The Attacker in Ubiquitous Computing Environments: Formalising the Threat Model*

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

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

Abstract. The ubiquitous computing paradigm will involve both the storage of increased amounts of valuable or sensitive digital data, and an increase in potential interfaces to networks and devices where such data is stored. Such information systems will be relied upon to provide adequate data security. It is imperative that techniques be developed to assure the trustworthiness of the technologies from which they are built. Formal verification provides the tools and techniques to assess whether systems are indeed trustworthy, and is an established approach for security assurance. When using formal verification for security assessment one of the most important concerns should be to be precise about the threat model. In the now extensive field of the formal analysis of crypto-protocols, the Dolev-Yao threat model is the *de facto* standard. This paper discusses how and when this threat model may be inappropriate for use in the ubiquitous computing environment and that it may be appropriate to assume the existence of multiple channels exposed to different threat models. We then present a selection of crypto-protocols for ubiquitous computing scenarios that require this heterogeneous approach.

1 Introduction

1.1 Ubiquitous computing environments

Future ubiquitous computing environments will consist of computational devices embedded in and pervading all parts of our environment. These devices will be capable of transmitting and receiving data, and providing varying degrees of computational processing power. Such huge numbers of communicating devices will provide and enable multiple dynamic networks at any one location. The

* This research is being 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

types of these networks will range from standard client-server architectures to dynamic ad-hoc peer-to-peer networks, including a spectrum of hybrids in between. In order to utilise future services users will be surrounded by intelligent and intuitive interfaces, some consciously controlled and others operating more-or-less autonomously, capable of providing information and communication facilities efficiently and effectively. These bespoke interfaces will be accessible from diverse geographic locations much like the Internet of today. The interaction of such bespoke interfaces with local dynamic networks will enable sophisticated ambient intelligence systems.

As ubiquitous computing technologies become common-place in our societies, companies, organisations and individuals will increasingly depend on them. It will be challenging to navigate through the copious amounts of information and services on offer. Various research initiatives such as the EU Framework 5 PAM-PAS project [10] and research funded by the U.K's Department of Trade and Industry T-SAS project [14], predict that autonomous computing agents will act and negotiate on users' behalf. This would further increase the range of critical and sensitive transactions.

The unobtrusive nature we expect from this new generation of computing, and its very ubiquity (almost unavoidable presence in every part of life), will make it easy for users to forget security requirements yet make these especially essential. This imposes an obligation on the engineers who create these systems to produce and satisfy appropriate security specifications, and to argue this persuasively to users and, hopefully, to standards authorities.

Whereas in some circumstances ubiquitous computing applications may behave like traditional ones and have readily available links to trusted third parties and a usable PKI, we believe that this will often not be the case. This might be through the use of these systems in isolated circumstances or simply because of problems with scalability, lack of resources, interoperability or lack of trust. In this paper we therefore concentrate on the more difficult case and see what can be achieved without a traditional PKI.

1.2 Security assurance and formalisation

The difficulty of building secure, trustworthy systems in this world of vastly more complex information relationships will be greater than ever. This highlights an increased need to be able to verify (or at least validate) that proposed designs and implementations are indeed sound. However, the increased complexity of ubiquitous computing environments will exacerbate the difficulty of achieving bolt-on information assurance. See [11] for a discussion on the difficulties of bolting security onto a system implementing certain fault-tolerance techniques.

Basic security properties such as authentication and secrecy are frequently achieved via key agreement protocols; these protocols are traditionally designed for an environment which differs from what we expect in ubiquitous computing in two vital respects:

1. Processes identify each other by long-term names, either because they have *a priori* knowledge of the partners they wish to do business with, or per-

haps in some cases because there is no better way to identify them. In a ubiquitous environment we are most unlikely to know the name of a process we encounter (the latter probably being a consequence of locality), and mass-produced processes may well not have names or unique public-key certificates. Even if we did discover a process's name, this would not in many cases be a basis for trust: see [6] for a discussion of attribute versus identity authentication.

2. Protocols often rely on the presence of trusted third parties, for example to judge the validity of partners in some public key infrastructure. This may not always be a realistic prospect as argued above.

Formal methods and formal assurance techniques provide the proven capability to verify systems, in order to support high assurance design, implementation or accreditation. A fundamental component of any formal analysis of security properties is to construct an appropriate threat model for the environment in which the system is designed to be working. If the threat model represents too weak an attacker then it may be that security can be proven in the formal model, but does not hold in the real implementation. Conversely, it may be that a stronger attacker than necessary is modelled, which could then place constraints on the implementation which could have been avoided.

In this paper the security service that we study is authenticated key agreement. Currently, the *de facto* standard threat model for analysing key agreement protocols is the Dolev-Yao model.

1.3 Is the Dolev-Yao threat model appropriate?

The bulk of classic crypto-protocol analysis is predicated upon an almost omnipotent attacker, limited only by (more-or-less perfect) cryptography. The Dolev-Yao model [7] supposes an intruder who effectively controls the communications network, and is therefore capable of overhearing messages between legitimate principals; of intercepting messages and preventing their delivery to their intended recipient; and – within the limitations of the cryptographic mechanisms involved – of synthesising 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.

This is clearly a worst-case scenario for a protocol to cope with, although it may unfortunately be realistic for the interned and similar environments. In any case, it is arguably best to err on the safe side: a protocol which exhibits no flaws under these assumptions will *a fortiori* be secure against a less potent attacker. But it is undoubtedly easier to come up with protocols which achieve their goals of confidentiality or authentication if one is reasonably able to assume a more restricted threat model: Broadfoot and Lowe [4], for example, have shown the security of transaction protocols relative to assumed properties of a secure transport layer. Equally, there may be intrinsic properties of the network under consideration which justify cutting back the Dolev-Yao capabilities.

The main thrust of this paper is to develop a more flexible threat model which allows us to handle ubiquitous applications better. We will demonstrate this with a number of examples and show that it helps to explain some existing literature.

1.4 Paper outline

In the rest of this paper we first provide a brief overview of the threat models assumed by some of the extant literature on ubiquitous computing security[1]. Then in Section 3 we present a range of restricted threat models and describe example use scenarios. In Section 4 we discuss how these revised threats may formally be modelled for automated verification. Finally, in Section 5 we present our conclusions and some ideas for future work.

2 A Brief Overview of Threat Models

A superficial survey of the rapidly growing literature in this area reveals that, from a formal perspective, the threat model is not well- specified. One reason for this might be that the extant work is dominated by approaches to *devising* security mechanisms and policies, while the problems of verification have, as yet, received little attention. Our brief look is based on three items:

1. Balfanz *et al.*'s work on authentication in ad-hoc networks, [2].
2. Bootstrapping security in mesh networks [1].
3. Proxy based security protocols [5].

The first two also provide added context for the presentation of our protocols; the three together provide a cursory sample of the subject. A commonality of all three is that it is difficult to distinguish the capabilities of the attacker (in other words what the attacker is able to do to attempt to compromise their mechanisms) from the constraints on the attacker (which are in effect the security properties achieved by their mechanism, against an attacker of uncertain capability). Below we attempt to outline the threat model utilised by each.

2.1 The Threat Model when Trusting Strangers

The problem addressed by Balfanz *et al.* in [2] is of a user wishing to print a confidential document, residing on a PDA, on a public printer – the printer may, for instance, be in an airport. Communication between the the printer and the user's PDA is via a wireless connection of some type (the exact communications protocol is not relevant here). The user requires some assurance that the printer

[1] In fact much of the work that we examine describes itself as security for ad-hoc networks, but, in essence, this is a part of the ubiquitous computing security field.

they are sending the document to is indeed the one they are looking at, as opposed to some other device elsewhere within communications range.[2]

Balfanz *et al.* eschew dependency on any trusted infrastructure; instead their proposal to secure the PDA to printer wireless link is based on the concept of a "location-limited channel", as found in 'The Resurrecting Duckling' of Stajano, [13]. Essentially, public key information is exchanged on physical contact between the PDA and the printer and then standard authenticated key exchange protocols secure the wireless printer to PDA link. No pre-existent authentication mechanisms, such as certificates, are needed, thus they are establishing trust between strangers. However, the location-limited channel requires users to touch their chosen printer with their PDA, on an appropriate physical interface (this being the location-limited channel).

They describe their solution as "secure against passive attacks on the privileged side-channel and *all attacks* on the wireless link". The emphasis is ours and we assume "the privileged side-channel" is the same as the location-limited channel. In summary:

- Attackers should not be able to transmit on location-limited channels, though this constraint is a little equivocal since they add "or at least to transmit without being detected".
- Attackers should not be able to eavesdrop on a location-limited channel, but it appears as if this is the security property achieved, against an apparently unspecified attacker.
- They further clarify that *active attacks* are where the attacker transmits, and they imply that *passive attacks* mean eavesdropping. Other types of attack, such as blocking communications, appear not to be considered.

The location-limited channels are the cornerstone of Balfanz *et al.*'s solution, but they are only the means to an end, namely a secure wireless link between the PDA and the printer. The paper's abstract makes the rather sweeping statement that this second secure channel will be secure against "all attacks", but it is not clear the domain over which *all* ranges.

2.2 The Threat Model when Bootstrapping Group Keys

For our second example imagine a set of people meeting in some place and wanting to work together securely. Of course, they will have their PDAs (laptops, or whatever) and so they will want these PDAs to form a 'secure' network[3]. Here

[2] One may not be entirely convinced of the plausibility of this scenario, or that there may not be more commercially attractive solutions, such as establishing a key by swiping the credit card to be charged. However, it is illustrative of a class of authentication and key-negotiation problems which is likely to become more important as ubiquity strikes.

[3] This is an example of promotion of human trust, to extend it to cover their electronic representatives.

secure may be taken to mean that a group key exists and is known only to the PDAs owned by the people holding the meeting.

This problem is discussed by Asokan and Ginzboorg [1]; their solution is inspired by the EKE protocol of Bellovin and Merritt [3], which uses weak password-based encryption to agree a strong encryption key. Asokan and Ginzboorg's work also extends this concept from point-to-point key agreement to group key agreement. The basic idea is that the users agree a password and then they type that password into their PDAs. This password makes the weak encryption key, which is, nevertheless, sufficient for the legitimate PDAs to agree a strong key - using the protocols that they present. Thus they provide a solution for bootstrapping security that requires no pre-existent electronic trust. In effect they have *manual initialisation* of trust, since the users are (implicitly) responsible for controlling the exposure of the password.

The well-known "Alice-Bob" notation of the crypto-protocol community is utilised, which usually implies the standard Dolev-Yao threat model, however, this appears not to be the case. The threat model is implicitly included in their discussion security requirements. The security properties the protocols are required to uphold include:

- Contributory key agreement.
- Tolerance to disruption attempts.

Contributory key agreement means that malicious principals cannot limit the size of the key space and so enable cryptanalysis attacks. As such it implies a cryptanalysis enabled attacker, which differs from a pure Dolev-Yao attacker.

The meaning of "tolerance" in "Tolerance to disruption attempts" is not clear, we assume it means that "disruption attempts" cannot violate the other security properties. In explaining what "disruption attempts" mean the authors state that:

> *The strongest attacker can disrupt any protocol by jamming the radio channel or modifying communication among legitimate participants. However, a slightly weaker attacker who can insert messages but cannot modify or delete messages sent by others is also of interest in this scenario. We will consider disruption tolerance against this type of disruption attacks only.*

The strongest attacker sounds similar to a Dolev-Yao attacker, but the weaker attacker is more limited. Thus the proposed attacker appears to be a hybrid of something weaker than Dolev-Yao, but with cryptanalysis capabilities.

2.3 The Threat Model in Proxy-Based Security Protocols

In the ubiquitous computing future many devices will have relatively limited computing resources. To overcome this limitation it is often touted that computationally limited devices should be endowed with virtual proxies. These virtual

proxies will have access to high-bandwidth communications and plentiful computing resources. In [5] Burnside *et al.* propose a security architecture for virtual proxies[4]. Security concerns in systems of this sort will of course be very many and very complex, their paper concentrates on presenting two security protocols:

- A protocol for device-to-proxy secure communication.
- A protocol for proxy-to-proxy secure communication.

We assume (though it is not stated explicitly) that "security" means that the communication is protected by *confidential* and *authenticated* key agreement[5]. Unfortunately, the paper does not discuss the device-to-proxy *protocol*. Rather the assumption is stated that "the proxy and device share 128-bit keys". The protocol for initialising such shared keys would perhaps have been of most interest to us. For this protocol the paper only discusses the messaging hashing and encryption techniques used, which imply that only a cryptanalysis attacker is being considered, though this is not stated.

The proxy-to-proxy protocol, is described as a "typical challenge-response protocol", which, to a formalist, might imply a Dolev-Yao attacker. However, the only explicit discussion of "the adversary" is as part of a brief discussion of how time-stamps are used to protect against replays. The state- of-affairs is made a little more confusing when the authors make clear that the protocol does *not* provide a range of security services or defences against attacks "from the network". The authors then discuss how security services can be layered, and indicate that added security may be achieved by layering their services on protocols such as SSL/TLS. Thus, if formal verification were to be applied to their architecture, then an approach such as that developed in [4] might apply.

2.4 The dual-interface assumption

The threat models presented in Sections 2.1 and 2.2 above utilise properties of a dual interface to provide security. For example, both require a channel involving physical contact or human interaction for *part* of their protocols (while "bootstrapping" a relationship, for instance). Such a dual interface seems to be a key to this genre of problem[6].

In the ubiquitous arena, where most communications are through the essentially broadcast medium of wireless, it may sometimes be sensible to restrict the powers of the attacker: in particular, while a hostile agent can undoubtedly overhear any message or perhaps, by some form of jamming, prevent its delivery, it is arguably unreasonable to suppose that it can do both at the same time. (This is only reasonable when we know the intended recipient is active and is

[4] In fact their work concerns using proxies for resources discovery.
[5] Interestingly, by authentication they implicitly mean *attribute authentication*, much as espoused by [6].
[6] Unfortunately a lack of information in [5] means that we are unable to tell whether the same dual interface would be required or not.

within range.) Thus we gain the advantages of some kind of atomicity in transmission: if anyone receives a given message, then the intended recipient (also) does. Note that this argument holds good only for direct, unmediated, radio communications: if the message is relayed by a router node, for instance, then it would be open to a nearby attacker to overhear the first leg of its journey and to interfere with its second. In this case it might be possible to design acknowledgement schemes which effectively restore atomicity, but these in turn would require careful verification.

Another variant might be appropriate to the airport-printer problem [2, 6], discussed in Section 2.1 : with suitable tamper-proof hardware (and a flashing light to show "secure" printing) a printer might be able to achieve a one-way channel that was secure against messages being faked on it (and, perhaps less crucially, against overhearing, in the absence of cameras focused on its output tray). It is probable that in many cases additional security may be obtained at higher cost: be that in monetary terms, lower bandwidth, or reduced ease of use.

The reliance on a duality of channels, where one can be relied upon more than the other to provide certain security charateristics, does not appear to be unreasonable. However, it goes without saying that where a weaker threat model is assumed, careful examination of an implementation is required to ensure the model accurately captures the danger. What follows is a formalisation of three different variants of such systems.

3 Variant Threat Models

3.1 Twin-Channel Threat-Model Variants

In our examples below we presume the existence of two communications channels: N and E. The first channel, N, is the 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 channel, E, represents the more costly channel with higher security, which may according to the details of the scenario be either uni-directional or bi-directional. E typically represents an *empirical* channel, such as reading a message on the printer display panel, physically inputing a code via the printer control panel, or checking that the flashing light is present as in the example above. We will generally want to minimise the use of channel E and seek to delay it until relatively late in a protocol since it is likely to involve the human user in some effort.

By varying how the attacker can manipulate these two channels (the attacker's capabilities on E being the most limited), a variety of hybrid threat models can be created. We will express threat model variants using the following notation, 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*.

Combined restrictions can be represented with a hyphen, such as: AOT_{C1}-NS_{C2}, which means the attacker cannot both block and hear messages at the same time on channel $C1$ nor can it spoof messages on channel $C2$. The channels will of course be either N or E. If no restriction is specified for a channel, then that means the standard Dolev-Yao model applies.

The following two scenarios will be used to demonstrate the usefulness of these variant threat models. We have considered the complete matrix indexed by the assumed properties of each channel and the direction of E, and we have found credible protocols for the majority of the different configurations. Due to space limitations, however, we present only three distinct models here; a further paper will set out our exploration of the broader space.

3.2 Scenario 1 - The wireless printer

We begin by returning to the airport-printer problem of 2.1. We will assume that all suitable printers are manufactured with a generic public/secret key pair. In addition we will consider the case where the printer possesses its own unique public/secret key pair. What is required is an authentication protocol which identifies the desired device as being the one on the receiving end of the communication channel, and the only one which can read the sent data. In addition one might want to verify some behavioural attributes of the device, but we will not consider that question here beyond checking the possession of the key belonging to the appropriate genus of printer.

NS_E-AOT_N Here we 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. 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 she can see, as in Figure 1.
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).

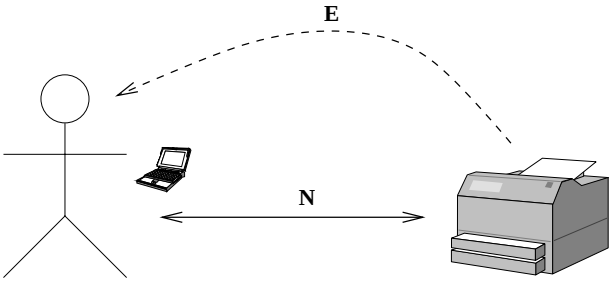

Fig. 1. A unspoofable channel E exists between the printer and the user; the network N gives a bidirectional, atomic link.

Our first protocol uses the fact that the attacker cannot both block and receive network messages at the same time, thus AOT_N. It also 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 denotes 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.

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

In Message 1 A sends the printer a message containing 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. We can, however, guarantee that the printer heard Message 1 as sent by A, thanks to AOT_N; and that an attacker has not sent another Message 1 because of our assumption about B aborting. Therefore, the $kc(A)$ which B holds was indeed just sent by A.

In Message 3 the printer sends A a message containing 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 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 a "printer" aborting a run after Message 2 should send an E-message to A saying so.

If we were to assume a conventional Dolev-Yao model for the high bandwidth medium then the above protocol would not be so secure since a printer-in-the-middle could act as an intermediary, sending A's values for N_A and N'_A on to the real B for printing. The point where he might have problems doing this is in understanding Message 3 if it really was A's real public key certificate that was sent to B in message 1. However in the absence of a usable PKI this may not be true, and it may be possible to send B a public key "for A" that he has made up.[7]

We do have protocols which we believe work with the same model for E but with full Dolev-Yao on the high bandwidth channel. They differ in having B print out hashes rather than literal values sent by A. The latter might be a disadvantage where a human is expected to check equality. These will be discussed in a later paper.

NOH $_{\boldsymbol{E}}$ As a variant of this scenario, illustrated by Figure 2, let us 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.

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

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

[7] In this case the concept of sending a public key *certificate* as opposed to just sending a public key may have limited value.

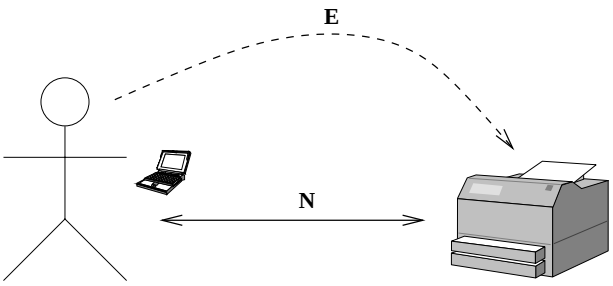

Fig. 2. Channel E now runs from user to printer, unobservable by the attacker. Channel N is bi-directional with the standard Dolev-Yao vulnerabilities.

Message 2 with a copy of its own unique key certificate and identity, and a copy 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 others' public keys so they can use these.

3.3 Scenario 2 - The PDA Mesh

In this example we use our hybrid threat model approach revisits the problem from Section 2.2, where a group of people want to agree a key across their electronic representatives (PDAs, or whatever). In our solution the E channel is used to agree a hash value, which will appear on the users' screens and which they can compare.

$\boldsymbol{NS_E}$ Here we again restrict the attacker on the E channel to NS, no spoofing. We will consider the example where each user possesses a unique key certificate pair, and a generic manufacturer PDA certificate. Each message is described as being sent from each user A to all other users: the first can be broadcast to *Every* B, but the rest are sent in turn to *Each* B that a Message 1 was received from:

1.Each A $A \rightarrow_N$ Every B : $\{A, kc(A), N_A\}_{pk(PDA)}$
2.Each A $A \rightarrow_N$ Each B : $\{$all Messages 1, $N'_A\}_{pk(B)}$
3a.Each A A displays : $hash(\{$all Messages 2$\})$, number of processes present
3b.Each A $A \rightarrow_E$ Each B : Users compare hashes and numbers on screens
4.Each A $A \rightarrow_N$ Each B : $hash'(\{$all Messages 2$\})$

We assume that the boundary between Message 1 and Message 2 is determined by some time-out. Tit is not necessary to include this in the security analysis since any failure to synchronise properly will only have the effect of causing the eventual hash values to differ, meaning that no node believes the protocol has completed successfully.

In Message 2 a given A knows that it has just sent nonce N'_A to a given set of identities, using the keys it received in Message 1. One purpose of Message 2 is to allow disagreements about the Messages 1's, in the most part, to be caught by the PDA's without troubling their users. When all the users agree on the hash of the values transmitted in Message 2, which they check on channel E in Message 3, and the number of participants displayed corresponds to their expectations, they know that there cannot be any additional devices that they are unaware of involved in the protocol. The number shows each of them that they are in a network of the right size, and the hash shows them they are in the same network. It follows that there can be no unwelcome participants. The nonces are present to ensure freshness and prevent replays of messages in previous sessions. Message 4 acts as a final message to close the protocol and confirm possession of the shared secret; this value may also be used as a session key. We have to use a different hash function because the previous value has been displayed, and in any case it reduces the probability of hash collisions.

4 Verification in the Presence of Restricted Attackers

One of the main challenges to using process-algebraic and model-checking approaches to verify (or, more strictly, to look for counterexamples to) the security of crypto-protocols was to find an effective way of modelling the attacker and his ability to recombine elements of the network traffic to date in decrypting incoming messages and synthesising messages to trick his targets. The solution devised for FDR and incorporated in Casper [9, 12] proved to be an enabling technology, and similar schemes have been used in many subsequent model-checking and related approaches, such as strand spaces [8].

Casper exploits the fact that the intruder intercepting a message and then "faking" it (unchanged) to its intended recipient as if from the sender amounts to much the same as eavesdropping on its uninterrupted transmission. This means

that all messages are in fact mediated by the intruder [12]. This simplifies some specification idioms, but makes it harder to implement the weaker attackers we are considering.

In the original concept [9], however, there were separate channels: *comm* to represent direct communication, and *take* and $fake$ to represent interception and introduction by the intruder, respectively. Renaming is used in the CSP model so that sending on *comm* and *take*, and receiving on *comm* and $fake$, are indistinguishable to the principals. 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.

Thus there is no difficulty to the mechanical checking of these weakened attackers, at least in the CSP/FDR/Casper paradigm. (Proponents of other technologies must answer for themselves! It would be interesting to learn whether the encoding of the attacker in alternative approaches is sufficiently flexible to cope with this kind of variation in its powers.)

5 Conclusions and Further Work

The problem of formalising the threat model is fundamental to the formal analysis of security. But a superficial look at the literature on security for pervasive computing shows that it appears to be informally (and perhaps a little loosely) specified. In our own examination of the problem, for authenticated key-agreement protocols, we found that the standard Dolev-Yao model was arguably unrealistic, and yielded a dearth of protocols which are simultaneously both secure and useful, in the absence of the assumptions and support infrastructure postulated for more traditional environments. This is consistent with the need that other researchers have felt to invent mechanisms outside the normal communications structure, to bootstrap trust in some way.

Among the redeeming features of the ubiquitous world are that there will often be physical proximity, which can sometimes be exploited to allow contact-based communication, and that the unique data-processing capabilities of human agents can also be brought into the equation as a last resort. The properties that such mechanisms win for us appear to be reasonably straightforward to capture and model. We believe that developing an understanding of the power of the full range of variants (and intuition as to how they may be implemented) and of their interactions will help in designing techniques for bootstrapping security in a world where no backbone infrastructure is required (or available).

Further work is needed in crystallising the notions introduced in this paper, and a subsequent paper is planned that will populate the matrix of attacker capabilities with appropriate countermeasures, and explore the ideas in this paper

more deeply. A number of interesting technical issues regarding the desirability of including the various potential component data in messages or hashes have arisen in the course of our initial investigations, and these aspects, too, merit further exploration.

6 Acknowledgements

The authors would like to thank give special thanks to Gavin Lowe for stimulating discussions.

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. Steven M. Bellovin and Michael Merritt. Encrypted key exchange: Password-based protocols secure against dictionary attacks. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 72–84. IEEE Press, May 1992.
4. Philippa Broadfoot and Gavin Lowe. On Distributed Security Transactions that use Secure Transport Protocols. In *Proceedings of 16th IEEE Computer Security Foundations Workshop*, Monterey, CA, June–July 2003.
5. M. Burnside, D. Clarke, T. Mills, A. Maywah, S. Devadas, and R. Rivest. Proxy-based security protocols in networked mobile devices, 2002.
6. 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.
7. D. Dolev and A.C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2), 1983.
8. F. Javier Thayer Fabrega, Jonathan C. Herzog, and Joshua D. Guttman. Strand spaces: Proving security protocols correct. *Journal of Computer Security*, 7:191–230, 1999.
9. Michael Goldsmith and Bill Roscoe. The perfect 'spy' for model-checking cryptoprotocols. In *Proceedings of DIMACS Workshop on Design and Formal Verification of Cryptographic Protocols*, Rutgers, 1997.
10. http://www.pampas.eu.org/.
11. Jan Peleska. Formal Methods and the Development of Dependable Systems. Technical Report 9601, University of Bremen, 1996. Project UniForM, www.informatik.uni-bremen.de/agbs/jp/papers/depend.html.
12. 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.
13. 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.
14. http://www.gpc.ecs.soton.ac.uk/research/TSAS.html.