Deriving Authentication for Pervasive Security

Dusko Pavlovic
Kestrel Institute and Oxford University
dusko@{kestrel.edu,comlab.ox.ac.uk}

Catherine Meadows
Naval Research Laboratory
catherine.meadows@nrl.navy.mil

Abstract

In this paper we discuss some of the challenges and opportunities offered to authentication by pervasive computing and discuss the work we are doing in developing formal and graphical systems for reasoning about and understanding the security of protocols in pervasive computing. We provide an example of the verification of a proximity authentication protocol that uses several different types of channels to achieve its goals.

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 the Internet.

There has been another aspect of pervasive computing that has not been as well acknowledged. 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 nature 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 [18]) and “what you see” (use of human-verifiable channels to boot strap secure communication, as in [13, 19]).

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 [7, 14]. 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. In Section 5 we illustrate our logic in an analysis of a distance bounding protocol that makes use of several types of 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 a description of 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, even in the presence of a hostile attacker who can monitor, insert, delete, and modify messages. But the means by which the nodes actually communicate (that is, the lower layers of the protocol) is generally not included in the model. Moreover, generally no distinction is made between different types of channels.

Security protocols have been thus naturally modeled formally within a process calculus, as in [14]. In order to model security protocols in pervasive networks, we extend the process model from [14], used for analyzing security protocols in cyber networks. The main complication is that now we must make the channels explicit.

2.1 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. 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}.

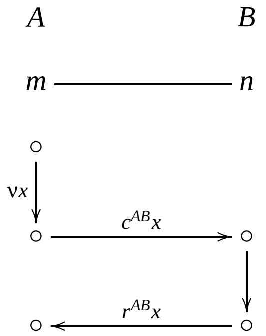

Figure 1: Challenge-Response Template

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)

3.2.1 PDL Syntax

PDL is a descendant of an early version of the Compositional Protocol Logic (CPL) [5] 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 [14] 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, 9] or "acts for" B in the sense of [12]. 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 [8], 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 $\langle\langle s\rangle\rangle_A$ to 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. [1]

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

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

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

In [10], 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.

[1] Note that the convention used here is a little different from that used in [10] in which the interpretation in terms of legal substitutions was only used for received messages.

In this paper, except where otherwise noted, we will assume that the term algebra we are dealing with is the free algebra augmented with exclusive-or, which we will need to reason about the Brands-Chaum protocol we use as an example.

3.2.2 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:

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

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 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. The template is now The point of spelling out the details of this rather familiar scenario is to emphasize the *role of the network link* in the authentication. The goal of the authentication is to assure that

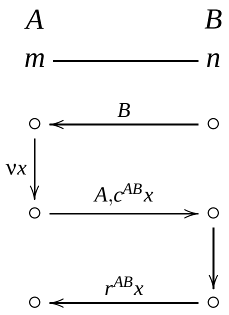

Figure 2: Gate Keeper Challenge-Response Template

- (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.

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.

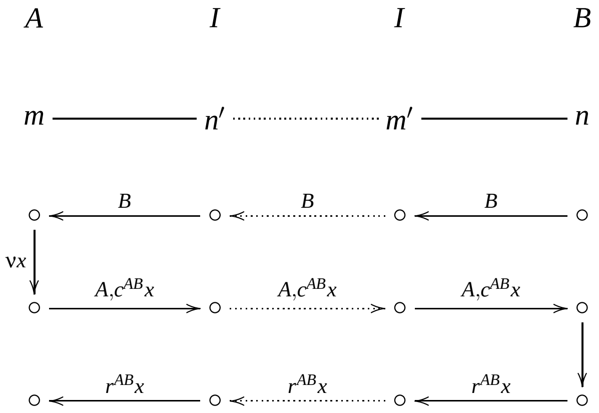

Figure 3: Attack on Gate Keeper Protocol

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 [16], 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.

4.1 Proximity verification by timing

4.1.1 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.

4.1.2 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:

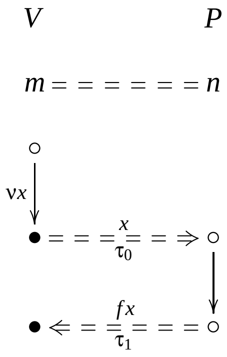

Figure 4: Timed Challenge-Response Template

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 [3]. 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.

4.1.3 Specifying timed channels in PDL

PDL has already been used to analyze distance bounding protocols in [10], 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 [10]. 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:

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.

4.1.4 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:

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

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.

5 Applying the Protocol Derivation Logic to Distance Bounding

5.1 Distance bounding protocols

The simplest way to achieve our goal is to combine templates cr and crt is to take $c^{VP}x = x$ and to send 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 [2]. 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 [17] 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. [10] 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 [10], 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.

In the next section we show how PDL can be used to analyze the Brands-Chaum protocol.

5.2 Brands-Chaum Protocol

In Brands-Chaum, the function f is exclusive-or. The equational theory E obeyed by the term algebra consists of the group-theoretic properties of exclusive-or : associativity, commutativity, identify, and cancellation. Thus, the timed portion of the protocol is as follows:

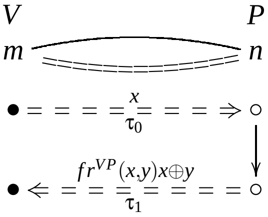

Figure 5: Timed Portion of Brands-Chaum

However, although Victor knows his own nonce, he doesn't know Peggy's nonce in advance. Thus, he can't verify that Peggy's response was computed using both Victor's and her nonces. Peggy could have sent a random term z. Victor could verify that $z = x \oplus y$ for his nonce x and some y , but this is true for any term, thanks to the cancellation properties of exclusive-or. Victor can't conclude that $x \sqsubseteq z$.

The solution to this is to use a *commitment*. A commitment is a pair of functions ct and ot such that $ct(y)$ does not reveal y, but the result of receiving $g = ct(y)$ and $ot(y,g)$ means that it is possible to verify that $ct(y)$ was computed using y. An example of ct would be applying a one-way function to y concatenated with a nonce x. The resulting ot is the revealing of x and y.

We describe commitment by the axiom

$$\langle\langle ot(y,g)\rangle\rangle_{A<} \Longrightarrow \exists Z(\mu x)_Z \leq (g = ct(y))_A < \langle\langle ot(y,g)\rangle\rangle_{A<} \tag{cmt}$$

This says, that, if A creates and sends the opening of a commitment, then she must have created the commitment, and then sent its opening, and x must have been created (not necessarily by A) prior to A's commitment.

We are now have the following template:

But we are still missing something. Victor still has no idea of *who* has initiated a challenge and response with him. We solve this problem by having Peggy sign her commitment the two nonces in the last message, thus producing a two-response template:

We specify the Brands-Chaum protocol in PDL below. Victor's role is as follows:

$$\nu x.\ (g)_V \prec \tau_0\langle x\rangle_{V<} \prec \tau_1(x \oplus y)_V \prec (S_P(x,y,ot(y,g)))_V \tag{bcv}$$

The role of an honest Peggy is as follows:

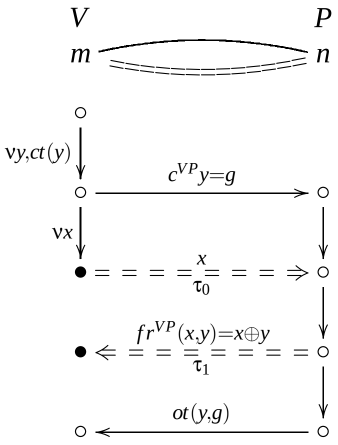

Figure 6: Brands-Chaum Protocol with Commitment

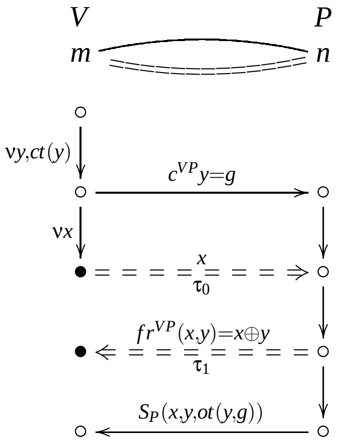

Figure 7: Brands-Chaum with Commitment and Signature

$$\nu y.\ \langle g = ct(y)\rangle_{P<} = \langle\langle y\rangle\rangle_{P<} \prec (x)_V \prec \langle x \oplus y\rangle_P \prec \langle S_P(x,y,ot(y,g))\rangle_P \tag{bcp}$$

We consider the case of an honest Victor interacting with a possibly dishonest Peggy. After participating in the protocol, Victor will know that the events in bcv occurred from his observations. From the rcv and sig axioms Victor learns that Peggy sent the final message. Victor also learns, from the rcv and cmt axioms, that whoever sent $x \oplus y$ to Victor sent it after x was created. Victor can also verify that $x \oplus y$ could have been constructed using x, but he must also verify that x remains a subterm of $x \oplus y$ for all legal substitutions to the variables in bcv. The only substitution that would make x disappear from $x \oplus y$ is $y = x \oplus z$ for some z. But, from the cmt axiom Victor can conclude that y was created before he sent x. Thus, this substitution is not legal. We conclude y is a subterm of $x \oplus y$ for all legal substitutions to bco and thus $(x \oplus y)_V = ((x))_V$. Victor can then use to the new axiom to conclude that whoever sent $x \oplus y$ sent it after receiving x.

But Victor still does not have enough information to prove that Peggy sent $x \oplus y$. One reason for this is that a dishonest Peggy could offload some of her tasks to a cohort Eve who is closer to Victor than Peggy is. Eve would send $x \oplus y$ over the timed channel, while Peggy would send her signed message over the conventional channel. This attack, coined the “terrorist” attack by Desmedt [6], was known of by Brands and Chaum, who considered designing distance bounding to be secure against it as an open problem [2].

When considering the terrorist attack, however, it is best to take the threat model into account. In the proximity authentication scenario described, if Peggy had a cohort Eve who satisfied the proximity requirement, she could simply hand over her keys to that Eve, and then have Eve run the entire protocol. Terrorist attacks are mainly relevant in the case in which cohorts are unwilling or unable to share keys. Thus, the solutions to the terrorist attack problem that have been produced require either tamper-proof devices (in which case Peggy is always honest), or sharing of long-term keys between Peggy and Eve.

Our approach to proving the security of the Brands-Chaum protocol will thus be to follow the approach of Schaller et. al [15] and assume that Peggy has no one available to be her cohort:

$$d(V,Q) < \delta \Longrightarrow \mathsf{H}Q \tag{hst}$$

But, we still do not have quite the assurance we need. Suppose that an honest prover Priscilla interacts with Victor according to the rules of the protocol, culminating in her sending the message $S_{Pr}(ot(c,x),x,y)$. There is nothing preventing Peggy from intercepting Priscilla’s message and substituting her own signature $S_P(x,y,ot(y,g))$, making Priscilla’s distance look like her own.

There is an easy fix: have the prover commit to her identity as well as her nonce. This is the approach followed in more recent distance bounding protocols, e.g. explicitly in Meadows-Syverson-Chang and implicitly in Hancke-Kuhn. But before we conclude that Brands and Chaum were in error, it may be useful to take a look at the assumptions they were making about the environment. Brands and Chaums were writing in the early 90’s when Peggy was a smart card and Victor was a reader, such as an ATM, which needed to be in physical contact with the smart card in order to communicate. When Peggy starts the protocol with Victor, she sets up a channel which no one else can send messages on until she finishes the protocol and removes the smart card. This is an *integrity channel* ι, which we can axiomatize as follows

$$\exists X,Y.\ \langle m:\iota\rangle_Q \implies Q = X \vee Q = Y \tag{int}$$

In other words, there are only two parties that can send messages along an integrity channel: the principals on either end of the channel.

We now continue with our analysis of Brands-Chaum, under the assumption that only one channel, which is both a timed channel and an integrity channel, is being used. The diagram of the protocol now looks like this: where the single dashed line indicates the integrity channel. Victor’s observations are now:

$$\nu x.\ \big((g:\iota)_V < \tau_0\langle x:\iota\rangle_{V<} < \tau_1((x\oplus y:\iota)_V < (S_P(x,y,ot(y,g)):\iota)_V\big) \tag{bco2}$$

We now amend the definition of the honest prover:

$$\begin{aligned}\mathsf{H}P \Longrightarrow \Big(&\langle m\rangle_P \Longrightarrow \langle P\rangle_{P<} \\ &\wedge \nu y.\ \langle g = ct(y:\iota\rangle_{P<} \prec (x:\iota)_P \prec \langle x\oplus y:\iota\rangle_P \prec \langle S_P(x,y,ot(y,g)):\iota\rangle_P\Big)\end{aligned} \tag{bcp}$$

That is, honest provers not only follow the rules of the protocol, but they do not forward messages from other principals along the integrity channel.

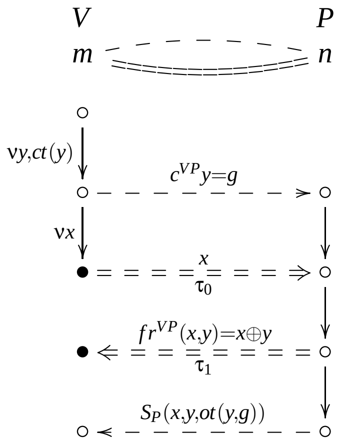

Figure 8: Full Brands-Chaum Using Integrity Channel

We are now ready to complete our proof. There are two cases: one in which the other party on the integrity channel is honest, and one in which she is dishonest. In the honest case, we can conclude from the definition of honest prover and the other observations and axioms that Peggy sent both timed response and signed message, and this gives us the result we need. In the dishonest case, we use the hst assumption to conclude that, if the sender of the timed response is not Peggy, then the round-trip time is greater than if Peggy had sent it. Thus, we can draw the same conclusion as in the honest case.

6 Conclusion and Further Work

We have outlined a procedure for logical analysis of authentication protocols that make use of multiple channels with different properties, and applied it to the case of distance bounding using timed channels. We also have shown how an apparent insecurity in the Brands-Chaum protocol arises from implicit assumptions about channel behavior, and showed how these assumptions can be formalized in our system. This demonstrates one of the chief advantages of this approach; it can be applied to *families of protocols* that use different mechanisms and channels, and used to compare behavior and security guaranties.

Timed and integrity channels are only a few of the channel types that arise in pervasive computing. Another type that arises is the *human-verifiable* channel [19], in which Alice sends Bob a short message and verifies that he received it by visually checking that it appears on his device. Human-verifiable channels are low-bandwidth, and must be combined with conventional channels to be useful. Generally, they have been proposed for use in bootstrapping key distribution in the absence of a public-key infrastructure [13, 19]. We have developed an axiomatization of human-verifiable channels that takes into account both their integrity properties and their limited bandwidth, and are beginning to apply it to these types of protocols.

Other channels that we believe would be amenable to this approach are *quantum cryptography* channels. Quantum key distribution usually relies on classical channels to provide services that the quantum channel is incapable of. Our approach could be applied to comparing different ways of integrating these channels and determining which ones are safe. As quantum crypto becomes more practical, more attention is being paid to the problem of integrating together with conventional networking systems, e.g. in [11], and we expect this will be come an important issue in the future.

Finally, we consider our work complementary to other ongoing work on providing explicit formal models of attackers [4] and physical properties of channels [15]. Indeed, we expect that such explicit models could be used to provide a semantics by which our axioms can be proved sound.

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] S. Brands and D. Chaum. Distance-bounding protocols. In *EUROCRYPT '93*, pages 344–359. Springer-Verlag New York, Inc., 1994.

[3] 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.

[4] 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.

[5] A. Datta, A. Derek, J. C. Mitchell, and D. Pavlovic. A derivation system for security protocols and its logical formalization. In Dennis Volpano, editor, *Proceedings of CSFW 2003*, pages 109–125. IEEE, 2003.

[6] Y. Desmedt. Major security problems with the 'unforgeable' Feige-Shamir proofs of identity and how to overcome them. In *Proc. Securicom '88*, 1988.

[7] 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.

[8] Leslie Lamport. Time, clocks, and the ordering of events in a distributed system. *Commun. ACM*, 21(7):558–565, 1978.

[9] 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.

[10] C. Meadows, R. Poovendran, D. Pavlovic, L.W. Chang, and P. Syverson. Distance bounding protocols: authentication logic analysis and collusion attacks. In R. Poovendran, C. Wang, and S. Roy, editors, *Secure Localization and Time Synchronization in Wireless Ad Hoc and Sensor Networks*. Springer-Verlag, 2006.

[11] 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.

[12] 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.

[13] 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/` .

[14] 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.

[15] P. Schaller, B. Schmidt, D. Basin, and S. Čapkun. Modeling and verifying physical properties of security protocols for wireless networks, April 2008.

[16] 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`.

[17] S. Čapkun and J. P. Hubaux. Secure positioning in wireless networks. *IEEE Journal on Selected Areas in Communication*, 24(2), February 2006.

[18] 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.

[19] F. L. Wong and R. Stajano. Multichannel security protocols. *IEEE Pervasive Computing*, 6(4), December 2007.