Bootstrapping Multi-Party *Ad-Hoc* Security

S.J. Creese[*] M.H. Goldsmith[†] A.W. Roscoe[‡] Ming Xiao[§]

November 16, 2005

Abstract

Increasingly pervasive computing throws up scenarios where users may wish to achieve some degree of security in their interaction with other people or equipment, in contexts where the entities involved cannot be confident of the others' long term identities and where there may be no PKI which can validate entities to each other. We examine the problem of bootstrapping security in an *ad-hoc* network formed by a group of users.

In [3] a number of protocols using low-bandwidth channels involving the human agent(s) "in the loop" to bootstrap secure communication over an untrusted infrastructure were presented, the point being that it is reasonable to suppose that the attacker's powers are more limited with regard to these *empirical* channels in one or more respects than in the standard Dolev-Yao view. The two-party protocols proved easy to verify [6] (relative to those weakened powers) by adapting the CSP/FDR-based approach of Casper [9], but the multi-party network-formation protocol (based on a combination of low bandwidth non-forgeable channels and an ordinary Dolev-Yao communication medium) proved more problematic.

Approaching the verification of this protocol [11] led to consideration of a number of simplified versions, which appear interesting in their own right. We here examine the consequences of these simplifications.

1 Introduction

In [3] the position was argued that:

- It is not realistic to assume that a universal PKI is available which can be used in the authentication of names and public keys.
- In any case it is highly unlikely that each entity will know the name of every entity it wishes to communicate with. Rather it will identify the entities by other attributes such as "the printer in front of me" or "the laptops belonging to the four of us in this meeting".
- Frequently the standard Dolev-Yao model, in which all communications can be overheard, trapped or faked, is too strong, in particular for low-bandwidth (and often unidirectional) *empirical* channels that may exist in addition to standard ones because of (for example) physical contact or proximity. Indeed, the existence of such a channel will often form the basis of authentication: A may want to form a secure link (over the standard network) with

[*]QinetiQ plc, St Andrew's Road, Malvern, `screese@qinetiq.com`
[†]Formal Systems (Europe) Ltd, 28 Temple St Oxford, `michael@fsel.com`
[‡]Oxford University Computing Laboratory, `bill.roscoe@comlab.ox.ac.uk`
[§]`Ming.Xiao@st-annes.ox.ac.uk`

the entity to which she has the empirical channel. Such channels will typically involve the direct participation of human agents, who can more easily bring these aspects of the physical world into play.

- It is therefore natural to devise a range of authentication protocols for the different possibilities of empirical channels classified according to the weakened threat models they provide (the lattice of properties in [6]) and their direction.

In [3] protocols were for presented for several different classes of empirical channel (it is expected that much of the remainder of the space may usefully be populated). One of these was a protocol for a scenario (after [1]) where a group of people are assembled in a room with their laptops or PDAs; the people know and trust one another (to whatever extent) and, by extension, wish their electronic counterparts to be able to share data securely. However the various devices have no idea of the others' identities or keys, and the (possibly wireless) network is outside their control. Here, verbal communication between users based on messages appearing on their screens (or showing those screens to one another) represents a set of low bandwidth, non-forgeable channels. The result is a protocol which gives a high degree of security, based on the standard assumption of perfect cryptography, and establishes a shared key between the users' devices, known to those alone.

2 Ad-Hoc Networks and Human Interaction

Imagine that some group of two or more systems has emerged that wish to set up a secure network amongst themselves. They can all identify who is meant to be in the group, and how many of them there are, but do not necessarily have the true identity of the participants: they might be identified by pseudonyms or by some other connection they have. In [5] it was argued that there may be good ethical reasons for seeking to set up such sessions securely and essentially anonymously and independently of any central authority. In our current example, no human/device combination knows the identity of the other devices (whether or not the humans know each others' names), but they have contextual information that allows them to identify the proposed membership of the group precisely.

Other examples might be a group of people participating in a conference call, all members of a club who are known to one another and are in touch by various means from time to time, and a pair of robotic machines that come within each other's view.

The important thing about all of these examples is that there will be a way of the different participants knowing somehow who the others are. Whenever a group has just come together and formed some sort of trusting relationship without an intrinsic knowledge of the others' names it seems fairly necessary that they have a way of passing information between them in a way that they can be confident is not being forged. The people in the room have this, since they can see and discuss things with each other. People on the telephone have it provide they discount the possibility of someone else being able to imitate them so as to convince the others. The robots could have this through line of sight or perhaps physical contact, and doing things the other could interpret like flashing lights. The members of a club would have it via meetings supplemented by individual conversations for spreading gossip.

In truth, of course, human interactions are subject to some risks of impersonation, snooping etc. What we will seek to do here is to make communication between their systems only vulnerable to the humans being impersonated successfully to their peers. The humans would be comforted that their devices cannot themselves introduce security flaws, provided that they do their jobs properly. It is beyond the scope of this paper to work out how to ensure that they do!

From here on we simply assume that the humans have some empirical channels for communicating with each other that are not forgeable: if A thinks that message M has come from B, then B really did send M to A. The humans can therefore implement such channels between their systems by reading from their screens and conveying the information to the owners of other systems. For obvious reasons these are quite low-bandwidth channels, even if we allow the screens' contents to be shared visually, rather than orally.

Even though the human interactions are assumed to be vulnerable to snooping (the attacker can hear everything that is said), we will achieve a common secret key between their systems, ensuring privacy at that level: the attacker cannot understand the data passing between the devices.[1]

We design our protocols to guarantee security over a normal Dolev-Yao network such as wireless or the internet subject to the existence of these empirical channels. So they will work wherever some means of communication with these properties exist, not just through human intermediaries.

3 The Network-Formation protocol

In this section we concentrate, for the purposes of discussion, on a group **G** of humans attempting to achieve a secure link between their devices. However the protocol would work wherever there is are low bandwidth non-forgeable channels between each pair of agents. A protocol[2] for just these purposes was proposed in [3]. In standard notation it is:

1.	Each A	$A \rightarrow_N$ Every B	:	$\{A, kc(A), N_A\}_{wkk}$
2.	Each A	$A \rightarrow_N$ Each B	:	$\{$all decrypted Message 1's, $N'_A\}_{pk(B)}$
$3a$.	Each A	A displays	:	$hash(\{$all decrypted Message 2's$\})$, number of active participants
$3b$.	Each A	$A \rightarrow_E$ Each B	:	Users compare hashes and check numbers on screens
4.	Each A	$A \rightarrow_N$ Each B	:	$hash'(\{$all decrypted Message 2's$\})$

Here, the boundary between Message 1 and Message 2 is determined by some means: a time-out; counting the incoming messages on-screen and allowing the user to press a key to proceed when the expect number are received; or by a parameter to the invocation of the protocol.[3] 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 communication over the normal Dolev-Yao network, where all messages are subject to overhearing, taking out and faking, whereas $\rightarrow_E$ means over the empirical channels which are, in this case, immune to faking.

The value $kc(A)$ is some kind of *key certificate* for A. 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 be generated by someone other than a recognised authority. Therefore one system hearing a name A in a Message 1 carries no particular

[1]The transition from unforgeable to confidential is far from new: compare for example [2].

[2]Here distinguished from the simplified versions that follow as "the Full Protocol".

[3]The protocol is clearly not safe against denial of service attacks, since an active attacker can always insert his own Message 1.

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. Note that an agent is entirely free to use different names and different public keys in different runs of this protocol.

We can argue the correctness of this protocol as follows, under the normal *perfect cryptography* hypothesis that an intruder cannot break a code without the key, and cannot guess a value that gives a particular hash.

The encryption in the first round of messages is under some widely-known key wkk, known at least to all potential participants in the protocol. It might be a symmetric key disseminated amongst some profession, or perhaps one of a tractably small number of manufacturer issued keys. It offers a limited degree of pragmatic security and integrity, but none at the level of our analysis since we, as standard, assume that all agents not in our group $\mathbf{G}$ may be corrupt. Of course if some *a priori* bound is placed on the range of agents who may participate in some version of the protocol, and consequently on the set of agents who know wkk, this would improve the pragmatic strength of this move.

Once Messages 1 and 2 have been exchanged, each node has a set of nodes to which it is connecting, and has verified that all of them agree on the set of names. It also has two nonces from each participant, one which is public and one that is apparently secret. At this stage we have no way of knowing that the participants in a particular A's session are actually the set $\mathbf{G}$ of agents with which she has the empirical channels within $\mathbf{G}$. Each user can be confident, under perfect cryptography, that anyone else with the same hash value agrees on the sets of all Messages 1 and 2, so if everyone in the group agrees on this value then all of them are in the session. There is nothing at this stage, however, that stops an extra participant being present that the devices are all aware of but the humans are not. This, of course, might well (indeed must) be an intruder. This the the reason for the devices displaying the number of participants so the humans can verify that it is the actual size of $\mathbf{G}$.

All the users know that the hash value is unpredictable since they have individually contributed to its "randomness" via their nonces, but not in a way anyone can predict. (This is a basic assumption about cryptographic hashing.) It follows that the group cannot be in two or more separate sessions that happen to have the same hash value (or rather that we have discounted that possibility with our perfect cryptography assumption). For there is no way for an intruder to ensure that the different sets see the same value. Therefore the checking of Message 3 ensures that N participants are in the same group of size N. They can thus validly deduce that they are the only participants and that all the new nonces sent in Message 2's make up a shared secret between them.

Naturally, all the above depends on the participants who are in the empirical group actually being trustworthy and implementing the protocol properly. That is an assumption more or less universally made in authentication protocols.

Message 4 in the protocol simply represents the devices performing an additional round of agreements amongst themselves prompted by their users pressing "OK" after Message 3. It is not evidently necessary to include all of the Message 2 bodies in the hash; the set of the various N'_A values should suffice.

Having a shared secret (the $\{N'_a \mid a \in \mathbf{G}\}$) allows the devices of the group members to communicate securely: they can use it to generate a session key. Unlike the humans' conversation, the contents of the encrypted messages cannot be understood by any intruder, so they have promoted their mutual trust to a higher degree of security amongst their devices. Provided that they have not trusted anyone incorrectly, they know that only **G** can understand messages encrypted under that key, and anything encrypted using it must have come from a member of **G**.

The above argument represents a convincing proof that the principles underpinning this protocol are correct. Nevertheless it is desirable that we can somehow prove this protocol mechanically in as general a sense as possible. It is, however, a large protocol when more than two nodes participate, and there is the additional challenge of addressing arbitrary group size.

In the following section we look at simplifications both with a view to getting more tractable automated proofs and in order to understand it better.

3.1 Simplifying the protocol

It is worth considering what the purpose of Message 2 is; its complexity is certainly a hindrance to automatic verification of the protocol. It does not appear to be strictly necessary to gain assurance that the group members are running the protocol only amongst themselves. Comparing hashes of Message 1's and the counts would do that. Doing only that, however, would not allow them to build a shared secret: anyone overhearing the Message 1's knows all the identities and nonces, and hence all hashes of them by public algorithms.

So Message 2 does achieve a useful purpose. Including all the Message 1's in it has one security function and one of a general nature. The first is that they ensure freshness. The second is to allow the devices to be reasonably confident that they all agree on the participants before making their human users do any workOf course a hash of the Message 1's would be as good as the messages themselves here. If we are prepared to sacrifice that advantage, we can achieve a similar effect with a rather simpler protocol[4]:

1.	Each A	$A \rightarrow_N$ Every B	:	$\{A, kc(A), N_A\}$
$2a$.	Each A	A displays	:	$hash(\{$all Message 1's$\})$, number of active participants
$2b$.	Each A	$A \rightarrow_E$ Each B	:	Users compare hashes and check numbers on screens
3.	Each A	$A \rightarrow_N$ Each B	:	$\{A, \{N'_A, N_B\}_{sk(A)}\}_{pkB}$

Message 1 is as before, except that for simplicity we drop the encryption under wkk, which as we remarked earlier adds only limited pragmatic benefit.

Message 2 plays the role of Message 3 in the original protocol, and achieves the same assurance that all and only members of **G** have contributed to the hash.

Message 3 exchanges new nonces securely; the inclusion of N_B serves to guarantee its freshness to the recipient. Once more, the set of the various N'_A constitute a shared secret.

We may remark that there is a rather odd kind of "attack" on the Simplified Protocol: since the keys exchanged in Message 1 are only used in Message 3, there is no guarantee that the entity purporting to be A actually knows $sk(A)$ until after the humans have been brought into the loop.[5] In the case (which we excluded above) that one of the members of **G** is malicious, this could lead to damage to A's reputation, as "she" can be identified as repeatedly responsible for failure of the protocol run. Note that this fate can befall a participant even if she is present and trying to behave according to the protocol: an attacker can abstract her Message 3 to achieve the same effect.

[4] "The Simplified Protocol"

[5] This is actually a good argument for swapping the order of Messages 2 and 3, even though this may require a degree of wasted effort in performing public-key cryptography in the case that the hashes don't match; it depends on the relative value assigned to battery life and human patience!

For this reason, it may actually be advantageous if there are no long-term names involved.[6] We can identify A with $pk(A)$, and drop the (now duplicate) name from Message 1. At first sight, one might suppose that the nonce N_A is necessary to prevent a replay of Message 3, which has no guarantee of freshness to B except the return of his nonce. If, however, we generate a new key-pair for each run of the protocol, then the encryption key itself can serve this purpose.

Thus we can arrive at a further simplification of the protocol[7], where for variety we adopt the alternative message order suggested in Note 5:

1.	Each A	$A \rightarrow_N$ Every B	:	pk_A
2.	Each A	$A \rightarrow_N$ Each B	:	$\{pk_A, \{pk_B, N'_A, \}_{sk_A}\}_{pk_B}$
$3a$.	Each A	A displays	:	$hash(\{\text{all Message 1's}\})$, number of active participants
$3b$.	Each A	$A \rightarrow_E$ Each B	:	Users compare hashes and check numbers on screens

where the (pk_A, sk_A) are a putatively fresh public-key pair formed just for this protocol run, and pk_A can serve as both name and nonce as well.[8]

3.2 Further simplification

Several protocols in the literature rely on acquiring the public-key of one's interlocutor, for example Lowe's 7-message version of the corrected Needham-Schroeder Public Key protocol [7]. Often this presumes the presence of a trusted server issuing certificates, but the current protocol offers an alternative:

1.	Each A	$A \rightarrow_N$ Every B	:	$[\ A,\ ]\ pk_A$
$2a$.	Each A	A displays	:	$hash(\{\text{all Message 1's}\})$, number of active participants
$2b$.	Each A	$A \rightarrow_E$ Each B	:	Users compare hashes and check numbers on screens

After executing this protocol[9] successfully amongst a trustworthy group of entities, all are in possession of a common set of public keys, and optionally identities, which by the assumption of trustworthiness each belong to one member of the group and can be taken to provide a means for confidential messages to and authenticated messages from that member. These can now form the starting point for any desired further protocol exchange that depends on exploiting that facility.

4 Lessons for protocol design

These protocols are related to the following principle enunciated in [6]:

> A low-bandwidth empirical non-forgeable channel can, via hashing, convert a Dolev-Yao channel in the same direction into a non-forgeable one.

[6]There are equally valid alternatives, of course, such as including $\{N_A\}_{sk(A)}$ in Message 1.

[7]"The Anonymous Protocol"

[8]It may be desirable therefore to include pk_B in clear in Message 2, simply as address information to prevent unnecessary decryption attempts; of course, an attacker can pervert this, but it does improve the best-case performance.

[9]"The Core Protocol"

This is because if A sends something to B via the network, she can back it up with a hash of the same value to B which, being unforgeable, proves that A really did send the original message.

The difference here is that instead of backing up a single transmission, a single hash is used in the two protocols to back up transmissions from many agents at once by arranging that all participants will have common – and hopefully identical – views of what has been sent. The role of the agreed count is to ensure that the correct instantiation (i.e. size) of protocol is being run.

Thus it is possible to "bootstrap" the security of a non-forgeable channel by utilising pre-existing empirical channels. Such channels must be either trusted and trustworthy (we cannot protect against a double-agent who is able to lie about the empirical evidence in a manner undetectable by device and user), or any cheating must be detectable by the devices or users[10]

Various mechanisms may be designed into the protocol to add confidence in the trustworthiness of users and devices. An example of this is the use of *wkk* in the Full Protocol, where possession of the key suggests that the holder has been authenticated previously at some level. Of course, use of such mechanisms for building trust should be treated with care as the evidence they provide may not be trustworthy in the operational context under consideration (e.g. it may be out of date). In some cases infrastructure services may be utilised to check the trustworthiness of evidence, however access to them may not be guaranteed in pervasive computing environments. The ability to bootstrap security via the combination of strong hashing and empirically checkable evidence enables protocols to be designed which can tolerate the changeable nature of communications connectivity in ad-hoc networks[11]

5 Verification and proof

The model checking community has become very expert at checking and verifying protocols that run between a fixed number of participants with distinct roles. There has been relatively little work on protocols which involve an arbitrary number of participants, and none that we are aware of where all these participants interact symmetrically with each other in rounds of communication. This means that for the protocols presented in this paper we have had to reconsider both the nuts and bolts of the network model, and also the question of how one can turn the verification of a particular finite model into one for any instantiation of the protocol. The most striking difference is that previously we have had to check that a protocol worked between a fixed-size collection of nodes playing certain roles, amidst other nodes and a network that are potentially correct. Now we have to allow for any size group participating in the protocol itself. In effect we have a family of protocols: one for each group size.

Our approach has been to build models of the protocols in CSP for finite checks by incremental development from traditional ones, but to think radically about methods of *verifying* arbitrary group-formation protocols.

As a target for model building we chose the Simplified Protocol from Section 3.1. Our first step was to build a model for the same protocol in the binary version (i.e. groups of two only) and where we had artificially introduced initiator and responder roles: thus each of the exchanges

[10]Achieving detection of cheating over an empirical channel may be relatively straightforward, since communications over such a channel are typically observable. But if the hashes are compared verbally, then there is nothing to prevent a malevolent participant claiming to agree with whoever reads out first; there is no advantage to such behaviour in these protocols, however, and we have in any case assumed that the users are willing to trust one another.

[11]Lowe has suggested an alternative possibility based on the same idea: agreeing a hash of the key derived from a Diffie-Helman exchange to provide end-to-end authentication.

of Messages 1–3 were fixed to happen in the order initiator sends to responder, and then vice-versa. The main difference from traditional protocol modelling was that (looking forward) we sent and stored information as sets, which required straightforward additions to our data and intruder models. The result was exactly as expected: no attack against the authentication of the nodes with empirical channels to each other, and secrecy of the final nonces.

Our second step was to move to making the exchanges symmetric. The two main issues here were as follows:

- How do we tell our computers who to form a session with in the first place? This is rather nonstandard since we do not know the other processes' names at the start. The solution we adopted was for each node to build up a group G of nodes based on its own identity and those of nodes from which it receives Message 1's.[12] Of course at the end the nodes should only think the protocol is complete if the identities gathered in G are correct and the same across all parties.

- How do we model the sending and receiving of the broadcast messages that this protocol requires (i.e. it sends and receives each message to and from each other node in G)? Simply allowing $2(\mid G \mid -1)$ messages in any order is fraught with combinatorial problems: 24 permutations for $\mid G \mid= 3$ and 720 for $\mid G \mid= 4$ with no intruder, and potentially those numbers squared if he is around.

  Since the intruder can in any case block or duplicate messages, there is not really much point in a broadcaster knowing how often it has sent a given message. Therefore we simplified our model so that a broadcaster will send his message arbitrarily often to anyone. This dramatically reduces the state explosion due to order of communication.[13]

The Simplified Protocol worked exactly as expected when restricted to group size 2 in this symmetric version, but group size 3 produced very long runs of FDR and it was clear that we would be unable to extend it either to larger groups or to incorporate the sort of data independence mechanisms necessary to *verify* the protocol in general (even for a fixed group size).

At this point we switched our attention to the Core Protocol, which simply provides authenticated transport of public keys without checking between nodes. In this case we sought to verify that, after a run of the protocol with all nodes trustworthy, each node has precisely the set of (identities and) keys of the nodes in the intended group. A side effect of the simplification step producing this protocol is that the intruder is now able to generate, from the very start, all messages that could ever be sent in the protocol. Related to this is the fact that nodes no longer have to generate fresh values for each run (provided they are using long-term public keys), and therefore that our CSP models are not limited to how many runs a node can perform in sequence.

It follows by fairly easy data independence arguments [8] that a CSP model of this protocol in which there are $n-1$ identities for the intruder to use, and n trustworthy nodes, can verify the protocol for group sizes up to and including n. This can easily be done for $n = 3$ and probably, with more patience, for 4 and maybe 5.

[12] In any practical case where the computers of the intended participants are not the only ones present, it would be as well for the humans to agree some informal name for their run or to swap Message 1's via some directed, if insecure, channel such as email.

[13] It can give rise to some implausible-looking error traces from FDR, since sometimes an agent has, thanks to this model of broadcast, communicated a message no times when the intruder knew it already. It is always easy to translate such traces into more conventional ones that are still attacks.

Of course, what we actually want to do is prove this protocol for arbitrary group size. Intuitively, it seems obvious that if such a simple protocol is correct for small groups then it will also be correct for large ones: there simply is not enough "room" for interesting attacks to appear for the first time on large groups. Unfortunately this seems to be hard to formalise using any model checking technique we are aware of: the problems are the arbitrarily large (in a symbolic sense) Message 2 and the fact that any proof will necessarily use facts about set theory and the natural numbers.

We have already stated the mathematical argument which underpins this protocol informally. Here we summarise it formally:

- Assuming that it is infeasible for the intruder to manipulate the hash, we can infer that when the trustworthy users have equal hash values, then the sets their computers are hashing are all equal to some set S.
- Since their computers are assumed to be implementing the protocol reliably, we can deduce that the size of S is the same as the number N which the users check is the size of the group.
- We can assume the likelihood of the Message 1's of two users being equal is negligible. Therefore they contribute N different Message 1's.
- The fact that user A hashes set S implies that A's own Message 1 is a member of A.
- Since this is true for N different users with different Message 1's, it follows that S consists solely of these messages.
- Therefore, when A has acknowledgements from the other $N-1$ users that they agree with A's hash, she can be certain that by using the public keys in the Message 1's she is only addressing the computers she intends to, namely those belonging to other members of the group.

It seems certain that this argument could be worked up for a theorem prover, and we invite people to do this, though at least in this version it is sufficiently simple not to need this.

All our protocols essentially contain the Core Protocol within them, which means that the above argument also works for them. This could be used as a lemma in any proof of a stronger property we wanted to prove of them (such as freshness or the secrecy of some data). It is noteworthy that the Core Protocol relies on no secrets other than secret keys, which means that it is hard to think of any ways in which an extension of it could lose the basic authentication of public keys property proved above.

This lemma could be used to support further manual proof, proof with a theorem prover, or even model checking. This might take the form of asserting that once the core protocol has been completed we know that nothing which has previously been sent only encrypted with the authenticated public keys is known to the intruder unless he knew it initially. This particular lemma would be supported by individual proofs that the node processes do maintain this degree of security (something very suggestive of rank function [9, Chapter 7] or strand space analysis [10] as well as model checking). We could then infer the secrecy of everything claimed for our various protocols as shared secrets, *once the core protocol has completed*.

For proving freshness, it is clear that any term F which has been generated freshly by any user's computer for this run *and which is included in the empirically agreed in such a way that the user can deduce F from the pre-hash term* is fresh. Therefore in the original protocol each node can be confident that all nonces are fresh.

The only one of our protocols where the freshness of anything presents a challenge is in the Simplified Protocol where the nonces N'_A are intended to be fresh but are not included in the hash (and may actually be generated after the hash agreement). The proof of freshness here is that these nonces have been signed and encrypted by their generators, linked with something that the recipient already knows to be fresh (his own nonce).

Our conclusion is that model checking is in itself not at present capable of verifying this style of protocol for arbitrary group sizes by itself. It may well be possible for theorem provers, and we lay this down as a challenge. On the other hand the basic correctness of the core protocol is easy to prove manually, and this is of great assistance in further analysis.

On the other hand building small implementations for model checking, as described above, confers two specific benefits:

- It adds confidence to the manual or theorem-proving analysis, since it is capable of demonstrating that there is nothing obvious that we have somehow omitted to prove.
- It allows easy experimentation with variations on the protocol, allowing one to discard ones that are too weak and to formulate correctness properties to prove of the other.
- In particular it allows one to discover to a high degree of confidence how much a protocol depends on a specific construct or assumption.

6 Conclusions and Further Work

The previous section highlights various challenges in terms of the development and application of verification techniques.

We believe that this style of protocol offers great opportunities for new applications of computer security simply because it is so independent of the infrastructure that usually imposes a high cost on security. We hope that the concept of promoting human trust to their computers will appeal to potential users. Of course it will be important to educate them that they do not gain trust in the users by running the protocol successfully with their computers.

Since we are involving humans in the work of our protocols, it is desirable that they should not be engaged in unnecessary work. We have already highlighted the issue of making it as difficult as possible for an intruder to cause the computers to ask their humans for agreement when agreement will not in fact occur. The most we can realistically hope for here is that this does not happen unless the intruder has participated actively by sending messages, and in seems that the Full Protocol and agreement-last Simplified Protocol achieve this. Another issue here is that of making it as easy as possible for humans to agree on a hash value which is sufficiently complex so as to justify the non-deducibility assumptions we have made in this paper. This is an issue to which we intend to return in a subsequent paper.

We believe that our approach to bootstrapping security offers much strength:

- We no longer require access to trusted third parties in order to authenticate, thus reducing our dependency on infrastructures and bandwidth.
- The dependency on empirical channels necessarily involves the “user”, in some sense. In cases where users are humans the empirical channel requires them to do what they do intuitively, *observe*, so offering the potential to greatly enhance the user experience for those who struggle to trust digital communications which they *can't see*.

- Our approach allows for the introduction of third party evidence should it be available.

We envisage such protocols will be utilised in peer-to-peer computing environments enabling users to quickly form secure groups without having to utilise third party band-width. Here security is bootstrapped either solely in their pre-existing mutual trust or in conjunction with third party evidence (such as that provided by a Public Key Infrastructure). Such third party evidence may be based upon certificates issued by professional services (such as financial or health care) which have long standing experience in managing identities. (We are already seeing evidence of such services in high-street banks.)

Empirical Channels The trustworthiness of the empirical channel is core to our ability to bootstrap security utilising such a channel. There are many ways in which to implement such a channel which include human-human and human-device channels. Whilst we have in-built mechanisms for determining how and when to trust another human, some call it intuition, the same is not true for human-device communications. In [4, 6] we utilise an empirical channel to design a protocol for bootstrapping security between a human and a wireless printer; specifically we authenticate that the printer we have established a shared secret with must be the printer we have within our direct line of sight. Here the empirical channel was a visual check of a display on the printer to verify the hash in conjunction with a visual indicator that the device had not been tampered with. Research is required into: the various options and permutations for creating such empirical channels; the various operational contexts for which they are suited; the degree to which they invoke trust in the user; the types of authentication they can provide (e.g. of entity identity or function).

Formal Verification Whilst we have some limited success in the verification of the protocols presented here, we believe that further research into automatic verification techniques suitable for multi-way protocols is required.

References

[1] N. Asokan and Philip Ginzboorg. Key-agreement in ad-hoc networks. *Computer Communications*, 23(17):1627–1637, 2000.

[2] D. Balfanz, D. Smetters, P. Stewart, and H. Wong. Talking to strangers: Authentication in ad-hoc wireless networks, Feburary 2002. In Symposium on Network and Distributed Systems Security (NDSS '02), San Diego, California.

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

[4] S. Creese, M. H. Goldsmith, Bill Roscoe, and Irfan Zakiuddin. Authentication in pervasive computing. In D. Hutter and M. Ullman, editors, *First International Conference on Security in Pervasive Computing*, Boppard, March 2003. Springer LNCS.

[5] S. 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] Sadie Creese, Michael Goldsmith, Richard Harrison, Bill Roscoe, Paul Whittaker, and Irfan Zakiuddin. Exploiting empirical engagement in authentication protocol design. In Dieter Hutter and Markus Ullmann, editors, *Proceedings of* 2^{nd} *International Conference on Security in Pervasive Computing (SPC'05)*, volume 3450 of *LNCS*, Boppard, Germany, April 2005. Springer.

[7] Gavin 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.

[8] A. W. Roscoe and P. J. Broadfoot. Proving security protocols with model checkers by data independence techniques. *Journal of Computer Security*, 7, 1999.

[9] P.Y.A. Ryan, S.A.Schneider with M.H. Goldsmith, G. Lowe, and A.W. Roscoe. *The Modelling and Analysis of Security Protocols: the CSP Approach.* Addison-Wesley, 2001.

[10] J. Thayer, J. Herzog, and J. Guttman. Strand spaces: Proving security protocols correct. *Journal of Computer Security*, 7, 1999.

[11] Ming Xiao. Verifying the PDA-mesh protocol. MSc, Oxford University Computing Laboratory, September 2005. Sumbmitted.