Deriving Ephemeral Authentication Using Channel Axioms

Dusko Pavlovic[1]* and Catherine Meadows[2]

[1] University of Oxford, Department of Computer Science, and
Universiteit Twente, EWI/DIES, dusko@cs.ox.ac.uk
[2] Naval Research Laboratory, Code 5543, Washington, DC 20375
catherine.meadows@nrl.navy.mil

Abstract. As computing and computer networks become more and more intertwined with our daily lives, the need to develop flexible and on-the-fly methods for authenticating people and their devices to each other has become increasingly pressing. Traditional methods for providing authentication have relied on very weak assumptions about communication channels, and very strong assumptions about secrecy and the availability of trusted authorities. The resulting protocols rely on infrastructures such as shared secrets and public key hierarchies that are too rigid to support the type of flexible ad-hoc communication we are growing accustomed to and beginning to rely upon.

Recently, different families of protocols allow us to weaken assumptions about trusted infrastructure by strengthening the assumptions about communication channels. Examples include proximity verification protocols, that rely, for example, on the round trip time of a challenge and response; and bootstrapping protocols that rely upon human-verifiable channels, that is, low-bandwidth communication between humans. The problem now becomes: *How do we ensure that the protocols are achieve their security goals?* A vast amount of literature exists on the formal analysis of cryptographic protocols, and mathematical foundations of protocol correctness, but almost all of it relies upon the standard assumptions about the channels in end-to-end, and so its usefulness for nonstandard channels in pervasive networks is limited. In this paper, we present some initial results of an effort towards a formalizing the reasoning about the security of protocols over nonstandard channels.

1 Introduction

Pervasive computing has become a reality. We have long been used to the idea that computers are everywhere, and that we interact with multiple devices that can interact with each other and with the Internet. But there is another important aspect of pervasive computing. Not only has the concept of a computer and a computer network changed, but the notion of a communication channel is changing as well. Wireless channels, of course, have been a common part of computer networks for some time. Quantum channels are appearing on the horizon. But what is really interesting is the way the nature

* Supported by ONR. Current address: University of London, Royal Holloway, Department of Mathematics/ISG.

of the information sent along these channels is changing. Information is no longer restricted to input typed in by users, but includes environmental information gathered by the network itself, including location, biometric information, and weather and motion data picked up by sensors.

These new concepts of channels have also resulted in new methods for authentication. The old mantra of "who you are, what you know, and what you have," has been extended to include concepts such as "where you are" (verification of location and/or proximity), "what you are" (use of techniques such as CAPTCHAs to verify that the entity on the other end is a human being [2]) and "what you see" (use of human-verifiable channels to boot strap secure communication, as in [20, 29]).

An important thing to note is that these new methods of authentication do not exist on their own. They are typically integrated with more traditional authentication and key exchange protocols that use more conventional channels. This is partly because the new channels may have particular properties that make them less practical to use than conventional channels except when absolutely necessary. Human-verifiable channels are limited in bandwidth. Channels used to implement proximity verification and CAPTCHAs rely on strict timing properties. And even when the new channels do not have these limitations, it will be necessary to integrate them with standard channels so they can be used to interface with traditional systems.

When integrating specialized channels with traditional channels for authentication, one is usually faced with a number of choices. One needs to choose when to use the specialized channel, what information to send on the specialized channel, and what information to send on the conventional channel. Different choices can have different effects on the security, applicability, and efficiency of an authentication protocol.

In this paper we introduce a system for reasoning about authentication using multiple types of channels with different types of properties. The system consists consists of two parts. The first is a graphical language for displaying cryptographic protocols that use different types of channels. This is based closely on the usual graphical methods for representing secure protocols. The second is a logic for reasoning about the security of authentication protocols that use different types of channels. This logic is an extension of the Protocol Derivation logic described in [6, 22]. Both language and logic are intended to be used to reason about, not only individual protocols, but families of protocols, in order to help us identify and reason about tradeoffs.

Outline of the paper. In Section 2 we discuss the problem of modeling pervasive security protocols. In Section 3 we describe the introduction of channel axioms into the Protocol Derivation Logic. In Section 4 we introduce the problem of distance and proximity bounding, provide axioms for channels, and discuss how it they can be reasoned about in PDL. In Section 5 we provide a similar discussion of human-verifiable authentication via social channels. In Section 6 we conclude the paper and discuss plans for future work.

2 Modeling pervasive security protocols

A protocol is a distributed computational process, given with a set of desired runs, or the properties that the desired runs should satisfy. To prove security of a protocol we usually demonstrate that only the desired runs are possible, or that the undesired runs can be detected through participants' local observations.

Security protocols are thus naturally modeled formally within a process calculus, as in [22]. In order to model security protocols in pervasive networks, we extend the process model from [22], used for analyzing security protocols in cyber networks. The main complication is that in this previous work, as in most work in the analysis of security protocols, the network itself was kept implicit in the process model, because every two nodes can be assumed to be linked, without loss of generality. The network infrastructure provides for that. More precisely, the network infrastructure provides the service of routing and relaying the messages, and hides the actual routes and relays (unless they change the messages, in which case they are considered to be attackers). From user's point of view, it looks like the messages are delivered directly from the sender to the receiver, and the network infrastructure is abstracted away.

In a pervasive network, the assumption that there is a link between every two nodes is not justified: some devices have a range, some have access to one type of channels, some to other type of channels, and they may not have any direct, or even indirect links to connect them. To express this, we must make the network explicit.

Moreover, the toolkit of security primitives and security tokens, available to establish secure communication, is essentially richer in pervasive networks.

2.1 Principals and security tokens

Principals are the computational agents that control one or more network nodes, where they can send and receive messages. A principal can only observe (and use in his reasoning) the events that happen at his own network nodes.[3]

Security tokens are the data used by the principals to realize secure communication. Informally, security tokens are usually divided in three groups:

- something you know: digital keys, passwords, and other secrets[4]
- something you have: physical keys and locks, smart cards, tamper-resistant devices, or
- something you are: biometric properties, e.g. fingerprints, or written signatures, assumed to be unforgeable

The difference between these three types of security tokens lies in the extent to which they can be shared with others:

- what you know can be copied and sent to others,

[3] We shall model his observations of the actions of others as messages of special kind, i.e. received over the *social channels*.

[4] i.e., data distributed to some principals and not to the others; data known to all are not very useful as security tokens

- what you have cannot be copied in general, but can be given away, whereas
- who you are cannot be copied, or given away.

Standard end-to-end security is in principle realized entirely by means of cryptographic software, and the principals only use the various kinds of secrets. This means that *a principal can be identified with the list of secrets that she knows*. If Alice and Bob share all their secrets, then there is no way to distinguish them by the challenges that can be issued on a standard network[5]. For all purposes, they must be considered as the same principal.

In pervasive networks, on the other hand, security is also supported by cryptographic hardware: besides the secrets, a principal is also supplied with some *security devices*. They are represented as some of the network nodes, given to the principals to control. A dishonest principal (or an honest certificate authority) can relinquish control of a security device, and give it to another principal.

To capture the third and the strongest kind of security tokens, and distinguish the principals by who they are, we need some *biometric devices*. They are represented as network nodes. Principals' biometric properties, on the other hand, are represented as some of the network nodes as well, available to respond to the challenges from the biometric devices. The only difference of a biometric property p from the other network nodes given to a principal A to control is that p always remains under A's control, and cannot be given away to another principal. We call the networks equipped with biometric devices and biometric properties — biometric networks.

2.2 Modeling networks

In modeling security, principals can be identified with their security tokens, since security tokens are the material that security is built from. Summarizing the preceding section, we can say that

- in end-to-end networks (or cyber-networks), the only security tokens are the secrets, and the principals are reduced to *what they know*;
- in pervasive networks, the security tokens also include some security devices, and the principals are identified not just by what they know, but also by *what they have*;
- in biometric networks, the security tokens furthermore include some biometric properties, and the corresponding biometric devices, needed to test them; the principals are identified not just by what they know, or what they have, but now we take into account *who they are*.

Communication networks. A *communication network* consists of

network graph $\mathcal{N}$, consisting of a set of nodes N, a set of links L, and a source-target assignment $\langle \delta, \rho \rangle : L \longrightarrow N \times N$, inducing the matrix representation $\mathcal{N} = (\mathcal{N}_{mn})_{N\times N}$ with the entries $\mathcal{N}_{mn} = \langle \delta, \rho \rangle^{-1}(m,n)$ for $m,n \in N$,

[5] We assume that they share the secrets dynamically: if a new one is sent to one, it will immediately be shared with the other. This implies that they also observe the events on the same set of network nodes.

channel types $\mathcal{C}$, and the type assignment $\theta : L \longrightarrow \mathcal{C}$,
set of principals (or agents) $\mathcal{A}$, partially ordered by the subprincipal relation $\leqslant$,[6].
control $© : \mathcal{A} \longrightarrow \wp N$, such that (1), and often also (2) is satisfied:

$$A \leqslant B \Longrightarrow ©A \subseteq ©B \tag{1}$$

$$A \nleqslant B \wedge A \ngeqslant B \Longrightarrow ©A \cap ©B = \emptyset \tag{2}$$

Remark. In a cyber network, the end-to-end assumption, that all security is done at the ”ends” and any route ”in-between” is as good as any other route implies that the network service can be reduced to an assumption that there is a single link between every two nodes, i.e. $\mathcal{N}_{mn} = 1$ for all m and n. Moreover, $\mathcal{C} = 1$, i.e. all channels are of the same type, insecure. So the only nontrivial part of the structure is $© : \mathcal{A} \longrightarrow \wp N$. But controlling one network node or controlling another one makes no difference, because a message can always be sent from everywhere to everywhere. So the only part of the above definition visible in the process model needed for cyber security is the poset $\mathcal{A}$.

Cyber networks: principals are what they know. The fact that the principals can be identified with the lists of secrets that they know is represented by an inclusion $\Gamma : \mathcal{A} \hookrightarrow \mathcal{T}^*$, which we call *environment*. However, since a principal may learn new secrets when a process is run (or during a protocol execution), her environment may grow: at each state σ, she may have a different environment $\Gamma_\sigma A$ such that for every transition $\sigma_1 \longrightarrow \sigma_2$ holds $\Gamma_{\sigma_1} A \subseteq \Gamma_{\sigma_2} A$. During a protocol execution, different principals may thus become indistinguishable if they learn each other’s secrets, since $\Gamma A = \Gamma B \Rightarrow A = B$. This means that the set of principals $\mathcal{A}$ may also vary from state to state in the execution: there is a family $\mathcal{A}_\sigma$, with the surjections $\mathcal{A}_{\sigma_1} \twoheadrightarrow \mathcal{A}_{\sigma_2}$ for every transition $\sigma_1 \longrightarrow \sigma_2$, induced by identifying the principals that become indistinguishable.

In the cyber network model, a principal may have a number of internal actions involving creating nonces, incrementing counters, etc. But the principal has only three types of external actions: send, receive, and match. In the last, the principal matches received data with what he or she is expecting. In some models, receive and match are identified.

Pervasive networks: principals are what they have. A *pervasive network* is obtained by distinguishing, within a cyber network as defined above, a set of mobile nodes (i.e. security devices) $\widetilde{N}$, from the fixed nodes $\overline{N}$, so that $N = \widetilde{N} + \overline{N}$.

Besides the send, receive, and match actions, the process calculus now has two new kinds of actions, which allow each principal to:

- move a mobile node under his control, and reconnect it elsewhere in the network;
- pass control of a mobile node to another principal.

This means that the network connections and controls of the mobile nodes can dynamically change during a process run.

[6] Briefly, A is a *subprincipal* of B is A “speaks for” B in the sense of [1, 14] or “acts for” B in the sense of [19]

Biometric networks: principals are what they are. A *biometric network* is obtained by distinguishing, among the nodes of a pervasive network as defined above, two more sets

- $B_r \subseteq \widetilde{N}$ of *biometric properties*, and
- $B_c \subseteq N$ of *biometric verifiers*.

The intended interpretation of these two sets of nodes is implemented by ther requirement that:

- control of the elements of B_r cannot be passed to another principal,
- the elements of B_c are related with the elements of B_r, so that the former can issue biometric challenges to the latter.

2.3 Message delivery modes

The main source of the new security phenomena in pervasive network is the fact that different types of channels have different message delivery modes.

In cyber networks, a message is usually in the form A to $B : m$, where A is the claimed sender, B the purported receiver, and m the message payload. As explained before, the network service is implicit in this model, so that A and B refer both to the principals and to the network nodes that they control. All three message fields can be read, intercepted, and substituted by the attacker. The point of the end-to-end security is that the receiver can still extract some assurances, even from a spoofable message, because the various cryptographic forms of m limit attacker's capabilities. Moreover, this message form is an abstract presentation of the fact that the message delivery service provided by the network and the transportation layers, say of the Internet.

In pervasive networks, different channel types provide different message delivery services. In general, there is no universal name or address space, listing all nodes. Annotating all messages by sender's and receiver's identities thus makes no sense, and the principal's identities are added to the payload when that information is needed.

There may be no link between two nodes, and no way to send a message from one to the other. On the other hand, a message can be delivered directly, e.g. when a smart card is inserted into a reader, without either of the principals controlling the card and the reader knowing each other.

The different message delivery modes determine the different security guarantees of the various channel types.

3 Templates and Logics

3.1 Templates

In this section we give an introduction to the use of templates and logics.

A *template* is a graphical specification of the desired behavior of a protocol that can be filled in with a number of actual protocol specifications. A template begins by describing the different types of channels available between principals. This is done simply by drawing a line indicating a channel between two principals that share the channel.

Different types of lines indicate different types of channels. Messages passed between principals along a channel are indicated by arrows between principals corresponding to the channel type. Internal transitions are indicated by arrows from a principal to itself.

For example, the following template gives a common situation in which Alice generates a nonce (νx), and sends a cryptographic challenge $c^{AB}x$ containing x to Bob, after which Bob sends a response $r^{AB}x$ to Alice. Note at this point we give no details about the operations c^{AB} and r^{AB}.

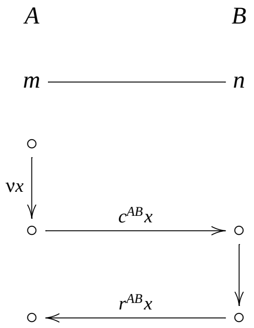

Fig. 1: Challenge-Response Template

What we would like to say, of course, is if that Alice creates x and sends $c^{AB}x$ to Bob, and subsequently receives $r^{AB}x$, she knows that Bob sent $r^{AB}x$ after receiving $c^{AB}x$. The Protocol Derivation Logic we describe below will give us a way of stating and proving this requirement.

3.2 Protocol Derivation Logic (PDL)

PDL Syntax. PDL is a descendant of an early version of the Compositional Protocol Logic (CPL) [9] and has certain of its axioms in common with it. Like CPL, it is intended to be used to prove security of protocols without explicitly specifying the behavior of the attacker. Unlike CPL, it is a logic about authentication only, although it can be interfaced with a companion secrecy logic [22] when it is necessary to reason about secrecy. In PDL principals are partially ordered sets where $A \subset B$ (A is a *subprincipal* of B) if A "speaks for" B in the sense of [1, 14] or "acts for" B in the sense of [19]. The logic makes use of cryptographic functions that only certain principals can compute; thus keys do not need to appear explicitly unless we are reasoning about key distribution.

In PDL principals exchange messages constructed using a term algebra consisting of constants, variables, and function symbols. The term algebra may obey an equational theory E, or it may be a free algebra. The constants and variables used in the term algebra may or may not obey a type system which is specified by the protocol writer.

We consider a protocol as a partially ordered set of actions, as in Lamport [13], in which $a < b$ means that action a occurs before action b. We let $(t)_A$ denote t being received by A, $\langle t \rangle_A$ denote a message being received by A. We let $\langle t \rangle_{A<}$ where $t = f(x_1, \cdots, x_n)$ denote A creating t by applying the function f to the arguments $x_1, \cdots, x_n$ and then sending it ; thus this is the first time A sends t. We let $x \prec y$ denote the statement "if an action of the form y occurs, then an action of the form x must have occurred previously." We let $\nu.\ n$ denote the generation of a fresh, unpredictable nonce, n, and $(\mu.\ n)_A$ denote the generation of some arbitrary term n that A has never generated before. We think of ν and μ as acting a binders and write them as such. Finally, we let $A : S$ denote A knows S, and $\mathsf{H}A$ to denote that fact that A is an honest principal following the rules of the protocol.

We let $\langle\langle s \rangle\rangle_A$ denote A sending a message that was computed using s. We let $((s))_A$ denote A receiving a message that was computed using s. Finally, we let $\langle\langle s \rangle\rangle_{A_\langle}$ denote A's computing s for the first time (that is, the first time for A) and sending it in a message.

We use certain syntatic subterm conventions to determine if a term was used to compute a message. Suppose that $\langle m \rangle_A$ or $(m)_A$is an event. We use the convention that if s occurs as a subterm of m then s could have been used to compute m. We conclude that s *must* have been used to compute m is for all legal substitutions σ to the variables in m, and for all $y =_E \sigma m$, s appears as a subterm of y. [7]

We define a legal substitution as follows:

Definition 3.1. *Let $\mathcal{P}$ be a protocol specification, together with a type system T. Let R be an description of a run in $\mathcal{P}$ where R is a set of PDL events partially ordered by the $<$ relation. We say that a substitution σ to the free variables in R is* legal *if*

1. *If a type system has been specified, then for any variable v in R, σR is well-typed, and the type of σv is a subtype of the type of v, and;*
2. *The run σR does not disobey any PDL axioms when s a subterm of event $(x)_V$or $\langle x \rangle_V$ occurring in R is interpreted as $((s))_V$ or $\langle\langle s \rangle\rangle_V$, respectively.*

To give an example, consider the run

$$(\nu x)_B.(z)_A < \langle x \rangle_{B_<}$$

The substitution $\sigma z = x$ is not legal, since it would violate the PDL axiom (which we will present later) that says that a fresh variable can't be sent or received until if is sent for the first time by its creator.

In the case in which the term algebra is a free algebra, we conclude that, for any message m sent or received, s is a subterm of m means that s must have been used to compute m. For other term algebras obeying some equational theory, this may not be the case. Consider the following:

$$\langle x \rangle_A < (x \oplus y)_A$$

where $\oplus$ stands for exclusive-or with the usual cancellation properties $x \oplus x = 0, x \oplus 0 = x$. It is easy to see that if $\sigma y = x \oplus z$ we get

[7] Note that the convention used here is a little different from that used in [16] in which the interpretation in terms of legal substitutions was only used for received messages.

$$\langle x \rangle_A < (z)_A$$

so that x was not necessarily used to compute the message that A received.

In [16], we develop a syntactical means of checking, for the basic PDL axioms, whether or not a message was created was created using a term x, where the term algebra in question is the free algebra augmented by exclusive-or.

PDL Axioms. PDL axioms are of three types. The first type describe basic properties of the communication medium. These are standard axioms that do not change; the main innovation of the work we describe in this paper is that we will be introducing new channel axioms for different types of channels. The second type describes the properties of the cryptosystems used by the protocols; these need to be augmented whenever a new type of cryptosystem is used. The third type describes the actions of honest principals in a protocol run. These, of course, are different for each protocol.

The basic channel axioms are as follows:

The receive axiom says that everything that is received must have been originated by someone:

$$A: \; ((m))_A \Rightarrow \; \exists X. \langle\langle m \rangle\rangle_{X<} < ((m))_A \tag{rcv}$$

The new axiom describes the behavior of the ν operator.

$$\begin{aligned}(\nu n)_B \wedge (a_A = ((n))_A \vee \langle\langle n \rangle\rangle_A) \Rightarrow (\nu n)_B < a_A \\ \wedge \; (A \neq B \Rightarrow (\nu n)_B < \langle\langle n \rangle\rangle_B < ((n))_A \leq a_A))\end{aligned} \tag{new}$$

where $FV(a)$ denotes the free variables of a. Thus, any event a involving a fresh term must occur after the term is generated, and if the principal A engaging in the event is not the originator of the term B , then a send event by B involving n and a receive event by A involving n must have occurred between the create and a events.

An example of an axiom describing the properties of a cryptographic function, is the following, describing the behavior of pubic key signature.

$$\langle\langle S_A(t) \rangle\rangle_{X<} \Longrightarrow \;\; X = A \tag{sig}$$

This simply says that, if a principal X signs a term t with A's digital signature and sends it in a message, then X must be A.

An example of a protocol specification is A's role in the challenge-response protocol:

$$\mathsf{H}A \Longrightarrow (\nu x)_A . \langle\langle c^{AB} x \rangle\rangle_A$$

In other words, if A is honest she creates a fresh value x and sends it in a challenge.

We are now able to express the Challenge-Response requirements template that we expressed in graphical form in Section 3.1 in PDL as follows:

$$\begin{aligned} A:\ (\nu x)_A.\ \langle\langle c^{AB}x\rangle\rangle_A \quad & < \quad ((r^{AB}x))_A \\ \Longrightarrow \langle\langle c^{AB}x\rangle\rangle_A < ((c^{AB}x))_B & < \langle\langle r^{AB}x\rangle\rangle_{B<} < ((r^{AB}x))_A \end{aligned} \tag{cr}$$

We consider it a proof obligation that will be discharged in PDL.

Suppose that we instantiate c^{AB} with the identity and r^{AB} with SIG_B. Then we can prove the challenge-response axiom in the following way.

1. We start out with what A observes: $A : (\nu x)_A.\ \langle x\rangle_{A<} < (S_B(x))_A$
2. Applying the rcv axiom, we obtain $A : \exists Q.\ \langle\langle S_B(x)\rangle\rangle_Q < (S_B(x))_A$
3. Applying the new axiom, we obtain $A : ((\nu x))_A.\ \exists Q.\ \langle x\rangle_{A<} < ((x))_Q < \langle\langle S_B(x)\rangle\rangle_{Q_\langle} < (S_B(x))_A$.
4. Applying the sig axiom, we obtain $q = B$, and we are done.

For the purposes of analyzing protocols that use different types of channels, we will need a means of specifying which channels we are using. Thus we extend PDL with a channel notation. We denote send actions taken along a channel κ as $\langle a : \kappa\rangle$ and receive actions taken along κ as $(a : \kappa)$. When no channel is specified we assume that standard cyber channel that obeys only the rcv and new axioms is being used.

4 Timed channel protocols

4.1 Proximity authentication

Our assumptions about cyber channels are very basic: 1) if a message is received it must have been sent by somebody, and 2) a few simple assumptions about the ordering of actions involving nonces. However, there are many cases where that is not enough. We consider for example the authentication problem in Section 3.1 as it might arise in a pervasive setting. In a pervasive network, Alice is, say, a gate keeper that controls a smart card reader m, which is a network node. Bob arrives at the gate with his smart card n, and creates a network link between m and n. Alice may not know Bob, but she is ready to authenticate any principal X who arrives at the gate, and links his smart card x to the reader m. She will allow access to anyone whose credentials are on her authorization list. Authentication with a fresh nonce bound to the secret credentials is necessary to prevent replay. This *Gate Keeper* Challenge-Response template extends Fig. 1 by one prior step, where Bob identifies himself to Alice by sending his name. The point of discussing this very simple scenario is to emphasize the *role of the network link* in the authentication. The goal of the authentication is to assure that

- (a) Bob is authorized to enter, and
- (p) he is at the gate.

While the authorization requirement (a) is emphasized, the proximity requirement (p) must not be ignored. If it is not satisfied, then an intruder Ivan may impersonate Bob. Ivan needs to control a smart card reader m' at another gate, where real Bob wants to enter; and he needs to establish a radio link between the card reader m' and a smart card n', with which he himself arrives at Alice's gate.

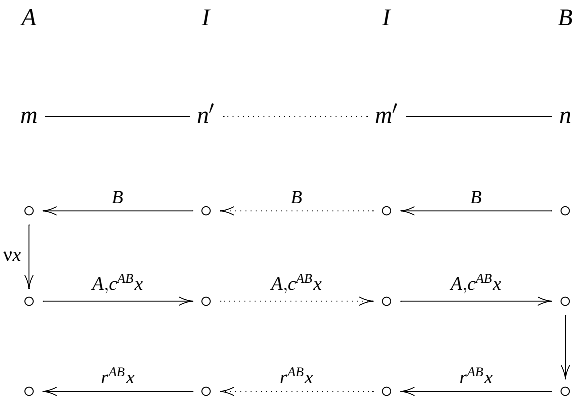

Fig. 2: Attack on Gate Keeper Protocol

In a cyber network (assuming that the radio link between m' and n' is realized as an ordinary network link), this would be a correct protocol run: Ivan is only relaying the messages. Indeed, Bob's and Alice's records of the conversation coincide, and the matching conversation definition of authenticity is satisfied.

In a pervasive network gate keeper model, the above protocol run is considered as a Man-in-the-Middle attack: although unchanged, the messages are relayed through the nodes under control of a non-participant Ivan. This would allow Ivan to impersonate Bob and enter Alice's facility unauthorized. These attacks, by the way, are not hypothetical; for example they have been demonstrated by Tippenhauer et al. in [27], in which location spoofing attacks on iPods and iPhones are implemented.

This example shows why pervasive networks require stronger authentication requirements than those routinely used in cyber security. The strengthening requires verifying not just that the principal Bob has sent the response, but also that he has sent it directly from a neighboring network node. This is the *proximity* requirement. It arises from *taking the network into account*, as an explicit security concern.

One way to verify whether a proximity requirement is satisfied is to use timed channels in distance bounding protocols. That is, if we can measure the time between the sending of a challenge and the receipt of the response, and we know the speed at which the signal travels, we can use this information to estimate the distance between two devices. The trick is to do this in a secure way.

4.2 Proximity verification by timing

Timed channels. We model a timed channel simply as a channel that allows the sender to time the message it sends and receives. More precisely, the difference between the timed channel and the standard channel is that

- the send and receive times can be measured only on the timed channel,
- the purported sender and receiver are a part of every message only on the standard channel.

Timed challenge-response. The main assumption about the timed channels is that the messages travel at a constant speed c, and that the length of network links is approximately d, say at most $d+\varepsilon$. By measuring the time t that it takes to a message x to arrive from a node m to a node n, one can verify whether x has travelled through a direct link by making sure that $ct \leq d+\varepsilon$. If Victor is the owner of the node m and Peggy owns the node n, then Peggy can prove to Victor that she is in the neighborhood by very quickly responding to x, say by fx. If Victor sends x at time τ_0 and receives fx at time τ_1, then he needs to verify that $c(\tau_1 - \tau_0) \leq 2(d+\varepsilon)+\theta$, where θ is the time that Peggy may need to generate and send fx.

The *timed challenge-response* template, capturing this idea, looks like this:

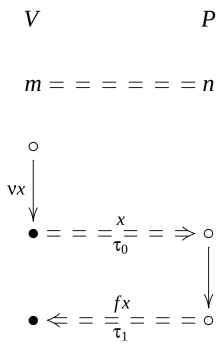

Fig. 3: Timed Challenge-Response Template

where the principals are now V (Victor the Verifier) and P (Peggy the Prover).

An action along a timed channel is called *timed* if the principal performing it notes the time (on its local clock) at which it was performed, *untimed* if the principal does not. A timed action a performed by a principal N is denoted by $\tau_i a$, where a is an action and τ_i denotes the time at which N performed the action. The taking of the time measurement in the diagram is noted by a bullet • under the name of the principal performing the action. Intuitively, this template says that after sending a fresh value x at time τ_0 and receiving fx at time τ_1, Victor knows that there is someone within the range of at most $\frac{c}{2}(\tau_1 - \tau_0)$. This template can be interpreted as a specification of the security property of the function f.

We make a comment here about implementation of timed challenge and response. For example, consider the case where f is the identity. Peggy can start responding to Victor as soon as she receives the first bit of x. Depending on the degree of accuracy needed, this can give Peggy a considerable advantage. Thus, unless Peggy is a trusted principal who will wait until she receives the entire nonce, it is advisable to use a bit-by-bit challenge and response, where the chance of Peggy guessing the correct bit response is bounded above by a constant. We can then choose x large enough so that the chance of Peggy cheating without being detected is negligible. A more thorough discussion of bit-by-bit challenge and response and the security issues involved in implementing it are found in [7]. In this paper we abstract away from these issues and make the notation $<$ stand for both the conventional and bit-by-bit notions of precedes.

Specifying timed channels in PDL. PDL has already been used to analyze distance bounding protocols in [16], for which we defined a timestamp function and some axioms governing it. However, this had the disadvantage that we could not specify in a natural way the actions for which timestamps were or were not defined. Moreover, specifying axioms in terms of which channels they apply to allows us to structure our specifications in a more modular way, in keeping with the spirit in which PDL was developed. In this section we describe how to do this, using a timestamp function similar in construction to the one used in [16]. However, in this case the function is used to describe properties of the channel.

An event a taking place along a timed channel is denoted by $\tau_i a$, where τ_i is a real number denoting the time at which the event takes place. Since recording of time can only take place on timed channels, and axioms involving timed channels involve time-recorded events only, we do not need any timed channel identifier. We have one axiom, saying that local times increase:

$$\tau_0 a_A < \tau_1 b_A \Longrightarrow \tau_0 < \tau_1 \tag{inc}$$

We also use the following definition of distance:

Definition 4.1. *Let A and B be two principals. We define the* distance *between A and B, or $d(A,B)$ to the minimum τ of all possible $(\tau_0 - \tau_2)/2$ such that the following occurs:*

$$(\nu n)_A.\ (\nu m)_B.\ \tau_0 \langle\langle n \rangle\rangle_{A<} < ((n))_B < \langle\langle m \rangle\rangle_{B<} < \tau_1\ ((m))\ _A$$

The inc axiom guarantees that this is well-defined.

Security goals of proximity authentication. The task is now to design and analyze protocols that validate the *proximity challenge-response template*, which is the timed challenge-response template augmented with a conclusion about time and distance. It is expressed in PDL as follows:

$$\begin{array}{ll} V: & (\nu x)_V.\ 0\langle x\rangle_V \qquad\qquad < \qquad\qquad \delta(f^{VP}x)_V \\ & \Longrightarrow 0\langle x\rangle_V < \quad (x)_P < \langle f^{VP}x\rangle\rangle_{P<} \ < (f^{VP}x))_P < \delta(f^{VP}x)_V \quad \wedge \quad d(V,P) \leq \delta \end{array} \tag{crp}$$

The (crp) template says that, if V creates a nonce, and sends it along the timed channel at time 0, and then receives a response at time δ, then P must have received the challenge after V sent it and then sent the response before V received it. Moreover, the distance between V and P is less than or equal to δ.

4.3 Distance bounding protocols

The simplest way to achieve our goal is to combine the cryptographic challenge-response from Fig. 1 with the timed challenge-response from Fig. 3. The authentication reasoning will then follow from the templates (cr) and (crp), taking $c^{VP}x = x$ in the former, sending both challenges together. The only part of the cryptographic challenge sent on the standard channel that is not sent on the timed channel are the purported sender and receiver. So they need to exchange some messages on the standard channel, to tell each other who they are.

Cryptographic response to a challenge usually takes time to compute. This means that it either (1) needs to be sent separately from the timed response, or (2) that it must be very quickly computable, as a function of the challenge. These two possible design choices subdivide distance bounding protocol in two families: those with two responses, and those with one response.

The easiest approach, from a design point of view, is to use two responses, since this allows one to avoid the challenging task of developing a function that both provides the necessary security and is fast to compute. However, in order to accomplish this the two responses must be linked together securely.

The approach of using two responses is that followed by the original distance bounding protocol of Brands and Chaum [3]. There, the response sent on the timed channel is the exclusive-or of the prover's nonce with the verifier's, sent as a sequence of one-bit challenge-response pairs. The response sent on the conventional channel is the digital signature on the two nonces. The binding message is a commitment (e.g. a one-way hash function) that the prover computes over its nonce, and sends to the verifier before the responder . In Čapkun-Hubaux [5] the binding function is the same, the timed challenge and response is a single exchange of nonces, and the response sent on the conventional channel is a hash taken over the nonces. In Meadows et al. [16] the response sent over the timed channel is a one-way hash function taken over the prover's name and nonce, exclusive-ored with the verifier's nonces. This combination of commitment with timed response reduces the message complexity of the protocol.

Hancke and Kuhn [16], who developed their protocol independently of Brands and Chaum, take the approach of using just one response. They do that by using a function $\boxplus$ which is quick to compute, but for which it not possible for a principal who has seen $x \boxplus y$ for only one value of x to compute y. This is obviously impractical to use if y is a secret key shared between verifier and prover, so instead prover and verifier use a keyed hash computed over a fresh values and a counter exchanged earlier in the protocol.

A detailed formal analysis of Hancke and Kuhn's distance bounding protocol has been presented in [24]. A sketch of a PDL analysis of the Brands-Chaum protocol is given in [23] [8]

5 Social channel protocols

Social channels arise from the fact that many pervasive devices, such as cellular phones, PDAs, and laptop computers, use their humans not only to promulgate in space, but

[8] The details of these analyses were written after the workshop version of the present paper was completed.

also to exchange information. For instance, a great part of the initial address books on many devices is usually received through this channel: a human manually enters some addresses of other devices. The devices often use their humans to exchange short messages.

5.1 A timed social protocol that we all use

Often, an address received through a human channel is authenticated using a timed channel: one device sends a message to the other one through a network channel, assumed to have some minimal speed, and then it observes through the human channel whether the message is received by the other device within a reasonable amount of time. This protocol can be interpreted as *binding a challenge sent on a timed channel with a response on a human channel*. This binding is assured by the human observing the caller's number on the receiver's device. The run is thus as on Fig. 4 where the notation

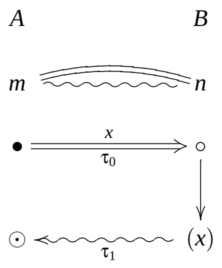

Fig. 4: Toy example

$b \underset{\tau}{\rightsquigarrow} \odot$ means that the $\odot$-side *sees* the other side perform the action b at time τ. In this case A sees B perform $b = (m)$, i.e. he receives m. This message may be just A's own identifier. Although m thus may not be fresh, if the waiting time $\tau_1 - \tau_0$ is sufficiently small, the chance that the message that B receives is not the same message that A has sent is assumed to be negligible.

5.2 Formalizing social channels

The non-local observations through social channels have deep repercussions on the problems of authentication. As pointed out in the beginning, the source of the problem of authentication in computer networks arises from the fact that all observations are local: a computer Alice can only observe her own actions (among which are the actions of sending and receiving messages). However, a mobile phone Alice can ascertain that another mobile phone Bob has received her message, if their humans are standing next to each other, observing both devices. The other way around, the mobile phone

Bob can ascertain that Alice has sent that message, because Bob's human has seen the message on Alice's screen, and Alice's human pushing the send button. Moreover, besides observing each other's actions, Alice and Bob can also send and receive brief messages through their humans, which are considered authentic because the humans observe each other.

Formally, we consider social actions in the form

$$\lessdot B \text{ to } A : \vartheta \gtrdot$$

which intuitively mean that Bob displays a term or an action ϑ for Alice to see. We attempt to capture this intended meaning by the following axioms

$$\lessdot B \text{ to } A : \beta \gtrdot \implies A : \beta_B \tag{sc1}$$

$$\lessdot B \text{ to } A : \beta \gtrdot \,\triangleright\, \lessdot C \text{ to } A : \gamma \gtrdot \implies A : \beta_B \triangleright \gamma_C \tag{sc2}$$

$$\lessdot B \text{ to } A : \beta(t) \gtrdot \vee \lessdot B \text{ to } A : t \gtrdot \implies \sigma t \in \Gamma_A \tag{sc3}$$

$$\forall T \in \Theta \forall t \in T \exists u \in T.\ u \neq t \ \wedge\ \sigma u = \sigma t \tag{sc4}$$

which should be read as follows:

- (sc1) If A observes the action β_B, then she knows that β_B really occurred.
- (sc2) If A observes the action β_B before γ_C, then she knows that β_B occurred before γ_C.
- (sc3) If A observes an action with a term t, or is shown the term t itself, then A knows the digest σt.
- (sc4) For every sufficiently large set of terms T, and every $t \in T$ it is feasible to find a different term $u \in T$ with the same digests $\sigma t = \sigma u$.

Intuitively, the prefix σ can be construed as a short hash function, leading to many collisions. Still more concretely, in the above scenario with mobile devices, this corresponds to the fact that Alice's human sees Bob's human receive a message, but if the message is long, he can only see a part that fits on Bob's screen; if the message is numeric, he can only discern a couple of digits. Therefore, many messages look the same, and the message that Bob has received may not be the one that Alice had sent after all.

The task is to design protocols to bootstrap authentication using these low bandwidth fully authentic social channels.

Graphic notation. In protocol diagrams, we use the following graphic elements:

- $\beta_B \rightsquigarrow \odot_A$ represent $\lessdot B \text{ to } A : \beta \gtrdot$
- $\beta_B \overset{\sigma t}{\rightsquigarrow} \odot_A$ represents $\lessdot B \text{ to } A : \beta(t) \gtrdot$
- $\circ_B \overset{\sigma t}{\rightsquigarrow} \odot_A$ represents $\lessdot B \text{ to } A : t \gtrdot$

The annotations by σ can be omitted, since they are redundant. Yet it is may be useful, at least in the initial derivations, to keep a reminder that all social communication is low bandwidth, and shold be viewed as digested through a short hash function σ.

5.3 Socially authenticated key establishment

We begin from one of the simplest protocol tasks: Bob announces his public key e (or an arbitrary message) on the standard channel, and his human displays a digest on the social channel to authenticate it. The two announcements can be made in either order, as on Fig. 5. Both cases are open to a Man-in-the-Middle (MitM) attack. The problem is that, by axiom (sc4), the Intruder can easily find a public/private key pair $(\breve{e},\breve{d})$ such that $\sigma(\breve{e},B) = \sigma(e,B)$. As always, the Intruder is also assumed to be in control of the standard channel. So in the first case, he can find $(\breve{e},\breve{d})$ such that $\sigma(\breve{e},B) = \sigma(e,B)$, and replace Bob's announcement of e,B by $\breve{e},B$. In the second case, the Intruder needs to intercept and hold Bob's announcement of e,B, compute $\sigma(e,B)$, find $(\breve{e},\breve{d})$ as above, announce $\breve{e},B$ in Bob's name, and wait for Bob's human to announce the digest $\sigma(e,B) = \sigma(\breve{e},B)$. In both cases, the Intruder ends up with the key $\breve{d}$ to read the messages sent to Bob, and Bob cannot read them.

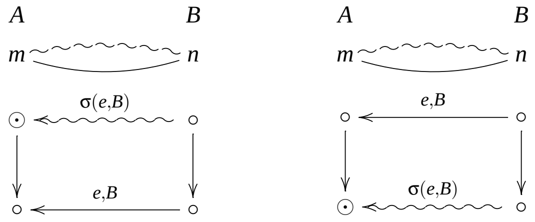

Fig. 5: Low and high bandwidth announcements

Social commitment. The MitM attack on the social channel authentication can be prevented in the same way as in the case of timed channel authentication: by binding the communications on the two channel types. For this purpose, Bob generates a fresh nonce, to be included in the social digest. The value of this nonce is publicly announced only after of the key, so that the Intruder, at the time when the key is announced, cannot know what the value of the digest will be, and cannot look for the collisions. On the other hand, in order to bind this nonce to the key (and to the protocol session, in order to prevent confusion), the nonce needs to be *committed* from the beginning, and, of course, decommitted in time to verify this binding. The two abstract templates from Fig. 5 thus refine to Fig. 6.

Authentication before decommitment. A particularly simple instance of the template on the left has been discussed by Hoepman [11]. The nonce y is also used as the private key, corresponding to the public key $e = g^y$. He takes the simple commitment schema where $ctx = Hx$ is just a sufficiently strong hash function, whereas the decommitment

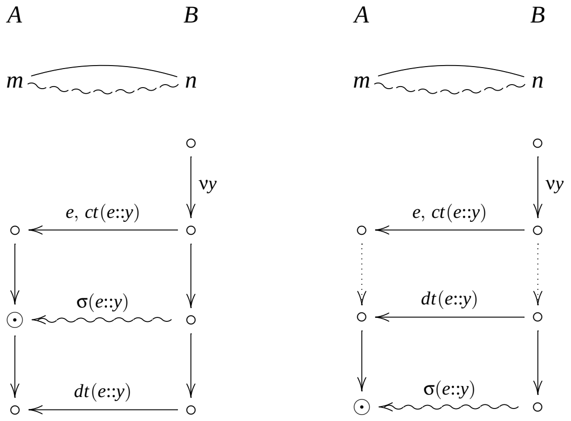

Fig. 6: Authentication before and after decommitment

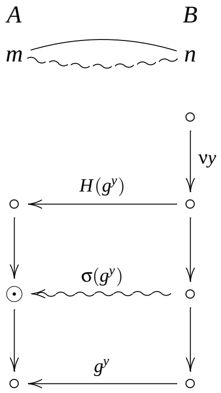

Fig. 7: The template for Hoepman's protocol

is simply $dtx = x$. The protocol boils down to Fig. 7. Of course, the main point of the Diffie-Helman exchange it to get a shared key g^{xy} by composing the above protocol with its mirror image, where Alice generates x and announces g^x. Hoepman also considers a final key validation phase. There are interesting possibilities to employ social channel here as well.

Authentication after decommitment. The problem with the template on the right hand side in Fig. 6 the is that the Intruder can simply hold the commitment message until the decommitment is sent, and then proceed with the MitM attack as before. To prevent this, we need to introduce a message from Alice at the dotted lines, confirming that the commitment is received. Note that this message needs to be authenticated as well: if the Intruder can fake it, he can get the decommitment from Bob, and again launch his MitM attack. But the only way to authenticate Alice's message, in absence of all other infrastructure, is the social channel again. So Alice needs to generate and send a fresh value as her acknowledgement, and this value also needs to be included in the social digest. Hence the following template

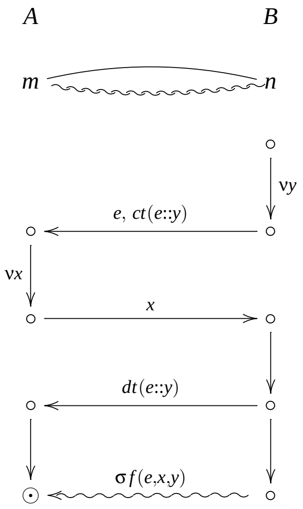

Fig. 8: Authentication after decommitment template

Remark. Authentication on the social channel does not follow the challenge-response template of authentication. There is no reason why it should: the challenge-response template implements *indirect* authentication, based on demonstrating a capability (to invert cryptographic functions, or to quickly respond on a timed channel), whereas the social channel implements a *direct* authentication, based on observing other principals' actions.

Mutual authentication. The analogous use of the nonces x and y in (8) is quite convenient when both Alice and Bob want to announce their keys. Alice's nonce x, used in (8) just to acknowledge receipt of Bob's commitment can now also be used to bind Alice's announcement to its social digest, whereas Bob's nonce y, which was used for this purpose in (8) can now also be used to acknowledge receipt of Alice's commitment. Composing (8) with its mirror-image version, where Alice and Bob exchange the roles, thus leads to a remarkably symmetric protocol.

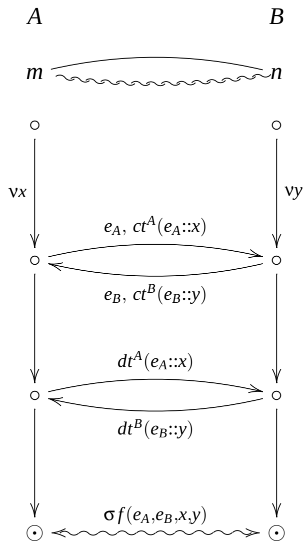

Fig. 9: Mutual authentication after decommitment template

The coupled messages from Alice to Bob and vice versa, as well the mutual social authentication at the end, are new graphic elements, corresponding to concurrent, or parallel actions of both principals. Note, however, that both concurrently sent messages in the second line must be received before either of the messages in the third line is sent. The social exchange in the end can be viewed in a similar way, as a pair of messages sent and received concurrently, this time on the social channel; but it may be more natural to view social authentication as a joint action of both principals.

Instances of the protocols from Figures 8 and 9. Vaudenay's SAS-authentication [28] (where SAS stands for "Short Authenticated Strings") is an instance of (8), with $f(e,x,y) = x \oplus y$. The special case of Nguyen-Roscoe's Symmetrized HCBK protocol, reduced to two principals is an instance of (9), with $f(e,x,y) = e{::}(x \oplus y)$, with the simple commitment schema[9] implemented by a short digest function $ct(y) = H(y)$, and $dt(y) = y$. Security of this protocol essentially depends on the special properties of this function, discussed in detail in [21].

HCBK protocol: matching conversations socially. The paradigm of *”matching conversations”* is, in a sense, the ultimate goal of authentication: if Alice’s and Bob’s views of all their conversation coincide, then they surely see their actions correctly, since Alice sees her own actions correctly, and Bob his. In order to match their views, each of them must derive the actions of the other from their own.

Nguyen and Roscoe’s HCBK protocol [21] directly attacks the problem of *simultatneous mutual authentication of whole groups* of principals. The strategy is to use social channels for *direct simultaneous matching of the conversations*: all principals announce on their social channels the digests of their views (records) of the conversation conversations, and their humans check that the digests match. This is a remarkably direct approach to authentication.

The point is that template (9) readily lifts from 2 to *n* principals: instead of just Alice and Bob sending their commitments in parallel, and then waiting for each other, all *n* principals can do that in parallel. The process expressions above remain quite similar to the two party case above. The technical proviso for this extension is that a suitable format for *n*-way matching on the social channel needs to be agreed upon. In particular, the principals must agree about

- (1) a method for each of them to arrive to the same ordering of the announcements that each of them has recorded, and needs to hash,
- (2) a social protocol to compare the values of all digests.

Both problems require ”breaking the symmetry” in a coordinated way. The first one can be deferred to the social channel, by adding an initial social message, announcing a linear ordering of all principals’ names. The second problem, minimizing the number of comparisons between the digests, requires imposing a tree structure on the group. Both problems have been major concerns in the Bluetooth design [12].

6 Conclusion

We have described some of the different types of channels that arise in pervasive networks, and the challenges and opportunities they give for authentication. We also give a formal description of several types of channels that arise in pervasive computing, and demonstrate a graphical template language for describing the behavior of protocols that make use of these different channels. We also describe some of the axioms that describe the behavior of some of these channels.

We are currently extending our work to develop logic for reasoning about the security of authentication protocols that make use of these different channels. We have

[9] They include Bob’s identity explicitly in the commitment. We keep principal’s identity implicit in every commitment.

found earlier that this approach, combined with the use of graphical templates that can describe both abstract and concrete specifications of protocols can be useful, not only in describing and organizing protocols that already exist, but generating new protocols that satisfy different types of requirements. We expect this to be the case here as well.

We do not intend to limit ourselves to proximity and social authentication. We expect this approach to work for other types of channels as well. In particular, we intend to investigate applications of our approach to quantum cryptography. Any quantum protocol, in order to be practical, will need to be harnessed together with conventional protocols as well. Although there as been a substantial amount of work on formal methods for quantum computation and quantum cryptography, very little of it addresses this aspect of the problem. We expect our methods to provide a natural way of doing this.

References

1. M. Abadi, M. Burrows, B. Lampson, and G. Plotkin. A calculus for access control in distributed systems. *ACM Transactions on Programming Languages and Systems*, 21(4):706–734, September 1993.
2. L. von Ahn, M. Blum, Ni. J. Hopper, and J. Langford. CAPTCHA: Using hard AI problems for security. In *EUROCRYPT 03*, pages 294–311, 2003.
3. S. Brands and D. Chaum. Distance-bounding protocols. In *Advances in Cryptology - Eurocrypt '93*. LNCS 765, Springer-Verlag, 1995.
4. Michael Burrows, Martín Abadi, and Roger Needham. A Logic of Authentication. *ACM Transactions in Computer Systems*, 8(1):18–36, February 1990.
5. S. Čapkun and J. P. Hubaux. Secure positioning in wireless networks. *IEEE Journal on Selected Areas in Communication*, 24(2), February 2006.
6. I.Cervesato, C. Meadows, and D. Pavlovic. An encapsulated authentication logic for reasoning about key distribution protocols. In Joshua Guttman, editor, *Proceedings of CSFW 2005*, pages 48–61. IEEE, 2005.
7. J. Clulow, G. Hancke, M. Kuhn, and T. Moore. So near and yet so far: distance-bounding attacks in wireless networks. In *European Workshop on Security and Privacy in Ad-Hoc and Sensor Networks (ESAS),*, 2006.
8. S. Creese, M. Goldsmith, A. W. Roscoe, and I. Zakiuddin. The attacker in ubiquitous computing environments: Formalizing the threat model. In *Proc. FAST '03*, pages 83–97, 2003.
9. Anupam Datta, Ante Derek, John Mitchell, and Dusko Pavlovic. A derivation system and compositional logic for security protocols. *J. of Comp. Security*, 13:423–482, 2005.
10. Y. Desmedt. Major security problems with the 'unforgeable' Feige-Shamir proofs of identity and how to overcome them. In *Proc. Securicom '88*, 1988.
11. Jaap-Henk Hoepman. Ephemeral pairing on anonymous networks. In Dieter Hutter and Markus Ullmann, editors, *SPC*, volume 3450 of *Lecture Notes in Computer Science*, pages 101–116. Springer, 2005.
12. Markus Jakobsson and Susanne Wetzel. Security weaknesses in bluetooth. *Lecture Notes in Computer Science*, 2020:176+, 2001.
13. Leslie Lamport. Time, clocks, and the ordering of events in a distributed system. *Commun. ACM*, 21(7):558–565, 1978.
14. B. Lampson, M. Abadi, M. Burrows, and E. Wobber. Authentication in distributed systems: theory and practice. *ACM Trans. on Comput. Syst.*, 10(4):265–310, November 1992.
15. Catherine Meadows and Dusko Pavlovic. Deriving, attacking and defending the GDOI protocol. In Peter Ryan, Pierangela Samarati, Dieter Gollmann, and Refik Molva, editors, *Proc. ESORICS 2004*, volume 3193 of *LNCS*, pages 53–72. Springer Verlag, 2004.

16. Catherine Meadows, Radha Poovendran, Dusko Pavlovic, Paul Syverson, and LiWu Chang. Distance bounding protocols: Authentication logic and collusion attacks. In R. Poovendran, C. Wang, and S. Roy, editors, *Secure Localization and Time Synchronization in Wireless Ad Hoc and Sensor Networks*, pages 279–298. Springer Verlag, 2007.
17. Catherine Meadows, Paul Syverson, and LiWu Chang. Towards more efficient distance bounding protocols. In *SecureComm 2006*, August 2006.
18. A. Mink, L. Ma, T. Nakassis, H. Xue, O. Slatter, B. Hershman, and X. Tang. A quantum network manager that supports a one-time pad stream. In *Pro. 2nd International Conference on Quantum, Nano, and Micro Technology*, February 2008.
19. A. C. Myers and B. Liskov. Protecting privacy using the decentralized label model. *ACM Transactions on Software Engineering and Methodology*, 9(4):410–442, October 2000.
20. L. H. Nguyen. Authentication protocols based on low-bandwidth unspoofable channels: a survey, 2008. Available at `http://web.comlab.ox.ac.uk/people/Long.Nguyen/` .
21. L. H. Nguyen and A. W. Roscoe. Authenticating ad hoc networks by comparison of short digests. *Inf. Comput.*, 206(2-4):250–271, 2008.
22. Dusko Pavlovic and Catherine Meadows. Deriving secrecy properties in key establishment protocols. In Dieter Gollmann and Andrei Sabelfeld, editors, *Proceedings of ESORICS 2006*, volume 4189 of *Lecture Notes in Computer Science*. Springer Verlag, 2006.
23. ———. Deriving authentication for pervasive security. In J. McLean, editor, *Proceedings of ISTPS 2008*. ACM, 2008.
24. ———. Bayesian authentication: Quantifying security of the Hancke-Kuhn protocol. *E. Notes Theor. Comp. Sci.*. 265:97–122, 2010.
25. P. Schaller, B. Schmidt, D. Basin, and S. Čapkun. Modeling and verifying physical properties of security protocols for wireless networks, April 2008.
26. D. Singleé and B. Preneel. Location verification using secure distance bounding protocols. In *International Workshop on Wireless and Sensor Network Security*. IEEE Computer Society Press, 2005.
27. N. Tippenhauer, K. Rasmussen, C. Popper, and S. Capkun. iPhone and iPod location spoofing attacks, 2008. `http://www.syssec.ch/press/location-spoofing-attacks-on-the-iphone-and-ipod`.
28. Serge Vaudenay. Secure communications over insecure channels based on short authenticated strings. In Victor Shoup, editor, *CRYPTO*, volume 3621 of *Lecture Notes in Computer Science*, pages 309–326. Springer, 2005.
29. F. L. Wong and R. Stajano. Multichannel security protocols. *IEEE Pervasive Computing*, 6(4), December 2007.