Exploiting Empirical Engagement in Authentication Protocol Design*

Sadie Creese[1], Michael Goldsmith[2,3], Richard Harrison[1], Bill Roscoe[2,4], Paul Whittaker[2], and Irfan Zakiuddin[1]

[1] QinetiQ, Malvern, UK
{S.Creese, R.Harrison}@eris.QinetiQ.com
I.Zakiuddin@signal.QinetiQ.com
[2] Formal Systems (Europe) Ltd
{michael, paulw}@fsel.com
http://www.fsel.com
[3] Worcester College, University of Oxford
[4] Oxford University Computing Laboratory
Bill.Roscoe@comlab.ox.ac.uk

Abstract. We develop the theme of an earlier paper [3], namely that security protocols for pervasive computing frequently need to exploit empirical channels and that the latter can be classified by variants of the Dolev-Yao attacker model. We refine this classification of channels and study three protocols in depth: two from our earlier paper and one new one.

1 Introduction

1.1 Pervasive Computing Environments and Security

The pervasive computing paradigm predicts a future in which wireless communicating and computing devices are ubiquitous throughout our environment. This will in turn facilitate the formation of dynamic networks operating independently of back-bone infrastructures, offering the capacity for short-lived relationships operating over hybrid resources and perhaps utilising local distributed processing services. It will also offer the ability to connect to infrastructures where they exist via a wide variety of means, providing major opportunities for service-based business models. The exact form of such devices will vary from the embedding of such functionality within already commonplace electronic objects, to enhancing the functionality of previously non-electronic devices, and to the creation of new bespoke devices designed to offer specific communications and computing services (perhaps on a nano scale).

* This research is conducted as part of the FORWARD project which is supported by the U.K. Department of Trade and Industry via the Next Wave Technologies and Markets programme: www.forward-project.org.uk.

It is clear that the pervasive computing paradigm offers huge opportunity, as evidenced by the level of international research being focused on the domain. Successful exploitation of such opportunity will require both a trust in the technologies on the part of the user community, and that the technologies are dependable and worthy of such trust. Information security is a fundamental component of dependability in pervasive computing environments.

However, it is also clear that there are significant challenges to be addressed if we are to secure information in such potentially complex computing environments. The connectivity offered and required in order to benefit from the pervasive nature of computing resources and services may also offer a malicious intruder more options for unauthorised access. Services offering bespoke functionality and information may in turn require access to increased stores of personal or organisation details, which once networked may potentially be accessible from an unknowable set of interfaces and users. Information may exist in unknown locations for indeterminate durations.

A key concept in security of any type is authentication: you must be able to decide whether a user (human or otherwise) is authorised to access a resource. Without the ability to authenticate we would be unable to implement a useful security policy[1]. It is likely that we will require mechanisms both for authenticating the identity of a device or user ("which" or "who"), and for authenticating how a device or user will behave ("what"). We are concerned here with the former, the challenge of identity authentication in pervasive computing environments and in particular where no previous knowledge of the device or user exists.

This is a particularly challenging problem as it necessitates either:

– that there exists an accessible trusted third party who can vouch for a claimed identity based upon some pre-agreed information or token (such as a public-key or a biometric),
– or that there is some mechanism for bootstrapping trust between strangers (human or devices) where there exists no trusted third party who can verify identities.

Neither of these options are entirely unproblematic. To implement a trusted third party solution one would first need to ensure that all devices can be guaranteed access when they require it. It would not be possible to construct a centralised model, where there exists one such authority, as the number of devices to be verified may be so large that it becomes impossible to guarantee bandwidth, freshness of mirror sites and even the existence of unique naming policies. If we are to implement many such identity verification authorities then we will require interoperability between authorities, and associated access. Finally, even if a device has a unique name it may not be possible for another user or device to establish what that name is, and so to refer to the device.

The FORWARD project is investigating how we might bootstrap security by utilising authentication protocol services built upon empirically verified proper-

[1] A security policy which simply specifies that everyone or anything has access would of course not require an authentication mechanism.

ties of our local environment. This has the benefit both of reducing overheads associated with network communications, and that users will be interacting as part of the service via their primary senses collecting empirical evidence (such as feeling, hearing, seeing)[2]. Core to our design methodology is the use of formal analysis in order to better understand protocols' behaviour and facilitate high-assurance design.

We previously presented [3] a variant hybrid threat model precisely for reasoning about and analysing such services. The threat model includes mechanisms for representing the various types of security property that we might assume of the various empirical communications. This in turn enables the formal analysis of any authentication protocols developed, and an understanding of the corresponding levels of resilience offered by such authentication methods. Whilst in [3] we presented a number of suggested protocols for example pervasive computing scenarios, we had at that time not utilised the variant threat model to formally analyse their behaviours.

In this paper we present refinements to the variant threat models, report on our analysis of two protocols specifically for authentication between two devices in a local (line-of-sight) environment, and present an overview of the methodology for formally analysing authentication protocols utilising such empirical engagements between two devices.

We begin with a discussion of formal analysis in general, and the refinements to the variant threat models. In §3 we present an overiew of variants of the two protocols designed for device authentication in local environments (presented in [3]), and discuss the potential for design modifications. In §2.1 we briefly discuss our chosen formalism for analysis – the process algebra CSP (Hoare's Communicating Sequential Processes [5, 7]), and the accompanying tool support from FDR and Casper [8] – we outline our analysis methodology, and we present the results of such analysis on these protocols. Finally, we present our conclusions and directions for further work.

2 Refinements to the Variant Threat Models

Formal analysis itself depends on a (formal) representation (*viz.* a model) of the subject that is both[3]:

- *faithful* in the sense of capturing the behaviour that needs to be analysed in a traceable way; and
- *clear* to an analyst that the model is correct and the analysis useful.

[2] The use of such human verifiable properties should help make the experience of using such services more intuitive and perhaps easier.

[3] Tractability of the model is an important concern as well; but tractability is closely linked to the verification technology that is used. Moreover, an unfaithful and tractable model is useless; and a faithful, tractable, but unclear model is only a little better.

If we do not achieve this then the benefits of the analysis may not be realised in the design.
Any solution that we might propose must be flexible enough to capture:

- The varying assumptions that determine how users and electronic devices may interact to ensure successful initialisation of the link.
- The range of entities that can form the participants in the protocol, and their associated behaviours.

As formal specification and analysis depend crucially on capturing these assumptions and the objectives that the protocol is trying to achieve, the first of these points is particularly important. A fundamental requirement in any formal analysis of security properties is to construct an appropriate threat model for the environment in which the system is designed to operate. The threat model encapsulates the capabilities of an attacker, and so an incorrect threat model may lead to either:

- too weak an attacker where security can be proven in the formal model, but does not hold in the actual implementation environment; or
- too strong an attacker, which can then place restrictive constraints on the design that impact resource usage and overall functionality, when that could have been avoided.

The type of security service with which we are concerned is that of authenticated cryptographic key agreement. Currently, the *de facto* standard threat model for analysing key-agreement protocols is the Dolev-Yao model [4].

The Dolev-Yao model supposes an intruder who effectively controls the communications network, and is therefore capable of

- overhearing messages between legitimate principals
- intercepting messages and preventing their delivery to their intended recipient
- synthesising – within the limitations of the cryptographic mechanisms involved – messages from data initially known to him together with fragments of any previous messages and delivering them, apparently originating either from an identity under his control, or indeed from any other principal.

In essence this is the most potent malicious attacker that a protocol can possibly need to cope with; in effect the worst-case scenario. However, designing protocols which can withstand attack of this nature is certainly erring on the safe side: a protocol which exhibits no flaws under these assumptions will *a fortiori* be secure against a less potent attacker.

Here we are concerned with the ubiquitous arena, where most communications are through the essentially broadcast medium of wireless. The attacker can interfere with communications, attempting to subvert or disrupt the protocol that the principals are using for authentication. The facilities at the attacker's disposal depend very much on the nature of the communications that he is trying to attack. When two people physically exchange PGP keys[4] (and thus use

[4] www.pgpi.org

empirical engagement to initialise an authenticated link), the scope for disruption of that authentication protocol is very limited. It is in effect assumed that the two participants share a “channel” that has very few vulnerabilities. If the keys were exchanged in a digital communication using clear text then the scope for disruption would be much wider. So, in this case, empirical engagement has been used as a secondary channel for bootstrapping security. We may remark that this approach is consonant with many less theoretically-inspired suggestions in the literature [9, 1, 2, 6–etc].

In [3] we observed that this type of empirical engagement may be increasingly possible for bootstrapping security in the pervasive computing environment. There we presented a series of protocols which depend upon the existence of such engagements for their correctness. The scenarios envisaged all relied on locally-verifiable empirical data, and accompanying assumptions regarding the restricted powers of the attacker.

When considering how empirical engagement might be used as part of an authentication protocol, it is clear that the vulnerabilities associated with empirical engagement are likely to be diverse and context-specific. Thus our strategy for developing clear and faithful models of empirical engagement in authentication starts by noting that such interactions can be captured as a form of communication channel with specific properties, and for security analysis the most important of these properties are the channel’s vulnerabilities. From the viewpoint of formal analysis these vulnerabilities are equivalent to the attacker’s capabilities.

We incorporate the vulnerabilities of a channel into a modified threat model, which restrains the capabilities of an attacker appropriately. In the pervasive computing environment we observe that there are potentially two types of communications channel:

– A high bandwidth bidirectional medium with low or unreliable security. This represents the *network* wireless communications medium, such as wireless LAN, Bluetooth, or IrDA.
– The second type represents *empirical* channels, such as reading a message on the printer display panel, physically punching in a code on the printer control panel, or checking that the flashing light is present as in the first example below.

We shall call the high-bandwidth digital communications channel N (for “network”) and the “empirical” channel E. This channel can be considered more costly, as a human is necessarily “in the loop” and the channel is therefore low-bandwidth and consumes the scarce resources of intellect and attention. However, the E-channel can thus offer various forms of higher security; it may, according to the details of the scenario, operate in either or both directions.

The two (or more, if multiple empirical mechanisms are available) channels can be supposed vulnerable according to a different threat model. This flexibility can be exploited to develop cryptographic protocols which are secure where they would not be under standard Dolev-Yao assumptions.

By varying how the attacker can manipulate these two channels (the attacker’s capabilities on E being the more limited), a variety of hybrid threat models can be created. In [3] we identified a number of reduced-threat variants, each specifying a restriction placed on the powers of the Dolev-Yao attacker:

- AOT_C : the attacker cannot both block and hear messages at the same time on channel C, where AOT stands for *Atomicity of Transmission.*
- NS_C : the attacker cannot spoof messages on channel C, where NS stands for *No Spoofing.*
- NOH_C : the attacker cannot overhear messages over channel C, where NOH stands for *No OverHearing.*

To this list we may add NB_C : the attacker cannot block messages on channel C, where NB stands for *No Blocking.*

These restrictions are not entirely independent of one another. Indeed, we may form a small lattice, making explicit the modality over time (or messages):

$$\begin{tikzcd}
& \square(NB \wedge NOH) \quad (\equiv \square NB \wedge \square NOH) \arrow[dl, dash] \arrow[dr, dash] & \\
\square NB \arrow[dr, dash] & & \square NOH \arrow[dl, dash] \\
& \square(NB \vee NOH) \quad (\equiv \square AOT \not\equiv \square NB \vee \square NOH) \arrow[d, dash] & \\
& \mathit{true} &
\end{tikzcd}$$

Here the top of the lattice is reliable and confidential, while the bottom allows the full Dolev-Yao powers over these aspects, with varying degrees of security in between; higher properties imply the lower. Adding NS along another dimension yields a rich variety of channel types that may be available[5]. A later paper will populate this design space with illustrative protocols.

The idea behind AOT, which is a dynamic blend of NOH and NB, is that in some circumstances it may be impractically hard for an attacker to "jam" a signal and at the same time hear the message that was being conveyed. We have received more feedback on this item than any other, mostly suggesting cunning noise-cancellation schemes which make it hard to justify the assumption that this property holds of radio traffic. We stand corrected, and seek to reduce our reliance on the notion.

2.1 Formal Analysis of Variant Threat Model Cryptographic Protocols

Our analysis is based around use of Formal Systems' refinement checker FDR, with the CSP models of the agents and the attacker typically generated by Lowe's Casper front-end tool. Space precludes a full presentation of the technology: the reader is referred to [8] for a complete exposition.

[5] Especially as NS is not as simple a concept as it may seem at first sight; see below.

Casper takes as input descriptions of security protocols written in a "journal" notation, together with specifications that assert the security properties claimed for the protocol. Casper compiles a given protocol description to CSP processes representing the possible behaviours of the agents when run in the presence of the Intruder – standardly a Dolev-Yao attacker. The security specifications are also compiled to CSP processes. These processes – representing the implementation and specifications of the protocol – can then be automatically compared using the CSP refinement checker FDR2.

The Casper model of the Intruder is that of the standard Dolev-Yao model, modulo perfect encryption: collision-free hash functions, strong message types, and so on. The Intruder is given complete control over the network and may *overhear*, *intercept*, *re-route*, *delay*, *reorder*, *replay*, *fake* or *obliterate* a message. Although this default model of the Intruder is sufficient for most analysis work, the tool is very flexible as regards deductions on the part of the Intruder and the user may easily extend the threat model to weaken the assumptions of perfect cryptography. However, it is harder to change the model to weaken the attacker as regards control over the network simply because the Intruder and network are one and the same entity within Casper. Lowe is working on incorporating (at least some of) the notions required here into the tool, but in the interim we have worked by modifying the resultant CSP scripts directly.

The special channel properties AOT, NS, and NOH that define the new variant threat models have been implemented approximately as recommended in [3] – we curb the powers of the Intruder by limiting its ability to *take*, *fake*, and *overhear* certain messages on the network.

The first step in implementing the special channels is to (re-)introduce, at least conceptually, a channel *comm* to represent direct, uninterrupted communication between agents. The Intruder has no control over this channel and may only overhear messages. This separates out the Intruder from the network and allows us to limit its powers by modifying the mapping of *send/receive* events for those messages sent over the special channels. To add the new channel we must first endow both agent processes and the Intruder process with the event. For agent processes, we add the channel by mapping *send/receive* events to both *send/receive* and *comm* events. For the Intruder process we do the same but for only the *hear* channel as the Intruder may only overhear the *comm* channel.

In this context it is straightforward to modify the definition of the intruder to reflect restricted powers:

– leaving the intruder's knowledge unchanged after a *take* gives the AOT semantics;
– barring communications on a subset of the message space on $fake$ can model a reliable one-way NS channel;
– disconnecting both *take* and the tap on *comm* on a subset of the message space captures the confidentially of NOH transmission.

In practice for various technical reasons we in fact replace the synchronous *comm* channel with two counterparts *scomm* and *rcomm* representing the sending and receiving of a direct, uninterrupted communication. These channels have the

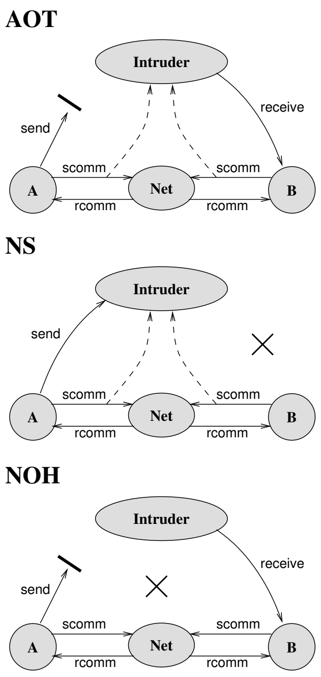

Fig. 1. Restrictions on the Intruder with a benign network process

same profiles as *send* and *receive* and are connected together with a new benign network process that adds some buffering and reordering of messages as in Figure 1. Finally, we modify the construction of the complete system process to ensure that agents and Intruder communicate over the new channels.

The next step is to define the set of network messages for each channel property *AOT*, *NS*, and *NOH*. As Casper provides names for the sets of input/output messages for each step of the protocol it is straightforward for the user to define a set-union for each property.

3 Example Protocols

In [3] we presented three protocols, addressing two different usage scenarios. Here we consider further the first of these scenarios, namely *The Wireless Printer Scenario*, the subject of the first two protocols there; and also present a variation on the first of these. This is the case of a user wishing to print a confidential document, residing on a PDA, on a public printer – imagined, for the sake of argument, to be located in an airport lounge. Communication between the printer and the user's PDA is via a wireless connection of some type (the precise communications technology is not relevant here). The user requires some assurance that the printer they are sending the document to is indeed the one they are looking at, as opposed to some other device elsewhere within communications range, and that that printer will be the only agent capable of successfully decyphering (in both the technical and colloquial senses of the word) the document.

3.1 Protocol 1.1 ($\mathtt{NS}_{\boldsymbol{E}}[+\mathtt{AOT}_{\boldsymbol{N}}?]$)

We begin by assuming that all suitable printers are manufactured with a generic public/secret key pair. This does not offer any particular additional security in the worst case, since we assume that an intruder may have access to a suitable printer, and possibly be able to subvert it; but it clearly makes life more difficult for an attacker in practice. There is clearly plenty of scope for research into desirable degrees of assurance and the related question of the cost-benefit trade-offs for the attacker, but we do not address that issue here. Throughout we also make the assumption that the user A has a unique key certificate, $kc(A)$; and that the printer is manufactured with knowledge only of the generic key pair associated with the class of printers to which it belongs, P.

Moreover we assume that it is fitted (in a reasonably tamper-proof way) with a light which flashes while (and only while) it is printing data that it itself is communicating as part of a protocol run. This effectively gives a no-spoofing assurance that the printer expelling the paper is the one generating the contents of the paper; we discuss below, when we come to the formal analysis (§3.1), the question of the printer itself being spoofed into doing so inappropriately. An alternative mechanism might be to use the LCD panel on the printer as a reliable medium between the printer and its user. Without some such facility, it would be hard to avoid the possibility of a suborned device-in-the-middle engaging in the protocol and simply using the intended printer as a slave to reprint what the protocol requires.

A wants to check that the printer she is communicating with over the N-channel is indeed the printer B that she can see, as in Figure 2.

More specifically, the security goal is to establish a shared secret known only to the user and that specific printer (which may then be used as a symmetric-encryption key for the document, for instance).

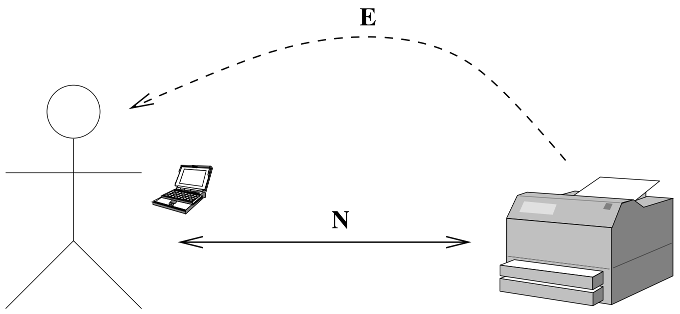

Fig. 2. A unspoofable channel E exists between the printer and the user; the network N gives a bidirectional link

Our first protocol depends on the channel E, running from the printer to the user, being impervious to spoofing, NS_E. The protocol proceeds as described below (the N or E subscript on the arrow denoting the channel over which the communication event occurs). If, however, at any stage before the protocol has completed either the user or the printer receives additional Messages 1 to 4 directed to them then the protocol will abort.

$$\begin{array}{ll}
1.\ A \rightarrow_N P(B) & : \{A, kc(A), N_A\}_{pk(P)} \\
2.\ B \rightarrow_E A & : \text{print } A, N_A \\
3.\ B \rightarrow_N A & : \{K, N_A, N_B\}_{pk(A)} \\
4.\ A \rightarrow_N B & : \{N_B, N'_A\}_K \\
5.\ B \rightarrow_E A & : \text{print } N'_A
\end{array}$$

Informally, the protocol proceeds as follows: In Message 1 A sends the printer her name, her key certificate and a random nonce, all encrypted with the generic printer public key. The printer then prints both A's name and the nonce, which A verifies empirically over channel E. At this stage, given that an attacker cannot spoof messages on channel E, A knows that the printer she is looking at has received Message 1 and is a printer, but she cannot exclude a corrupt printer-in-the-middle having overheard and understood Message 1. Assuming that the certificate $kc(A)$ is unforgeable, an honest B will only proceed to print a Message 2 acceptable to A if he received the intended Message 1 (from somebody).

In Message 3 the printer sends A a session key, K, the previous nonce N_A from Message 1, and a new nonce N_B, all encrypted using A's public key, extracted from the $kc(A)$ received in Message 1. A then sends a message to the printer which contains the second nonce N_B and a new nonce N'_A, encrypted using the key sent in Message 3. Since only A can have read the contents of Message 3 (it was encoded using her unique public key), it follows that only A and the device which sent Message 3 know the key K. However since an intruder knows N_A and may have blocked Message 3 from B and sent his own, A does not know that the Message 3 that she heard was actually from B.

The printer then prints the new nonce N'_A, which A verifies over channel E. Since this new nonce was sent in Message 4 and was encrypted using the key K in a message containing N_B, the printer would not print N'_A unless the Message 3 that A received really was from B. It follows that at this point A is certainly connected to B (the desired printer).

Note that the first printed output also serves to guard against spoofing on channel N, since if an attacker were to force B to abort after this point in the run, then the attacker could not get beyond this point without a further Message 2 being spotted, which A could see. This observation is somewhat application-specific, since it may not always be the case that the E-channel that B would use with an intruder would be observed by A. For this reason we additionally require that any entity in the role played by the printer here which aborts a run after Message 2 should send an E-message to A saying so.

Verification: As expected, under Dolev-Yao assumptions the protocol fails to meet any of its goals. For the basic system of one user and one printer, Casper finds an attack on secrecy in which the Intruder uses an honest printer to decode Message 1 before spoofing Messages 2 and 5 in order to get the user to accept *his* session key, K_I. Interestingly, without corrupt printers it takes a more complicated system (one in which the printer runs twice) for the Intruder to learn a real session key and for the user to claim it as secret, i.e. to think she has completed a correct protocol run.

For the authentication, Casper uses similar tricks to produce several attacks on the basic system of one user and one printer. One attack fakes Message 3 to provide the user with the Intruders own session key and nonce, then again spoofs Message 5 to convince the user that the session key originated from the intended printer.

When we consider the possibility of corrupt printers and users (by giving the Intruder appropriate secrets – namely the generic printer private key $sk(P)$), Casper finds many more attacks and clearly does not need an honest printer to learn the user's nonce N_A sent in Message 2.

Under the variant threat model with the special channel properties AOT_N and NS_E, Casper fails to find any attacks on the protocol even in the presence of corrupt users/printers, within the scope of two printers and two sequential runs.

This initial modelling (naturally) used the interpretation of the NS property as specified in [3], that is that a message received on a channel with this property must have been freshly sent by its apparent sender *to the recipient* (although it may be overheard, or indeed not received because blocked).

If, however, a weaker interpretation that omits the italicised phrase above is adopted, and so allows a message to be diverted (or in this case, the recipient to be deluded that a message physically directed to her was logically intended by the sender to be so), we may uncover the following "printer-in-the-middle" attack (assuming, perhaps unreasonably, that E-messages can also be suppressed):

$$\begin{array}{lll}
1. & A \rightarrow_N B(P) & : \{A, kc(A), N_A\}_{pk(P)} \text{ (overheard)} \\
2. & B \rightarrow_E A & : \text{print } A, N_A \text{ (overheard)} \\
1'. & I \rightarrow_N B(P) & : \{I, kc(I), N_A\}_{pk(P)} \text{ (abort 1st run)} \\
X. & B \rightarrow_E A & : \text{print } A, \text{abort (suppressed)} \\
2'. & B \rightarrow_E I & : \text{print } I, N_I \text{ (suppressed)} \\
3'. & B \rightarrow_N I & : \{K, N_A, N_B\}_{pk(I)} \\
3. & I(B) \rightarrow_N A & : \{K, N_A, N_B\}_{pk(A)} \\
4. & A \rightarrow_N I(B) & : \{N_B, N'_A\}_K \\
4'. & I \rightarrow_N B & : \{N_B, N'_A\}_K \\
5. & B \rightarrow_E A(I) & : \text{print } N'_A
\end{array}$$

To avoid this problem, Message 5 should (of course, by all rules-of-thumb for good protocol design) make explicit the identity A.

Within the range of instances of the scenario analysed, the property NS_E alone is sufficient to ensure the security of the protocol, and this seems likely to be the case in general; AOT_N appears superfluous. If, however, we allow the intruder to forge a key certificate for A (as would be the case if the association between identity and key is self-certified, as it is with PGP), then this exposes another printer-in-the-middle attack, which *is* ruled out by AOT_N.

$$\begin{array}{ll}
1.\ A \rightarrow_N I(P) & : \{A, kc(A), N_A\}_{pk(P)} \\
1'.\ I(A) \rightarrow_N B(P) & : \{A, kc_I(A), N_A\}_{pk(P)} \\
2.\ B \rightarrow_E A & : \text{print } A, N_A \\
3.\ B \rightarrow_N I(A) & : \{K, N_A, N_B\}_{pk_I(A)} \\
3'.\ I(B) \rightarrow_N A & : \{K, N_A, N_B\}_{pk(A)} \\
4.\ A \rightarrow_N I(B) & : \{N_B, N'_A\}_K \\
4'.\ I(A) \rightarrow_N B & : \{N_B, N'_A\}_K \\
5.\ B \rightarrow_E A & : \text{print } N'_A
\end{array}$$

3.2 Protocol 1.2 ($\mathtt{NS}_E$)

The above protocol requires two communications on the empirical channel. With the same physical scenario, this can be reduced to one at the expense, potentially, of making the task of the user harder (assuming that the user has a part in the implementation of this channel, as is the case in these examples). In the example above the user has to check that various pieces of data output on the printer are correct and appear while the printer is in "security mode". The main constituents of this data are the two nonces N_A and N'_A. Conventional nonces consisting of many characters of random data are not ideal for humans to check – and they might quickly get fed up and not do it properly! However there is the opportunity to use other types of values here, perhaps pictures, words and patterns, of a type chosen to map better onto humans' capabilities.

The following protocol requires only one communication on the empirical channel, but that is a cryptographic hash value. If a human was involved in implementing the empirical channel then we would have the following choices:

– Give the user a lot of tedious checking to do.
– Use relatively small hash values that humans can check conveniently[6].
– Devise some form of hashing into user-supplied pictures, words or patterns (or combinations of these) which enables a satisfactory range of values to be discriminated between in a satisfactory manner.

Of course the practicality of doing this, and whether any consequent greater risk is acceptable, would depend on the circumstances of the particular scenario. The protocol itself is considerably simplified:

[6] One can argue that the presence of a human in the loop makes this less dangerous than in many circumstances, since he or she is likely to smell a rat (rather than just resetting) if presented with an incorrect value.

$$\begin{array}{ll} 1.\ A \rightarrow_N B(P) & : \{A, pkA, N_A\}_{pk(P)} \\ 2.\ B \rightarrow_N A & : \{A, B, pkA, N_A, K\}_{pkA} \\ 3.\ B \rightarrow_E A & : \text{print } hash(Message\ 2) \end{array}$$

As in the first example, this is reliant on a no-spoofing empirical channel. In the above, N_A is a nonce, pkA is a public key chosen by A (which may or may not be the same for different sessions and other types of communication requiring public-key cryptography), and K is a session key chosen by the printer.

When A sees Message 3 (and that it agrees with the value that A has herself generated by hashing the Message 2 received), she can be sure that Message 2 really was from the printer she has the empirical channel with (namely the one she wishes to communicate with, and trusts). For the trustworthy printer B would not have sent this message (which she knows it has by non-spoofability) unless it had sent M2, which firmly ties its interest to A (as it contains A and pkA), and to this session (as it contains N_A). Furthermore A knows that B will only have sent K in Message 2, necessarily encrypted under pkA, and that therefore K is a secret shared between A and B.

Notice that if we are free to use hashing on a non-spoofable empirical channel, then we can effectively convert any full Dolev-Yao channel C in the same direction into a non-spoofable one by using the above trick. Namely each message along C is followed up by a hash of that message plus enough information to identify the recipient (if the latter is not already included). The message-then-hash order has the advantage that it bounds the time available for any intruder to look for hash collisions, unless the empirical channel is delayable. This, of course, is an advantage if the hash range (thanks to the human-factors argument above) is not as large as one would ideally like.

Here the explicitness in Message 2 is sufficient to establish the tie between A and B as being intentional, and the question of AOT_N and of the variability in the notion of NS_E does not impact the analysis.

3.3 Protocol 2 (NOH$_E$)

As a variant of this scenario, illustrated by Figure 3, we now assume that the printer has its own unique public/secret key pair, and that the E-channel communicates from the user to the printer (in the opposite direction to the first example), perhaps by discreetly pressing buttons on a front panel. If we consider a different threat model, namely that of no overhearing on channel E, we can achieve the same goal by means of a different protocol.

$$\begin{array}{ll} 1.\ A \rightarrow_N B(P) & : \{A, kc(A), N_A\}_{pk(P)} \\ 2.\ B \rightarrow_N A & : \{B, kc(B), N_A\}_{pk(A)} \\ 3.\ A \rightarrow_E B & : N'_A \\ 4.\ B \rightarrow_N A & : hash(N'_A, pk(A), kc(B), N_A) \end{array}$$

In Message 1 A sends a copy of her key certificate and identity, and a nonce, encoded with the generic printer key, to the printer. The printer then replies in Message 2 with a copy of its own unique key certificate and identity, and a copy

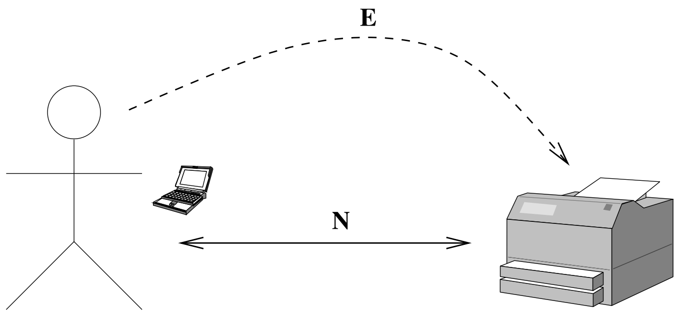

Fig. 3. Channel E now runs from user to printer, unobservable by the attacker. Channel N is bi-directional with the standard Dolev-Yao vulnerabilities

of the nonce it received from A in Message 1, all encoded with A's public key. At this stage A knows that someone has received the original message, but she cannot be sure that it is printer B.

In Message 3 A inputs a nonce N'_A into the printer directly, over the E channel. In this threat model we are assuming that this input cannot be overheard, and therefore only the printer physically interacted with can know N'_A. So in Message 4 the printer sends a hash of the nonce N'_A, A's public key, the printer's key certificate sent earlier in Message 2 and the first nonce, N_A, sent by A. As a result, A knows that the printer she is communicating with over N is also the printer being communicated with on E. This message certainly originated at the physically present hardware, since only one printer can know N'_A; the inclusion of evidence of the identities of both A and B in the hash is necessary to prevent an intruder acting as a man-in-the-middle for Messages 1 and 2, and persuading each that they are engaged in the protocol with a different principal. If, for example, we were to have encrypted Message 4 under $pk(A)$, this would have opened up a man-in-the-middle attack. Of course, it is essential in this protocol that E cannot be overheard, as otherwise an imposter who had been taking part in the rest of the protocol could forge Message 4.

Since N'_A is a shared secret here, it can be used as a session key; alternatively the protocol leaves each of A and B with knowledge of each other's public keys, so they can use these.

Verification: As expected, under Dolev-Yao assumptions the protocol again fails to meet its goal. However, to find an attack Casper requires corrupt printers, i.e. Intruder knowledge of the generic printer private key $sk(P)$. This is not surprising as only Message 3 uses a special channel, and the nonce N_A that must end up in the final hash originates from the encrypted Message 1 (where it is associated with the users identity). When we do allow for corrupt printers, the Intruder overhears the nonce N_A and has only to wait for the user to send Message 3 (N'_A) before faking the response in Message 4.

With the special channel property NOH_E, Casper failed to find any attacks on the protocol.

4 Conclusions and Further Work

- We have established the utility and viability of mechanical checking of protocols using empirical engagement to bootstrap authentication.
- The multi-party scenario from [3] does not fit so neatly into the Casper paradigm, and its verification will be the subject of a future report.
- Similarly, we plan a further paper on the population of the variant-threat lattice with plausible mechanisms and illustrative protocols.
- All the mechanisms considered so far use locality (and location-limited channels) to establish firm identification of parties to one another. We also intend to explore the possibility of using reduced-threat channels over a distance.

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. D. Dolev and A.C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2), 1983.
5. C.A.R. Hoare. *Communicating Sequential Processes.* Prentice Hall International, 1985.
6. Tim Kindberg and Kan Zhang. Validating and securing spontaneous associations between wireless devices. In 6th *Information Security Conference (ISC'03)*, number 2851 in LNCS. Springer-Verlag, October 2003.
7. A.W. Roscoe. *The Theory and Practice of Concurrency.* Prentice Hall, 1998. ISBN 0-13-6774409-5, pp. xv+565.
8. 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.
9. Frank Stajano and Ross Anderson. The resurrecting duckling: Security issues for ad-hoc wireless networks. In B. Christianson, B. Crispo, and M. Roe, editors, *Security Protocols, 7th International Workshop Proceedings*, pages 172–194. Springer LNCS, 1999.