Efficient group authentication protocols based on human interaction

L.H. Nguyen and A.W. Roscoe

Oxford University Computing Laboratory
{Long.Nguyen, 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. We propose several protocols which achieve a high degree of security based on a combination of human-mediated communication and an ordinary Dolev-Yao communication medium. In particular they resist combinatorial attacks on the hash or digest values that have to be compared by human users, seemingly optimising the amount of security they can achieve for a given amount of human effort. We compare our protocols with recent pairwise protocols proposed by, for example, Hoepman and Vaudenay.

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 pieces 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 this goal is unachievable, since it is impossible to prevent some impostor I playing the man-in-the-middle between the participants. 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 [13] (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 second author, Creese, Goldsmith, Zakiuddin and others [9–12] 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 may be 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 interact with securely. In pervasive computing this feature is frequently one of position or proximity [18]. 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 [9–12] 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.

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

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.

We show how these combinatorial attacks can be overcome by several related methods, all based on the twin ideas:

- Have the human users confirm that all systems involve agree on a final hash or digest value that is formed from the entire details of the session. In an ideal world the likelihood of a successful attack would be bounded above by $1/H$, where H is the number of possible hash values.
- Commit nodes to this final hash or digest value before they (or any potential attacker) actually know it.

We show that the amount of computation required by the members of the group can be a trade-off depending on what they seek to achieve from the protocol, and how much trust they have in other members of the group.

Our protocols were, in the main, developed before we were aware of several other recent (pairwise) protocols reported in [15, 19, 6]. We compare ours against these others, comparing both the amount of human and computational resources required by the different styles.

We introduce a number of different protocols in this paper that are suitable for many different sizes and types of group, all the way from pairwise connection between the systems of people who completely trust each other, to a lecture theatre full or more, for example. This, and the range of potential implementation technologies, mean that in this paper we largely abstract away the details that are not immediately important to security.

This paper is organised as follows: in the next section we show how protocols for this type of scenario can be vulnerable to combinatorial attacks, and analyse how others have solved this problem. In Section 3 we introduce a class of efficient group formation protocols that rely on trust and which seem particularly appropriate for a user's equipment to set up secure communications with a group of simple devices. In Section 4 we show how to build groups securely even in the presence of corrupt participants in the protocols, using extra computation to dispense with the need for trust. Finally we analyse the relative efficiency of our protocols and those of [15, 19, 6], and look to future verification work.

2 Analysis of Existing Protocols

It has been widely realised that it is impossible to bootstrap security from nothing. Nevertheless, as we have discussed in the introduction, it is necessary to be able to bootstrap it from minimal assumptions. So what is it reasonable to assume exists prior to an attempt to acquire high-quality security? There have been (at least) two separate types of approach to this. One has been to assume that suggested that the pair of parties, who are seeking to exchange a strong secret key, already share a short or low entropy secret information such as a password. The second approach assumes the existence of a low bandwidth empirical channel that is not susceptible to spoofing. Based on the either assumption, the parties can agree on a strong shared secret from scratch.

The first method, which is based on shared password, has been studied intensively in the last decade. One typical example is Bluetooth [5] , which has been found insecure due to off-line PIN crunching attack by Jakobsson in [16]. After that people in the cryptography community came up with various formal frameworks presented in [2–4, 7, 8] by Bellare, Pointcheval, Rogaway, Canetti, and Krawczyk and others that focus on preventing off-line dictionary attack. That means the only way that an attacker can find out whether his random guess of the password is correct or not is by interacting with the legitimate player. It leads the protocol to be only vulnerable to a random guess that is successful with a probability of $1/H$ where H is the space size of the password. Sharing the password has the advantage of simplifying the problem of bootstrapping security as well as security analysis. In contrast, it has completely abstracted away the process in which the private information is exchanged between the two parties and therefore does not provide a complete solution for this problem. Even worse, it might have ignored weaknesses that might be present in the methods of initial key exchange.

Taking a different approach that makes use of the empirical channel, in [9–12] Roscoe, Creese, Goldsmith, Zakiuddin and others attempted to form a secure network for both two parties and multi-party scenarios. However, as shown below, their scheme is vulnerable to a combinatorial attack, due to the birthday paradox. There has been also other work that concentrates on the application that involves two parties, presented in [15, 19, 6, 1, 14] by Hoepman, Vaudenay, Balfanz, Gehrman, Mitchell, Nyberg and many others.

2.1 Combinatorial Attack on Security Bootstrapping Protocol for ad-hoc Group

The following protocol was originally proposed in [11]:

1. $\forall A \longrightarrow_N \forall A'$	: $A, pk(A), N_A$
2. $\forall A \longrightarrow_N \forall A'$	: $\{$all Messages $1^d, N'_A\}_{pk_{A'}}$
$3a.$ A displays	: $hash(\{$all Messages $2^d\})$, number of processes
$3b.$ $\forall A \longrightarrow_E \forall A'$	: users compare hashes and check numbers
4. $\forall A \longrightarrow_N \forall A'$	: $hash'(\{$all Messages $2^d\})$

Here, $\forall A$ means that a message is sent or received by all parties in the group **G** who attempt to achieve a secure link between their laptops or PDAs. $pk(A)$ stands for an uncertificated public key for A. The protocol uses two types of channel:

- $\longrightarrow_N$, the normal Dolev-Yao network where all messages transmitted between the laptops in this channel can be overheard, deleted or modified by the intruder.
- Whereas $\longrightarrow_E$ indicates the low bandwidth empirical channel, typically implemented by human users, which is similar to the authentic channel used in [15], and not susceptible to spoofing.

This protocol introduced, implicitly, the first of two principles which underlie the new protocols we will be describing in this paper:

P1 Make all the parties who are intended to be part of a protocol run empirically agree a cryptographic hash or digest of a complete description of the run.

Once that agreement has occurred then, unless there is a hash anomaly – different nodes in the group computing the same hash value from different antecedents – then all the parties agree on all the data transmitted during the protocol. Usually, as in the protocol above, this will bind data to the various members of the group which, since each of them (including the owner of each piece of data) is agreeing with this data, must be correct.

The crucial part of this protocol is where all the parties display and compare hashes in Messages 3a and 3b. Since these hash values are compared manually by human effort, and not by computer, the length of the hash can only be up to a few digits or characters. Thus it is not hard for an intruder, in a limited amount of time, to search for a collision that might cause the parties to agree on different secret information; and thereby to force an anomaly in the sense described above.

The attack can be described as follows: the intruder will run two parallel sessions with two disjoint subsets S_1 and S_2 of the group $\mathbf{G}$ of N parties. During the two parallel runs, the intruder impersonates all parties of subset S_1, by modifying messages sent from subset S_1 with different public keys where she knows the corresponding secret keys, to talk to parties in the other subset, S_2, and vice versa. Therefore, any one in group $\mathbf{G}$ is still thinking that s/he is running the protocol with the other $(N-1)$ parties. The intruder only allows messages to be passed unaltered within each subset, but when a message is intended to go across the boundary between the two subsets its content will be changed appropriately.

If $S_1 = \{P_1, P_2\}$, $S_2 = \{P_3, P_4, P_5\}$, and $\{N'_1, N'_2, N'_3, N'_4, N'_5\}$ are the original nonces randomly created by all the parties in Messages 2 then $\{N''_1, N''_2, N''_3, N''_4, N''_5\}$ are the nonces generated by the adversary, in whatever time it has, such that:

$$hash(N'_1, N'_2, N''_3, N''_4, N''_5) = hash(N''_1, N''_2, N'_3, N'_4, N'_5)$$

This can be done by picking the nonces N''_1, N''_3, N''_4 randomly and then, based on the birthday paradox, the intruder can search for values of N''_2 and N''_5 such that the hashes come out to be equal to each other.

If we are to attain our goal of optimising the amount of security obtained from a given amount of empirical communication (essentially hash width) we need to eliminate not only birthday attacks but also ones in which the intruder is able to achieve something if a basic combinatorial search for a value v such that $hash(a, v) = b$ succeeds, where a and b are fixed. In other words we need to attain security in the case where the hash function that the humans compare is not *collision resistant*.

2.2 Other peer-to-peer Key Exchange Protocols

In [1, 14], Balfanz, Gehrmann, Mitchell and Nyberd proposed two protocols for key exchange. Both of the schemes, however, require a large number of bits to be communicated over the authenticated channel and to be compared manually by using human effort. Taking a further step, in [15, 19, 6], Hoepman, Vaudenay, and Čagalj proposed protocols that can get around the problem of bandwidth of authenticated channel. They all share many similarities: for example, all of them concentrate on the case where there are two parties a *peer-to-peer* network. In addition, they contain the idea of pre-committing two parties to some random secrets by sending the corresponding cryptographic hash, in [15], or the output of the commitment scheme, in [19, 6], to each other at the start of the protocol.

In [15], Hoepman required each party to compute and manually compare short authenticated strings that are transmitted over the empirical channel:

---

1. $A \longrightarrow_N B : longhash(g^x)$
   $B \longrightarrow_N A : longhash(g^y)$
   Where x and y are randomly picked by A and B
2. $A \longrightarrow_E B : shorthash(g^x)$
   $B \longrightarrow_E A : shorthash(g^y)$
3. $A \longrightarrow_N B : g^x$
   $B \longrightarrow_N A : g^y$
   A and B then share the key $k = g^{xy}$

This is not optimal in the amount of work required by the humans implementing the empirical channel, since the same amount of security (i.e. improbability of a successful attack) can be obtained in several ways by them comparing a single string of the same length. We will demonstrate two of these later: a further has been devised by Vaudenay [19], and adapted by [6]. Vaudenay and Čagalj require a commitment scheme that is at least as secure as a standard cryptographic hash function.[2] By this we mean that it must be computationally infeasible to find, with greater than infinitesimal probability, collisions, or inverses to the "commit" values. It should therefore be assumed that these values are as hard to compute as, and have as many bits as, a strong cryptographic hash.

---

1. $A \longrightarrow_N B : m, c$
   Where $c \parallel d = commit(m, R_A)$, and R_A is a random nonce of A.
2. $B \longrightarrow_N A : R_B$
3. $A \longrightarrow_N B : d$
   B computes $R_A = open(m, c, d)$
4. $A \longrightarrow_E B : R_A \oplus R_B$

Note that the final value compared in this protocol is the XOR of the short entropies devised by A and B. In particular it does not depend functionally on the message m being sent; in other words it does not follow our principle **P1**. The guarantee of authenticity of m that this protocol delivers is as a consequence of:

[2] We note that there is an error in the specification of the commitment scheme in Vaudenay, since the security specification there fails to bind it to the message m, as was obviously intended. The definition there is satisfied by $commit(m, r) = Hash(N, r) = c$, and $open(m, c) = (N, r)$, for N a fresh nonce.

– The fact that this exchange guarantees the value for R_A that B has discovered from the commit scheme is the one that A intended.
– The way the commit scheme has strongly bound the message m to R_A at a point where R_A is itself unknown to any attacker.

We will see later that this indirect binding of m to the final agreement makes this protocol relatively expensive relative to others we will introduce.

In this paper, we shall extend the idea of [9–12] in constructing an arbitrary-sized secure network, but without the trustworthiness of the entire network. We also believe that the degree of security obtained is essentially optimal for the amount of empirical (human) communication required.

In the mean time, we also try to reduce disadvantages of [15, 19, 6] with respect to efficiency when it comes to implementing the commitment scheme and computing the long and short hashes, or digests.

3 Some Protocols for Bootstrapping Groups

We will introduce our protocols in order they were discovered. The ones presented in this section are based on one discovered by Roscoe in June 2005.[3]

Both in this section and the next we will assume the existence of a set of processes that are connected by a standard Dolev-Yao network and also by non-spoofable empirical channels. By and large we will consider the *symmetric* case where there is an empirical channel between each (directed) pair of nodes, and where we are seeking to authenticate each node to every other. However we will also, in this section, consider the *asymmetric* case in which there is a particular node I to which there are empirical channels from the others, and who wishes to have authenticated connections to the others.[4] In each case we will assume that each node A has some information $INFO_A$ that it wants to have authenticated to other members of the group: this might include (i) name and addressing information, (ii) its uncertificated public key or Diffie-Hellman token g^{x_A}, (iii) contextual information to help identify it, such as its location or human owner, and (iv) certificates relating to its functionality. Nothing in this information should be secret.

[3] In the notation of this section, that was HCBK3.

[4] A model for the asymmetric case is a human user trying to connect his or her laptop to a number of wireless devices, each of which have some display that the human can see.

$INFO_A$ might be attached to A permanently or for the long term; alternately some of it might be relevant to this particular run only. The goals or our protocols will always consist at least in part of authenticating pairs $(A, INFO_A)$ as members of the network.

It is usual in describing authentication protocols to demand the following: *If a party A believes it has completed a run of the protocol with a second party B, then in fact B has been running the protocol with A, and A and B agree on the underlying variables of the run. This is on the assumption that A, B and any trusted third party that plays a role in the protocol are trustworthy, even though all other parties and the network may be corrupt.*

This leaves a slightly grey area for protocols building *groups* of more than 2: should we or should we not be content if the presence of a corrupt party in a group means that communications that result between trustworthy members of the group are themselves compromised? In some of the circumstances where we may wish to use *ad hoc* group formation protocols it would be much better if the protocols were tolerant of corrupt members. We will therefore be careful about our assumptions on this front.

It is obvious that any protocol which creates a shared secret is at least partially compromised by the presence of a corrupt participant. However protocols which merely authenticate public-key-like information to nodes are not globally compromised in the same way: they could be said to be establishing a *local PKI*.

In this section we will assume that there is one participant I in the protocol whom all agree is trustworthy. This could be because all participants are known to be trustworthy, because I has some special status amongst them, or because I is the only one requiring authentication. I will be called the "initiator", and the other nodes will be termed "slaves". We will first give a protocol that is designed for the case where there are empirical channels both from all nodes to I and from I to all nodes. (It will be obvious that in some aspects the protocol might work more naturally if there were empirical channels between all nodes.)

In the following description S represents a typical slave node, A a typical node (either A or S), *longhash* is a strongly collision-resistant and inversion-resistant hash function and *digest* is a digest function producing as many bits as we expect the humans to compare. $init(I, A)$ is $true$ if $I = A$ and $false$ otherwise.

0.	$I \to_N \forall S$	$: I$
1.	$\forall A \to_N \forall A'$	$: (A, INFO_A)$
$2a$.	$I \to_N \forall S$	$: longhash(N_I)$
$2b$.	$\forall S \to_E I$	$:$ *committed*
3.	$I \to_N \forall S$	$: N_I$
$4a$.	$\forall A$ displays	$: digest(N_I,$ all Messages 1$), init(I, A)$
$4b$.	$\forall A \to_E \forall A'$	: Users compare and check presence of I

The meanings of these messages are as follows:

- **Message 1** publishes the information that all the nodes want to have attached to them, via the insecure channel. Therefore they do not know upon receiving it that it is accurate.
- **Message 2a** has I devise a nonce N_I with sufficient entropy that $longhash(N_I)$, which it publishes here, has no more than an infinitesimal likelihood of any combinatorial attack on it succeeding.
- **Message 2b** has all the slaves communicate to I that they have received Message 2a and are therefore *committed* to their final digest value (though none of them know it yet).
- **Message 3** has I publish the nonce N_I after it has received commitments from all members of the group over the empirical channels. All slaves now have the duty to check if the values of Messages 2a and 3 are consistent.
- **Message 4a** has all the nodes compute what should be the same digest value.
- **Message 4b** has them compare these values: this could be done either through the single point of contact at I or more generally. Once a node knows that all have agreed this value it has completed the protocol and can enter group mode. It also guarantees that one of the nodes doing the agreeing has been playing the initiator role.

The following analysis proves this protocol achieves its aim subject to one further assumption that we will state below.

1. If, in Message 4, the agreement between digest values implies the agreement of the antecedents of the digest in all the nodes, then the protocol has clearly achieved the aim of authenticating all the $INFO_A$s to the corresponding names A, for each such A has agreed to this digest value.
2. However this message does not preclude the possibility that the group formed might contain more parties than the humans intend: an intruder can join the group on the $\to_N$ communications and remain

silent on the $\rightarrow_E$ communications. This would not do any harm provided that the participants do not assume that all the members of the protocol run are trustworthy: rather, trust has to be deduced in some other way. If there were an untrustworthy party (whether or not part of the overt group $\mathbf{G}$), it could claim $INFO$ or a name that really belonged to one of the other parties. That party could then make the run abort.

If it is assumed that all parties are trustworthy, then a worthwhile addition to the protocol is to have each node display the number (and perhaps, in a small group) the identities of the participants as an addition to Message 4a. Each user would then check if this information was as expected before proceeding to check the digest.

3. We can infer that our protocol is only attackable if it is feasible for an intruder to make different members of the group digest different combinations of values to the same digest value in Message 4. This is not impossible, since the intruder can partition $\mathbf{G}$ into two parts, and feed both of them different sets of values (building them up to the right size, and with the right names, if the checks in the past paragraph are implemented). It would then act as a silent "initiator" in one of these subgroups. Picking a random value for the nonce this node introduces will give it a $1/H$ chance of the two digest values agreeing, where H is the size of the range of the digest function.[5]

   Our hope is that it is impossible for an attacker to have a better chance than this. To demonstrate this we analyse the positions the various nodes are in when they first become committed to their final digest value.

4. I is committed when, following its acceptance of Messages 1, it creates the nonce N_I.

   A slave S is effectively committed once it has accepted Message 2a, even though at that stage it cannot know what the digest value is. For it has all information other than N_I, and it has $longhash(N_I)$, meaning that there is no better than an infinitesimal chance that it will accept a different N_I' in Message 3.

5. So let us examine the state the network is in just before I publishes N_I in Message 3. The trustworthy node I is the only one that actually knows enough to compute the final digest – in particular no intruder can know the value h_I that I will compute in Message 4a. Further-

[5] We will analyse the probabilities of digest function collisions in some detail in Section 5.1.

more, I knows – and therefore we know – that each slave S has been committed to some final digest value h_S.

Some or all of the h_S may be different from h_I, and even equal ones may be based on different antecedents. But there is no constructive way in which our intruder could have manipulated any set of different antecedents that some S has heard so that $h_S = h_I$, for the simple reason that the intruder does not know h_I and cannot guess it with more than $1/H$ likelihood of success.

6. It follows that if, in Message 4, the various nodes go on to compare precisely these values then the intruder has no better than the $1/H$ chance that we have aimed for.
7. There is still one potential avenue of attack open: can the intruder change the mind of one or more participants about the final digest value so that it equals the others. The only way it could do this would be to make them abandon this run and bring them to the point in a subsequent run where they are ready to agree the final digest.

   This would be impossible with the initiator. I is the final determiner of its own hash value by constructing N_I. So re-starting it would not give greater than $1/H$ chance of achieving any particular value. Also, and conclusively, the initiator expects to get empirical signals in Message 2b from the slaves, and these would not be available from the slaves.

   On the other hand, if a slave S could be re-started after h_I was known, then the intruder could perform a combinatorial search for a value N_I' which would yield the digest value h_I (with the combination of $INFO_A'$ of which he wants to persuade S), then a potential attack is open provided the second empirical Message 2b from S to I can be blocked. This would lead to an attack. We therefore make the following specification for the implementation of the protocol:

   > *The implementation must be designed so that agreement is impossible between final digest values other than those whose commitment has been signalled by the Messages 2b that I received.*

   The most obvious way of achieving this is via timing limits: an upper bound on the time between I sending Message 3 and agreeing Message 4, and a lower bound between I receiving a Message 2b from S and S sending another Message 2b. One could also use run numbers that are included in empirical communications, but of course that would add to the empirical effort.

   On the assumption that the above is achieved, we conclude that the nodes will never seek to agree final digest values to which they were committed later than the issue of Message 3 by I. Therefore our protocol achieves its goal of limiting the chance of a successful attack to at most $1/H$.

We will call the above protocol HCBK1, standing for *Hash Commitment Before Knowledge*, the principle on which it works. Recall its goal: to agree a set of information of the form $\{(A, INFO_A) \mid A \in \mathbf{G}\}$ amongst the members of $\mathbf{G}$, and hence authenticate each such $INFO_A$ belonging to a trustworthy A to the node that is declaring it.

We will call the protocol in which each node checks that the number of participants corresponds to the size of $\mathbf{G}$ HCBK2, and note that *provided all members of* $\mathbf{G}$ *are trustworthy*, it guarantees that no further node is present in the run of the protocol.[6] (It is in fact sufficient for one node, say I, to do this check)

If each $INFO_A$ contains a way of sending A data privately, say a public key (which need not be certificated or long term) or a Diffie-Hellman token, then we could replace the broadcast Message 3 by some means of propagating N_I securely. This could be a separate message from I to each S, or some tree of propagation amongst the S rooted at I. Upon successful completion of the protocol the group would then have a shared secret, namely N_I. Since it is vital that a shared secret is not shared with untrustworthy nodes, variants of this form are only useful on the assumptions that (a) all members of $\mathbf{G}$ are trustworthy and (b) that the number of participants is trustworthy as in HCBK2. Clearly this represents a class of potential variant protocols, but we can name them all under the heading of HCBK3.

The fact that these variants all create a shared secret is a consequence of the correctness of HCBK2: the different means of propagation of N_I is irrelevant since in HCBK2 this was essentially in the hands of the intruder anyway. That means that, following successful completion, all the aims of HCBK2 have been achieved. This means that we can look back at how N_I was propagated, and if it was in fact sent only in ways that (as we no know) members of $\mathbf{G}$ could understand, then we know that no-one outside the group knows it. In other words it is a shared secret.

Recall that these protocols depend crucially on the initiator I being trustworthy: a corrupt initiator could use a birthday attack essentially like the one we described earlier.

One situation where this is definitely not an issue is when the slave devices themselves have no need of security, as when the user of the initiator

[6] If there were an untrustworthy party present, it could pretend empirically to be running the protocol while actually some other corrupt party performed the messages over $\rightarrow_N$.

is seeking to connect his or her laptop to a number of wireless peripheral devices. That person must be sure that the connection is precisely to those devices that are trusted because of their context, labelling etc. In that case there is no need for empirical channels *from* the initiator *to* the slave devices. All we require is that these devices can signal the initiator (probably via some display that the initiator's user can see) to convey Message 2b and the digest value from Message 4.

This would work for all three of the variants described above: HCBK1, 2 and 3. We will call the resulting, simplified protocols AHCBK1,2 and 3, on the grounds that they are definitely *asymmetric*. (The original protocols are neither properly symmetric, thanks to the role of the initiator, nor asymmetric, since their overall goals are symmetric.)

4 Symmetrised Group Protocol

The protocol we present in this section was devised by the authors in February 2006.

The protocols in the previous section all rely crucially upon I being trustworthy: what are we to do if there is no node that is uniformly trusted or it is hard to select one, but we still want a *local PKI* which authenticates the $INFO_A$s of all trustworthy nodes? What we would like to achieve instead, is that a successful run of the protocol correctly authenticates all the trustworthy parties to each other irrespective of what the others may have done.

In order to do this we identify the following second principle, derived from the design of HCBK:

P2 Suppose a party A knows it is committed to a specific final hash or digest value h, and furthermore has invented a piece of fresh information f which is known to A but nobody else, and which is one of the antecedents of h. Then none of the information that A has been sent could have been designed to force its computation of h to be a specific value.

In HCBK A is the initiator and f is the nonce N_I. That protocol relies on much more subtle reasoning in respect of the slave nodes, as shown by our reasoning in the previous section and the principle of Messages 2 and 4 being aligned that we had to adopt. If the slave nodes had been able to follow **P2**, there would have been no need for this.

Our second sort of protocol is designed so that all nodes can rely on **P2**. Therefore each node will now need some value, either taken from its $INFO_A$ or made up specially for this purpose, which is fresh and unpredictable. Let us call this value hk_A (noting that it may well be a nonce) and the remainder of $INFO_A$ will be called $INFO'_A$. The protocol is now:

1. $\forall A \longrightarrow_N \forall A' : A, INFO'_A, longhash(hk_A)$
2. $\forall A \longrightarrow_N \forall A' : hk_A$
3. $\forall A \longrightarrow_E \forall A'$: users compare $digest(hk^*, \{INFO'_A | A \in \mathbf{G}\})$
   where hk^* is the XOR of all the hk_A's for $A \in \mathbf{G}$

The following notes explain these messages.

- **Message 1**: introduces the information, $INFO'$, each party wants to authenticate and a long hash of its hk. After this message each node should have all the information it requires about the other nodes except for the values hk_A, and furthermore should be committed to each of these values in the sense that when told the hk_A's it will be able to check each one.
  At the point when the sending and receiving of this message is complete, it follows that every node A is committed to some final digest value h_A, knows one of the antecedents of this final digest (hk_A) that no-one else does, but does not yet know h_A. We see that **P2** applies.
- **Message boundary**: There has to be some moment at which a node decides it is finished inputting new Message 1's. This might be determined by some timeout, or some message sent from one of the nodes (empirically or over the general network). It is clearly in nodes' interest that they all make correct decisions on this, for otherwise they will not agree. One can imagine them attempting to synchronise by agreeing on a hash of the Message 1's they know about over the Dolev-Yao channel: that might well serve a useful purpose since it would guard against involving humans in empirical communication when there is no point.
  Whatever mechanism they choose does not matter provided it does not involve them revealing hk values to each other. For it is absolutely vital that none of them accepts any further Message 1 after any of these values are revealed.
- **Message 2**: Each node broadcasts its unguessable hash key to all other nodes once it is committed to its final digest value. Having received all these hash keys, each node can check the correctness of all the long hashes received from Messages 1. If there is any thing unmatched regarding the long hash values, the node will abort and presumably tell the rest of the group that this has happened.

Essentially these broadcasts expand the hk parts of the Messages 1 into something the nodes can understand.
- **Message 3**: has the members of **G** display and compare the value of digests through the empirical channel. Notice that, like both the previous protocols we have considered, this digest follows **P1** and includes the whole data of the protocol.

4.1 Protocol Analysis

We shall call this the SHCBK protocol, for *Symmetrised Hash Commitment Before Knowledge*. The final result is that the members of **G** are authenticated to each other as the owner of the information they have introduced. We can analyse the protocol as follows:

1. Before its Message 2 is sent, every node has N long hashes that are computed from the N different unguessable hash keys, but trustworthy node A has not revealed its own hash key to the group. Due to the Dolev-Yao model of the network, the intruder can modify Messages 1 to transmit its own long hashes to other parties, but he has never got any idea of what value of the final digest of any node still receiving Message 1's is going to be at this stage. Therefore, the intruder cannot constructively manipulate the long hashes in Messages 1 for his/her own purposes. This is essentially the effect of **P2**.
2. Once a node A has received all Messages 1, the value of the final digest has been completely determined. However, no other party can know what A's view of the final digest will be.
3. Assuming that trustworthy parties A and B have agreed on the value of digest in Message 3, then there are two possibilities:
   - If A agrees all the antecedents of the digest with B then all is well, since this data is certain to include all correct data about A and B.[7]
   - Otherwise, the intruder has caused A and B to have different antecedents that lead to the same digest. However, since each of A and B was committed to its digest before anyone else could have known this value (or any relationship between the two of them) the best the intruder can do is to commit them blind, having no better than a $1/H$ chance of success. Here, H is the number of digest values.

[7] If an intruder has introduced extra *false* information about either A or B, then that node will be in a position to spot this at this stage or earlier and therefore refuse to proceed.

We conclude that, at best, the intruder has a $1/H$ of any attack succeeding if there are at least two trustworthy nodes present. Furthermore, if the attack does not succeed then all trustworthy nodes T share their views of all $INFO_A$'s, which are, in turn, correct for all $A \in T$.
4. The above analysis still applies even if there are corrupt participants in the protocol. The fact that they send values that contribute to the final digests of every node means corrupt participants can cause trustworthy ones to disagree, but since they have no way of knowing the digests at the point of sending these values means that there is no strategy for them to force agreement other than running the protocol properly.

Calling the basic protocol SHCBK1, it can be extended by a count of nodes to create SHCBK2 for the case where all nodes are assumed to be trustworthy.

5 Implementation

In this section, we are going to discuss how to design a good digest function as well as comparing our protocols against Vaudenay's scheme.

5.1 Computing the Digests

There are some important points we wish to make about the computation of the digest values. Let us first consider the non-symmetrised protocols. Then we have to compute the digest of one nonce and N collections of $INFO_A$.

The first thing to observe is that the nodes are actually asked to digest a *set* of $(A, INFO_A)$. This is on the grounds that it would not necessarily be possible for a node to distinguish between identical $(A, INFO_A)$ sent by different nodes, and re-sends by a single node A, and since the nodes may obviously pick up the Messages 2 in different orders. It would be normal to require stronger separateness conditions on the $(A, INFO_A)$ than just that they were distinct: for example one would normally require that all the names A and public keys (if in $INFO_A$) were distinct.

So we might expect all the nodes, or at least I, to perform these integrity checks, and for all for them to sort the pairs into the same order before digesting them. In the following we will assume that this sorting approach is taken, and refer to the list of $(A, INFO_A)$s as $INFOS$.

Our second comment relates to the randomising effects of the digests. It would be a great mistake to compute $digest(N_I, INFOS)$ as some function of N_I and $digest(INFOS)$, which it might be tempting to do. This is because an intruder could then – during the exchanges of Message 1 – manipulate the sets $INFOS_A$ heard by the different nodes A provided they will all compute the same $digest(INFOS_A)$. The fact that the intruder cannot predict at this stage what the final digests will be (not knowing N_I) would be irrelevant, since it would know they will all calculate the same value. As a result, what we need to compute is the keyed digest of $INFOS$ with respect to key, N_I, denoted as $digest_{N_I}(INFOS)$.

What we actually require is that for essentially all pairs of distinct values $INFOS_1$ and $INFOS_2$, the probability of

$$digest_{N_I}(INFOS_1) = digest_{N_I}(INFOS_2)$$

as N_I varies, is never significantly different from $1/H$ ($H = 2^b$ being the number of distinct digest values). More precisely, it must be intractable to compute distinct $INFOS_1$ and $INFOS_2$ that respectively contain specific $(A, INFO_A)$ and $(B, INFO_B)$ such that the above relationship does not hold. This is the only property that our parameterised family of digest functions must have.

It is this idea, which distinguishes the idea of being a digest from that of being a cryptographic hash function. The latter is usually expected to have the properties of being collision free and non-invertible. Neither of these concepts make any sense at the typical length of digest we will be using. In any case, these properties neither imply nor are implied by the specification above.

In order to achieve the above specification, we need to have the probability of $digest_{N_I}(M)[i] = digest_{N_I}(M')[i]$ be essentially equal to $\frac{1}{2}$ for $i = 1, \ldots, b$ when $M \neq M'$, and that the probabilities for different i are essentially independent. This means that a change in any non-zero number of bits of M must have a distinct random effect on every bit of the output. Let consider the following idealised framework: for $i = 1, \ldots, b$ and $j = 1, \ldots, B$ suppose that $R_{i,j}$ are independent uniform boolean-valued random variables, based on N_I[8]. If we define $hb_i = \bigoplus_{j=1}^{B}(R_{i,j} \wedge M_j)$, where M_j is the j^{th} bit of the input M, then $digest_{N_I}(M) = [hb_1, \ldots, hb_b]$. Formulated like this the function indeed seems to be closer to a *digest* than a *hash*.

[8] For them to be truly independent N_I would have to have far more bits than it actually does. In practice we should aim to have them only subtly dependent, and in ways that are impossible to predict without knowledge of N_I.

We can see that the bit d_i of $digest_{N_I}(M) \oplus digest_{N_I}(M')$ is equal to $\bigoplus_{M_j \neq M'_j} R_{i,j}$, which has the same distribution as $R_{i,j}$ whenever $M \neq M'$. Thus our idealised framework meets our specification completely. As a result, in order to get a good digest, we need to get close to this model. One possible solution is described below, based on multiplication.

Let us divide $INFOS$ and N_I into b-bit blocks $[m_1, \ldots, m_{t=\frac{B}{b}}]$ and $[n_1, \ldots, n_{k=\frac{K}{b}}]$. We then generate pseudo random b-bits blocks r_i based on hk[9]. If we define $S = \bigoplus_{i=1}^{b}(m_i \times r_i)$[10], we finally set $digest_{N_I}(INFOS) = S_1 \oplus S_2$, where S_1 and S_2 are two halves of S.

We could move closer to our idealised model by increasing the amount of calculation, for example by replacing a single multiplication of m_i by several of functions of m_i. Further work is required to decide if this is worthwhile.

For the symmetrised protocols, each party A creates its own random unguessable data hk_A, and therefore the value that corresponds to the random nonce in the non-symmetrised protocols might be the XOR of all hash keys. And the rest of the algorithm is the same with the previous case.

For the purposes of computation above one would almost certainly wish to round b up to the length of a whole number (usually 1 or 2) of half words. The final value for the humans to compare would then be truncated.

5.2 Efficiency

It seems reasonable to measure the efficiency of protocols in this class in two ways: the amount of empirical, or human, effort required to complete them; and the amount of processing required at the nodes.

The major item of work for the humans is probably the sending and receipt of the final digest value, and the effort required to check equality. In the case where a user can broadcast empirically to all other nodes (as with a set of people in a room), the most efficient way of performing this check is for one person to announce his/her value h_0 and the rest to check that their values all equal h_0. Depending on circumstances they might then each have to announce definite equality, or only announce inequality.

[9] Any high quality pseudo-random generator could be used here, One possibility is feedback shift register seeded with hk, or several seeded with parts of hk, this can be implemented extremely fast in hardware.

[10] The integer multiplications $m_i \times r_i$ lead S to have $2b$ bits.

It seems clear, as argued in [19] and [17], that it is impossible to bound the intruder's chance of success to 2^{-k} by comparing (explicitly or implicitly) less than k bits of information. Given pre-knowledge of the size of the group, the size check in protocols labelled 2 and 3 is essentially free; it seems impossible to account for the difficulty of performing it in other circumstances (though in the protocols with an initiator it simply means that the number of Message 2b's received by I corresponds with the number of nodes' $INFO$s that are digested).

It therefore seems that all our protocols are essentially optimal in the amount of security they provide for a given amount of human effort, *except* that in the non symmetrised cases there is the work involved in the sending and receipt of Message 2b, which is a constant and certainly less than the effort required for Message 4. We will see shortly that this represents one side of an interesting trade-off.

Let W and B be the number of words and respectively bits required to hold a long hash value: Vaudenay suggests perhaps 160 bits, so we assume $W = 5$. He also suggests that 15 or 16 are reasonable choices for b, the width of the digest, and we will adopt that too. We also assume that nonces and other strong cryptographic values have the same length B. Aside from the protocols labelled 3, the only processing effort required in implementing our protocols is the computation of long hashes and digests. In order to assess the complexity of our protocols we have to have a model of the complexity of computing hashes and digests. It is clear that the cost of computing the k-bit $hash_k(M)$ increases linearly with the length of M. It also seems clear that it will increase significantly with k, and a simple model in which each word of a running temporary value of length k is combined with each input word suggests our overall model might be $k \times length(M)$, as indeed does the idealised model presented in the previous section. Therefore we will adopt that assumption in the following analysis. Since well-known hash algorithms tend to be fixed width, and vary significantly in their individual costs, it is hard to be too definite about this rule. Our analysis of SHA-256 shows it to have a cost perhaps 20 times that of the digest algorithm we described above, based on the random number generator quoted in the footnote earlier.

It follows that the total processing cost of the non-symmetrised protocols labelled less than 3, with a group of size N and where the total size of all the $INFO$s is M, is, at every node $W \times W + M = 25 + M$.

In the symmetrised case there is more work to do since now each node has to check $N - 1$ long hashes and create one. Therefore the above quantity increases to $N \times W \times W + M = 25N + M$ This, of course, is the other side of the trade-off mentioned above.

Vaudenay's protocol, in its basic form, relates only to the transmission of a message from one party to another. In order to compare it with ours we need either to restrict our protocols to this function or to expand Vaudenay's so that it achieves the broadcast of a message from each member of a group to each other. We can do both of these things.

We will assume that the commit scheme used in [19] to commit a message of length M and r of length ≤ 32 takes $max(M, W)$ (either to generate or check) since it seems to require randomisation that introduces additional nondeterminism to that introduced by r equivalent to adding a hidden variable of length $B - b$. We will assume, for ease of calculation, that $M \geq W$.

With $W = 5$ it follows that for transmission of a single message of size M this protocol requires, at each of the two nodes, processing of order $W \times M = 5M$. Our symmetrised protocol does this in $25 + M$.

We observe that Vaudenay's protocol can be extended to a group protocol that achieves the same goal as our schemes: each node has to commit once and open (or decommit) $N - 1$ times, and no digest is required. (The users will finally compare the XOR of one short random string per node.) If M is the total size of all the $INFOs$ in our protocols, then the equivalent message that each party in Vaudenay's group version commits to will be of length $\frac{M}{N}$. In order for the commit scheme to have an equal level of security as our long hash, the lengths of both the random data of the input of the commit scheme and its output need to be W. As a result, the processing cost of each party in Vaudenay's group version is approximately $N \times W \times (\frac{M}{N}) = M \times W = 5M$. This will normally be significantly more expensive than our protocols.

It seems clear that our protocols are the more efficient in terms of computational power because we followed **P1**: we have only had to bind the messages cryptographically to the level required for human interaction. Vaudenay chose to bind the messages to random values earlier, which would have been subject to combinatorial attack had he not done so with more complex cryptography. The probability of a successful attack on either his protocol or ours is essentially 2^{-b}.

6 Conclusion and Future Work

In this paper, we have analysed the strengths and weaknesses of a number of protocols that form a secure network using the empirical channel. We further extend these protocols to propose two group protocols that resist combinatorial attack and work in different levels of trustworthiness. In addition, both of them are less expensive in computing power compared to [19].

We have exposed some clear principles which underlie our protocols and demonstrated their correctness. We hope that they will find use in a wide variety of applications.

We have introduced the concept of a *local PKI*, that is in effect the result of the run of one of our protocols, since they bind information such as public keys, identities and context together in an authenticated way. It is natural to ask how one can extend this analogy to allow for adding nodes, forming the union of two such groups etc. This of course raises interesting questions of how trust based on confidence in particular (initiator) nodes or perhaps subgroups of **G** can extend in transitive ways. This will be a topic for future research.

It is natural to ask how protocols of this form fit into the standard models and analysis tools for cryptographic protocols. The answer is that our protocols are rather outside the standard models for two orthogonal reasons. The first is that they are *group* protocols with an arbitrary number of participants: most methods are only fully developed for protocols with a small fixed number. The second is that they are intended to counter a much stronger attacker model than exists in the standard models: one who can perform combinatorial searches. We are developing a modified version of the standard CSP model for protocols that incorporates such a strong attacker and expect to report on that in a subsequent paper.

References

1. D. Balfanz, D. Smetters, P. Stewart, and H. Wong. Talking to strangers: Authentication in Ad Hoc Wireless Networks. *Proc. 9th Annu. Network and Distributed System Security Symp 2002*, San Diego, California, USA, 2002.
2. M. Bellare and P. Rogaway. Entity Authentication and Key Distribution. *Advances in Cryptology - Crypto 1993*, LNCS vol. 773, Springer-Verlag, pp. 232-249, 1993.
3. M. Bellare, R. Canetti and H. Krawczyk. A Modular Approach to the Design and Analysis of Authentication and Key Exchange Protocols. *30th STOC, 1998*
4. M. Bellare, D. Pointcheval and P. Rogaway. Authenticated Key Exchange Secure against Dictionary Attacks. *Advances in Cryptology - Eurocrypt 2000*, LNCS vol. 1807, Springer-Verlag, pp. 139-155, 2000.
5. www.bluetooth.com/developer/specification/specification.asp
6. M. Čagalj, S. Čapkun, and J. Hubaux. Key agreement in peer-to-peer wireless networks. *Proceeding of the IEEE, Special Issue on Security and Cryptography*, vol. 94, no. 2, February 2006.

7. R. Canetti and H. Krawczyk. Analysis for Key-Exchange Protocols and Their Use for Building Secure Channels. *Advances in Cryptology - Eurocrypt 2001*, LNCS vol. 2045, Springer-Verlag, pp. 453-474, 2001.
8. R. Canetti, S. Halevi, J. Katz, Y. Lindell, and P. MacKenzie. Universally Composable Password-Based Key Exchange. *Advances in Cryptology - Eurocrypt 2005*, LNCS vol. 3494, Springer-Verlag, pp. 404-421, 2005.
9. 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, *Proceeding of* 2^{nd} *International Conference on Security in Pervasive Computing* $(SPC'05)$ volume 3450 on $LNSC$, Boppard, Germany, April 2005. Springer.
10. S. J. Creese, M. H. Goldsmith, A. W. Roscoe, and M. Xiao. Bootstrapping multi-party ad-hoc security. In *Proceeding of IEEE Security Track*, 2006, to appear.
11. 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.
12. 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.
13. Gavin Lowe. Breaking and Fixing the Needham-Schroeder Public-Key Protocol using FDR. *Tools and Algorithms for the Construction and Analysis of Systems*, LNCS vol. 1055, Springer Verlag, pp. 147-166, 1996.
14. C. Gehrmann, C. Mitchell, and K. Nyberg. Manual Authentication for Wireless Devices. *RSA Cryptobytes*, vol. 7, no. 1, pp. 29-37, 2004.
15. Jaap-Henk Hoepman. Ephemeral Pairing on Anonymous Networks. In d. Hutter and M. Ullmann, editors, 2^{nd} *International Conference on Security in Pervasive Computing* $(SPC'05)$ volume 3450 on $LNSC$, pages 101-116, Boppard, Germany, April 2005. Springer.
16. M. Jakobsson and S. Wetzel. Security Weaknesses in Bluetooth. *CT-RSA 2001*, LNCS vol. 2020, Springer-Verlag, pp. 176-191, 2001.
17. A. W. Roscoe. Human-centred computer security. See `http://web.comlab.ox.ac.uk/oucl/work/bill.roscoe/publications/113.pdf`, 2005.
18. F. Stajano and R. Anderson. The resurrecting duckling: Security issues for ad-hoc wireless networks. *Security Protocols 1999*, LNCS vol. 1976, Springer-Verlag, pp. 172-194, 1999.
19. S. Vaudenay. Secure Communications over Insecure Channels Based on Short Authenticated Strings.*Advances in Cryptology - Crypto 2005*, LNCS vol. 3621, Springer-Verlag, pp. 309-326, 2005.