Context Authentication Using Constrained Channels

Tim Kindberg, Kan Zhang
Hewlett-Packard Laboratories
1501 Page Mill Road
Palo Alto, CA 94304, USA
{timothy,kzhang}@hpl.hp.com

Narendar Shankar
Computer Science Department
University Of Maryland
A.V. Williams Building
College Park, MD 20742, USA
narendar@cs.umd.edu

Abstract

This paper presents practical protocols for authenticating the parameters that characterise a principal's context. Context features in what are known as mobile, ubiquitous, pervasive and nomadic computing systems. Location, in particular, is a highly significant contextual parameter. We present a model of context authentication based on the characteristics of communication channels. Then we present protocols for location authentication that are based on physical channel characteristics, including a protocol that we have implemented over HTTP for use from any Web browser. We conclude with a summary and discussion of the work.

1. Introduction

This paper describes a model for *context authentication*: authentication of a principal's status in a certain context; in particular, its physical location. This work is part of the CoolTown project [1,2,3,4], which is investigating 'nomadic' computing systems: ones in which users, carrying wirelessly connected devices, enter places and use local services associated with those places, as well as remote services.

Conventional authentication protocols establish the identity of a principal *p* based upon the premise that only *p* possesses some secret *K*. But in some circumstances we are interested in the characteristics of a principal's context, such as its location, in addition to or instead of its identity. For example:

1. To attract customers, the Kardomah Coffee House wishes to provide a service *S* only to those who are, or who have recently been, on their premises.
2. The President of Coolania wishes to take calls on the 'Red Phone' only from those who are physically inside the inner sanctum of Hotria's centre of government.
3. A computer that provides a service inside a company's headquarters is to cease to operate if taken outside the building.
4. A public 'kiosk' computer in an airport is to erase its memory if the user, who may have downloaded personal data onto it, walks away for more than *T* seconds.
5. Only users who click an 'I agree' button on a certain web page with certain contents—i.e., users who visit that virtual location—are provided with service *S*.

We consider a principal's context to be characterised by a set of *contextual predicates*, such as 'the location of *p* is the Kardomah cafe', 'the date of *p*'s action is 2002/1/1 or later', 'the temperature in *p*'s environment is 15C or more'. To authenticate such a contextual predicate ϕ for a principal *p* is to verify securely that $\phi(p)$. In particular, we wish to know that the principal who issues a given request message satisfies ϕ. For example, we may require that a principal that requests a service is in a certain location, or exists within some particular interval of time.

Our main contribution is a set of practical protocols for verifying contextual predicates in situations that do not require strong authentication—e.g., a coffee house rather than presidential headquarters. The protocols implement a model of *constrained communication channels*, which we introduce to enable us to specify and discuss the abstract properties of the systems that we build. We relate our problem to other work in Section 2. Section 3 presents our model, which captures the essential features of context authentication in terms of constrained communication channels. Section 4 describes location authentication protocols based on the model. Section 5 concludes.

2. Related work

Context-awareness has been identified as a key issue in nomadic computing with location being the most prominent contextual parameter [4, 5, 8].

Focardi [14] uses location as an extra criterion of identity when analysing authentication protocols but does not consider how to authenticate a principal's location itself.

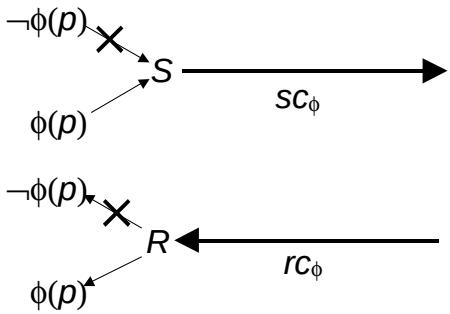

Figure 1. Send- and receive-constrained channels

Location and other contextual data are usually collected via sensing technologies, e.g., GPS and the Active Badge [5]. However, some sensing-based approaches are subject to spoofing. For example, in the Active Badge system the badges are easily separable from the owner. The outputs of conventional (code correlating and differential) GPS receivers can be easily forged since there is no way to tell whether they were actually calculated by a GPS receiver. To make forgery difficult, Denning, et al. [6] introduced location signature sensors to compute a location signature from the microwave signals transmitted by the GPS satellites. However, this system is vulnerable if an attacker is able to record the GPS satellite signals and re-assemble the aggregate signal with the appropriate delays for the location he is trying to spoof.

Gabber and Wool [12, 13] discussed several techniques for tracking the location of set-top terminals used for receiving satellite TV. Those techniques include: using the CallerID feature of telephone exchanges; the enhanced 911 location feature of cellular networks; GPS and differentials in the arrival times of the satellite signal at different set-top boxes. These solutions are unsuitable for a general nomadic computing environment because of their requirements for specialised infrastructures and/or tamper-proof modules integrated with user devices.

Other techniques, which we follow from, exploit the physical characteristics of network propagation. Brands and Chaum [10] introduced a distance-bounding technique to determine an upper-bound on the distance between two communicating parties by timing the delay between sending out a challenge bit and receiving back the corresponding response bit. Ensure Technologies' XyLoc system [11] uses short-range radio transmission to establish the proximity of a PC to its owner as a criterion for its accessibility.

We are continuing from the work of Caswell and Debaty, also within the CoolTown project [4]. Caswell and Debaty introduced the idea of "establishing a user's presence by proving proximity to a known reference point within a place". They went on to present a timestamp-based protocol for location authentication. A short-range wireless beacon is used to emit a time-varying token for location authentication. A shortcoming of this approach is that the clocks of the beacon and the authenticator have to be synchronised to avoid replay attacks.

In this work, we are interested in a general model of context authentication without assuming a particular technology. As an instantiation of our model, we will present new location authentication protocols that do not use time.

3. The model

In this section we describe the abstractions of constrained channels and channel proxies, which are used to connect constrained channels. We present some techniques for composing and reversing constrained channels. Then we outline how our abstractions can be realised.

3.1. Constrained channels

We are interested in channels that implement a contextual constraint: ones that allow us to make inferences about the context of sending or receiving principals.

Any one-way channel *c* has *send* and *receive* operations for a message *m* as follows:

m = *c*.*receive*()

c.*send*(*m*).

We denote the principals that perform those operations for a uniquely identified message *m* as *receiver*(*m*) and *sender*(*m*), respectively.

A *constrained channel* is a one-way communication channel that is either *send-constrained* or *receive-constrained* or both (Figure 1):

send-constrained channel sc_ϕ on predicate ϕ: if $m = sc_\phi.receive()$ then ϕ (*sender*(*m*)).

receive-constrained channel rc_ϕ on predicate ϕ: ϕ(*receiver*(*m*)) for any message *m* appearing in an operation $rc_\phi.send(m)$.

Constrained channels are distinct from the notion of restricted channels in the π-calculus [15]. The latter incorporates restriction only with respect to sending on channels; and it restricts access according to possession of capabilities, not according to extrinsic factors such as location.

Our definitions capture some properties established by conventional security protocols. For example, consider two principals connected by a Transport Layer Security (TLS, also known as SSL) connection [7], who send and receive clear-text messages that are encrypted and decrypted by the connection. Then that channel is both send-constrained and receive-constrained on the predicate

'possesses secret key *K*' for some *K* negotiated by the TLS protocol.

But constrained channels are designed to capture a wider class of contextual predicates: any that can be established by construction of suitable hardware and software systems. Among the possibilities, an important example is a channel that imposes constraints upon the location of the communicating parties at one or other end of the channel.

3.2. Channel proxies

A *channel proxy P* is a process that connects exactly two one-way channels. The channel proxy receives messages from its input channel and selectively forwards the messages to its output channel. It does so according to its application: it is not simply a transport-layer entity. For each message it receives, it may either discard or forward it, possibly after a delay but without modifying it. The proxy may only send messages that it has received.

The simplest type of channel proxy forwards all the messages it receives. But we are free to use channel proxies that delay or discard messages. By delaying messages, proxies can implement temporal constraints: for example, a proxy could delay all messages until midnight 2002-01-01.

Another type of proxy decides whether to discard or forward messages based upon some property of a context. Suppose that, for Tim to access the Kan filing cabinet service, (a) he must be present in Kan's office and (b) Kan must also be present (so that he can observe Tim's activities). Kan installs a proxy in his office that uses a wireless network to detect the presence of users in the office. When Tim sends a message to the filing cabinet service on that channel, the proxy knows that Tim is present. But it holds onto the message until it can establish, using the same wireless channel, that Kan is also present. If so, it forwards the request. Otherwise, it discards it.

In general, we can construct channel proxies that can independently evaluate a contextual predicate ψ that applies to any principal p such that $\phi(p)$. For example, it could be a proxy that measures the set of people in a room or the temperature in the room. Employing an input channel that is send-constrained on ϕ, we can use the proxy to implement a channel constrained on $\phi \wedge \psi$.

3.3. Composing channels

We can use constrained channels as building blocks for making further constrained channels. We do so by inserting a channel proxy between two constrained channels.

Suppose a channel proxy *P* connects two one-way channels c_1 and c_2. We denote the complex of the proxy and two channels as $c_1.P.c_2$. The complex $C = c_1.P.c_2$ behaves as a (possibly lossy) channel, if we identify $C.send \equiv c_1.send$, and $C.receive \equiv c_2.receive$.

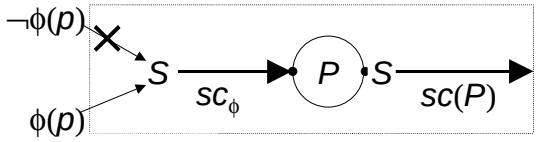

Figure 2. A channel proxy interposed to construct a new send-constrained channel.

One configuration in which we are interested is where a channel proxy *P* is connected to a channel that is send-constrained on the predicate 'sender is *P*'. We shall denote that channel as *sc*(*P*). Similarly, we can consider a proxy whose input side is a channel that is receive-constrained on the predicate 'receiver is *P*'. We shall denote that channel as *rc*(*P*).

We can construct new send-constrained or receive-constrained channels from others, as follows.

If sc_ϕ is a send-constrained channel on the predicate ϕ, then so is the channel $C = sc_\phi.P.sc(P)$ (see Figure 2). To verify this, we must show that all senders of a message received from *C* satisfy ϕ. *C.receive*(*m*) means *sc*(*P*).*receive*(*m*). That implies, by the definition of *sc*(*P*), that *P* sent *m*. But *P* can only send what it receives, so *m* arrives through an operation $sc_\phi.receive()$. Therefore, $\phi(sender(m))$.

Similarly, we can create a new receive-constrained channel from a receive-constrained channel and a channel proxy. If rc_ϕ is a receive-constrained channel on the predicate ϕ, then so is the channel $C = rc(P).P.rc_\phi$. We must show that all receivers of the message outside *C* satisfy ϕ. If *m* is sent on *C* then it is sent on *rc*(*P*). Therefore, by definition, only P receives it. Since *P* may only forward *m* along rc_ϕ, we have that $\phi(receiver(m))$—if *P* forwards *m*.

3.4. Reversing constrained channels

Let rc_ϕ be a receive-constrained channel on the predicate ϕ. We shall show how to construct a channel $s(rc_\phi)$, which is send-constrained on ϕ.

We use a trusted node *N*. When it receives a message *m*, it uses the receive-constrained channel to return to the sender a signed hash $sig\{h(m, t)\}$, where *t* is the time by *N*'s clock.

Let *c* be any (possibly unconstrained) channel connecting the parties that we wish to be able to communicate. We construct $s(rc_\phi)$ from *c* and *N*. The rules for sending and receiving on $s(rc_\phi)$ are as follows:

To send *m* on $s(rc_\phi)$:

send m to N

receive < t, sig{$h(m, t)$}> from N over rc_ϕ

send <m, t, sig{$h(m, t)$}> on c.

To receive m on $s(rc_\phi)$:

receive <m, t, h> on c

verify h = sig{$h(m, t)$}

verify currency of t and freshness of h

Discard m if verification fails, else receive m.

We assume that the receivers' clocks are synchronised to N's clock. The timestamp (and hence state of the sender) is deemed current if it is within a given bound of the time on the receiver's clock. To prevent replay attacks, the receiver need remember the hashes for only a limited time: older messages will have a non-current timestamp. In the next section we shall use a challenge-response protocol to remove the synchronisation assumption for the case of location authentication.

In a similar fashion, we can implement a receive-constrained channel from a send-constrained channel. All messages are sent (over any channel) to a trusted node, which stores them. Receivers must use a particular send-constrained channel to reach that node, which responds with the next message for them.

3.5. Implementing constrained channels

An example of a constrained channel would be one that is mechanically secure, such as a tamper-proof series of links and logic circuits connecting one terminal to another. Such a channel will have the property that a receiver can be sure the sender is at the other terminal.

Other physical characteristics of communication channels also enable us to construct constrained channels: the speed of signal propagation and the decrease in signal strength as it propagates.

One way of establishing location information is to use network time-of-flight. In principle, with sufficiently accurate instrumentation and knowledge of real-time system parameters, we can use round-trip times to bound the location of a network node. To gauge a bound on the distance to node M, node N sends a 1-bit message to it, which M is to return to N immediately. If the speed of signal propagation is c then, if M can return the message to node N in time $t < (l + 2d/c)$, it is within a distance d of N, where l is the total communication latency imposed by software and hardware.

We can also employ wireless network segments whose range (the distance over which messages may effectively propagate) is bounded. This includes radio (e.g. Bluetooth or IEEE 802.11), with a range of 10cm-1km; infrared (IR), with a range of 10cm-10m; and ultrasound, up to 10m in range.

Radio permeates walls, in general, so its reach often does not match the physical territory that we think of as a distinct location. On the other hand, radio at 60 GHz is strongly attenuated by many building materials, as are IR [3, 5] and ultrasound [8].

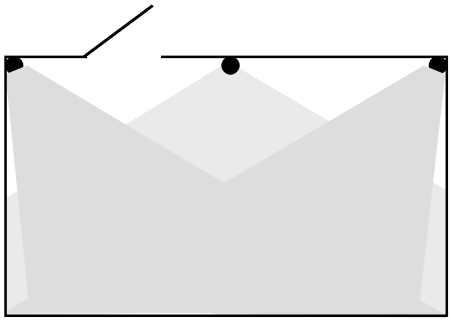

Figure 3. A room covered (mostly) by three IR beacons, which do not penetrate walls.

We may combine several short-range transmitters, if necessary (see Figure 3). Transmitters can be placed so as to cover (most of) a place such as a room, without it being practically possible to receive their signals from anywhere outside that place.

Constraints based on signal strength may be inadequate in certain circumstances, so these techniques are intended for 'good enough' rather than strong security. For example, infrared may be reflected off a user's shirt and thus propagate through the door in Figure 3. Radio signal strength may be a complicated function of, for example, girder placement in buildings. Moreover, an attacker that has an arbitrarily powerful transmitter or an arbitrarily sensitive receiver may be able to flout what are normally considered to be the limits of wireless technologies. However, we can make attacks reasonably difficult so as to thwart attempts in many circumstances. For example, we can control line-of-sight access to thwart directional antennae; we can use highly attenuating materials; and in general we can conduct experiments with source positioning to satisfy ourselves about practical limits to propagation.

4. Location Authentication Protocols

In the preceding section we developed a model of constrained channels and outlined how they can be constructed—from components with appropriate physical characteristics, and from other constrained channels. We now look more closely at protocols for location authentication.

In a location authentication protocol, the location of the principal in question is asserted; the task is to find out whether the assertion is true. The statement 'principal p is at location L' can be seen as a contextual predicate, denoted $\lambda_L(p)$. In general, we take p to be the sender of a given message. Note that $\lambda_L(p)$ is a statement about the

current location of *p* (or, rather, its location in a bounded interval up to the present).

As we have discussed earlier, a send- or receive-constrained channel can be used to authenticate contextual properties of the principal who uses the channel. If we can find a receive-constrained channel or a send-constrained channel on the predicate $\lambda_L(p)$, we can design location authentication protocols by requiring the principal in question to communicate over the constrained channel.

The basic idea of our approach to authenticating a principal's location is to employ a challenge-response protocol. The authenticator can choose a nonce and either ask the principal in question to send it back to the authenticator over a send-constrained channel; or send the nonce to the principal in question over a receive-constrained channel and check if the principal has received it.

If the authenticator has direct access to a physically constrained (e.g. range-bounded) channel, that is, if the authenticator is located inside location *L*, it is trivial to implement location authentication. For example, the authenticator can send a nonce using a Bluetooth transceiver located at *L* and check if the principal receives it. If the principal is actually within the range of the Bluetooth transceiver, he/she should be able to receive the data from the Bluetooth transceiver. Here, the Bluetooth radio link is used as a receive-constrained channel.

What we are interested in is the more general case where the authenticator does not have direct access to a physically constrained communication channel at location *L*, i.e. the authenticator is remote from location *L*. In such a case, we need to use a trusted channel proxy to connect the authenticator with the constrained channel or to turn a local constrained channel into a remote constrained channel.

Before we describe the protocols, we should clarify one assumption we make about our constrained channels. That is 'whoever uses a physically constrained channel must be physically located within the transmission range of that channel'. It is conceivable that our protocols can be defeated by an attacker who sends an agent to the place of the constrained channel and uses the agent to relay the communication between the attacker and the constrained channel. Such an attack works unless the underlying communication system allows sufficiently accurate measurement of the response time to find out if a message has traveled 'extra' miles. This is not generally feasible on the Internet.

But there are many cases where it is either difficult or not worthwhile to send an agent or put a relaying device at the place of interest. For example, to use a relaying device means that the attacker has to put it there beforehand but the whereabouts of 'there' is often unforeseeable. In these cases, it is safe to make the assumption that whoever communicates with the constrained channel at a certain location is physically there. In this sense, we are providing reasonable solutions for some practical problems.

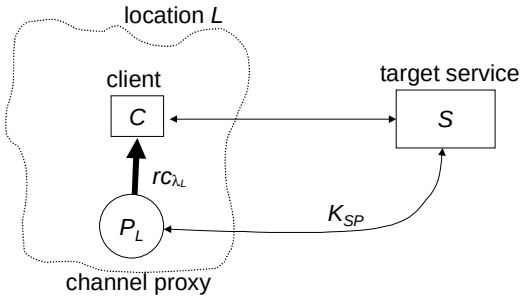

Figure 4. System model for the basic protocol

4.1. The basic protocol

This protocol uses a receive-constrained channel for location authentication (constructed using, e.g., Bluetooth or infrared). The principals in our protocol are the following (Figure 4):

- A server *S*
- A channel proxy P_L
- A client *C*

Server *S* wants to verify that client *C* is at a certain location *L*—that is, that $\lambda_L(C)$—before providing services. Channel proxy P_L is a process that connects securely to the send-end of a channel rc_{λ_L} at location *L* that is receive-constrained on λ_L. Channel proxy P_L shares a secret key K_{SP} with *S* and is trusted by *S* to behave correctly. Channel rc_{λ_L} could be a network whose physical reach is limited to *L*, such as one or more wireless LANs or infrared beacons. The constrained channel is such that data may be received from the channel only if the receiver is physically inside location *L*.

The goal of the protocol is for *S* to verify that $\lambda_L(sender(R))$, where *R* is a service request and *L* is the asserted location of $C = sender(R)$. The protocol proceeds using a challenge and response. The exchange occurs via a channel proxy, so that the authenticating entity does not need to be connected directly to the constrained channel. The protocol is as follows (we use the standard notation $\{M\}K$ for the encryption of *M* with key *K*):

(1) $C \rightarrow S$: C, R, L

(2) $S \rightarrow P_L$: $\{C, N\}K, \{K\}K_{SP}$
/* *N* is a nonce, *K* is a randomly chosen one-time session key */

(3) $P_L \rightarrow C$: C, N
/* broadcast over receive-constrained channel rc_{λ_L} */

(4) $C \rightarrow S$: C, R, N
/* S checks whether the received N is equal to the N sent to P_L */

In Step (2), the message sent is an encryption of $\{C, N\}$ using key K_{SP}. Session key K is used to defend against known-plaintext attacks on K_{SP}. Since only P_L knows K_{SP}, only P_L can receive $\{C, N\}$. Hence, the protocol effectively set up a receive-constrained channel $rc(P_L)$ on the predicate 'receiver is P_L' between S and P_L (see Section 3.2). Moreover, Step (2) and (3) together can be viewed as a single step in which S sends $\{C, N\}$ to C over the aggregated receive-constrained channel $rc(P_L).P_L.rc_{\lambda_L}$.

4.2. The private protocol

The basic protocol, above, does not protect client C's privacy since the identifier C is sent in the clear and may be received by others located in L, e.g., over a Bluetooth or 802.11 link. We can protect client C's identity by two measures: first, we employ a private channel (e.g. using SSL) between C and S; second, we avoid transmission of the identifier C over the constrained channel, which is assumed to be open. In all other respects we assume the same set-up as in the basic protocol. The following 'private protocol' is designed to protect C's privacy:

(1) $C \rightarrow S$: C, R, L, N_1
/* private channel; N_1 is a nonce generated by C */

(2) $S \rightarrow P_L$: $\{N_1, N_2\}K, \{K\}K_{SP}$
/* K is a randomly chosen session key and N_2 is a nonce generated by S */

(3) $P_L \rightarrow C$: N_1, N_2
/* broadcast over receive-constrained channel rc_{λ_L} */

(4) $C \rightarrow S$: C, R, N_2
/* private channel; S checks if the received N_2 is equal to the N_2 sent to P_L on behalf of C */

The only message not sent over a private channel is the one in step 3; however, that message does not contain C's identifier: C recognises N_1 and thus sends the message of step (4), which S can easily verify. In the course of the private protocol, others may observe at most a correlation between a service-access and an origin MAC and IP address—the latter (and the former, in some technologies) being typically dynamically and anonymously assigned.

4.3. The offline protocol

The above two protocols assume that S and P_L can communicate directly. But this is not always so: for example, the client may be a mobile phone that accesses S over GSM and makes a Bluetooth connection to P_L, but with no connection between S and P_L. In such cases, we can use an 'offline' protocol to achieve our goal.

We assume the same principals and the same set-up as in the basic protocol except that S and P_L cannot communicate directly (although they do share a secret key K_{SP}). The offline protocol follows:

(1) $C \rightarrow S$: C, R, L

(2) $S \rightarrow C$: $\{N_1\}K_{SP}, \{N_1 \otimes N_2\}K_{SP}$
/* N_1, N_2 are nonces; '⊗' denotes the exclusive-or operation */

(3) $C \rightarrow P_L$: $\{N_1\}K_{SP}, \{N_1 \otimes N_2\}K_{SP}$

(4) $P_L \rightarrow C$: $\{N_2\}K_{SP}$
/* sent over receive-constrained channel rc_{λ_L} */

(5) $C \rightarrow S$: $C, R, \{N_2\}K_{SP}$
/* S checks whether the received N_2 is equal to the N_2 sent in Step (2) */

Since P_L is the only party (except S) that can compute $\{N_2\}K_{SP}$ from $\{\{N_1\}K_{SP}, \{N_1 \otimes N_2\}K_{SP}\}$, the fact that C can show the correct $\{N_2\}K_{SP}$ means that C is able to receive it from the receive-constrained channel rc_{λ_L}. Therefore, it verifies that C is at location L. Note that no nonce is sent in the clear, thus we avoid known-plaintext attacks.

In Step (4), C receives $\{N_2\}K_{SP}$ over a receive-constrained channel rc_{λ_L}. However, if we view Steps (3) and (4) combined as the precondition for Step (5), the protocol is effectively turning a receive-constrained channel from P_L to C into an aggregated send-constrained channel from C to S—as we outlined in Section 3.3.

4.3 Implementation over HTTP

We implemented the offline protocol to enable transparent authentication of the location of a standard web browser used to access services. The implementation (Figure 5) is independent of the target service in order to be reusable between service implementations. It largely consists of a servlet that acts as a location-authenticating 'stub' local to the service, and a servlet that implements the location authentication proxy (not to be confused with an HTTP proxy) at the desired location.

Suppose a client's browser invokes the target service's URL (message 1 in Figure 5). The service invokes its local stub to authenticate the request, which sends a response (3) in the form of an HTTP redirection to the location authentication proxy (the channel proxy at the location in question), containing the challenge encoded in its URL. Similarly, the response from the proxy (5) is an HTTP redirection back to the stub, containing the response encoded in the URL. Hence, the browser is transparently directed to the proxy to authenticate its location.

The protocol establishes the location-authenticity of the client as an 'https' (SSL/TLS) connection endpoint. Multiple requests could appear from that client and the target service would know that they were all from the

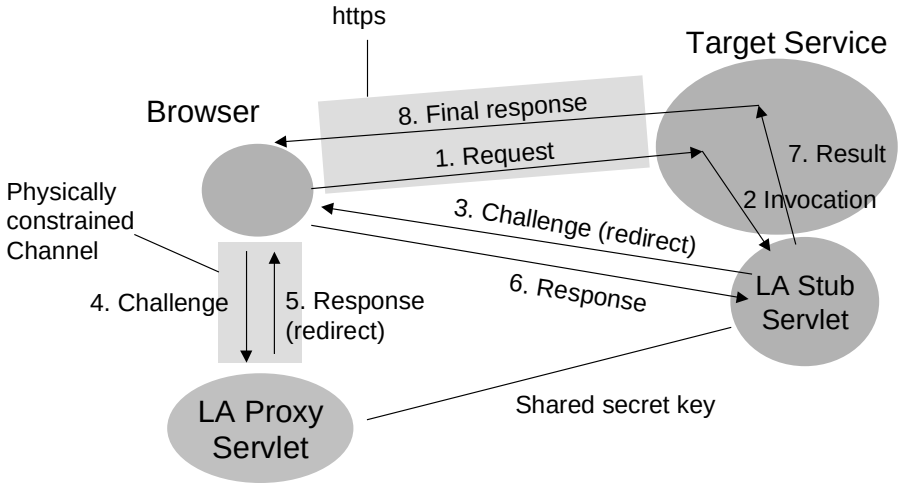

Figure 5. Location authentication ('LA') over HTTP

same client, which at the time of the initial request was in the required location. In a world with mobile IP, or a world in which the network used for the SSL connection (which is not necessarily the same as the physically constrained network) extends beyond the location, the target service could not tell whether the client remained at the location. However, the target service could time out the client and force it to undergo location authentication again.

With regard to performance, SSL imposes encryption overheads that may be beyond the capabilities of some small devices but we found that it was well within the capabilities of an HP Jornada 540 Pocket PC. Furthermore, the protocol's set-up costs may be amortised over multiple requests.

In some situations, the client could be in any of several locations to obtain the service, e.g. any of the Kardomah coffee houses where customers may print as long as they visit the establishment. In such a case, the target service needs to know which proxy to address itself to, to establish location-authenticity. That can be achieved in several ways, e.g.:

1. The target service sends back a form with a pick-list of possible locations and the user specifies it.
2. In CoolTown [2], the target service is offered directly through a URL emitted by a beacon in the location, or indirectly from a page obtained from such a beacon. In either case, it can be arranged that the URL of the service already contains, in its query component, the identity of the location.

5. Conclusion

We have described protocols for location authentication based on physical communication channels that are subject to signal propagation constraints. We described our implementation of the private protocol over HTTP, which is a generic implementation suitable for a variety of services accessed from a standard Web browser.

We also contributed a general model of send- and receive-constrained channels for context authentication. We showed how we can 'lengthen' channels constrained on location predicates. Moreover, channel proxies enable us to broaden constraints to temporal and other contextual predicates. We showed how to construct send-constrained channels from receive-constrained channels and *vice versa*.

In Section 1, we listed five problems that exemplify the types of context authentication in which we are interested. The first four are examples of location authentication; the third and fourth involve self-verification of location (a computer inside a building, a kiosk near to a human carrying a short-range radio transceiver). That can be achieved straightforwardly using our techniques, by having the device act attempt to authenticate its own location.

The fifth asks us to authenticate that a principal is in a certain 'virtual location'. That problem falls under the techniques we have given for physical location. A server can place a nonce on the web page, of which the client (which must 'visit the URL') must demonstrate knowledge.

Our examples show that our constrained channel concept and protocols are powerful enough to capture certain contextual properties of communicating principals, and consequently can be used to authenticate those properties. We believe that protocols that integrate physical properties and logical (cryptographic) properties will have increasing importance for ubiquitous computing. We are not yet at the stage where suspects can

appeal to an ‘alibi service’ but we expect such a service to arrive.

References

[1] John Barton & Tim Kindberg (2001). “The challenges and opportunities of integrating the physical world and networked systems”. HPL Technical report HPL-2001-18.

[2] Tim Kindberg & John Barton (2001). “A Web-Based Nomadic Computing System”. Computer Networks, Elsevier, vol 35, no. 4, March 2001, pp. 443-456.

[3] Tim Kindberg et al. (2000). “People, Places, Things: Web Presence for the Real World”. In proceedings WMCSA2000, IEEEE Computer Society, Dec. 2000, pp. 19-28.

[4] Debbie Caswell & Philippe Debaty (2000). “Creating web representations for places”. Proceedings Handheld and Ubiquitous Computing 2000, Springer, pp. 114-126.

[5] R. Want, A. Hopper, V. Falcao, and J. Gibbons (1992). “The active badge location system”. ACM Transactions on Information Systems, vol. 10, pp. 91-102.

[6] Dorothy E. Denning and Peter F. MacDoran (1996). “Location-Based Authentication: Grounding Cyberspace for Better Security”. In Computer Fraud & Security, February. http://www.cosc.georgetown.edu/~denning/infosec/Grounding.txt.

[7] T. Dierks and C. Allen (1999). “Transport Layer Security”. RFC 2246. www.ietf.org.

[8] A. Harter, A. Hopper, P. Steggles, A. Ward, P. Webster. “The Anatomy of a Context-Aware Application”. *Proc. 5th Annual ACM/IEEE Int’l Conf. on Mobile Computing and Networking*, Seattle, Washington, USA, August 1999, pp. 59-68.

[9] Y. Desmedt. “Major security problems with the ‘unforgeable’ (Feige)-Fiat-Shamir proofs of identity and how to overcome them,” SecuriCom’88, SEDEP Paris, 1988, pp 15-17.

[10] S. Brands and D. Chaum, “Distance-Bounding Protocols,” Proc. EUROCRYPT ‘93, Lecture Notes in Computer Science, no. 765, Springer-Verlag, pp. 344-359.

[11] Ensure technologies. www.ensuretech.com.

[12] Eran Gabber and Avisahi Wool, "On Location-Restricted Services", IEEE Network, November/December 1999, pp. 44-52. Special Issue on Networking Security.

[13] Eran Gabber and Avishai Wool, "How to Prove Where You Are", Proceedings of the 5th ACM Conference on Computer and Communications Security, San Francisco, California, November 3-5, 1998, pp. 142-149.

[14] Riccardo Focardi. “Using Entity Locations for the Analysis of Authentication Protocols”, in proceedings of Sixth Italian Conference on Theoretical Computer Science (ICTCS ’98), Prato, 9-11 November 1998.

[15] Robin Milner. “Communicating and mobile systems: the π-calculus”, Cambridge University Press, 1999.