDA's registration number is, or anything like that, I just want a secure session with that box right there, look, I can see it. But of course, evil Moriarty is going to try and be the person in the middle and get session secrets shared with him. OK so far?

Well the two weapons that we usually have are either Diffie-Hellman, or a variant of Diffie-Hellman called encrypted key exchange. Everybody knows Diffie-Hellman, we just give each other a public key and we magically have a shared key that no-one else can work out. Encrypted key exchange is a little bit more complicated, instead of exchanging the raw values we super-encipher them with a shared key, and then we calculate the magic number, and the simplest way to verify that each of us is looking at the same key is to just expose some of the low order bits of the shared secret. So we expose some of these bits in order to be sure that we really do share the remaining ones with the person we thought we were sharing with.

Now the problem is that we're not in a position to do either of those things. If we have a channel between A and B that's reasonably high bandwidth, and has high data origin authenticity, so that we know exactly where the bits are coming from that we're receiving, then Diffie-Hellman answers the problem that we have. If you have a high bandwidth channel and you don't have high data origin authenticity, but you can share a secret, even if the super encipherment key k is only a weak secret, you can use encrypted key exchange. EKE is a very good way of leveraging a weak shared secret, up into a strong shared secret, and you can do it as many times as you like with the same weak secret k, but it relies on you sharing some secret bits to begin with [9]. And the difficulty that we have is that Alice and Bob have never met before, so they've had no opportunity to share a secret before the protocol starts, and they have no privacy. We're assuming that Alice and Bob are being watched by Moriarty, Moriarty can overhear not just electronic communications, there's also a security camera in the room, possibly there's spyware inside their PDAs, which is going to take whatever they type into the keyboard and broadcast it to Moriarty.

Michael Roe: Can you have a hash value displayed on one PDA screen? Sorry if that was going to be the punchline.

Reply: No, you're absolutely right, we've got to assume something, there's got to be some integrity someway. Recently there's been quite a lot of interest in what are called multi-channel protocols, where you say, OK, well we don't have one channel with all the properties we want, we have several channels, each of which has some of the properties that we want, can we somehow leverage up from that to do key agreement? So channel one is a high bandwidth channel with absolutely no guarantees about integrity, it makes no guarantees about data origin authenticity, there's no guarantees about anything except that there's high bandwidth. The second channel is a low bandwidth channel, but it does have integrity, and it has endpoint authenticity, it has exactly the properties we want, except the bandwidth is extremely low, and think about 20 to 40 bits in each direction, not nearly enough to do Diffie-Hellman with a decent modulus.

Before I go any further, I ought perhaps to say, you've got to be a little careful in problems like this about where the endpoints of the channel actually are, this is one of the classic traps. Here's what's inside Alice's PDA: we assume that we've got some input and output devices, perhaps a little keyboard and a little display, we've got some sort of crypto module in here, and we're going to assume that the crypto module really, really is invulnerable to attack, we've secured the crypto module somehow, so that Moriarty can't get inside. We've either got a way of generating random numbers, or more likely we've got some random numbers that we've got from somewhere else, and just stored in there, we've got a way of accessing those without Moriarty seeing, we can do encryption inside here without Moriarty seeing it, we can store session keys in there without Moriarty finding out what they are, but the channels from the display and the keyboard to the crypto module can be overheard, we don't have a way of communicating data from the keyboard to the crypto module without Moriarty finding out about it. We do have to assume that at least sometimes we have integrity on

this path, at certain times we can be sure that the keyboard really is connected to the crypto module, and at other times we can be sure that the display really is under the control of the crypto module, but possibly not all the time, it may well be that there are other times when the display is under the control of some other application which Moriarty actually wrote.

Alf Zugenmaier: Do you assume that Alice and Bob will know when the display is under control?

Reply: Yes. We have to assume that there's some sort of red light that says, when this light is on, the keyboard and the display are under the control of the crypto module, and the red light really is under the control of the crypto module, and there's no doubt about it, and Moriarty can't spoof that.

George Danezis: Maybe it's low what I'm going to say, but isn't that basically assuming that you have a secure PDA, i.e. the crypto module? I mean, let's assume that the red light is always on.

Reply: If the red light is always on then you can't share the PDA's resources with anything else, so while people like us might want to do that, other (more normal) people, rather than carrying around hundreds of boxes all linked to each other, would prefer to have multi-function PDAs.

George Danezis: So is that just saying that you have a high and a low, and most of the time you run low, and when you run high the red light is on?

Reply: Yes, you can think of it very much in terms of the multi-hat model that Frank Stajano spoke about here three years ago[1].

Michael Roe: Or, for example, on a Windows system when you type control-alt-del you know you're really talking to the login programme, because the rest of the time you could be talking to anything.

Reply: Yes, it's a similar situation. Instead of having the red light you might just have a little button that, when you click it, puts the system into a ground state in which the display and the keyboard are connected, and you have to do something else to allow any other application to take control.

OK, so we have two channels. You can think of the first channel as being a conventional wireless channel, if you like, there's various ways that people have played about with realising the second channel. The easiest one from the management point of view is some kind of physical contact, you either have a little stud that you physically put the PDAs together, and they have physical contact through that stud, or you connect them by a wire. Of course if you could do that, bandwidth isn't really a problem, and that's also generally not very convenient. Other possibilities are some sort of infrared optical link; music, you can get one of them to play a little tune to the other one, the other one records it, similar to what Michael Roe was alluding to yesterday[2]; or you can get one of them to display a barcode on the screen, and then the other one can photograph the barcode, and extract some bits out of it. In the last resort you can say, well your PDA is going to display something on the screen, and I'm going to type it into the keypad on my PDA, but I bore very easily: I don't mind typing

[1] LNCS 3957, 51–67.

[2] Aura et al., these proceedings.

in something that's about the length of a telephone number because I frequently have to do that anyway, but much longer than that and my motivation starts to wander. So the second channel bandwidth is somewhere between 20 and 40 bits in each direction, about the size of a reasonably good password.

So here's the approach. It's just the encrypted key exchange, except that A and B are using different keys to do the super encipherment. So we do Diffie-Hellman, super enciphered by two weak keys that have been randomly chosen by A and by B, and the interaction through the keyboard is designed to ensure that each PDA is committed to the particular value that it's received before the weak key is revealed. So I start off the protocol in my PDA, you start of the protocol in your PDA, I say, did you get the long super enciphered Diffie-Hellman key, you say (on the second channel) yes I did, and at that point I click the button that says, go for the next step. At that point we exchange the values of the super encipherment key over the second channel, that consists of typing in something a bit like a telephone number, probably I'll put some Hamming forward error correction in it as well, just in case I make a typing mistake. And then we do exactly the trick that we did before, the PDAs can now work out what the values of g^x and g^y were, we do the usual computation, and then we burn some bits of the shared secret so each side can be satisfied that the other side really did lock in the value that they sent and not some value that was intruded by Moriarty. And it's worth noting, these verification communications here are not bit limited, they're happening over the RF channel, you can have as many bits of verifier as you like.

A perhaps obvious observation is that you can get from this protocol back to the EKE protocol very straightforwardly, you just take out all the communications over the second channel, and equate k_A and k_B. So if you have an implementation of this protocol, then you have an implementation of EKE.

Michael Roe: Then don't you have to worry about patents?

Reply: Which may get into licensing problems, yes. We do have some patent-circumventing variations of this from a paper we did sometime ago [9].

So for EKE, we assume that A and B can agree a secret offline, here we are assuming that A and B have no opportunity agree a weak secret, they've never met before, and they have spyware on their PDAs. In the case of EKE the k's are pre-shared and they're long-term secrets, the intention is that the same shared secret can be used to agree many different session keys. In the case of this multi-channel protocol, the k's are short-term, they are initially unshared secrets, and by the end of the protocol they aren't even secret, they are revealed during the course of the protocol. So very different semantics on some of the protocol elements, very different threat model, but in some sense the same protocol.

The protocol also has the interesting feature that the commitment is done on the open channel, but the revelation is done on the closed channel, the low bandwidth, high integrity channel, and a consequence of this is that no amount of pre-computation by the attacker will help. With a lot of multi-channel protocols, if the attacker is willing to put a great deal of work into pre-discovering collisions then a correctly prepared attacker can spoof the participants. Here, the verifiers,

the nonces that we exchange at the end of the protocol, are not bit limited, so we can make them as long as we like, and consequently no amount of precomputation by the attacker will help. Another feature of this protocol which differs from some of the other varieties is that it actually requires a secret to be transferred, well it's not a secret anymore, but it used to be, to be transferred from one PDA to the other over the second channel, it doesn't merely require two things to be checked to see if they're the same. A lot of these protocols in the last step, you end up with two PDAs, each displaying a barcode, or a picture of Mickey Mouse with various accessories, or something like that, and Alice and Bob are supposed to compare them and press OK, or not OK, depending on whether or not they think the pictures are the same. Now what's the chance of them actually doing that, do you think? If a human can be lazy, they will, so although you might regard the fact that the telephone number has to be transferred manually as a liability, it also ensures that it is in fact done: if Alice doesn't type the number on Bob's display into her PDA, the protocol run will fail, she can't just not bother to look at two numbers and say that they're the same when in fact they're not.

There's a general point here about honest-seeming participants. Quite a good model to have is that people will be lazy in a protocol if they think that no-one will notice, so if the protocol says to check g^{xy} to see if it's equal to some particular value or not, they probably won't bother to do it. If the protocol requires some pieces of the value of g^{xy} to be used in the next step, then they have to compute it, or they can't appear to cooperate with the protocol. It's always better to design protocols in such a way that the honest-seeming participant has to actually have done the steps that the protocol says they have to.

Alf Zugenmaier: You do assume that there's no competition between protocols, that your protocol is the only one they can use. If they have an alternative protocol they can use which only requires them to do the comparison, they'll do that instead, except they probably won't, because in most cases the communication is not that important.

Reply: Yes, and if they have the option of turning security off completely, and just getting the job done without any of these annoying pop-up boxes, they'll do that. One of the hard aspects of security is persuading users that it actually adds some value to what they're doing, and that's why I'm also quite keen on protocols which, for example, allow users to automatically retrieve some session state, or something like that, that otherwise they would have to scroll through and recall manually. If that's piggybacked on the back of a security protocol then people say, oh well it's worth doing because it's convenient. So yes, in real deployment of security protocols you do have to be sneaky, to persuade the user that they are getting some benefit from it, otherwise you're just in the position of the religious cult leader who says, well last year our prayers averted five major earthquakes in the San Francisco Bay area.

George Danezis: Maybe I add something controversial to the last point. Would you advise that forcing users to make some effort, also goes along with my advice that we should design a protocol so that it fails spectacularly if users

don't follow it. Because that will create evolutionary pressures for people to do it.

Reply: Oh that's a really nice idea, some sort of security terrorism that says, if you set your password to be a dictionary word, your computer monitor blows up, or something like that.

Japp-Henk Hoepman: It seems almost like a sure way to stop people using security protocols altogether.

Reply: Yes, although it's generally an issue about what you should do when a security protocol fails, how visible should you make it, and to what extent should you allow people to carry on anyway. Or is it enough if the PDA just gives an annoying beep every two seconds if you haven't completed the protocol. It's a good point, we should think about that issue more.

I now want to point out an alternative context in which you can use this protocol. The point is, having got this protocol, you can now push it into various other threat models and see what it does there.

Another thing that's quite fashionable now is trying to do time-bounded protocols, where I say, well I want to establish an association between this PDA and that PDA, and the way I'm going to be sure that I've got the correct PDA is because I think that's the PDA that's closest to me, and so I'm going to do a protocol that relies just on the fact that the speed of light is fast, and Moriarty won't have time to interpose. OK, so when I send a challenge I get a response, and the other side has to be able to prove that it committed to that response in less time than it would take for a light signal to go significantly beyond where the PDA physically is. Well, you can use exactly the protocol that I just showed you to meet this new requirement, but this time the second channel is the time critical channel. You do the same protocol, but now these communications here of k_A and k_B are done in a time critical fashion. Probably you interleave bits from k_A with k_B, A sends a one bit challenge, B sends back the first bit of k_B xored with the challenge, A sends B the first bit of k_A xored with the bit of k_B that B sent, B sends back the next bit of k_B xored with that, and we both have to do that very, very quickly, and at the end of it there's a requirement to have got more than a certain proportion of the bits correct. Because we're going to be talking over a channel as fast as we possibly can, we're going to expect to get some errors, so we just have a security threshold, a trade-off between the number of bits we exchange, and the proportion of those bits that we expect to be got right. So we might say, well you've got to get 90% of the bits right. There are other applications of the same protocol, I think I talk about a couple in the position paper.

So to conclude, on one level this is just yet another multi-channel key agreement protocol, although it does have some nice properties, and some slight differences with the other ones that are on the market. Second, a general plug for the point that sometimes what you want to do is to talk to strangers, and the security requirement is to talk to the correct stranger, and in these cases identity based authentication is at best solving the wrong problem, and at worst is actively counterproductive. Thirdly this protocol travels quite well, it meets a lot

of different specifications in different threat models under different assumptions. And finally to make this point that, working out where the endpoint of a channel is, and how that interacts with local resource management, is something that perhaps more attention should be paid to. The last inch is often the hardest to make secure.

Jaap-Henk Hoepman: Just a comment, actually you have studied almost exactly the same problem that I've been studying a few years ago, and wrote a few papers about, and one appeared in Financial Crypto, except I called it ephemeral key exchange, or ephemeral pairing problem. You're only assuming that the small bandwidth channel is authentic, and I also studied the case where the channel is either uni-directional or bi-directional, and it could either be authentic, or it could be private in the sense that, if I send something to you, you will not know whether it came from me, but I can be sure that you are the only one getting it, and actually if you're using that kind of small bandwidth channel it allows you to do this kind of stuff also. I'll send you the references[3].

Reply: Excellent, please send me the papers, that's brilliant.

Micah Sherr: This is perhaps an obvious point, but if you have a time-critical channel, since you are dealing with PDAs, the speed of light may not be the operative constraint. You obviously have processing delays because you have lower power devices, so if Moriarty is running a Cray supercomputer, and Alice and Bob are using a 20 MHz processor, then they'd definitely go for Moriarty even if he's across the far side of the room, and Alice and Bob are right next to each other.

Reply: Yes, and you know, Moriarty can be doing all sorts of things like listening to the processor noise, because these PDAs are probably unshielded, and so forth. The threat model for the speed of light protocol is very different, and makes very different assumptions about what's in the boxes.

Micah Sherr: I think there are probably similar channels you could come up with.

Reply: I'm suggesting moving the protocol to a completely new environment; I'm not suggesting that Alice and Bob can just take the PDAs that they started with and use them to do this instead.

James Malcolm: Bruce, do you need a new random number for each run of the protocol?

Reply: Yes.

James Malcolm: So you need a lot of random numbers if you might meet a lot of people at this party.

Reply: You need a lot of random numbers to do Diffie-Hellman, and this is no better than Diffie-Hellman from that point of view. But I can always get my PDA to phone home securely and get some more.

[3] J.-H. Hoepman, The Ephemeral Pairing Problem, 8th Int. Conf. Financial Cryptography, LNCS 3110, 212–226, 2004; J.-H. Hoepman, Ephemeral Pairing on Anonymous Networks, 2nd Int. Conf. Security in Pervasive Computing, LNCS 3450, 101–116.