Reverse authentication in financial transactions

A.W. Roscoe, Oxford University Computing Laboratory
Bill.Roscoe@comlab.ox.ac.uk

Chen Bangdao, Oxford University Computing Laboratory
Bangdao.Chen@comlab.ox.ac.uk

L.H. Nguyen, Oxford University Computing Laboratory
Long.Nguyen@comlab.ox.ac.uk

ABSTRACT

New families of protocol, based on communication over human-based side channels, permit secure pairing or group formation in ways that no party has to prove its name, which is particularly suitable for authentication on mobile phones. Rather, individuals are able to hook up devices in their possession to others that they can identify by context. We examine a model in which, to prove his or her identity to a party, the user first uses one of these "human-interactive security protocols" or HISPs to connect to it. Thus, when authenticating *A* to *B*, *A* first authenticates a channel she has to *B*: the reverse direction. This can be characterised as bootstrapping a secure connection using human trust. We find that this offers new opportunities for convenient and secure electronic payment.

Keywords

HISP, Authentication, Reverse

1. Introduction

This is a paper about trust, security and identity management in the world of pervasive computing.

Over the past few years a number of what we might term "Human interactive security protocols", or HISPs, have been developed that permit one or more humans to bootstrap strong security between two or more devices based on the non-fakeable transmission of a minimal quantity of data between them to supplement a normal insecure communications medium. Because the humans know between which systems they have communicated this data (typically a few characters long and which we will refer to as a *check-string*) they know which systems are connected securely. There is an important difference between these protocols and those that bootstrap security from passwords, namely that the check-string does not have to be secret.

This class of protocols allows two or more parties who trust one another, or a single party who trusts one or more others, to bootstrap a secure network using no more than an ability to communicate a small number of bits over the human-based, non-fakeable channel. Another way of looking at them is that if the human(s) involved create an insecure channel between their devices, and already have an unfakable way of passing a small amount of information amongst them, then they can either turn the insecure channel into a secure one or discover the presence of an intruder who is trying to subvert it.

The best of these protocols, for example those of [2, 4–7, 8, 9], enable these humans to be assured that there is no attack that allows an intruder to get the system into an insecure state (where the connections established are other than what the humans believe) with probability meaningfully greater than 2^{-b} where b is the number of bits in the check-string. In addition, to have such a chance, the attacker will have $1–2^{-b}$ chance of his presence being revealed by the difference between the strings. In particular, these protocols prevent any combinatorial searching by the intruder improving its chance of success.

They thus provide a convenient way to bootstrap security that can be used in a wide variety of ways, in contexts both where all the devices are co-located and where they are not, and where the authentication is provided to all devices or asymmetrically to one, because only that device's user has observed the equality required of the check-strings. Similarly they can be used in convenient consumer devices or as part of the security process in a more elaborate type of system.

In this paper we try to understand the sort of authentication given by HISPs and the uses to which it can be put. We look in particular at a case study immediately relevant to everyday life: paying for goods via some electronic transaction in which the payer has to prove their identity to be allowed to use the account, and show how the model it gives for authenticating the payer can be adapted to other applications.

We next give a summary of the challenges that HISPs and this integration with identity management bring to the formal modelling of security, noting that the concepts of trust and authentication are closely intertwined here.

We briefly describe some of the prototype implementations that we have created.

2. Example protocol

The following protocol was designed and implemented by the authors as the first phase of a larger one designed to perform a financial transaction securely. It is closely based on the SHCBK protocol of [4, 5]. Because of its intended application we call the two parties *C* (customer) and *M* (merchant). Before the protocol is run these two parties have no shared knowledge that helps them achieve security, except, naturally, the ability to run the protocol itself.

In order to run the protocol each party must create two values of sufficient strength to achieve the cryptographic goals they have:

- Each party creates a *hash*, or *digest* key: we call these hk_C and hk_M. These are needed to randomise the final

check-string and we assume these are in the range 160-511 bits[1].

- *C* creates a session key *k* whose role is as set out above. This would normally be in the range 120–160 bits, but it could be increased to 512 bits (input width of the basic hash function) without penalty.
- *M* either creates freshly, or re-uses, an asymmetric key pair (*pk*,*sk*). There is no need for the "public" key *pk* to be certified. The length of these keys will depend on the desired level of security[2], the amount of available computing power, and the cryptosystem in use.

The protocol also involves a standard cryptographic hash function which possesses 3 main properties [3]: collision resistance, 2nd preimage resistance and inversion resistance. It depends heavily on the following property of such a function:

> If a party *A* has knowledge of *hash*(*V*) for some value *V*, then while *A* cannot constructively compute *V* other than with infinitesimal probability, it can always check in future whether a value alleged to be *V* actually is *V*.

In this state *A* is *committed* to *V* but *without knowledge* of *V*. In the first pair of steps of the protocol, *C* and *M* both commit each other without knowledge to values. The only one of the four parameters hk_C, hk_M, *pk* and *k* communicated openly is *M*'s public key *pk*:

1. $C \longrightarrow M$::$hash(0{:}hk_C), hash(k)$
2. $M \longrightarrow C$::$hash(1{:}hk_M), pk$

When these messages have been received, both parties are committed to values of all four parameters, but each lacks some knowledge. Thanks to the communication channel being insecure, they have no reason to believe that they are committed to the *same* values of the parameters – but of course they hope they are! Importantly, no intruder can know all four of the original (as opposed to hashed) values as created by the appropriate one of *A* and *B*.

The tags 0: and 1: are added to the hash keys hk_C and hk_M to ensure that the contents of these hashes can be distinguished as coming from a customer or merchant. This avoids the intruder reflecting hk_C back to *C* as a supposed hk_M in a way that *C* would accept. While this attack would not allow the intruder to break the protocol, it makes the analysis more complex – hence the tags.

Even if the intruder has participated in the protocol and impersonated one or both of the parties, it does not know the complete set of parameter values to which either *C* or *M* is committed. This is because no-one except *C* knows its value of hk_C, and similarly for *M* and hk_M.

The protocol now proceeds:

3. $C \longrightarrow M$::$hk_C, \{k\}_{pk}$
4. $M \longrightarrow C$::hk_M

The second part of Message 3 is to tell *C* the actual value of the session key, which is now checked against the hash. In fact this transmission can be delayed until after the check-string comparison if the computation of the public encryption $\{k\}_{pk}$ is time consuming on a low-powered customer device.

It is the transmission of the unencrypted keys hk_C and hk_M at this stage that represents the core of the protocol. Firstly, of course, the participants must check that these are the same values that were represented in Messages 1 and 2. If not, the run is abandoned. Secondly, they (and anyone else who has been listening in) can compute a value for

$$digest(hk_C \oplus hk_M, (pk, hash(k)))$$

where $\oplus$ is bit-wise exclusive or and (X,Y) is an ordered pair. The protocol completes successfully if *C* (or *C* and *M*) are convinced that their two versions of the value – the check-string of this protocol – are equal: in becoming convinced they must not use a channel which can be "spoofed" by an intruder. Typically one will read their value to the other, or *C* will read *M*'s value directly and compare it with her own. Whichever of them knows that the two values are equal can conclude that the link is authenticated. Typically this is either *C* or both of them.

3. Supporting a financial transaction

In many cases such as a financial transaction over the Internet or with a vending machine, or someone seeking to prove his identity and gain access to some service via a machine, there will only be one human present to perform the check of the equality of the digests or similar value used by other HISPs.

For high integrity applications involving a potentially complacent human in this way, there is a strong argument for having the human transfer a value manually rather than simply check that two displayed values are equal. So in fact the device actually performs the comparison – between the one its user has copied from *M* and the one it has computed itself. There are still some interesting variants possible on this. In the following, we will split the entity *C*, representing the combination of a human customer Alice and her security device *SD*, into these two parts.

CA Most obviously and probably easiest in many applications: Alice reads a value from *M* and types it into her own device *SD*.

CS As a variant on this: when *SD* says that the two values agree, Alice presses a button on *M* to signal this agreement.

[1] The upper bound takes account of the common hash block size of 512 and the extra initial bit inserted by the protocol.

[2] The key certainly needs to be strong enough so that there is no realistic chance of it being broken during the life of the session being established. Further strength is required to ensure that the contents of that session remain secret after it ends.

MA Alice reads the value from her *SD* and types it into *M*. In this case it is virtually certain that *M* will signal to the human whether the two values agree. This information may or may not be passed explicitly by Alice on to *SD*. (iii) will be the case where it is not.

MS This is the case like (iii) but where this information is transmitted by Alice to *SD*.

The pairs CA and MA, and CS and MS are clearly mirror images of each other: the first letter tells us who does the comparison in a method, and the second tells us whether the resulting authentication is asymmetric or symmetric. In CA and MA only a one-directional *empirical* channel is used between *SD* and *M*, in opposite directions. In CS and MS, both sides are assured of equality provided our human is trustworthy and reliable.

Note that in cases CA and MA one or other device proceeds without *knowing* that the check-strings actually agreed. Nevertheless these two cases are potentially useful: for example CA can be used when everything confidential and of value that passes through the transaction moves in the direction from *SD* to *M*.

The most obvious case of this is Alice using *SD* to pay *M* for some goods and services: *M* wants to get paid but does not care that much who pays him, whereas Alice only wants to pay the merchant to which she has an obligation to pay.

Notice that this reasoning does not apply when the merchant is passing value to a customer. Imagine that the merchant wants to pass value to the customer who is standing at a particular till, is on a particular phone call etc. If the protocol is run in mode MA it gains the assurance that it is this person's device that is connecting to it, since only she was in a position to make the empirical communication. In practice, however, this situation might well require the customer to pass (e.g. ID) information to the merchant that she would not want to give to *anyone*, and it would be better for Alice not to need to learn a very different protocol for a relatively unusual case. Therefore a protocol giving a symmetric outcome would probably be used here, in particular CS with a warning to Alice about the consequences of pushing the final button incorrectly.

One way of more-or-less ensuring that she does behave correctly and not push the button without knowing that the digests agree is to have both sides compute a bit from the protocol parameters, and use it to have *M* only tell Alice which of *two* buttons to press once it knows the digests/check-strings agree. The essential thing here is making the customer look at the device that has been able to check equality before confirming anything to the other one.

We might note that it will almost always be Alice who identifies that *M* is what she wants to connect *SD* to, and that the connection process itself gives neither party any proof of the identity of the other. Therefore, at the end of the HISP run:

- Unless MA is used, Alice knows that *C* is connected to *M*, as identified by being at the other end of the empirical channel.
- Unless CA has been used, *M* "knows" that the party it is connected to is the one at the other end of its empirical channel.
- Both know they have a shared secret symmetric key with the other, that can be used to secure and authenticate communication between them in a subsequent session.

The connection above has very little in common with the way that most personal financial transactions are performed, at least those involving banks[3]. For the current methods for doing these, whether manual (e.g. typing details from a credit card into an https site) or electronic (e.g., logging in to Internet banking; using Chip-and-PIN terminals at point of sale) are designed simply to authenticate the payer (Alice and/or her credit card) to the payee and/or bank. Typically also, traditional methods of payment give the payee a great deal of sensitive information about the payer: note that anyone who has been paid on-line with a particular card has the information he (or anyone to whom it is unwittingly compromised) to pay for goods with that same credit card; and that nothing proves to Alice that the Chip-and-PIN terminal in her hands will not clone her card and remember her PIN.

What the above protocol does is allow Alice to connect her *SD* to the particular merchant she has been shopping at, whether in person, on-line, or on the telephone. Clearly this authentication is in the reverse direction to the usual sort as described in the preceding paragraph. That explains the title of this paper: it can be viewed as *reverse authentication*.

What we have allowed Alice to do is to create a secure electronic connection with the merchant that she wants to pay, and furthermore where she is assured that the connection has been made within the context of the *particular* transaction for which she is willing to pay.

This electronic connection will allow a great deal more information to pass between her device (card/SD) that is otherwise possible unless she puts her card into the hands of the payee, or at all on-line.

What the reverse authentication achieves, in summary, is not to *replace* the usual ID check on Alice and her device, but to make it potentially more thorough, particularly on-line, easier for Alice (because in most cases she will have to do less), and to remove the danger of Alice's secrets getting compromised. We will demonstrate this below.

We believe that beginning a financial transaction with the HISP authenticating merchant to customer makes sense in all of the following cases.

(A) Electronic cash: the customer has some device with her that contains value, and she wishes to transfer some of that value to the merchant. It might well be the case that she wishes to do so anonymously.

(B) Credit card or cheque: the customer wishes to give the merchant the right to take some sum of money from her account. Note that, although the mechanisms are rather different, both conventional credit card transactions and paper cheques have this logical effect.

(C) Electronic banking: the customer wishes to give her bank a direct instruction to pay the money into the merchant's account. The logical difference with (B) is that the bank

[3] As we will see, good old fashioned cash transactions resemble our connection.

must be involved directly, and the merchant never holds a token that is good for money.

This first dimension influences the way the payment proceeds after a secure link has been established, and how identity issues arise. (A) is different from the others because our customer will not have to prove her identity (Alice) to anyone, while in the other two it is desirable[4] that she (separately from her device) proves that she is entitled to use the account by the entry of a secret PIN or biometrics.

In each of these we imagine a variety of payment situations, which influence how the authenticated connection is made.

(i) The customer is sitting at a desk and shopping on-line. Here we assume that the customer and the banking system do not trust the PC except possibly through an https windows displayed on browsers. [This is not to say that this last mechanism is 100% secure, but since e-commerce and digital banking rely on it currently, it seems reasonable that we can also provided we do not increase the risks inherent from using it in present methods[5].]

(ii) The customer is trying to pay the merchant in person: in present technology she would hand over cash or credit card, or place her card in some reader presented by the merchant. Here, the merchant might be a machine, or might be a manned till.

(iii) The customer is shopping over the mobile phone.

In case (C) (mobile banking) we do not concern ourselves with how the secure connection between phone and bank is made, since we can reasonably assume that there is a long-term key which achieves this. In all cases we assume that a HISP is to be used to connect the phone to the merchant that is to be paid, thereby proving to Alice that she is paying the correct entity within the correct transaction.

Except in the case where the paying device SD is the same telephone over which the transaction is being conducted, we need to get some connection between SD and merchant established to that it can be authenticated with a HISP. In technology used for everyday payments, both of these phases need to be very easy. One advantage of using a HISP is that the customer's view of the second phase can be the same in every case. The following sets out a few options for making the initial insecure connection in the on-line and point-of-sale cases.

In the on-line case there are two main options for this connection: the first of these is using the home PC on which the shopping is being done as a link between SD and merchant. The link could then be made by wire (e.g. USB), wireless (e.g. WiFi or Bluetooth) or infrared. None of these is technologically difficult, and the session on the browser can instruct the PC about where to route the communications.

The second is using telephony to make the connection: this may be the only option when Alice is forbidden to connect any personal device to the PC. The only problem then is giving one or other side of the connection (phone=SD or merchant) the information required to connect telephonically to the other. This will be the combination of a telephone number and a (probably one-time) token that identifies the particular transaction. The merchant's number can be transferred to the phone from the PC by (e.g.) Bluetooth, but of course this would fall foul of the no-connection rule if this applied. The user's number can be pre-loaded into the browser (and there are strong arguments for this being a separate number from the one used for ordinary phone functions), and this sent together with a one-time token to the merchant when a button on the payment site is pressed.

In the point-of-sale case, the same two options (local and telephonic) connection apply. A literally wired local connection is unlikely, but it would be possible to place a phone into a special cradle. In that case it is probably not necessary to use a HISP, since the phone is obviously connected to the merchant. Similarly, if a connection is bootstrapped from a physical connection that Alice can *see*, this is still probably not necessary, and the same may apply for low-value transactions if it is bootstrapped from very short range radio as used for example, in Oyster cards [1], provided this generates a session key. A HISP will provide additional assurance in this last case for high-value transactions. Other options include Bluetooth connection (where it may be necessary for Alice to select which till she is at) or telephony (where the number can be transferred using any of the methods described above, or perhaps via scanning of a bar-code displayed on the phone).

In all payment methods, we assume the first action after the secure session is established would be for the merchant to send the SD details of the transaction it wishes to be paid for plus secondary security information such as its name and logo. If Alice agrees to the payment, she will either press a button or enter the personal information (e.g. PIN or biometric) needed to confirm her presence. Whatever payment token is then sent by her SD will then contain the secondary security information so that a fake merchant who has "borrowed" these should not be able to obtain payment from them.

An electronic cash payment would simply follow the appropriate protocol over the secured session.

For a credit card transaction we either have greatly increased the communication possibilities between SD and merchant or (particularly when Chip-and-PIN terminals are replaced) ensured that there is much less availability of customer information to merchant. In either of these cases it makes sense to replace present payment methods by the SD giving the merchant an e-cheque containing

- Payee, payer, amount, credit card details and time-stamp, as on a conventional cheque.
- Transaction ID.
- Any secondary security information about the payee that Alice has confirmed. *The bank will confirm that this ties up with the payee.*

[4] We note that in some present credit card transactions, especially on-line ones, she does not have to do this.

[5] In fact we argue that our methods provide a higher degree of security than the traditional use of https sites, since we are only relying on the communication through it being authenticated, not secret. Thus neither screen-shot grabbing nor key-sniffing would benefit an attacker.

- Evidence that Alice has correctly proved her own identity. This might be either the actual information (e.g. PIN) she has input, or evidence both that the SD/card has confirmed this information (noting that at present PINs are typically confirmed by a credit card and not transmitted) and that the SD/card itself is genuine and behaving properly.

This would be encrypted a under key that merchants cannot understand (e.g. a symmetric key specific to this SD/Card or the public key of the banking system) and sent to the merchant to be forwarded and authorised by the banking system.

Perhaps the most attractive scenarios for using HISPs for payment comes in the context of *mobile banking* (i.e. on-line banking on a mobile phone). Systems implementing this with limited functionality are rapidly being developed by banks and rolled out to customers, but none that we are are aware of allow the user to make a payment a general point-of-sale or on-line merchant. The deficiency can be remedied once the phone is securely connected by HISP to the merchant. For then *M* sends details of the transaction for Alice to confirm, plus bank account details to which the money is to be paid. When *C* confirms and gives whatever authorisation code is required by her bank, the on-line banking session automatically generates a transfer to *M*'s bank, and an unforgeable certificate that this has occurred is sent by *C*'s bank to *M* via *C*. The value of the HISP here is that it ensures that the bank account details really come from *M*.

It is worth noting that the total effort that Alice has to make in running a HISP and confirming the transaction on her SD is substantially less than is required of her in conventional on-line purchases using credit cards, whether these are performed by entering card details onto a web-site or by entering her PIN into a secondary device provided by the card issuer and then copying a one-time authentication code into the web site. (Devices such as these are, of course designed to help Alice prove her identity – they do not help Alice to create an authenticated connection to the merchant.)

4. Implementations

We have implemented various mobile phone applications using versions of SHCBK, including the one described above. The following is a brief of some of these.

We have implemented two main variations on payment using mobile phones, A and B. In the first, the payee is assumed to be a merchant, as is the case for today's credit and debit card payments. The second is peer-to-peer payment between mobile phones. In both cases we have assumed that the payment is managed through online banking accessed through the mobile phone.

In scenario A, we have assumed that the purchase is made online. However, it would be easy to convert the following to Point of Sale (POS).

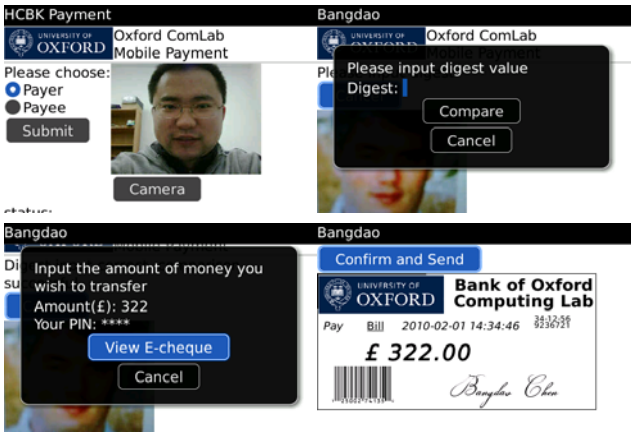
1. The customer C has come to the point of paying on an internet session and is confident that the HTTPS session is connected to the merchant M.
2. C presses a button on the website for mobile payment and starts (*) the payment application on his mobile phone. The button gives C's phone payment number to M securely via HTTPS.
3. M calls C's mobile phone and runs the initial messages of the protocol with it.
4. M calculates the digest and displays it on existing HTTPS window.
5. Assuming C wishes to carry on; he types this number into phone which then decides if numbers agree. Agreement gives secure connection.
6. M sends details of the payment it wants over the secure (authenticated and encrypted) connection including amount, name, possible logo and bank information.
7. The payment is displayed on mobile phone (in our implementation, in the form of a cheque) and C is asked to confirm payment (*).
8. Payment is processed by e-banking, which generates a "receipt" to send to M.

As discussed earlier in this paper, it will be necessary in practice to have the customer prove his identity as part of this process. One or both of the points marked (*) are appropriate.

The second works similarly. In our implementation, the mobile phones connect via Bluetooth, but telephony and other routes are also possible.

The two users will connect their phones much as C and M are connected above. Now C will probably enter the amount to pay rather than confirming the payee's request. In our implementation, the payer sees a cheque appear on his/her mobile phone to confirm paying it.

Screen shots of B:



We note that in neither scenario is any confidential information given by C to the payee. This will considerably reduce the opportunities for fraud.

The cryptography functions we have applied in the applications comply with the guidance published in FIPS 186 -3, FIPS 196, SP 800-78.

5. Conclusion

We have seen how, in transactions involving Alice proving her identity to some party Bob whom she can identify by context, it often makes sense for her to get a connection that she knows is with Bob, even if she does not know Bob's name. She can then use that connection to prove her identity securely, and perhaps perform other functions, and has no need to place a credit card, identity card etc in the hands of another party, thereby enabling

her to control what information is taken. In other words, before she authenticates herself in one direction, she performs an authentication in the *reverse* direction.

We believe that this technology will have many applications both within the area of financial transactions highlighted here and more widely.

6. Acknowledgements

We are grateful to Ronald Kainda and Ivan Flechais [10] for their work with Roscoe on the human factors of HISPs. Several MSc students, in particular Qiulu Zhao and Keith Awyong, helped us in assessing the efficiency of various cryptographic schemes. Emma Sceats of ISIS Innovation Ltd and a number of researchers from the banking industry have helped us to understand what is required there and enabled us to understand the real-life problems that protocols for financial transactions need to solve.

7. REFERENCES

[1] See: http://en.wikipedia.org/wiki/Oyster_card

[2] S. Laur and K. Nyberg. *Efficient Mutual Data Authentication Using Manually Authenticated Strings.* Volume 4301 on LNSC, 90-107, 2006.

[3] A.J. Menezes, P.C. van Oorschot and S.A. Vanstone. *Handbook of Applied Cryptography*. ISBN: 0-8493-8523-7.

[4] L.H. Nguyen and A.W. Roscoe. *Efficient group authentication protocol based on human interaction.* In Proceedings of FCS-ARSPA 2006, 9-31.

[5] L.H. Nguyen and A.W. Roscoe. *Authenticating ad hoc networks by comparison of short digests.* Information and Computation 206 (2008), 250-271.

[6] L.H. Nguyen and A.W. Roscoe. *Authentication protocols based on low-bandwidth unspoofable channels: a comparative survey.* Submitted to Journal of Computer Security. See: http://www.comlab.ox.ac.uk/files/2104/Compara.pdf

[7] L.H. Nguyen and A.W. Roscoe. *Separating two roles of hashing in one-way message authentication.* Proceedings of FCS-ARSPA-WITS 2008, 195-210.

[8] A.W. Roscoe. *Human-centred computer security.* (2005) See: web.comlab.ox.ac.uk/oucl/work/bill.roscoe/publications/113.pdf

[9] S. Vaudenay. *Secure Communications over Insecure Channels Based on Short Authenticated Strings*. Advances in Cryptology - Crypto 2005, LNCS vol. 3621, 309-326

[10] R. Kainda, I. Flechais and A.W. Roscoe. *Usability and Security of Out-Of-Band Channels in Secure Device Pairing Protocols*. In the Proceedings of SOUPS 2009