Context-Aware User Authentication — Supporting Proximity-Based Login in Pervasive Computing

Jakob E. Bardram, Rasmus E. Kjær, and Michael Ø. Pedersen

Centre for Pervasive Computing
Department of Computer Science, University of Aarhus
Aabogade 34, 8200 Århus N, Denmark
{bardram,rkjaer,michael}@daimi.au.dk

Abstract. This paper explores computer security in pervasive computing with focus on user authentication. We present the concept of *Proximity-Based User Authentication*, as a usability-wise ideal for UbiComp systems. We present a context-aware user authentication protocol, which (1) uses a JavaCard for identification and cryptographic calculations, (2) uses a context-awareness system for verifying the user's location, and (3) implements a security fall-back strategy. We analyze the security of this protocol and discuss the tradeoff between usability and security. We also present our current implementation of the protocol and discuss future work.

1 Introduction

The notion of ubiquitous and pervasive computing implies a shift in the use of computers. We are going from the personal computing paradigm, where each user is using a personal computer, to the pervasive computing paradigm, where computers are available in huge numbers, embedded in everyday artifacts, like phones, furniture, cars, and buildings. Hence, each user is using many personal computing devices, and at the same time, the same publicly available device is used by many users. This shift from a 'one-to-one' to a 'many-to-many' relationship between users and computers sets up some new usability challenges for computer security, especially user authentication. Contemporary user authentication schemes involve typing in usernames and passwords. When using a personal computer, typing in username and password is straightforward, but still it poses substantial usability problems in some work environments, like hospitals [4, 23]. In the pervasive computing paradigm, these usability problems are increasing, because the user is using many computers. Imagine that a user would need to type in username and password on all 'pervasively' available computers before he could start using them. Clearly, if the pattern of login and logout is not considered a usability problem today, it will most certainly become one in the years to come.

In this paper we describe our response to such usability problems of user authentication in a pervasive computing environment. Our aim is to support what we have termed *proximity-based login* (see also [11]), which allows users to be authenticated on a device simply by approaching it physically. The idea of enabling users to access a computer by simply walking up to it has a long history in ubiquitous computing research. This idea of proximity-based login can be traced back to the pioneering work on

the Active Badge System, which could be used to 'teleport' an X Window session to a display located in front of the user [27, 5]. Similarly, the AT&T Active Bat system [28] was used to create 'Follow-me Applications' where the user-interfaces of the user's application can follow the user as s/he moves around. The application is shown on the display that is deemed to be in front of the user as established via the user's Active Bat [16]. The idea has later been adopted by the Microsoft 'EasyLiving' project [7]. In other pervasive computing environments different types of physical tokens are used as user identification. In the BlueBoard project at IBM, a HID brand reader is used [22, 21]. In the AwareHome at Georgia Tech RFID tags provide identity of individuals near commonly used monitors [19], and at FX PAL the Personal Interaction Points (PIPs) System uses RFID cards that stores the users identification as well as passwords (the latter in encrypted form) [26]. Ensure Technologies' XyLoc system [15] uses short-range radio communication between a personal token and a PC to establish the proximity of a user and to unlock the PC by pressing a button on the token, not unlike the ones used to remotely unlock cars.

Common to these systems is the lack of proper security mechanisms that can effectively ensure a secure user authentication. In case of theft of a token, or by recording and replaying the communication between the token and the reader, an adversary can access the system and impersonate the legitimate user. The Smart Badge platform [24] uses some improvements to reduce the problem of stolen tokens. The method is to use a badge that can detect when it is no longer being carried. The link between the badge and a particular user can then be removed, and the badge can no longer be used for authentication purposes. It is however difficult to judge from the paper exactly how such a smart badge can sense whether is is being carried by its legitimate user. The Zero-Interaction Authentication method of [12] takes an approach similar to the XyLoc system. A token is used to gain access to a laptop with an encrypted file-system. To verify the user, s/he is required to enter a PIN code on the token before s/he can start using it.

There is often an inherent tradeoff between usability and security. User authentication mechanisms tend to be either secure, but less usable, or very usable, but less secure. It is our aim to try and combine the two standpoints and suggest a *context-aware user authentication mechanism* that is very usable as well as sufficiently secure for use in settings, where security matters – like a hospital environment. Traditionally a user authentication mechanism is considered secure if it is a combination of something the user *has* (e.g. a smartcard), something the user *knows* (e.g. a password), or something the user *is* (i.e. a fysiological trait) [25]. Our design of a user authentication mechanism is based on supplementing well-known user authentication mechanisms with knowledge about the context and location of the user. In line with Denning [14] we thus suggest location-based authentication and introduce *'location'* as a fourth element in a user authentication mechanism. The paper starts by outlining the background and motivation for the design of context-aware user authentication. Section 3 presents our design of a proximity-based user authentication mechanism and its degree of security is analyzed in section 4. Section 5 describes our current implementation of the proposed protocol, and section 6 discusses our work before the paper is concluded.

2 Background and Research Methods

The work on proximity-based user authentication takes place within the research area of *Pervasive Healthcare* [10]. A central area of research is to design and develop pervasive computing technologies for the use at hospitals. We approach this challenge in two ways. On the one hand we conduct ethnographic studies of clinical work in hospitals and how technology is used here. And on the other hand we engage clinicians in experimental design and development of technologies in various types of workshops in our laboratory.

2.1 Troubles with Login

In a study of the use of Electronic Patient Records (EPR) at a large metropolitan hospital, we observed a number of usability problems associated with user authentication [4]. The EPR was accessed through PCs distributed within the hospital, and it had a traditional login system with usernames and passwords. Thus, whenever a clinician should access patient information s/he had to log in and out on different PCs. Due to the way the PCs were deployed and the nature of the work in hospitals, it was not uncommon that a nurse, for example, would log in 30 times a day. Because this was a highly cumbersome thing to do in a hectic environment, workarounds were established. For example, users would avoid logging out, enabling them to return to the PC without logging in later; passwords were shared among users and made very easy to remember ('1234' was the most used password at the hospital); and users would often hand over user sessions to one another, without proper logout and login. Hence, what was designed to be a secure system (with traditional username and password user authentication) was suddenly turned into a highly insecure system, because of obvious usability problems.

2.2 Activity-Based Computing

Even though this EPR system in no way can be termed as 'pervasive technology', our study of its use has highlighted how essential user authentication is to the design of pervasive computer support for medical work in hospitals. In our second line of research we actively design and develop pervasive computing technologies for hospital work. A central component in this effort is a basic runtime infrastructure, which supports Activity-Based Computing (ABC) [11]. The basic idea of activity-based computing is to represent a user's (work) activity as a heterogeneous collection of computational services, and make such activities available on various stationary and mobile computing equipment in a hospital. Clinicians can initiate a set of activities, and access these on various devices in the hospital. For example, a nurse can use the computer in the medicine room to get some medicine, and later when giving this medicine to the patient she can restore the patient and medicine data on the display in the hospital bed. We have built prototypes of wall-size displays, displays embedded in tables, built-in computers in hospital beds, and we are using various mobile equipment like TabletPCs and PDAs. Figure 1 illustrates how a physician is using a wall-based display in a conference situation. Thus, activity-based computing allows users to carry with them, and restore, their work on heterogeneous devices in a pervasive computing environment. Central to this

is clearly that users need to be authenticated on every device they want to use, and easy login is hence a core challenge in the concept of activity-based computing.

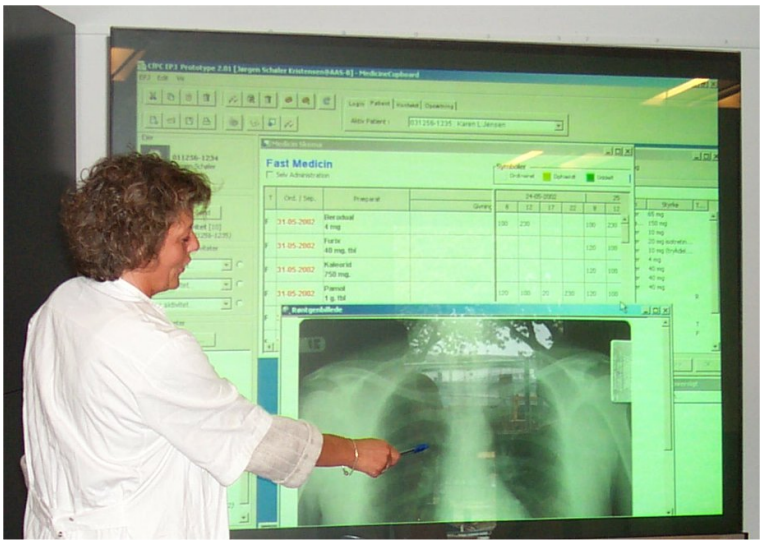

Fig. 1. A physician is using a wall-based display in a conference situation. In her hand she is holding a *Personal Pen*, which is used to authenticate her to the computer. An active badge woven into her white coat (not visible) is revealing her location to a context-awareness system.

Our design is based on participatory design sessions and workshops with a wide range of clinicians, including physicians, radiologists, surgeons, and different types of specialized nurses. All in all 12 such workshops were conducted, each lasting 4-6 hours having 6-10 participants each. Various aspects of designing support for clinical work were discussed, including the login mechanisms. Several user authentication mechanisms were designed, implemented, and evaluated in these workshops.

2.3 Requirements for a Pervasive Computing User Authentication Mechanism

Based on existing research within UbiComp, our studies of medical work, and our experimental design effort with end-users, we can list the following requirements for a user authentication mechanism in a pervasive computing environment.

– *Proximity-based* – Work at a hospital is characterized by busy people who are constantly moving around, and are engaged in numerous activities in parallel. Easy and fast login was thus deemed a fundamental prerequisite for the success of a distributed, pervasive computing infrastructure, embedded in walls, floors, tables, beds, etc. The usability goal in our workshops reached a point where the user should do nothing to log in – s/he should simply just use the computer, and the computer would know who the user was.

– *Secure* – Clinical computer systems store and handle sensitive, personal health data for many patients. It is therefore of utmost importance that these systems are protected from unauthorized access. Hence, pervasive computer systems in a healthcare environment require secure user authentication. This is used to set up the right user authorizations for reading and altering clinical data. For example, clinicians may only access clinical data related to patients, they are treating, and may not, for example, access clinical data on arbitrary persons. This is done to ensure the privacy of patients. Similarly, only physicians may prescribe medicine. One of our early designs for a user authentication mechanism was based on RFID tokens alone. This mechanism, however, was abandoned for obvious security reasons.
– *Active gesture* – We also experimented with a login mechanism that automatically would transfer a user's on-going session to a display nearby him – much like the 'Follow-me' application using the Bat system [16]. This, however, turned out to be a less useful design. The problem was that often a clinician would enter a room, where numerous computing and display devices would be available. For example in a radiology conference room, there would be several wall-based displays, a wide range of desktop computers, and an interactive table where images can be displayed and manipulated. It was unclear from monitoring the location of the user, which of such displays he would like to use – or whether he wanted to use a computer at all. Therefore the authorization mechanism must be based on an active gesture nearby the display or devices that the user wants to use.
– *Support for logout* – During our experiments we discovered that the process of logging out a user is equally important. Clinicians would often have to hurry on, and would simply walk (or run) away from an ongoing session. In this case, automatic logout was deemed important. Even though this is normally not considered to be part of a user authentication mechanism, we argue that logout has to be considered as a part of the design as well.

3 Context-Aware User Authentication

There are three key principles in our design of a context-aware user authentication mechanism. First, it uses a physical token used for active gesturing and as the cryptographic basis for authentication. Second, it uses a context-awareness system to verify the location of the user, and to log out the user when s/he leaves the computer(s) in a certain place. Third, it contains 'fall-back' mechanisms, so that if either of the two components in the system falls out, the user authentication mechanism switches to other mechanisms. If the context-awareness infrastructure is unreachable for any reason, the user is requested to enter his password when trying to log in. If the token cannot be accessed for any reason, the user is requested to enter both his username and password, as usual. Hence, in case of system failure, security is not compromised, and the system is still usable. We shall return to a more detailed security analysis below.

Our current design uses smart card technology and in the rest of the paper we will present this design based on a smart card as the physical token. Furthermore, we shall only consider the part of the authentication protocol that involves the smart card. Standard authentication using username and password is not further discussed. The two

basic components in the context-aware user authentication mechanism are (1) a secure user authentication protocol between a computer and a smart card, and (2) a distributed computing context-awareness infrastructure.

3.1 Authentication Protocol

The authentication protocol is running on a JavaCard [3]. Each client is equipped with a card reader and the protocol is executed every time a user inserts his card into the reader on the client. In order to use the card for authentication, the following information is stored on the card:

- An id for the user the card belongs to.
- The user's $password$.
- The user's pair of a secret key (K_S) and public key (K_P).

When the card is issued, an applet[1] for authentication is stored on the card, and initialized with the user's $password$ and the id of the user. When the applet is initialized, the card creates an RSA key-pair. The secret key (K_S) is stored on the card and the public key (K_P) is stored in a central server along with the id of the user. The authentication protocol is illustrated in figure 2 and consists of the following steps:

1. The client receives notification that user P is in the room (optional).
2. The user places his smart card in the card reader.
3. The client requests the id from the smart card.
4. The client looks up the person in the Context Server based on the id from the card.
5. There are two distinct cases based on the probability that the user is in the same place as the client.
   Case A: The probability is greater that a certain threshold.
   - The smart card is asked to verify that it holds the user's secret key, K_S.

   Case B: The location of the user is not sufficiently sure.
   - The computer asks the user to enter his $password$.
   - The smart card accepts or rejects the user based on the password.
6. The user is either denied or allowed access.

In case A, where the user is known to be in the room, the client verifies that the smart card knows the user's secret (private) key K_S by generating a random 20 byte "nonce", N, and sends it to the card. The card then sends back the signature under the private key, $sig\,(K_S, N)$, of N, and the client uses the corresponding public key, K_P, to verify that the signature is correct.

In case B, the user is not known to be in the room. The client asks for the user's password, concatenates it with a 20 byte nonce N, encrypts it under the user's public key, $E\,(K_P, password + N)$, and sends it to the card. Since the card knows the secret key, it can decrypt this message. It then compares the received password with the one stored on the card. If they match the card returns a byte R and a signature on R concatenated with N, $R + sig\,(K_S, R + N)$, where R equal to 1 is accept and 0 reject. The client uses the public key, K_P, to verify the signature.

[1] Programs running on JavaCards are called applets. They bear no resemblance with the applets running in web browsers, except for the name.

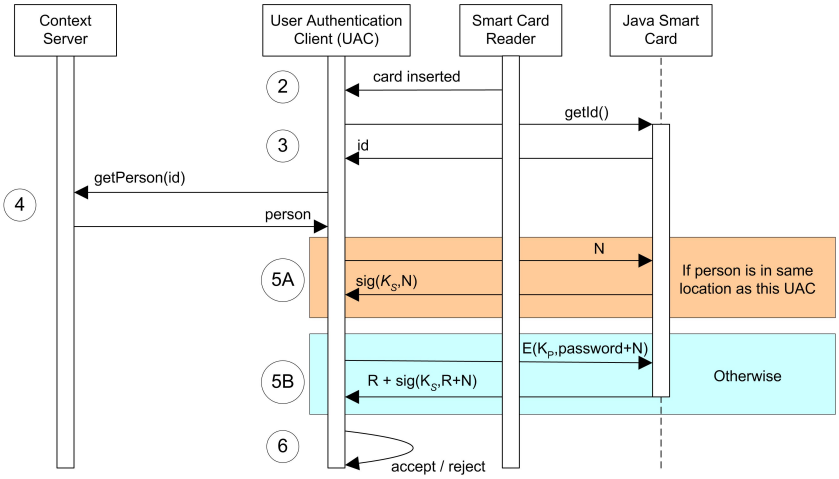

Fig. 2. Interaction Diagrams for the Authentication Protocol. Case A – The person is in the same location as the User Authentication Client (UAC). Case B – The person is not in the same place, and the user is requested to enter his or her password.

3.2 Infrastructure

The system architecture for the context-awareness infrastructure is illustrated in figure 3, and consists of the following main components:

- *Context Monitors* – A range of hardware and/or context data specific processes, which register changes in the environment. The monitor adapts this context information according to the data model in the Context Server. Examples of context monitors are location monitors based on monitoring RFID tags attached to items in the environment, or WLAN monitors that try to locate WLAN-based equiptment. Other monitors might gather information about temperature, planned activities in users' personal calendars, or try to identify people in a room based on their voices. Currently we mainly use a Portal RFID antenna (see figure 5) for locating people. When a portal monitor scans an RFID tag, the RFID adapter translates the 64 bit tag id into telling the context server that a specific entity has been seen at a specific location. Even though our current location mechanism (also) is based on something the user has (a RFID tag), the architecture of our context-awareness infrastructure enables us to create other location monitors based on e.g. audio or video.
- *Context Server* – The Context Server contains a simple data structure that stores information about 'Entities' in the environment. Entities are basically people, places, or things, but this structure is extensible and all kinds of context data can be stored by implementing some simple interfaces.

From a client's point of view there are basically two ways to connect to the Context Server. On the one hand, a client can look up an entity and ask for its context information, like location. For example, a client can request a person and ask for the location.

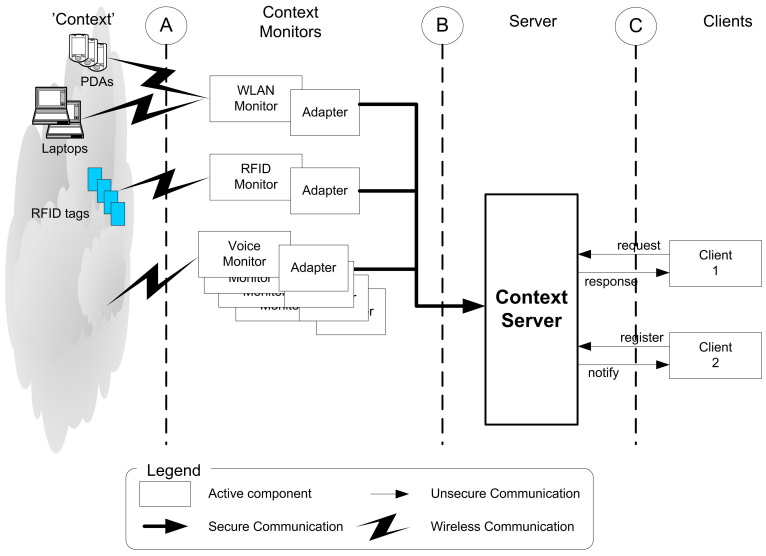

Fig. 3. System architecture for the Context-Awareness Infrastructure.

On the other hand a client can register itself as a listener to an entity, and it will hereafter receive notifications when changes to this entity take place. For example, a client running on a stationary computer (e.g. a wall-sized display) can register itself as listener for changes to the place in which it is located. Whenever entities enter or leave this place, the client will be notified.

In our authentication protocol, the context server is asked for a person's location. The confidence in the answer from the server can be divided into addressing two questions:

1. *How accurate is the location data?* – Whenever the context server provides a location of an entity, it estimates the accuracy of this location. Thus, the context server embeds an *accuracy algorithm*. In step 5 of the protocol, the accuracy of the location estimate is compared against a configurable threshold value.
2. *Do we trust the location data?* – This is a question of whether we trust the information that is stored in the context server. Because monitors are the only way location data can be altered in the server, this reduces to a question of trusting the Context Monitors. To prevent non-authorized monitors to access and update the context server we require the monitors to authenticate themselves to the server. Hence, our context-awareness architecture supports secure authentication and access in layer B in figure 3 between the monitors and the server. However, there is currently no security in layer A between the tokens and the monitors, because RFID tags do not have processing power to support a cryptographic setup. We shall return to the security consequences of this below.

The communication layer C between the Context Server and the clients is not secure. Hence, a non-authorized client can get access to the server and read data from it. This might be a problem from a privacy point of view, but from a security point of view clients cannot alter location data in the context server. We shall return to the security analysis below.

4 Security Analysis

We will now take a look at the security of this authentication mechanism. The goal of a person trying to break the system is to authenticate as a legitimate user. This person is called the adversary. We will make the assumption that the adversary has access to all information stored about the user, except from access to information stored on the smart card (the private key and the user's password). We will also assume that communication between the Context Monitors and the Context Server is secure and that only legitimate Context Monitors can access the Context Server. Finally we assume that the information stored on a smart card cannot be read or changed except through the designed interfaces[2].

4.1 Passive Attacks

In the passive attack scenario we assume that the adversary can monitor all communication between the smart card and the terminal. In the case where location data is based on a token the user has, the adversary can also monitor all communication between that location token and the Context Monitor. Using the information he acquires during this phase he will now try to impersonate the legitimate user after having acquired the user's smart card, both the smart card and the location token (if one is used) or none of them. We have identified the following passive attacks:

1. If the adversary acquires the smart card and is able to fake the location of the legitimate user (by stealing the location token or cheating other devices used to establish the location of the user) he can authenticate as the legitimate user since to all parts of the system he is that user.
2. If he acquires only the smart card he can authenticate as the legitimate user only in the location where the user is actually present.
3. If the user is not present he cannot authenticate as that user even though he has the smart card, unless he also knows the user's password. This is because the protocol used is a secure one-way authentication protocol. Proof of this can be found in appendix A.
4. If he does not have the smart card, it is impossible for him to do anything, since the terminal will ignore him when he approaches it.

[2] It is up to the manufacturer to protect against physical attacks on the card [2, 18]

4.2 Active Attacks

Where the passive adversary could only look at messages between the location token and the Context Monitor and the smart card and the terminal, the active adversary can drop, change, inject or completely replace any of these messages. He can also create his own smart card using the information he obtains. The goal is the same, though: After having done this for as long as he wants he will now try to impersonate the legitimate user. We have identified the following active attacks:

1. The adversary can retransmit the "I'm here" message from a location token to a Context Monitor or he can trick some other part of the location system to make the Context Server believe that the legitimate user is present. If the adversary has the smart card he can now authenticate as the legitimate user.
2. Replaying a value created by the legitimate user in another run of the protocol between the smart card and the terminal does not work since the protocol is a secure one-way authentication protocol and hence not vulnerable to replay attacks. More specifically it comes from the fact that the probability that the terminal will use the same value of N in another run of the protocol is negligible. This is true for both case A and case B.
3. An idea could be to trick the smart card into decrypting the value containing the password. Since signing using RSA is encrypting with the private key we could send $E(K_P, N+password)$ from case B to the legitimate user's smart card instead of N in case A. The smart card will then sign this value and return the signature, which will actually be the decryption of the password. This attack does not work since the smart card will never decrypt and return anything it receives. This means that the smart card cannot be used as a *Decryption Oracle*. In this case the smart card signs a hash of that value instead of the value itself. Since cryptographic hash functions are believed to be one way, the adversary cannot control what value is signed.
4. Creating a new smart card with a known password will not work since the public key will not be known to the server. If we use a legitimate user's public key we would not have the private key and could not make a smart card that can decrypt the first message in case B or make a signature that can be verified using a legitimate user's public key, which is required in both case A and B.

Another attack against this protocol is a *proxy attack* where the adversary tricks the user into using a fake terminal. This allows the adversary to read his password or to perform a man-in-the-middle attack where he can authenticate as the legitimate user without having the user's smart card. This requires that both the user and the adversary is running the same version of the protocol.

If they are both running the protocol from case A, the adversary simply forwards N to the legitimate user's card, which will return a signature on that value, and the adversary can now send this to the real terminal. The adversary is hereby granted access in the name of the legitimate user. Figure 4 illustrates this attack.

If we are in case B, this is a bit trickier since the adversary must know the correct password in order to succeed. This is because the nonce the real terminal sends is combined with the password and encrypted. In order for this attack to work, the smart card

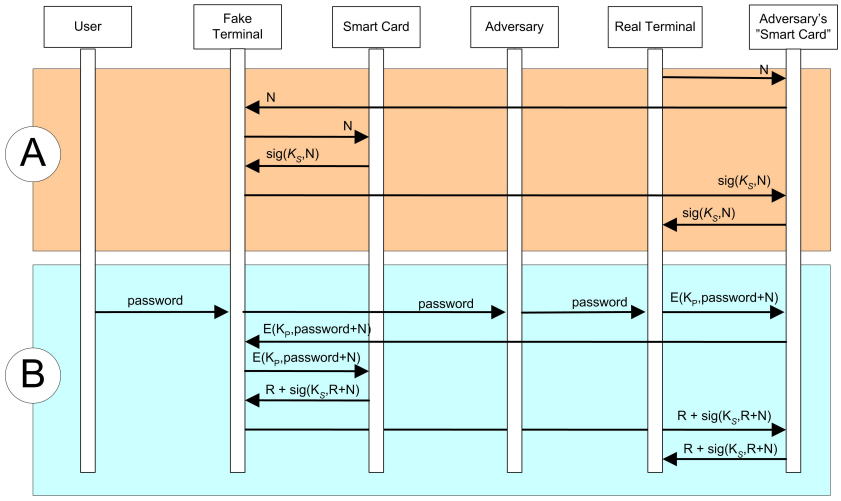

Fig. 4. Proxy attack where an adversary has put in a fake smart card terminal to be used by an ignorant user.

has to sign the nonce the real terminal sent, which means that it must also accept the password entered on the real terminal by the adversary. However this is not a problem for the adversary since the fake terminal can also read the user's password. This attack is also illustrated in Figure 4.

4.3 Summary

It is possible for the adversary to authenticate as a legitimate user by doing one of the following:

1. Steal the smart card and fake the location of a legitimate user by
   (a) Replay the "I'm here" message from a location token
   (b) Trick other parts of the location system in various ways
2. Steal the smart card and be in the same room as the legitimate user.
3. Steal the smart card and acquire the user's password somehow.
4. Perform one of the two proxy attacks mentioned above.

We realize that the real weakness in this protocol is the location data. If that can be faked, all you need is to acquire the smart card of a legitimate user, but without knowing the details on how location data is obtained you can not do a thorough security analysis of that problem. If location data is only based on some token the user has, it can be stolen, if such a location token does not use some kind of cryptography it is vulnerable to a reply attack, if you use voice recognition it might be fooled by a recording of the user's voice, etc. The proxy attacks are bad as well, but require more resources and knowledge to succeed and there are ways to prevent them (see section 6).

5 Current Implementation

Our current implementation consists of five parts:

- An installer.
- The authentication applet.
- A rough prototype of a Personal Pen
- The Context Server and Monitors
- A client that runs the authentication protocol.

The *Installer* installs the applet on the card with the help of IBM JCOP tools [17]. Then it retrieves the public key from the card, and stores it as a key-value pair on the person object in the Context Server. In our current implementation, the authentication applet uses 512 bit RSA keys. This can easily be changed since the cards also support 1024 and 2048 bit keys. Encryption is done in PKCS#1 mode and signatures are made by taking a SHA1 hash of the data to be signed and then this hash is encrypted with the private key in PKCS#1 mode. We run the applet on the dual-interface OpenPlatform compliant JavaCard designed by IBM, which supports both contact-based and contactless connections to the card reader. We use Philips Semiconductors' MIFARE PRO contactless reader [20] (see figure 5) as well as standard OpenPlatform Smart Card readers in e.g. keyboards.

Our current design of the *Personal Pen* is just to glue a JavaCard to a Mimio Pen, which is used at the wall display in figure 1. This form factor is not particularly appealing (see figure 5), but it is hard to change the form and size of the card, because the antenna is embedded in the plastic that surrounds the chip itself. However, the size of the chip in these cards does not prevent it from being embedded in a pen at the factory. Our ideal hardware design would be a smart card chip embedded in a pen, and the reader embedded in the touch-sensitive layer on a display. Putting the tip of the pen at the display would then correspond to inserting a card in a card reader. In this way we would be able to authenticate the user every time s/he is doing anything on the display. The *User Authentication Client* runs as a part of the Activity-Based Computing infrastructure, and it simply waits for a card to be inserted in the reader, and then runs the protocol.

As for the *Context Server*, we have currently implemented the two monitors shown in figure 3. The WLAN monitor monitors the WLAN base station infrastructure in our lab and can tell the cell-based location of IEEE802.11b networked devices. Various types of RFID monitors can monitor passive RFID tags in the environment. We currently use the Portal antennas shown in figure 5 to determine the location of persons equipped with RFID tags. We use the Long Range RFID Antennas from Datatronics [13].

The current implementation of the *Accuracy Algorithm* is very simple. It reduces the accuracy of the location estimate by 1% every minute. Thus, if a person has passed a portal 10 minutes ago, s/he is in this location with a probability of 90%. The User Authentication Client is also considered a trusted monitor (we call it a 'Login Monitor') and can hence reveal the user's location every time s/he logs in. The secure authentication of monitors has not been implemented using proper cryptograpic protocols.

Fig. 5. *Left* – The ScanGate RFID Long Range Portal Antennas from Datatronic. *Right* – The dual-interface OpenPlatform JCOP JavaCard designed by IBM using the Philips Semiconductors' MIFARE PRO contactless reader. The JavaCard is glued to a Mimio pen, which is used at the wall-display in figure 1

However, since monitors run on standard PCs there are already well-known ways of doing this using e.g. a PKI setup over secure IP. Hence, this was not necessary in order to make a proof-of-concept.

6 Discussions and Future Work

By combining a context-awareness sub-system with a personal smart card, we have designed and implemented a proof-of-concept of a proximity-based user authentication mechanism, which is both user-friendly and secure (c.f. section 2.3). One could argue that it seems like an unnecessary effort to implement a context-awareness system in an organization just to help users log in. However, on the one hand we argue that providing easy login is essential to maintaining a smooth flow of work in a hospital (see also [4]). And on the other hand, we envision that a context-awareness sub-system is already in place in a hospital for many other reasons, and we have demonstrated how such a system can help realize the vision of 'proximity-based user authentication'.

From a security point of view, the authentication mechanism presented here, allows for some additional attacks as compared to the traditional use of smart cards in conjunction with passwords (attack no. 3 is equally relevant in both cases). We consider attack no. 2 highly unlikely, which leaves us with attack no. 1 and 4. Our current implementation cannot cope with attack no. 1.a, except for simple revocation of smart card and location token (the RFID tag) when the theft is discovered. However, determining a person's location could be based on something that is not token-based, for example

video or audio. We currently plan to develop a voice-print monitor that can identify and locate a user based on voice. Such a voice monitor is not subject to theft, and methods for avoiding playback exist. In attack no. 1.a the adversary tells a Context Monitor that the user is present by actively replaying the user's ID. While we cannot completely prevent this given the computational power available in RFID tags, we can reduce the possibility of this attack succeeding by doing some additional checks in the Context Server. For example, the user cannot be present on two different locations at the same time, move from one location to another in less than a specified amount of time, or be present in the hospital outside his/her normal working hours. If any of these events occur, uncertainty in location estimation is increased and the authentication protocol will 'fall back' to password authentication. Attack no. 4 could be considered the most dangerous attack since it does not require access to the smart card or the location token. The attack against case A is easily prevented by disallowing the user to be present in two different locations simultaneously. Against case B the attack is more difficult to prevent. One solution could be to use biometrics instead of the password. That way the adversary cannot enter the password on the real terminal. This would also reduce the risk of the adversary aquiring the user's password or even if he could aquire the binary data generated by the biometric equipment he cannot feed it to the real terminal easily. Another solution to the proxy attack could be to use some kind of *Distance-Bounding Protocol* [9] to ensure that the user is close to the terminal he is trying to log on to.

A solution to the active replay attack (attack no. 1.a) could be to have location tokens to make a secure authentication when revealing itself to the context monitors (communication link A in figure 3). This could be done by using public key cryptography, almost similar to the smart card authentication. Another approach would be to use an authentication protocol that relies on secret key cryptography since it does not require the same amount of processing power. We are currently designing radio-based tokens that carry enough processing power and memory to make a secret key authentication to their context-awareness monitors.

7 Conclusion

In this paper we have pointed to the need for considering security, and especially user authentication, as a central aspect of UbiComp technologies and applications. It is of crucial importance that users can access pervasively available computers easily, if the vision of UbiComp shall become alive. In the paper we introduced the concept of *proximity-based user authentication* in a pervasive computing environment, where a user is authenticated to a device by just approaching it. Our aim was to create a user-friendly user authentication mechanism, which is sufficiently secure to be used in settings where security is important – like in a hospital. The suggested solution is based on using two different mechanisms for user identification and verification. A user is identified by presenting his Java Smart Card to the computer whereafter his correct presence is verified through a context-awareness system. If the context-awareness system is unavailable or cannot localize a user, the user is required to enter his password – this is identical to the standard use of smart cards.

Through our security analysis we have argued for the degree of security in our solution. This analysis showed that our solution is not completely secure. Especially, the solution is vulnerable to an active replay attack, where an adversary steals a user's smart card and at the same time is able to replay a false "I'm here" message to a nearby Context Monitor. To accommodate this attack, the Context Server implements an accuracy algorithm that calculates the accuracy or probability of the estimated location of a person. In case this probability is below a certain threshold, the user is requested to enter a password. We argue that this authentication mechanism is sufficiently secure because it combines something the users *have* (the smart card) with the *location* of the user. This authentication protocol is hence and example of location-based autentication [14]. This is traditionally taken as sufficiently secure, if each of the two mechanisms is secure [25].

Acknowledgments

Many thanks to Michael Osborne at the IBM Zurich Research Laboratory, who has provided us with the JCOP JavaCard technology as well as with much practical help. Thanks to Kim Kortermand for help with the RFID Portal Antennas. This research has been partly funded by the Competence Centre ISIS Katrinebjerg.

References

1. G. D. Abowd, B. Brumitt, and S. Shafer, editors. *Proceedings of Ubicomp 2001: Ubiquitous Computing*, volume 2201 of *Lecture Notes in Computer Science*, Atlanta, Georgia, USA, Sept. 2001. Springer Verlag.
2. R. Anderson and M. Kuhn. Tamper resistance – a cautionary note. *The Second USENIX Workshop on Electronic Commerce Proceedings*, 1996.
3. M. Baentsch, P. Buhler, T. Eirich, F. Höring, and M. Oestreicher. Javacard – from hype to reality. *IEEE Concurrency*, pages 36–43, Oct.-Dec. 1999.
4. J. E. Bardram. The Trouble with Login – On usability and Computer Security in Pervasive Computing. Technical Report CfPC 2003–PB–50, Center for Pervasive Computing, Aarhus, Denmark, 2003. Available from http://www.pervasive.dk/publications.
5. F. Bennett, T. Richardson, and A. Harter. Teleporting – Making Applications Mobile. In *Proceedings of the IEEE Workshop on Mobile Computer Systems and Applications*, pages 82–84, Los Alamitos, CA, USA, 1994. IEEE CS Press.
6. G. Borriello and L. E. Holmquist, editors. *Proceedings of Ubicomp 2002: Ubiquitous Computing*, volume 2498 of *Lecture Notes in Computer Science*, Göteborg, Sweden, Sept. 2002. Springer Verlag.
7. B. Brumitt, B. Meyers, J. Krumm, A. Kern, and S. Shafer. EasyLiving: Technologies for Intelligent Environments. In P. Thomas and H. W. Gellersen, editors, *Proceedings of Handheld and Ubiquitous Computing, HUC 2000*, volume 1927 of *Lecture Notes in Computer Science*, pages 12–29, Bristol, UK, Sept. 2000. Springer Verlag.
8. M. Burrows, M. Abadi, and R. Needham. A logic of authentication. *ACM Transactions on Computer Systems (TOCS)*, 8(1):18–36, 1990.
9. L. Bussard and Y. Roudier. Embedding distance-bounding protocols within intuitive interactions. 2003.
10. Center for Pervasive Healthcare. http://www.cfph.dk.

11. H. B. Christensen and J. Bardram. Supporting human activities – exploring activity-centered computing. In Borriello and Holmquist [6], pages 107–116.
12. M. D. Corner and B. D. Noble. Zero-interaction authentication. In *Proceedings of the eighth annual international conference on Mobile computing and networking*, pages 1–11. ACM Press, 2002.
13. Datatronic Long Range RFID Portal Antennas (297C-AT3M/S). http://www.datatronic-rfid.com.
14. D. E. Denning and P. D. MacDoran. Location-Based Authentication: Grounding Cyberspace for Better Security. *Computer Fraud and Security*, February 1996. Available from http://cosc.georgetown.edu/ denning/infosec/Grounding.txt.
15. Ensure Technologies. http://www.ensuretech.com.
16. A. Harter, A. Hopper, P. Steggles, A. Ward, and P. Webster. The Anatomy of a Context-Aware Application. In *Proceedings of the 5th Annunal ACM/IEEE Conference on Mobile Computing and Networking (MobiCom99)*, pages 59–68, Seattle, WA, USA, 1999. ACM Press.
17. IBM JCOP Smart Card Technology. http://www.zurich.ibm.com/csc/infosec/smartcard.html.
18. P. Kocher, J. Jaffe, and B. Jun. Differential power analysis. *Lecture Notes in Computer Science*, 1666:388–397, 1999.
19. K. Nagel, C. D. Kidd, T. O'Connell, A. Dey, and G. D. Abowd. The Family Intercom: Developing a Context-Aware Audio Communication System. In Abowd et al. [1], pages 176–183.
20. Philips Semiconductors' MIFARE Demonstration System (MF EV500). http://www.semiconductors.philips.com/markets/identification/products/mifare/.
21. D. M. Russell, C. Drews, and A. Sue. Social Aspects of Using Large Public Interactive Displays for Collaboration. In Borriello and Holmquist [6], pages 229–236.
22. D. M. Russell and R. Gossweiler. On the Design of Personal & Communal Large Information Scale Appliances. In Abowd et al. [1], pages 354–361.
23. B. Schneider. *Secrets and Lies : Digital Security in a Networked World*. John Wiley & Sons, 1 edition, 2000.
24. M. T. Smith. Smart Cards: Integrating for Portable Complexity. *IEEE Computer*, pages 110–115, Aug. 1998.
25. A. S. Tanenbaum. *Modern Operating Systems*. Prentice Hall, 2 edition, 2001.
26. J. Trevor, D. M. Hilbert, and B. N. Schilit. Issues in Personalizing Shared Ubiquitous Devices. In Borriello and Holmquist [6], pages 56–72.
27. R. Want, A. Hopper, V. Falcão, and J. Gibbons. The active badge location system. *ACM Transactions on Information Systems (TOIS)*, 10(1):91–102, 1992.
28. A. Ward, A. Jones, and A. Hopper. A new location technique for the active office. *IEEE Personal Communication Magazine*, 4(5):42–47, October 1997.

A Security proof

In BAN logic [8] the authentication protocol looks like this:

Case A	**Case B**
1. $B \rightarrow A : N$	1. $B \rightarrow A : \{N, P\}_K$
2. $A \rightarrow B : \{N\}_{K^{-1}}$	2. $A \rightarrow B : \{N, R\}_{K^{-1}}$

Where A is the smart card, B is the terminal, N is a nonce, P is the password entered by the user, K is the public key of the smart card, K^{-1} is the private key of the smart card and R says whether or not P matched the password stored on the smart card. The proof will be split in two different cases.

We want to show that in both cases the terminal can be convinced that the legitimate user is present in the current run of the protocol. In the first case this is done by showing that the smart card knows the private key and the password that the user entered. In the second case this is done by showing that the smart card knows the private key. In both cases we also show that no replay attacks can occur.

We assume the following in both cases:

$$
\begin{gathered}
fresh\,(N) \\
B \text{ believes } fresh\,(N) \\
\xrightarrow{K} A \\
A \text{ believes } \xrightarrow{K} A \\
B \text{ believes } \xrightarrow{K} A
\end{gathered}
$$

Case A

Claim: B believes A believes N

Proof: In the first step of the protocol we know that A sees N. After this we have:

$$
\frac{\dfrac{B \text{ sees } \{N\}_{K^{-1}} \quad B \text{ believes } \xrightarrow{K} A}{B \text{ believes } A \text{ said } N} \quad B \text{ believes } fresh\,(N)}{B \text{ believes } A \text{ believes } N}
$$

Which is what we wanted to show.

Case B

Claim: B believes A believes N, R

Assumptions: The smart card will always return the correct value of R.

Proof: In the first step of the protocol we know that A sees $\{N, P\}_K$. This gives us:

$$\frac{A \text{ sees } \{N, P\}_K \quad A \text{ believes } \xrightarrow{K} A}{A \text{ sees } N, P}$$

So A knows the password P, the nonce N and will now generate R based on the value of P. After A has sent its message we have:

$$\frac{\dfrac{B \text{ sees } \{N, R\}_{K^{-1}} \quad B \text{ believes } \xrightarrow{K} A}{B \text{ believes } A \text{ said } N, R} \qquad \dfrac{B \text{ believes } fresh\,(N)}{B \text{ believes } fresh\,(N, R)}}{B \text{ believes } A \text{ believes } N, R}$$

Which is what we wanted to show.