Context Authentication Using Constrained Channels

Tim Kindberg, Kan Zhang
Internet and Mobile Systems Laboratory
HP Laboratories Palo Alto
HPL-2001-84
April 2nd , 2001*

E-mail: timothy@hpl.hp.com, kzhang@hpl.hp.com

context-aware computing, location-based computing, security, authentication

This paper presents a paradigm shift from conventional authentication—of a principal's identity—to authentication of parameters that characterise a principal's context. Location, in particular, is a highly significant contextual parameter. It is one that features in what are known as mobile, ubiquitous, pervasive and nomadic computing systems. We present a model of context authentication based on the characteristics of communication channels. As an example, we present protocols for location authentication that are based on physical channel characteristics. We conclude with a summary and discussion of the work.

* Internal Accession Date Only
Approved for External Publication

Context authentication using constrained channels

Tim Kindberg & Kan Zhang

Internet and Mobile Systems Laboratory
Hewlett-Packard Laboratories
1501 Page Mill Road
Palo Alto, CA 94304, USA
timothy@hpl.hp.com, kzhang@hpl.hp.com

Abstract

This paper presents a paradigm shift from conventional authentication—of a principal's identity—to authentication of parameters that characterise a principal's context. Location, in particular, is a highly significant contextual parameter. It is one that features in what are known as mobile, ubiquitous, pervasive and nomadic computing systems. We present a model of context authentication based on the characteristics of communication channels. As an example, we present protocols for location authentication that are based on physical channel characteristics. We conclude with a summary and discussion of the work.

1 Introduction

This paper describes a model and protocols 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*. What is really proved is possession of *K*, and the association between *p* and *K* is a given that lies outside the protocol.

However, in some circumstances we are interested in the characteristics of a principal's context, such as their location, in addition to or instead of their 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 has 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 to be provided with service *S*.
6. A government document is not to be accessible before January 1st, 2002.

We can 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 ϕ (*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. We shall show how communication channels can be the trusted artifacts that verify contextual predicates as they apply to communicating principals.

Section 2 outlines related work. Section 3 presents a model that captures the essential features of context authentication in terms of constructs called constrained communication channels. Section 4 demonstrates the application of the model to the particular case of authenticating a principal's location. 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]. Contextual data are usually collected via sensing technologies, e.g., GPS and the Active Badge [5]. However, to our knowledge, very little has been published on authenticating contextual data. Many sensing-based approaches are intrinsically difficult for use in authentication. 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 (LSS) to compute a location signature from the microwave signals transmitted by the GPS satellites. Location signatures are hard to forge since GPS observations are unpredictable. 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. Moreover, there is commercially available test equipment to simulate GPS satellites, which transmit a signal appropriate for any location entered into them.

Location or distance information has been found useful in designing cryptographic protocols. For example, the so-called "mafia frauds" [9] against identification protocols work when a fraudulent prover is able to use an honest verifier as an oracle without them noticing it. Knowing the physical distance between the prover and the verifier can prevent a fraudulent prover from using a distant honest prover. 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. They showed how to integrate their technique into common identification protocols and also some three-party protocols.

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 some new location authentication protocols that do not use time.

3 The model

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. In this section, we first define such channels and then show how they can be realised.

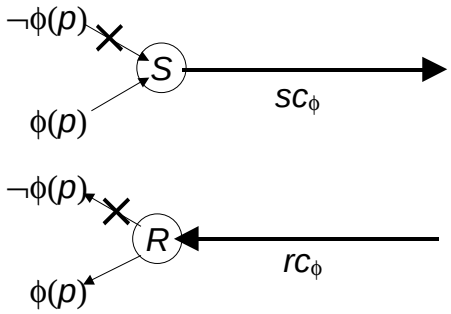

Figure 1. Send-constrained and receive-constrained channels

Any one-way channel *c* has message *send* and *receive* operations 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 the predicate ϕ: if $m = sc_\phi.receive()$ then ϕ (*sender*(*m*)).

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

These definitions capture some properties established by conventional security protocols. For example, consider two principals connected by a Transport Layer Security (TLS) 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 much 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.

The telephone system

The telephone system provides a simple example of constrained channels—at least, assuming that we were to trust certain aspects of its implementation. First, it provides receive-constrained channels on predicates of the form 'is in the location *L*'. If a principal wishes to impart some information to any principal who is in a particular place, they can do so by knowing the

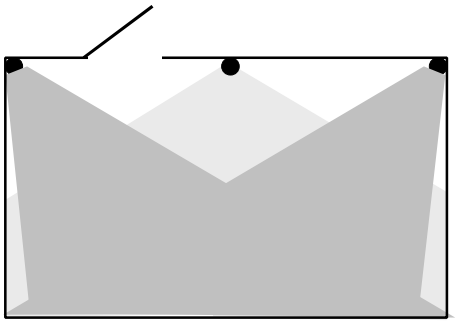

Figure 2. A room covered (mostly) by three IR beacons, which do not penetrate walls.

appropriate number to dial for a telephone fixed in the place. Assuming that we trust the integrity of the system, anyone who answers and receives that message is near to the phone. (In films, this is a technique favoured by kidnappers arranging to pick up a ransom payment!)

Caller-id provides us with send-constrained channels on predicates of the form 'is in the location *L*'. The user of a telephone may choose to answer only calls that originate from a specified telephone number—in particular, one that is wired in a fixed location. The user plus the phone system together implement a send-constrained channel.

3.1 Implementing constrained channels

The telephone system shows how we can exploit a channel with appropriate mechanical characteristics. Our telephone examples assume that the series of links and logic circuits connecting one telephone to another is proof against tampering, and that telephone circuits are not physically moved. That type of assumption is more likely to hold reliably in the case of short-range wired links in physically secured environments.

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.

Constrained propagation

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 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. IR [3, 5] and ultrasound [8], on the other hand, have the useful property that walls tend to attenuate their signals to a negligible level.

We may combine several short-range transmitters, if necessary (see Figure 2). 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.

In principle, 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 it reasonably difficult for an attacker to do so. For example, we can control line-of-sight access to thwart directional antennae, and use highly attenuating materials.

3.2 Constrained channels as building blocks

We can use constrained channels as building blocks for making further constrained channels. We do so by inserting a proxy between two constrained channels; and by running a protocol that turns a send-constrained channel into a receive-constrained channel, or *vice versa*.

A *channel proxy P* is a process that connects exactly 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 channel proxy receives messages from channel c_1 and selectively forwards the messages on to channel c_2. 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 complex $C = c_1.P.c_2$ behaves as a (possibly lossy) channel, if we identify $C.send \equiv c_1.send$, and $C.receive \equiv c_2.receive$.

One particular 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*).

Appending channels

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

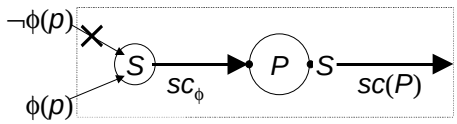

Figure 3. A channel proxy interposed to construct a new send-constrained channel.

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 3). 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 ϕ ($receiver(m)$)—if P forwards m.

Channel proxies and contextual parameters

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: a proxy could delay all messages until midnight 1/1/2002.

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 $\phi \wedge \psi$.

Reversing 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 $\text{sig}\{h(m)\}$.

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 $\text{sig}\{h(m)\}$ from N over rc_ϕ
> send $<m, \text{sig}\{h(m)\}>$ on c.

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

> receive $<m, h>$ on c
> verify $h = \text{sig}\{h(m)\}$
> Discard m if verification fails, else receive m.

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.

4 Location authentication protocols

In the preceding, 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 the particular case of location authentication, filling in the more important details that are necessary for practical purposes.

A location authentication protocol enables an authenticator to verify their own location or the location of another principal. The statement 'principal p is at location L' can be seen as a contextual predicate, which we denote $\lambda_L(p)$. 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.

Note that location authentication is different from location identification, where the aim is to determine a principal's location, e.g., using a GPS system. In a location authentication problem, the location of the principal in question is asserted; the task is to find out whether the assertion is true.

In this section, we shall introduce several location authentication protocols based on constrained communication channels. As we stated above, there are many short-range communication technologies that can be used to implement constrained communication channels for location authentication, with various degrees of precision. For example, a Bluetooth radio or an IR transceiver can be used as either a receive- or a send-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. We outlined such protocols in Section 3.2. In the following, we give more detailed protocols for this general case.

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. Finally, we want to point out that it does not mean that the model we presented earlier is flawed. It only means we need some assumptions about the physical world to get the desired constrained channels.

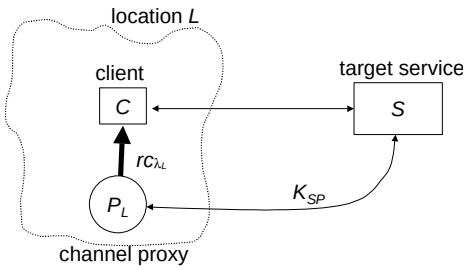

Figure 4. System model for telephone protocol

4.1 The telephone protocol

This protocol uses a receive-constrained channel for location authentication. The principals in our telephone protocol are the following (see 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 (device) 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 telephone 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 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 and *K* is a randomly chosen 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 $rc(P_L).P_L.\ rc_{\lambda_L}$.

We call this the 'telephone protocol' because a telephone could, in principle, serve to implement rc_{λ_L} (it rings and C must answer it to prove his presence in L).

Our protocol does not address issues of identity authentication or anonymity. Those are important for nomadic computing systems but they are orthogonal to our current purposes.

4.2 The private telephone protocol

The above telephone protocol does not protect client C's privacy since $\{C, N\}$ sent in Step (3) may be broadcast to all receivers located in L, e.g., over a Bluetooth link. An eavesdropper could pick up $\{C, N\}$ and learn about C's identity. Suppose we have the same principals and the same set-up as in the telephone protocol, except that, for privacy reasons, C and S share an encrypted channel. We can protect client C's identity by using the following private telephone protocol:

(1) $C \rightarrow S$: C, R, L, N_1
/* 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
/* S checks if the received N_2 is equal to the N_2 sent to P_L */

This protocol is similar to the telephone protocol. The difference is that a random identifier (N_1) is used to protect the identity of client C.

4.3 The offline protocol

The above two protocols assume that S and P_L can communicate in real-time. In cases where this is not true, we can use an 'offline' protocol to achieve our goal.

We assume the same principals and the same set-up as in the telephone 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; '$\otimes$' 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 3.2.

This protocol can be implemented transparently for a standard web browser. Suppose a client's browser C contacts a web server S for services in Step (1). The response coming back from S in Step (2) includes an HTTP redirection pointing to a local channel proxy P_L. Similarly, the response from P_L in Step (4) includes another HTTP redirection pointing back to S. Hence, the browser can be transparently directed to contact P_L for authenticating its location.

4.4 The self-verification protocol

Interestingly, a constrained channel can be used to verify a principal's own location. In the case of a receive-constrained channel, a principal can send a message to a receive-constrained channel rc_{λ_L} for location L and check if he can receive the same message from this channel. The principal has to know how to send a message to rc_{λ_L} but the self-verification protocol itself is trivial once the receive-constrained channel is available. The difficulty lies in how to construct rc_{λ_L} so that its guarantees are securely implemented.

Alternatively, a principal can send a message to a local send-constrained channel sc_{λ_L} at location L and check if

the same message can be received from sc_{λ_L}. Again the self-verification protocol itself is trivial once the send-constrained channel is available.

A real-life example arises from using the fixed-line telephone system (again, assuming that that system were to be sufficiently secure). If a user knows the telephone number of the fixed phone located at a certain place, the user can check if they are in that place by calling that number using, for example, a mobile phone and seeing whether the phone at their current location rings. In this way, they are using the fixed-line telephone system as a receive-constrained channel.

Alternatively, the user can employ the fixed-line telephone system as a send-constrained channel by calling their mobile phone from the fixed phone at their current location. The user checks whether the caller-id shown on the mobile phone matches the phone number of the place whose verification is in question.

5 Conclusion

We have described a model of send- and receive-constrained channels for context authentication. We have shown how to construct simple examples of constrained channels on location predicates. For this purpose, we use physical communication channels that are subject to mechanical or signal propagation constraints.

We further showed how, by inserting channel proxies, we can 'lengthen' channels constrained on location predicates. Moreover, proxies enable us to broaden channel constraints to temporal and other contextual predicates. We showed how to construct send-constrained channels from receive- constrained channels and *vice versa*.

Finally, we gave practical protocols for location authentication, including one that protects clients' privacy. We outlined how components can use similar protocols to authenticate their own location.

Applicability

In Section 1, we listed six 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). We have shown how to achieve those types of authentication in Section 4.

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 (who must 'visit the URL') must demonstrate knowledge.

The final example illustrates a temporal predicate. We have shown how to use a channel proxy to achieve that.

Status

We are engaged in an implementation of a location authentication protocol for CoolTown places. Users with handheld devices running standard web browsers will be able to prove their presence in, for example, a coffee shop or bookshop, and thus obtain privileged services such as printing-on-the-go, or a discount.

In this paper, we have contributed a new abstraction—the send- or receive-constrained channel—to help us formulate a notion of context authentication that can be realised. We have given protocols for some simple cases involving location, based on assumptions about signal propagation. It remains to show the true practicality of implementing a location authentication system in a real situation, and extending the work to a broader notion of context beyond toy examples such as time, to complex notions of, for example, user-presence.

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, available as http://www.champignon.net/TimKindberg/MobicomChallengeAsTR.pdf.

[2] Tim Kindberg & John Barton (2001). "A Web-Based Nomadic Computing System". To appear, Computer Networks, Elsevier. Available as http://www.cooltown.hp.com/papers/nomadic/nomadic.htm.

[3] Tim Kindberg et al. (2000). "People, Places, Things: Web Presence for the Real World". In proceedings WMCSA2000. Available as http://www.cooltown.hp.com/papers/webpres/-webpresence.htm.

[4] Debbie Caswell & Philippe Debaty (2000). "Creating web representations for places". Proceedings Handheld and Ubiquitous Computing 2000, Springer, pp. 114-126. Available as http://www.cooltown.hp.com/papers/placeman/placesHUC2000.pdf.

[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. Available as 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.