Human-centred computer security

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

Abstract

We re-examine the needs of computer security in pervasive computing from first principles, specifically the problem of bootstrapping secure networks. We consider the case of systems that may have no shared secret information, and where there is no structure such as a PKI available. Nevertheless we propose a protocol which achieves a high degree of security based on a combination of human-mediated communication and an ordinary Dolev-Yao communication medium. In particular it resists combinatorial attacks on the hash values that have to be compared by human users, seemingly optimising the amount of security they can achieve for a given amount of work. A variant of this protocol achieves one-sided authentication in a scenario similar to that of authenticating one or more bluetooth devices. We discuss the properties and verification of these protocols.

1. Introduction

Imagine that a group of people come together and agree that they want to transfer data between them securely, meaning that they want it to be secret and of authenticated origin. They all have some piece of computing hardware (e.g., a mobile phone or a PDA). Unfortunately none of them knows the unique name of any of the others' equipment, and in any case there is no PKI which encompasses them all. How can they achieve their goal in the context that their machines are connected by an insecure network (whether created by WIFI, the internet, telephony, or a mixture of these)?

The conventional answer to this question would be that it is impossible, since it is impossible to prevent some imposter I playing the man-in-the-middle between the participants A. For an example see Figure 1. Here, 5 users are trying to form a group and believe they have the connections on the left. But the intruder has partitioned them into two subgroups and invented 5 identities of its own to make both of these up to the right size. It then sets up group sessions involving, respectively, $\{A, B, C, D', E'\}$ and $\{A', B', C', D, E\}$ and can then pass transactions between the two groups or, if it wishes, impersonate a member of the group. As far as they know, $\{A, B, C, D, E\}$ are connected to each other, and cannot tell that the intruder is present since they have no security framework to distinguish, say, A from A'.

However a little creative thinking can easily solve the problem: if each person tells the others (using human conversation) a public key for his or her machine, they can then use something like the Needham-Schroeder-Lowe protocol [11] (over the insecure network) to establish secure, authenticated communications. (If there are more than two participants then they would either have to adapt that protocol or to use it multiple times.)

They will have bypassed the intruder for the crucial step of exchanging electronic identities.

Unfortunately this idea would require a serious amount of effort on the part of the humans unless, perhaps, they all carried a card with them containing their public key that other machines could read – but of course that would again introduce a compatibility problem as well as limiting the range of applications. What we shall demonstrate in this paper is that high quality security can be obtained using this same idea of human activity bypassing the intruder for crucial steps, but with a greatly reduced amount of work. Indeed, it seems that we essentially minimise this.

Putting this in the context of earlier work, the author, Creese, Goldsmith, Zakiuddin and others [4, 3, 1, 2, 5] have developed the idea that the concept of a PKI is not ideal for many pervasive computing applications, for the following reasons. It is not realistic to assume that one is both sufficiently universal and sufficiently available to cover all scenarios. In any case, in the pervasive world, processes will not know the names of the others they wish to contact – essential for a conventional PKI. And finally, it is unrealistic to expect the average user to understand, and react properly to questions posed by, PKI such as "Do you trust certificates issued by authority X?"

If we are not identifying a system by name, there must be some other feature which identifies the ones we wish to

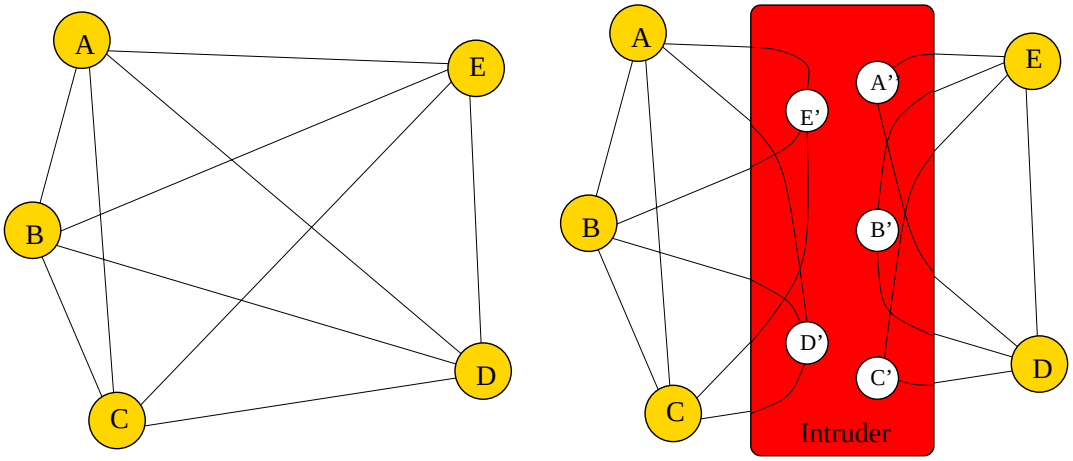

Figure 1. Man in the middle

interact with securely. In pervasive computing this feature is frequently one of position or proximity [10, 13]. We have argued that these features can often be captured as low bandwidth *empirical channels* between systems in addition to the insecure (Dolev-Yao) high bandwidth channels used for most communication. In the scenario set out earlier, these empirical channels would be implemented by the human users and can be assumed to be non-spoofable[1]: an intruder can't persuade A that an empirical message is from B if it is not.

We gave several protocols in [3, 1, 2] for different classes of empirical channel, an exercise we intend to continue. These included several related protocols for the scenarios of a group of people attempting to form a secure network, and of a single user connecting to an external device, such as a printer, with a high degree of security. The assumption was that the human or humans implement non-forgeable channels between their systems.

We emphasise that the protocols discussed in this paper are designed for groups of systems that already trust each other: we enable these to gain authenticated connection to each other. The protocols do not of themselves prove that agents are trustworthy or have particular names. In the case where the empirical channels are human based, our protocols essentially lift trust that the human(s) have, to computer security. This explains the name of this paper: *Human Centred Computer Security*.

In this paper we show that such protocols can be vulnerable to combinatorial attacks, meaning that the humans have to handle larger data items than ideal. The main innovation of this paper is a pair of protocols, one for the group scenario and a variation for the user-plus-devices one, which appear to be essentially optimal in the amount of security they offer in return for a given amount of human effort: we analyse this issue in Section 4.

Since these are protocols for arbitrary-sized groups and are intended to defend against a new and stronger attacker model they pose new problems for the verifier. We discuss these issues and set out some progress. a given amount of human effort to supplement Dolev-Yao channels.

2. Protocols for *ad hoc* groups

In this section we concentrate on a group **G** of humans attempting to achieve a secure link between their laptops. Throughout this paper, $N \geq 2$ is the size of **G**. A protocol for just these purposes was proposed in [3]:

1.	$\forall A \rightarrow_N \forall B$	: $\{A, kc(A), N_A\}_{wkk}$
2.	$\forall A \rightarrow_N \forall B$	: $\{\text{all Messages } 1^d, N'_A\}_{pk(B)}$
3*a*.	$\forall A$ *displays*	: $hash(\{\text{all Messages } 2^d\})$, number of processes
3*b*.	$\forall A \rightarrow_E \forall B$	: Users compare hashes and check numbers
4.	$\forall A \rightarrow_N \forall B$	: $hash'(\{\text{all Messages } 2^d\})$

Here, $\forall A$ means that a message is sent/received by all the participants A. In some cases this might be a broadcast, in others a special version will be required for each recipient. M^d denotes the decrypted contents of message M. The boundary between Message 1 and Message 2 is determined

[1] "Spoofing", "faking" and "forging" are synonymous terms used in the literature for the same activity by the intruder.

by some time-out. The purpose of Message 3 is to allow all the users to compare the hashes and only to proceed if these are the same and the number of participants is the same as the size of the group who are trying to form a connection (and whom they intend to constitute the group).

In the notation of [3], $\rightarrow_N$ means the normal Dolev-Yao network, where all messages are subject to overhearing, taking out and spoofing, whereas $\rightarrow_E$ means the empirical channels which are, in this case, immune to spoofing. Note that actually a two-way exchange is necessary for the agreement in Message 3b: each user tells each other his or her values, who then indicate their agreement. See Section 4 for more discussions on patterns of communication.

$kc(A)$ means A's *key certificate*. We have to be slightly careful here about the concepts of *name* and *public key* here since it is in the nature of our *ad hoc* networks that these are not generally known in advance, and public keys may not be generated by any recognised authority. Therefore one system hearing a name A in a Message 1 carries no particular information to the recipient other than that an entity calling itself A is trying to make contact. In many circumstances $kc(A)$ will simply be a public key, and indeed the name A might also be replaced by the key and the two fields conflated. An agent is free to use different names and different public keys in different runs of this protocol, and there is no need for public keys to be certified by any authority.

Under standard Dolev-Yao assumptions about the $\rightarrow_N$ channel and about encryption and hashing, for example that given a hash value it is infeasible to create a preimage that will map to it, it is straightforward to show that this protocol is secure. This was done in some detail in [2]. The essential point is that the agreement on Message 3 demonstrates that all the participants are present in the protocol run, and that there are no more participants other than the intended group. For under our assumptions, agreeing the hash value is tantamount to agreeing the things hashed (though without making them public).

The assumption that all cryptographic constructs are, in effect, unbreakable is fairly standard in the world of cryptographic protocol modelling and is termed the *perfect cryptography hypothesis*. It is made in the natural desire for separation of concerns. The idea of perfect cryptography has been weakened in some instances to allow for the special algebraic and deductive properties of individual cryptosystems such as Vernam and RSA, but still in an entirely symbolic way: creating perfect cryptography under different sets of rules.

In essence, the perfect cryptography hypothesis is a specification that must be satisfied by a combination of able cryptographers and sufficiently long keys, nonces, hashes etc. In the context of the protocol above, it is a reasonable assumption to make about all the cryptographic objects that are transmitted on $\rightarrow_N$.

There is, however, the question of whether it is valid for the hash that the human users compare. Let us imagine our human users comparing their hash values. They are not likely to want to handle more than a few characters. In some scenarios they might have a better way of comparing these values than just reading them out, but we need to consider the worst case, for example on the telephone. The hash length we are likely to be able to use conveniently may be vulnerable to a combinatorial attack as long as an intruder (in the guise of A'–E') can try many nonces so as to get a particular value in Message 3. An attack achieved by exploiting this would not be found under the perfect cryptography hypothesis. Let us assume that the set of possible human-compared hash values has size H.

There is indeed a such an attack on the above protocol. This involves partitioning a group $\mathbf{G}$ of N agents who want to form a single session into two nonempty subsets S_1 and S_2, with the intruder engaging in a separate session with each. For each, it uses exactly the right number of identities to expand the subset to size N, in each case using identities for which it has the secret keys. Therefore it uses the scheme we have seen in Figure 1. The two runs of the protocol operate in parallel, with the intruder manipulating the Dolev-Yao network so that only the messages within each subset get through. In each case it runs Message 1 with the appropriate group S_i and waits for all the nodes in $\mathbf{G}$ to contribute their Message 2's to the two runs. It then must invent N different nonces N'_A to transmit its Message 2's. Using whatever time it has, it can search for combinations of values which make the two hash values in Message 3 the same (note that it has all the information to compute these hashes for any choice it may make of nonces). Note that, since the intruder can work with the two hashes at once, it can expect to get a coincidence if it creates just over $\sqrt{H}$ hashes on each side, thanks to the "birthday paradox". For example, for a hash consisting of six digits, only about three thousand hashes must be computed for the intruder to have a 90% chance of success. If it succeeds in doing so it transmits the appropriate Message 2's to the respective members of S_1 and S_2. This is illustrated in Figure 2. In this figure, the line connecting A–E with only an arrow *to* the intruder represents the no-spoofing empirical channels. We see the intruder trying to make the communications on these consistent with the man-in-the-middle structure derived from Figure 1. The box at the bottom right represents the intruder looking for, and finding, a coincidence between two lists of hashes.

Upon reaching Message 3, all the humans (from the two groups combined) will then see identical hashes and correct counts on their laptops, and so will agree to proceed beyond Message 3. The intruder then just has to manipulate the network for Message 4 and contribute the correct final message in each of the two sessions. This leaves our

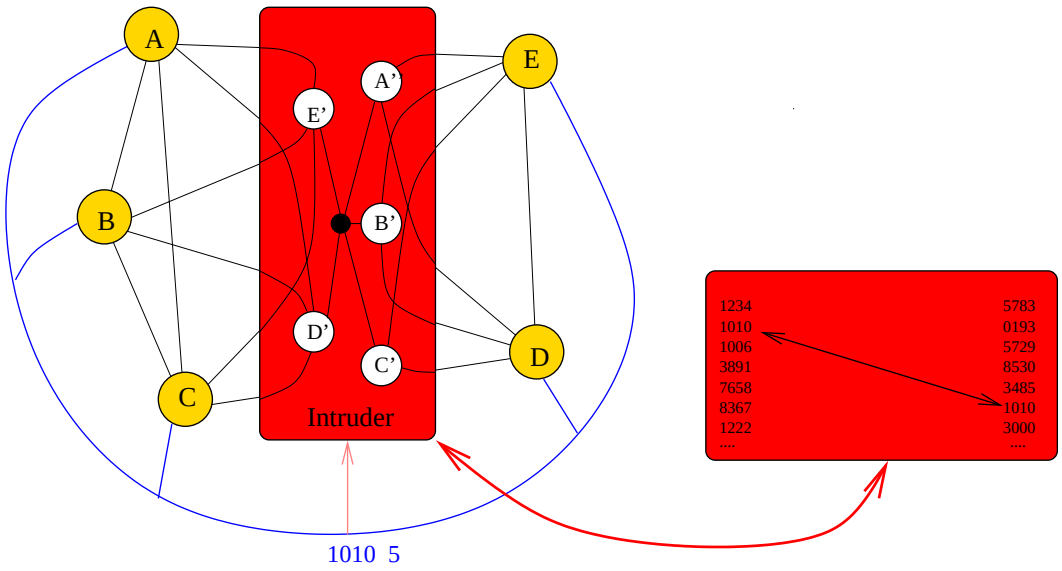

Figure 2. Birthday attack in progress

intruder with connections to the two subgroups, including knowledge of all the secrets arising from the runs. They, on the other hand, believe there is a single authenticated session between the whole of **G**. The intruder is then in a position to transmit the group's messages while understanding them or to impersonate group members to each other.

"Birthday attacks" like this are well known in other arenas where cryptographic hash functions are used. See [12], for example. They are, however, brought into particular focus by the human role in the protocols we are now discussing. We can take account of an attacker who can work out 2^M hashes by adding $2M$ bits to the hash function, but of course this means that a more powerful attacker could make headway, and in any case it involves the humans in extra work. So if k hash bits would have been required had combinatorial attacks been impossible, $2M+k$ are now actually required.

Similar attacks exist against all the protocols of [2]. In [1], several protocols are proposed for authenticating a device such as a printer to a user. Of these, only the following one (Protocol 2) addresses the objectives set out above, since others are either for a different sort of empirical channel or require confirmation of more data than we would like (including a user's unique ID). It provides an interesting half-way house as regards this type of combinatorial attack. In the following, A is a user and B is a printer, and $pk(P)$ is the public key of all printers, which plays essentially the same role as *wkk* above. K is intended to be used as a session key.

1. $A \rightarrow_N B \quad : \quad \{A, pkA, N_A\}_{pk(P)}$
2. $B \rightarrow_N A \quad : \quad \{A, B, pkA, N_A, K\}_{pkA}$
3. $B \rightarrow_E A \quad : \quad hash(Message\ 2)$

This is secure under perfect cryptography. It does not appear to be subject to a classic birthday attack since these can only work if the hash value is fixed by the attacker, whereas in the above it is always fixed by the printer (through the values it sends in Message 2). However, if the attacker has the ability to compute the same order of magnitude of hashes as the size of the hash space (far from inconceivable if only a few digits or letters are agreed) there is still an attack. The intruder acts as a man in the middle, switching his own public key *pkI* into the Message 1 received by B. Then, when he receives Message 2, he seeks to find K' such that $hash(\{A, B, pkA, K'\}_{pkA}) = hash(\{A, B, pkI, K\}_{pkI})$ The number of hash bits required to counter this would here grow like $M+k$ rather than $2M+k$.

The first protocol could be strengthened to $M+k$ growth as opposed to $2M+k$ if it were de-symmetrised so that a member of **G** declares themselves to be the finaliser of the hash. (To do this, this node must not issue its Message 2 until it has received Message 2's from all other participants.)

One can, however, do better than that. We now propose a different protocol for the first scenario, but which seems to avoid combinatorial attacks altogether. For this protocol we assume our hash functions have a key such that, for each choice of this key, they represent separate hash functions, and we distinguish between $sh(hk, m)$ and $lh(hk, m)$: short and long hash functions. The first of these is what the

humans compare, the second has enough bits to defeat any combinatorial attack. In both cases it must be infeasible to compute *hk* or *m* from the values (or indeed any number of hashes in which one or other is fixed and the other varying). In the long case it is infeasible given any reasonable time to find *hk* and *m* (or one given the other) which hash to a given value. We will discuss the length of the short hash later.

We give a special role to one of the members of **G**, who initiates the protocol and who makes all members of the group aware that he is playing this role. It is crucial that all members of **G** know, by the end, that one of their number has initiated this run of the protocol. The protocol is as follows, where A represents a typical non-initiator node, B a node of either type, and M^d again represents message M decrypted.

0.	$I \rightarrow_N \forall A$	: I
1.	$\forall B \rightarrow_N \forall B'$	: $\{B, kc(B), N_B\}_{wkk}$
2.	$I \rightarrow_N \forall A$	: $lh(hk, \text{all Messages } 1^d)$
2*b*.		See note
3.	$I \rightarrow_N \forall A$	: $\{hk\}_{pk(A)}$
4*a*.	$\forall B$ displays	: $sh(hk, \text{all Messages } 1^d)$
		$init(I, B)$, participant count
4*b*.	$\forall B \rightarrow_E \forall B'$	: Users compare
5.	$\forall B \rightarrow_N \forall B'$	: $\{lh(hk_{pub}, S)\}_{dual(pk(B))}$

The following notes explain these messages.

- Message 0 gets the protocol going and informs the participants of the identity of I. [2]
- Message 1 has the same role as the same message in the first protocol of introducing fresh data and a public key from each participant.
- Message 2 has I invent a new hash key *hk*. It then sends each node the hash of something which every node knows, and knows to be fresh. On the basis of this information no-one other than I knows the value of *hk*, but no-one who has accepted this message can be fooled into thinking that any other value of *hk* is correct.
- Message 2b is some way of ensuring that I will not send a Message 3 while any other agent in **G** is still waiting for a Message 2. The most obvious way of doing this is to use the empirical channel at this point: each laptop displays a signal which the users then convey to the user of I.

  One of the functions of Message 2 of the first protocol was to act as preliminary test of the probable coherence of a run of the protocol in advance of the humans becoming involved. There is no such check included in the present protocol. Therefore, both to save the humans unnecessary effort and to prevent incorrect diagnosis of an attack (see Section 4) we might want to add a Message 2a in which all nodes broadcast a public hash of the Message 1's for this purpose.
- In Message 3, I sends each node the hash key under its own public key. Since this message is firmly bound to Message 2 there seems to be no reason to add any further contextual information. Note that at this point each node can check the value of *hk* by testing to see if it produces the correct value for Message 2. They only proceed if this is true.
- Message 4a makes a similar display to Message 3a in the first protocol. The only difference is $init(I, A)$, which gives information about the initiator, at least whether this node is the initiator of the session.
- Message 4b has the members of **G** compare hashes and check the number of participants as before. The difference this time is that one of them must be able to say "I am the initiator". If these fail the run is abandoned.
- Message 5 acts as a confirmation amongst the laptops, like Message 4 did in the first protocol. At this point *hk* is a shared secret amongst **G**; assuming that some shared secret S is needed to take the session forward from here they can construct one in a systematic way from the run data – necessarily involving *hk* since the rest of the data is public. This message consisting of a hash of this secret (the hash key hk_{pub} is publicly known) simply acts as a confirmation. The signature using the duals of the public keys established makes this quite strong: since authentication is actually established by Message 4 it could be simplified without compromising security.)

We will call this the HCBK protocol, for *Hash Commitment Before Knowledge*. Note that though this protocol is asymmetric, both the set-up of empirical channels and the final result are symmetric. The final result is that the members of **G** (identified by the empirical channels) are authenticated to each other as the owners of the public keys they have introduced, and have the shared secret S.

We can analyse the HCBK protocol as follows:

1. After Message 4 all participants know that I, one of their number (and therefore trustworthy) has generated a value *hk* and that the value they have just agreed to (created using their various views of *hk*) all coincide with I's own. Under the perfect cryptography hypothesis this, together with the same reasoning used for the first protocol, would prove security. [In essence this is that each group member knows that it, and, thanks to

[2] In practical implementations it might be a good idea to have I introduce some public nonce that will act as an identifier for this particular run of the protocol: it could be used to reduce confusion rather than increase security by being added as a tag to all the other $\rightarrow_N$ messages.

the agreement, the other $N-1$ nodes, all belong to this hash of N sets of values. It follows that they are all present and no-one else is.]

2. Before Message 3 is sent, all members of **G** have a long hash which has apparently been computed using *hk*, but *I* has not divulged *hk* itself to anyone. Of course the intruder might have sent any of the Message 2's, but he has no way of knowing at this stage what the hash value computed in Message 4 will be. Therefore there is no constructive way in which the intruder can manipulate the Message 2's for his own purposes.

3. Since it is computationally infeasible for the intruder to find other inputs for which a long hash comes up with a given answer, the fact that a member of **G** gets past Message 3 means that he or she has the same view of *hk* and of the Message 1's as the initiator of its run.

4. At the point at which a laptop *A* accepts a Message 2, the hash value it will generate in Message 4 is completely determined; however neither it nor any other entity but *I* is in a position to compute that value yet.

5. Supposing that *A* has agreed the hash value *HV* in Message 4, there are two possibilities:

   - It is part of the same run as *I*, so all is well.
   - It is part of another run that happens to have the same hash value. But for this to be true the intruder has fixed another run (which *A* is in, similar to the attack on the first protocol) to have value *HV* in advance of knowing what *HV* is. This can only be done via a single guess, which we have assumed has negligible probability of success. It cannot be multiplied by attacking several members of **G** since as soon as one of them spots a discrepancy the run will be abandoned.

This reasoning depends crucially on having a trustworthy participant declaring that he or she is the initiator. Otherwise the intruder could act as initiator in two separate runs with members of **G**, giving it the opportunity to select *hk* values for which the short hashes will co-incide.

It is vital for the correctness of this protocol that the non-initiator nodes never start another protocol run between the acknowledgement of the receipt of Message 2 (namely Message 2b) and the final agreement about the hash values. For otherwise an intruder might persuade one of them to abandon the first run, and start a second one where he manipulates the final hash using a similar combinatorial attack to the one described earlier against the printer protocol. This could be achieved, for example, by putting timing constraints on how fast a second run can be started after one is abandoned and how long the initiator will wait for the final agreement. Or it could be achieved by the addition of run numbers to some of the messages. In some human-mediated implementations of the empirical channels we might find it reasonable to assume they cannot be blocked, which would mean that a node whose run is abandoned can communicate this fact to the others.

3. Asymmetric authentication

We have already seen that the HCBK protocol is much safer against combinatorial attacks than the earlier one. It is natural to ask whether similar ideas could lead to stronger protocols than others that have been based on the quote above. One such example is Protocol 2 from [1] which we quoted above.

We cannot simply apply HCBK to this scenario for two reasons. The first is that we only have a one-directional empirical channel. The second is that this new scenario is not one where the concept of mutual trust makes any sense. Instead we are assuming that the user *A* has some grounds for trusting the device *B*: maybe it is owned by her or someone she trusts. We are not considering the authentication of *A* to *B*. In summary, the scenario is no longer symmetric.

In fact a very similar protocol works, which is the same as HCBK except that all the empirical communications that are not to the user *A* – who always plays the role *I* – are deleted. Note that this is a group protocol in that it can authenticate any number of devices to a single user. We pay the penalty that the users other than *A* do not have anyone authenticated *to them*. The authentication provided is that *A* is sure that the session really is with the devices from which he has empirical channels (which will usually be provided by displays on the devices that he can see).

Once again it is vital that a device will not participate in a second run, apparently with *A*, while *A* is still running the first. For obvious reasons this version of the protocol will be termed *Asymmetric* HCBK, or AHCBK.

4. Optimality?

Our protocols are designed to initiate a secure and authenticated session between the owners of a set of empirical no-spoofing channels. Where these channels are implemented by humans it is natural to want to keep the amount they do to a minimum. We might therefore pose the following question:

> From a position where nodes in a group have no shared secrets or knowledge of each other except for a complete set of no-spoofing empirical channels, what is the *most* security that can be achieved in a group-formation protocol for a

given amount of communication on the empirical channels?

In order to answer this question we first have to decide what "the most security" means. The answer we propose to that question is:

> The security offered by a given protocol instance is the minimum, over all potential attacks with a more than infinitesimal chance of success, of the probability that the attack is discovered divided by the probability that it succeeds.
>
> Here, an *attack* is a strategy by an intruder intended to make some agreed security specification of individual runs fail, and it *succeeds* if – on a particular run – the specification does fail.

This, or the *log* of this value, seem to the author to be reasonable definitions since, upon discovering an attempted attack, the participants can make subsequent runs more secure by putting more resource into them. If there were an attack with no chance of being discovered, then presumably it could be repeated *ad nauseam* until it eventually succeeds. For that reason this seems to him to be a safer definition than just the reciprocal of the probability of success.

In the following analysis we will discount the possibility that an attacker of HCBK or other candidate protocol can make the perfect cryptography hypothesis fail except on the empirical comparison of hash values. More precisely, we assume that the probability of such an approach working is infinitesimal, hence the use of this word above.

Since it is readily provable that HCBK is secure if we had perfect cryptography, and the only place we are saying this can fail is on the short hash values, it follows that for an attack to succeed we must have different members of the group **G** producing the same hash value from different antecedents. The group will not proceed without an initiator I present, so we can concentrate on the case where another node A has different antecedents for the same hash value as I.

I knew that A was committed to the antecedents of some final hash value at the point when it first sent Message 3 to anyone; further we know, thanks to an assumption we made, that A cannot have become committed to another value since.

An intruder could only know whether the final hash value of A coincided with I's after I has sent a Message 3. Assuming that our hash functions etc have the properties we expect of them (namely that the short hash is uniformly distributed as a function of hk for all values of the other parameters, and that the intruder cannot infer any information about that hash from Message 2), the intruder will have at best $1/H$ chance of success, where H is the size of the hash space. (This is the chance of a single A with different antecedents matching I on the final comparison.)

If the intruder was lucky then presumably he would carry on. If he were unlucky then this attack is not going to succeed, so we should investigate how the intruder might be discovered and try to evade discovery.

If the intruder simply lets the nodes from **G** carry on then they will discover something wrong. If the intruder ensures (as he can) that they all get Message 3's consistent with their Message 2's then final agreement will fail, and if any of them gets a Message 3 that is inconsistent it can report intrusion. The most drastic action our intruder can take is to prevent any of the Message 3's getting through, but then all nodes will know that something has interfered with their run.

The only way of avoiding this would be if the intruder could somehow modify the states of the nodes so that they would in fact agree, perhaps on the correct values that the nodes from **G** actually intended. This could only be done if at least one of them were forced to abandon its present session and start a new one that would agree. But we have specified that, once they have got beyond Message 2 this cannot happen and that it would lead to an attempted attack being registered.

It therefore seems that if an attacker allows his attack to proceed to the point when he knows if it will succeed then he will be discovered whenever it does not. Therefore, to have a $\frac{1}{H}$ chance of success he must have a $\frac{H-1}{H}$ chance of discovery. So our measure of security seems to be $H-1$.

How much human/empirical work is required to do this? We will disregard the work done in comparing the number of participants, since (i) these do not have to be communicated (each user independently checks the number on his or her own screen) and (ii) these numbers serve to factor our general protocol into different N-ary specific protocols. Assuming we are using empirical messages for 2b, it is clear that this requires $N-1$, so let us consider the final agreement phase. The protocols themselves envisage all the participants sending the hash values to each other, but all that is actually required is that there is a connected graph of agreement exemplified by

- The initiator communicating the hash value to one other node.
- If this node agrees it sends the value to a further node.
- This is repeated until all nodes have seen this value: the last one in the chain then knows they all agree and so sends an acknowledgement message back to its predecessor.
- This is repeated until the initiator gets its acknowledgement, at which point all nodes know they all agree.

Note that since it is the initiator I who sets this chain going, this strategy would also act as the confirmation required that the initiator is a member of **G**.

It seems reasonable to measure the work required to send a message as $1 + \lceil log\, k \rceil$, where k is the number of potential message values and $\lceil x \rceil$ is the smallest integer $r \geq x$. The 1 is for the overheads of the message and the rest is the number of bits. Given this, our agreement strategy requires $N-1$ communications of cost $1 + \lceil log\, H \rceil$ and $N-1$ of weight 1. It follows that the total cost including 2b is $(N-1)(3+\lceil log\, H \rceil - 2$. If the empirical channels are broadcast rather than point-to-point (as in members of a group talking) agreement can be achieved in a single announcement of weight $1+\lceil log\, H \rceil$ by I followed by $N-1$ responses of weight 1 from the rest. (All of these have had to do the work of comparing the hash value independently.) So the total cost is $\lceil log\, H \rceil + 2N - 1$.

Intuitively, it seems that this must be close to minimal amongst all conceivable protocols with security measure $H-1$.

Let us examine this question briefly: we will start with the binary ($N = 2$) case. We do not claim that the following argument represents a formal proof; rather a strong indication that one exists.

In the absence of the two nodes A and B having any pre-existing knowledge of each other, or any security infrastructure, an arbitrary protocol P is certain to have a man-in-the-middle attack against it unless the empirical channels are used. For our attacker can simply set up parallel sessions with A and with B; each will know they are in a session, and the intruder is free to translate the messages each send the other from one session to the other.

Each of A and B has to receive an empirical message from the other for them to be mutually authenticated, since either that did not receive could not tell who they were connected to. It follows that the total work required is at least two plus the number of bits they communicate in their messages[3].

If A and B send a total of k bits of information to each other on empirical channels during a run of the protocol, then in order for the above attack to succeed the messages expected in the two attack sessions must be the same as are expected in the proper session that A and B imagine they are having.

The worst case for the intruder is when the protocol gives him no information about what empirical communications are coming from A and/or B in the two sessions, and when the structure of the two sessions does not allow him to transmit information between A and B in such a way that these expectations match up. (In the HCBK protocols the use of the nodes' public keys in the hash achieves this.)

Whatever values the intruder has chosen to inject into the two runs (noting that in the two combined it injects a complete set since it is playing all roles, and that the intruder is *trying* to get agreement) will have a chance of the empirical communications agreeing no worse than the reciprocal of the number of value combinations communicated. That is precisely what is achieved in HCBK. Since that protocol (with 2b) uses three messages as opposed to the two proved above, the cost of HCBK for two nodes exceeds the theoretical bound we have established by only one.

Now let us consider the case of a group of $N > 2$ agents who wish to be mutually authenticated. If broadcast communication is being used, it is clearly impossible to achieve agreement in less than one k-bit communication followed by $N-1$ null communications, since all must send an empirical communication, and k bits of information have to be compared somehow.

If point-to-point communication is used, then there must be a first node A that knows that the run is secure. From that point on clearly exactly $N-1$ basic communications are necessary and sufficient: all others have to receive something that tells them, and A can simply communicate the fact by one communication to each (either directly or via a deeper tree). The author conjectures that no node can know the run is secure until $(N-1)k$ bits have been communicated in the network, but has yet to find a proof of this.

5. Verification

Our protocols offer new challenges to automated verification. These are the fact that they are group protocols where the size of the group is checked as part of the protocol, and the use of a new and stronger attacker model. Additionally, though this had been solved previously (e.g. [2]), we need to model empirical channels specially so they cannot be spoofed. This last one is very straightforward.

The author cannot claim to have made much progress on the first of these as yet: it may well be that this, with its implicit use of arithmetic, is better suited to theorem proving than model checking. All he has been able to do is to build CSP models for FDR in which the group size was bounded above by some small natural number (e.g. 2 or 3).

The agreement implied by the short hash under perfect cryptography is, as we have observed, so strong that it would appear that the only possible source of attacks will be from the attacker's ability to create a fresh value so that some construction involving it will hash to an already-known value. (This is strong enough to allow both for birthday attacks and for the cruder style of attack we discussed for the printer protocol.)

The main difficulty this creates for the CSP/FDR model of protocols is that hashing can no longer be a simple free

[3] Note that it would be possible for them to send information to each other via patterns of content-free communication, but that the cost of such communication would exceed sending it as bits.

data-type constructor. For the usual way of incorporating it into the protocol data-type *Fact*:

$$\begin{array}{rcl} \mathit{Fact} & ::= & \textsc{Atom}.\mathit{Atom} \\ & & \mid \textsc{Sq}.\langle \mathit{Fact} \rangle \\ & & \mid \textsc{PK}.\mathit{Fact}.\mathit{Fact} \\ & & \mid \textsc{Encrypt}.\mathit{Fact}.\mathit{Fact} \\ & & \mid \textsc{Hash}.\mathit{Fact} \end{array}$$

makes all hashes of distinct objects distinct (the effective assumption in perfect cryptography). The solution that the author adopted was to retain the basic structure of this data-type with two modifications:

- Since sets are used in the protocol as a natural consequence of modelling groups of members with the same role, a SET construct was added.
- Two hashing constructs were used, SHORTHASH and LONGHASH, each taking two arguments reflecting the usage in HCBK.

but then to modify how the hashes are computed over this type.

Long hashes are worked out in the same simple way as previously, namely

$$lh(hk, f) = \textsc{Longhash}.hk.f$$

since they are still assumed to satisfy perfect encryption. The function *sh*, on the other hand, is computed via a lookup table that can be manipulated by the intruder. The lookup table is implemented as a separate process which maps hash keys and facts to short hash values in which each combination (hk, f) defaults to $\textsc{Shorthash}.hk.f$, but if the intruder creates a fresh atom *A* which is a part of the *fact f* or is *hk* (both of which the intruder knows given *A*), then the intruder is allowed to make the lookup table send (hk, f) to *shv* for any short hash value *shv* it already knows.

In runs of FDR, the lookup table was limited to have a single manipulated hash value, to reduce the state space.

As we would have expected, this model finds the same attack we illustrated earlier on our first protocol. It finds no attacks on HCBK or AHCBK.

6. Conclusions

We have seen how a new approach to computer security allows this to develop from trust that may have arisen in subtle and non-technical ways. This will often just be a human trusting hardware he or she can see and has decided to trust (in the case of the asymmetric protocol) or which is in the hands of other humans in the same group.

We believe that this new paradigm should be helpful in many circumstances, including the following.

- It provides a lightweight mechanism for a group – perhaps in a military or business context – to set up a highly secure network where the membership is precisely them, and no external user or piece of hardware needs to be trusted.
- It can perform the same role where the users are so heterogeneous that it is difficult or impractical to build an overarching security management system.
- It can re-inforce an existing security structure, for example because
  - It is feared that some hardware might have fallen into enemy hands. Our protocols can prevent connection to parties unless there is human-to-human trust.
  - Users want reassurance that they are connected to the particular member of a security structure that they want to be.

  In this context we should perhaps regard the optional use of *wkk* in the protocols as a cliché for the idea that the messages of our protocols might be sent within some wider security structure.
- It can be used by parties to any radio, telephone or video-phone conversation to achieve secrecy quite independently of any carrier, provided we discount the possibility that an attacker might successfully impersonate each human to the other(s) in the agreement phase.
- The user of any device in line of sight[4] can be assured of a secure connection with it (or a group of these in only one run).

We have given a relatively informal demonstration that our protocols seem to be near optimal in the trade-off between human/empirical effort and the chance of a successful attack. We have also shown how the stronger attacker of this new protocol can be modelled on FDR. The author would like to find a better way of modelling this that does not have to be limited to a single manipulation – this development would be very similar to the evolution of the "perfect spy" of [6] from our early intruder models that had a small finite memory. Particularly in that case, he hopes that it might find use in other circumstances in computer security where combinatorial attacks on hashes are an issue.

It is worth comparing what our protocols achieve against the secure mode of Bluetooth. The latter also uses an empirical channel in that both ends of a secure communication are primed with a PIN. The key for the transmission of secure

[4] Or in any other context where there is a non-spoofable channel from the device to the user.

information within that protocol is then computed from this PIN and some broadcast random information. It is well documented (for example [9]) that it is vulnerable to a number of attacks. These both make it imperfect within its defined context and severely limit the extent to which it can be used outside this. Firstly, unlike our protocols where we allow the empirical channel to be heard by an attacker, this would be fatal to Bluetooth. If the PIN becomes known to an attacker he can simply compute the key.[5] Even if the PIN is not revealed directly, Bluetooth is vulnerable to a *post hoc* attack in which the attacker just tries each candidate PIN in turn. This is in some ways worse than the situation with our first protocol, since there at least the combinatorial attack has to be contemporary with the protocol run, and is therefore harder. The HCBK protocols, of course, eliminate this entirely.

It would be interesting to take further the idea of quantifying the security provided by a protocol. One can imagine a number of alternative measures, and a protocol having different measures relative to different specifications (maybe secrecy and authentication). It should ultimately be possible to automate it using probabilistic model checking tools such as those described in [8, 7].

It is clear that any use of our protocols would benefit from several sorts of human factors research. Most obviously, how should we present our hash values so that human users will find it both easy and tolerable to compare as large a range as possible. Might they be composed of characters, letters, digits, words, or be sentences, or symbolic in some way? If the computers are close together, should our humans perhaps compare images, or have their hardware play a sequence of musical notes together? Obviously the optimal way of doing this might vary from context to context. How safe is the voice or video-telephone communication of hash values against attack; what is the best strategy for making an attack difficult in such circumstances?

Other questions in this area include how to explain to users their important role in the protocols, and present the data in such a way that they will not ignore the needs of security.

Acknowledgements The conceptual model of computer security proposed in this paper emerged in a series of discussions the author had with Sadie Creese, Michael Goldsmith and Irfan "Zak" Zakiuddin. In particular it was Zak who introduced, and emphasised the importance of, the problem of bootstrapping security from a minimal starting point. This joint work was reported in our earlier papers. I have had useful discussions with Gavin Lowe and Michael Goldsmith relating to this paper.

[5] This would seem to make the Bluetooth protocol susceptible to attack whenever the attacker might be physically present in some way, or overhearing the PIN if it is spoken.

References

[1] S. J. Creese, M. H. Goldsmith, R. Harrison, A. W. Roscoe, P. Whittaker, and I. Zakiuddin. Exploiting empirical engagement in authentication protocol design. In D. Hutter and M. Ullmann, editors, *Proceedings of* 2^{nd} *International Conference on Security in Pervasive Computing (SPC'05)*, volume 3450 of *LNCS*, Boppard, Germany, April 2005. Springer.

[2] S. J. Creese, M. H. Goldsmith, A. W. Roscoe, and M. Xiao. Bootstrapping multi-party ad-hoc security. In *Proceedings of IEEE SAC Security Track*, 2006. to appear.

[3] S. J. Creese, M. H. Goldsmith, A. W. Roscoe, and I. Zakiuddin. The attacker in ubiquitous computing environments: Formalising the threat model. In T. Dimitrakos and F. Martinelli, editors, *Workshop on Formal Aspects in Security and Trust*, Pisa, Italy, September 2003. IIT-CNR Technical Report.

[4] S. J. Creese, M. H. Goldsmith, A. W. Roscoe, and I. Zakiuddin. Security properties and mechanisms in human-centric computing. In P. Robinson, H. Vogt, and W. Wagealla, editors, *Privacy, Security and Trust within the Context of Pervasive Computing*, Kluwer International Series in Engineering and Computer Science. Springer, 2004. Proceedings of Workshop on Security and Privacy in Pervasive Computing, Wien, April 2004.

[5] S. J. Creese, G. M. Reed, A. W. Roscoe, and J. W. Sanders. Security and trust for ubiquitous computing. In *ITU WSIS Thematic Meeting on Cybersecurity*, Geneva, 2005.

[6] M. H. Goldsmith and A. W. Roscoe. The perfect 'spy' for model-checking cryptoprotocols. In *Proceedings of DIMACS Workshop on Design and Formal Verification of Cryptographic Protocols*, Rutgers, 1997.

[7] M. H. Goldsmith and P. Whittaker. A CSP frontend for probabilistic tools, 2005.

[8] A. Hinton, M. Z. Kwiatkowska, G. Norman, and D. Parker. PRISM: A tool for automatic verification of probabilistic systems. In *Proceedings of TACAS'06*, 2006. to appear.

[9] M. Jakobsson and S. Wetzel. Security weaknesses in bluetooth. In *Proceedings of 2001 conference on cryptology*, volume 2020 of *LNCS*. Springer Verlag, 2001.

[10] T. Kindberg and K. Zhang. Validating and securing spontaneous associations between wireless devices. In 6^{th} *Information Security Conference (ISC'03)*, number 2851 in LNCS. Springer-Verlag, October 2003.

[11] G. Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In *Proceedings of TACAS*, volume 1055 of *LNCS*, pages 147–166. Springer Verlag, 1996. Also in *Software – Concepts and Tools*, **17**:93102, 1996.

[12] B. Schneier. *Applied cryptography*. Wiley, 1996.

[13] F. Stajano and R. Anderson. The resurrecting duckling: Security issues for ad-hoc wireless networks. In B. Christianson, B. Crispo, and M. Roe, editors, *Security Protocols, 7th International Workshop Proceedings*, pages 172–194. Springer LNCS, 1999.