Title
Protecting user data in ubiquitous computing: Towards trustworthy environments

Journal
Privacy Enhancing Technologies, 3424

ISSN
0302-9743

Authors
Duan, Yitao
Canny, John F

Publication Date
2005

Peer reviewed

Protecting User Data in Ubiquitous Computing: Towards Trustworthy Environments

Yitao Duan and John Canny

Computer Science Division
University of California, Berkeley
Berkeley, CA 94720, USA
{duan, jfc}@cs.berkeley.edu

Abstract. In a Ubiquitous Computing environment, sensors are actively collecting data, much of which can be very sensitive. Data will often be streaming at high rates (video and audio) and it must be dealt with in real-time. Protecting the privacy of users is of central importance. Effective solutions for controlling access to data in ubicomp settings remain to be developed. Dealing with these issues will be a central challenge for ubicomp for some time to come. Here we propose some simple design principles which address several of these issues. We illustrate them through the design of a smart room capture system we are building. The main design principle is "data discretion:" users should have access and control of data about them, and should be able to determine how it is used. We show how data discretion supports both personal *and* collaborative uses. In our implementation, the data discretion principle is enforced with cryptographic techniques. Unlike ACL based access control systems, our scheme embeds access rights of legitimate users *within* the data. We have implemented a prototype system in the smart room equipped with several cameras, and we give data throughput rates under various degrees of protection. Finally we describe ongoing work towards a trustworthy ubicomp environment whose discretion is realistically checkable.

1 Introduction

A ubiquitous computing (ubicomp) or pervasive computing environment is typically envisioned as a space populated with large number of invisible, collaborating computers, sensors and actuators interacting with user-worn gadgets. Data about individuals who are in the environment is constantly being generated, transmitted and stored. Much of the data can be quite sensitive. Protecting private data is a major concern for users. There are a few challenges that make data security in such settings different from other system protection:

1. The environment is often unfamiliar to the users. They will not have a trust relationship with the owners of the environment as they might with their local system administrator appropriate for handling their private information.
2. Data are often generated dynamically and streaming at high rates (video and audio) and must be processed in real-time.

3. Users' access rights change dynamically with respect to their relationship with the mechanisms by which data are generated and sometimes the users cannot be predetermined. For example, a number of users can form an ad hoc group and record their meeting using a camera that is administrated by the environment. They should only have accesses to the video produced during the meeting period but not others. The system must be able to associate a piece of information with the correct set of users while it is being produced.
4. The system is typically decentralized, with multiple data sources/sinks. Data caching or replication can be common. There is usually no single point where access control can be enforced.
5. Data usage often involves sharing among a group of people [1]. Any protection scheme must allow efficient sharing among legitimate users.

Unless there is a clearly stated protection policy and scheme in place, together with provisions for convincing verification, users will not be comfortable trusting the infrastructure with their data. And this in turn will hinder the acceptance and limit the usefulness of such systems.

For the data that are generated by the user (e.g., documents produced on users' laptops), protection is relatively easy. Clients can generate proper keys and encrypt the files by themselves and upload the encrypted files to the server. They can also store the keys on the server using a variant of the scheme that will be described in Section 5. Since the server never sees the keys or the clear data, it has no way to leak them.

In ubiquitous computing settings, however, the data are often generated by another party. In this paper we are concerned with protection of user data that are generated by the infrastructure. This case is tricky because the users have to trust the system to some degree (e.g. the fidelity of the sensor data) since they rely on the system to perform certain operations (e.g. data generation, transmission, and encryption). We do not want to encourage "blind trust" from the user. Rather, we would like to design the system in such a way that it promotes user trust and confidence. We consider a typical ubicomp environment, in this case a smart room augmented with a variety of sensors. The issues we address in this paper are:

1. Protection of the user data generated and maintained by the environment.
2. Privacy of individuals who use the environment.
3. Ability of legitimate users to make use of data recorded in the environment.
4. Dealing with high-speed streams of data.
5. Trustworthiness of the environments. (This is work in progress)

We argue that traditional access control schemes employed by stand-alone computers and small networks don't provide the right protection models for ubiquitous computing, and propose an approach to protecting user data in a dynamic and ad hoc environment. Our scheme makes use of both symmetric and public-key cryptography and can achieve efficiency high enough to be implemented using today's commodity hardware and software, and deal with streams

of audio and video data. Our scheme embeds access rights in the *data* and offers a simpler and more efficient alternative to ACLs and provides a natural model for defining access policies and mechanisms for enforcing them. It is by no means a complete solution to data security in ubiquitous computing environments, nor do we claim to have solved the privacy problem that has been a major concern for ubicomp. Rather, our scheme provides technological primitives that, together with legal enforcement and social and economic considerations for privacy, can be used to build secure and privacy preserving systems. We consider it an important step towards making a ubiquitous computing environment trustworthy.

1.1 Privacy Principles

User data protection is closely related to privacy. Not only protection techniques can be used to ensure privacy, privacy considerations also influence protection policy and implementation. This is especially true in a fluid setting like ubiquitous computing. For instance, before we design any protection scheme, we need to determine who should have the access right to what data under what conditions, and this will be drastically shaped by privacy considerations. Here we state some privacy principles for ubiquitous computing that motivate our approach to data protection. Later we will derive design principles of our own that support these privacy principles and directly dictate our protection scheme.

In [2], Marc Langheinrich laid out some principles for design of privacy-respecting ubicomp systems. Langheinrich stresses the importance of including privacy considerations in the early stage of system design process. He develops six principles for guiding privacy-aware ubiquitous system design:

1. Notice: users should always be aware of what data is being collected.
2. Choice and Consent: users should be able to choose whether it is used.
3. Anonymity, Pseudonymity: should apply when identity is not needed.
4. Meeting Expectations: systems should mimic real-world norms.
5. Security: different amounts of protection depending on the situation.
6. Access and Recourse: users should have access to data about them.

Langheinrich's principles provide very specific guidance for our work. Later, we describe our design approach that supports several of the principles directly, and makes it easy to build systems which support all of them.

The rest of the paper is organized as follows. In Section 2 we survey related work on ensuring user privacy and data protection in ubicomp environments. In Section 3 we describe personal and collaborative applications of smart spaces which leverage our protection system to ensure privacy. Section 4 illustrates a smart room setting that we are using as a testbed for our principles and implementations. In Section 5 we present the data discretion principle and describe the protection scheme that enforces this principle and how it is applied to our smart room. Section 6 gives performance evaluation of our prototype system. Section 7 discusses the issue of trustworthiness of ubicomp environments and presents extensions and possible techniques for improving and validating privacy compliance in such environments. It also describes directions for future research. Finally we give summarizing remarks in Section 8.

2 Related Work

User privacy has been a major concern in the design of many ubicomp systems. Several recent projects seek to protect user anonymity in communication and data capturing [3, 4]. They basically follow the rules and practice established in Internet Privacy [5, 6], which focus on obscuring user's IP address, and extend them into ubiquitous computing context. The Mist Project at UIUC strives to anonymize users' communication in ubicomp environments [4]. It utilizes a hierarchy of "Mist Routers" that perform "handle-based routing" to preserve privacy and hide information about the source and destination. Users will expose their identities to part of the system, but their locations will be concealed. The EuroPARC's RAVE [7] presents a networked node model where interactions are defined by connections established between nodes. It emphasizes two principles in preserving privacy: control and feedback. The former empowers users to stipulate what information they project and who can access it while the latter allows the users to be informed of the capturing and usage of their information. Privacy preserving is achieved by controlling connection capabilities. While this is a reasonable model for protecting transient information, it is not clear how permanent data is protected. Also it represents an "untrustworthy" environment for visitors who have no knowledge of how it works.

Motivated by user privacy, as well as system security, concerns, data protection in ubicomp systems is attracting more attention. Among the many types of information, user location is the focus of many studies probably due to its close tie to user privacy. In [8] Mike Spreitzer and Marvin Theimer describe their design of an architecture for providing, and controlling access to, location information in a ubiquitous computing environment. In their architecture, each user owns an "User Agent" that collects and controls all personal information pertaining to its user (including location) and any request for such information must be routed through the "User Agent" which enforces predetermined access policies. One problem with their architecture, however, is that location, and other personal information, can be derived from a variety of sources (e.g. a camera owned by a mall) and it is infeasible for an user's agent to have control over all of them.

Another location data protection scheme described in [9, 10] anonymizes user location in sensor network by reducing the spatial accuracy of sensor readings and perturbing the count of users in the covered area. In addition, [11] describes the protection of user location information based on policies that are defined to reflect various privacy requirements of users and spaces. The implementation is based on digital certificate. This scheme provides a framework for protecting user location data as an output of the People Locator service in their system.

All these schemes assume that requests for user location information are directed towards some *service* or *agent* where access control can be naturally enforced. However, in many distributed system applications, especially in ubiquitous/pervasive computing environments, user location information can be derived from raw data (e.g. sensor data or images taken by a camera in a room) in a variety of ways that bypass the service. For example, by gaining access to

an image taken by a camera in a particular room and checking the time stamp of the file, one can easily obtain the location of some user(s) at certain time. And this unauthorized access could be quite often in a fluid environment like ubicomp (e.g. a stolen or lost device). This shows that existing schemes for controlling access to user location are inadequate without proper data protection. Our scheme can be used to complement existing ones to make them more secure. For example, our scheme can safeguard the image data so that no one except the owner(s) of that data can decipher them thus protecting the user(s) location information against illegitimate access.

Some access control architectures were proposed specifically for ubiquitous computing [12–15]. Most of them focus on formalizing and automating policies and their enforcement. For example, Role Based Access Control (RBAC), introduced in 1992 by Ferraiolo and Kuhn [16], is probably one of the best-known method for access control for distributed systems and is attracting increasing attention because it reduces the complexity and cost of security administration in large networked applications [17–19]. In RBAC systems, entities are assigned roles and access rights are associate with each role. Similar to the location systems described earlier, these schemes are primarily concerned with controlling access to the *services* provided by the system and many are based on authentication/authorization models which are used extensively in traditional centralized systems. This model is inadequate for ubiquitous computing in a number of ways: (1) it is difficult to specify the security policies in a fluid environment in a way that matches the dynamic access relationship users can have with the data; (2) it is difficult to make data sharing, which can be essential in collaborative ubicomp applications, efficient and safe; (3) data are not protected if the authorization agent is down or faulty, or bypassed. And such situations can be quite common in ubiquitous computing environments (e.g. a lost device).

In contrast, both our design principle and the enforcing scheme focus on data protection. The security of our scheme does not rely on any access control. Rather, it uses cryptographic techniques to make data safe by themselves. Even if an adversary gains access to the data, he cannot take advantage of them because they are encrypted and are not different from random data to those who don't have access right. Access control policies are useless if the code enforcing them is compromised or bypassed. As we argued, such situations can be common in ubicomp settings. Given this circumstance, our approach seems to be the only solution.

3 Applications

Our scheme can enable new ubicomp applications, or enhance existing ones with privacy. In this section we describe two types of applications, one stand-alone and one collaborative, that can leverage our scheme to ensure privacy.

3.1 Personal History

A number of researchers have explored the idea of lifetime personal histories collected electronically. Gelernter's "Mirror Worlds" is an early example [20]. More recently, wearable computing researchers Steve Mann [21] and Bradley Rhodes [22] (the "remembrance agent") built systems to record and retrieve their daily histories. Other current explorations of lifetime history include "Stuff I've seen" and Cyberall (resp. Susan Dumais, Gordon Bell, Microsoft), and the "Personal Server" (Roy Want, Intel).

A wearable computer provides one source of lifetime history. Wearables may or may not be adopted by most users. Sensor-equipped spaces provide an alternative. Data recorded in the space can be sent to all the users who are in the space at the time. In this way, the room becomes a "virtual personal recorder". As the user moves, they carry an ID tag, which might include their encryption key and the URI of a server to which the space should send their recorded data. If they enter another sensor-equipped space, it takes over the task of recording and shipping data to their server. Advantages of this approach are that the room may include more sensors, e.g. cameras in many positions, that the data may be streamed over a high-bandwidth fixed link to the users server, and that this server can have higher storage capacity (and backup) than a wearable.

We are implementing a simple video recording service along these lines in a smart room. We use RFID tags to determine who is in the room, although because of memory limits we use a lookup table to map from the tag number to an encryption key and target URI.

3.2 Collaborative Applications

Surprisingly, individual private data collection also allows some important *collaborative* applications that support privacy. In other words, users can share their information without losing privacy. We explain how briefly.

Collaboration is a major challenge to privacy. By its nature, collaboration involves exchange of information between collaborators. It may also involve automation – the best example of which is collaborative filtering [23]. We also have several ongoing projects which make use of history data from several individuals to support other types of collaboration. In particular, we are using personal history data to compute collaborative "activities". Activities are shared patterns of communication, document and web access which are mined using a clustering algorithm from histories of the participants.

Normally, collaborative computation would require user data to be stored and processed on a single server computer. This could raise serious privacy concerns. As mentioned before, in ubicomp settings, users generally do not have strong trust relationships with the individuals who manage the hardware. This lack of strong trust applies also between individuals and the owners of community-based services. The problem exists already on the web - many sites monitor users' progress around their sites, and their electronic purchases. The problem

is at least limited to the scope of a particular web site. But ubicomp provides much richer and more invasive data.

To deal with this problem we explored encrypted computation. The idea is that raw user data should remain accessible only to the user in question. When collaborative computation is needed, it should be done only on encrypted data, thereby protecting individual privacy to the maximum extent possible. This approach is indeed feasible, and in [1] Canny showed that it is practical for interesting collaborative tasks. That paper described a new collaborative filtering (CF) algorithm based on encrypted computation that matched the accuracy of the best-performing algorithms at that time. It also showed that basic clustering (based on SVD) is possible on encrypted data.

Subsequent work on collaborative filtering [24] showed that encrypted computation need not penalize either performance or accuracy. A second collaborative filtering algorithm, based on sparse factor analysis (SFA), was introduced in [24]. Not only does that algorithm support encrypted computation, but through several experiments it showed that it is the most accurate CF algorithm to date and one of the most efficient. This method also supports *community creation and maintenance* of collaborative groups, and so addresses the needs for social participation emphasized by the social sciences perspectives on privacy.

A very interesting application of CF in ubicomp is to location data. For instance, by tracking their location and aggregating with others, users can obtain recommendations about restaurants, shops, places to see and things to do. But gathering such information creates great risks to privacy and, as mentioned before, existing protection schemes for protecting location information are insufficient for ubicomp. The general thread of our work is to explore cryptographic and AI techniques to compute from user data only the information needed for a particular task, and to protect the rest of the data. And the scheme presented in this paper is the basis of such computation.

Note that not all data are suitable for encrypted computation. In the rest of the paper we will describe a prototype system that uses video data to represent user data that are generated by a ubicomp environment. The purpose of such prototype system is to demonstrate the feasibility and efficiency of our scheme. We are not proposing performing computation on encrypted video data. However, we stress that such computation is possible with some user data (e.g. user location, activity log, etc.) that are protected with our scheme [1]. In other words, our scheme does not hinder collaborative use of the data. On the contrary, because the data are properly protected, with access rights embedded in themselves, our scheme makes collaborative use of sensitive data privacy preserving.

4 Smart Room Testbed

We have begun building a smart room equipped with sensing and able to record user activity. The room is a small meeting room with a conventional whiteboard, a Smart Technologies Smartboard^{TM} with projector, four video cameras, and a Philips I-CODE^{TM} RFID tag reader. There is a single entrance to the room, and

the tag reader antenna is mounted around this entrance so that it can record entry and exit by any user carrying an I-CODE tag. The tags provide reliable recognition of users entering the room, although users without tags will not be noticed by the tag reader. It can of course be fooled by tags which move through the doorway when not attached to their users. The cameras capture images from the room continuously, and send them to a data server. Another computer drives the Smart-board, and this computer runs a logging program that records all significant user activity using the board.

The smart room is representative of typical ubicomp environments and serves as a good testbed for our privacy principles. According to Lessig, the "most serious violations of privacy by anyone's standard" are in the workplace due to email and web monitoring. Physical workplaces involve a mixture of individual and collaborative work. As those spaces are fitted out with sensing and recording devices, workplace monitoring is likely to extend to the physical world. To us therefore, smart workplaces are an excellent candidate for the most serious future privacy violations in ubicomp. The images and log data represent the dynamically generated user data that we seek to protect. We will provide a simple playback application that explores the privacy protections. Figure 1 shows a photo of the smart room testbed. Section 6 will give more details about the prototype system and the experiments we conducted on the smart room testbed.

Fig. 1. Smart room testbed

5 System Design: the Data Discretion Principle

If we take Langheinrich's principles 4 and 6 together, meeting expectations and access and recourse, we derive a principle we call data discretion:

Data Discretion: Users should always have access to, and control of (recorded or live) information that would be available to them in "real-world" situations. They should not have direct access in other situations.

So for instance, users should have access to information recorded in a smart room that while they were in the room. They should not have access to information recorded in that room while they were not present. They should also have control of this information, and be able to use it however they please.

By making access to the user a requirement, we can make it easier to satisfy the other privacy principles 1, 2, 3 and 5. That is because we can require all accesses to user data to be routed to the user, and therefore we involve the user in all decisions about use of their data. However, this does not mean that users literally participate in every decision about their data. They can certainly establish policies, use agents or proxies etc., as others have proposed. Our goal was rather to make sure that others do not override the user's current or earlier intentions regarding their data. Our proposal is a low-level architecture on which applications and user-centered information management systems can sit.

There is some subtlety in making the discretion principle work. Particularly in a smart room setting, we would like to keep the identity of the people who were in the room a secret from those who were not in the room at that time. This means that users who were not in the room at the time should not have access to information recorded in the room at that time, nor should they be able to find out who does have access, because that is equivalent to knowing who was in the room. This rules out many access control methods, which expose the set of users who have access to files to system administrators or others who can penetrate the operating system. We prefer methods based on cryptography which make it impossible to gain access to sensitive information even with control of the machine and operating system.

5.1 Basics

Our method makes use of both symmetric and public-key encryption. We assume that all data are stored in files that represent short time intervals. For instance, each video file contains a short sequences of images of one second or so, or logs from the Smartboard in files that span about a second, etc. Every user file is encrypted with a randomly generated secret key. This secret key is then encrypted by the public keys of the file owners. We will describe the scheme in details in section 5.3.

A user joins our system by registering his or her public key [1] and choosing a recognition tag that the system can detect once it is within the vicinity of a sensor. The recognition tag is used by the system to detect user's presence so that it can retrieve related information (e.g., user's public key). The user is

[1] We hope to eliminate this step in future. We would prefer the user's public key to be carried in their tag, along with (possibly) a URL to which their data should be sent. Our current tags do not have enough writeable bits.

assumed to wear the recognition tag. Those who fail to wear the tags will not have access to the data, which is a safe default from privacy point of view.

Note this approach is different from access control lists (ACL) that are typically used to protect user data whose owner is predeterminable. Our system only needs to know the association between user data and its access key(s) and can function perfectly well without knowing identification of the holder of a particular tag. No attempt is made to connect a tag with a user's real-world identity, or even his or her system-wide electronic identity. User anonymity is therefore protected and users can maintain multiple pseudo-identities with multiple tags. This of course is limited by their visible presence to other users in the space who may know them by name.

5.2 Establishing Information Ownership

The first step in protecting user data is to ascertain the natural owners of the information. In a dynamic context such as ubiquitous computing, this has to be done while the information is being generated. We assume that the system has the ability to detect a user's recognition tag when they are present in the environment. Other identification technologies (e.g., face and fingerprint recognition) are expanding rapidly, and there already exist several that are mature enough to offer non-intrusive detection at reasonable cost. However, some of these technologies rely on recognition of information that are unique to a user (e.g., fingerprint) and are not suitable for privacy. RFID tags are good technologies from a privacy perspective because they have no unique attributes of particular users. The decoupling of user's identity and their recognition tag provides anonymity. A user can also have several pseudonyms via multiple tags. Biometric data is more sensitive because if it were extracted from the system, it would provide outsiders with cues to this individual's identity. RFID tags also allow users who are not interested in using the system a way to opt out by not carrying a tag.

The issue of who should be the rightful owner of data is a controversial one and the legal dispute involved has not been resolved. While some believe that the party that invests in gathering personal data has a right on it, we have taken an ideological stance (based in part on European law and on principles provided by many researchers) that users have first rights to this data, as they would in real-world use of a collaborative space. This is in accordance with our "data discretion" principle.

The smart room is equipped with an ambient display that always shows the number of users in the room, according to readings of the tag reader. If this does not match what the users see, then users know there is a rogue tag in the room. This prevents an attacker from hiding his tag in the room to obtain access to data recorded while he is not present.

5.3 Encryption Scheme

The basic output of the system is a sequence of files and key-tuples. File F_i is the data logged at time i. Associated with F_i is a tuple of user keys $(k_{i1}, ..., k_{in})$, denoted as *key set*, which determine who has access to this file. The length of the tuple of user keys, n, is always fixed for that environment. The value of n is larger than the maximum number of users who could plausibly be in the environment at the same time. Using a fixed number of keys is necessary if we would like to hide the number of users who are actually in the room at a given time. This many keys may seem extravagant, but in our smart room application, the key set is much smaller than its data file.

Changes of access right are modelled by the system through transitions between *sessions*. A session is defined as a short duration of activities with a fixed set of users. Users' access rights to the data generated at that time remain invariant during one session but would change across sessions. The system is responsible for detecting events that, according to "data discretion" principle or other considerations, may trigger session change (e.g. a user entering or exiting the room, or a periodic timer expires). A session is assigned a unique ID.

Encryption on user data is performed in the following steps:

1. The system performs appropriate key generation algorithm such as ANSI X9.17 [25] and obtains a sequence of pseudorandom keys $d_1, d_2, ...$ One key, denoted d_i, will be selected from the sequence to be used to encrypt files generated in current session. Symmetric key encryption is used for efficiency. Each key will be used only for one session and a new key will be selected for new session. The system will perform key generation periodically when the pool of keys is exhausted.
2. d_i is then encrypted with the public keys of the m people in the room (determined by recognition tag reader). These encrypted keys are placed in m locations among the n key positions in the key set, in pseudo-random fashion. The other $n - m$ key positions are filled with random numbers.
3. Users who were in the room can recover the keys and review the video while they were in the room.

Although there are n possible keys associated with each file, it is not necessary to search through them all in steps 2 and 3. We use a family of n hash functions $h_1, \ldots, h_n$ to perform a pseudo-random search. At step 2, the system places user j's key in the position specified by one of the hash functions applied to the encrypted file and the user's public key. The first hash function which hashes to a free key location is used. If we assume that n is at least twice the maximum number of people in the room at one time, then at least half these locations will be free, and on average only $\log_2 m$ such steps will be needed to find a free location. The same method is used to retrieve the key at step 3. This scheme is illustrated in Figure 2, where $hash \in \{h_1, \ldots, h_n\}$.

This scheme grants equal access to all users involved. This might not be appropriate in some applications. A more general version would specify a general access structure among the users, which can easily be done by secret-sharing the

session key among the occupants and embedding the shares in the key set. This is left as an option for applications.

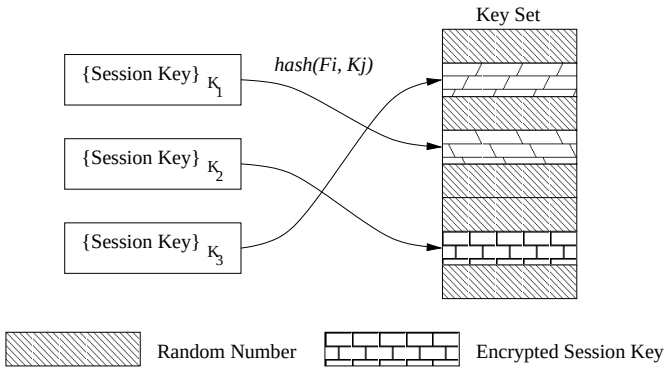

Fig. 2. Key set construction

5.4 Master Key Escrow

It is sometimes necessary for a few privileged parties, e.g., police, to access data stored in the system. However, it is not desirable that a single party be granted full access right to the data since there is a danger of a malicious power party misusing his privilege and compromising users' privacy. Our solution to this dilemma is a shared master key and threshold decryption scheme. The master key consists of a globally known El-Gamal public key and a matching private key that is not held by any single party but instead secret shared among a number of authorized "masters". Masters would normally be distributed to people like the local police department, the building security manager, the corporate president or safety officer, etc. Threshold decryption allows a subset of those individuals (say any two of them) to retrieve data for safety or law enforcement reasons. But it avoids the risks of single individuals accessing data inappropriately.

Each file's key d_i is encrypted with the public master key. A group of masters whose number must exceed a pre-specified threshold can collaborate to retrieve d_i and access the data[2]. Pedersen's key generation protocol [26] or its variants/enhancements [27, 28] can be used to securely generate the public key and distribute the secret shares of the private key among participants

5.5 Unlinkability

Assuming both encryption algorithms used for encrypting the data files and the keys are secure, our scheme is "zero knowledge" in that it reveals no information

[2] Note that since the encryption keys are one-time only and only valid for one session, it is safe to reveal it to the collaborating "masters".

about the data, not even the number of users who have access. It enforces the "data discretion" principle, i.e. only the rightful owners have access to, and control of the information.

One possible attack is on "unlinkability". I.e. an attacker may observe the public keys read by the tag readers installed at different places and link the data produced by these spaces to the same user(s). This enables the attacker to potentially track users. However, such an attack is only possible by compromising the system. The data themselves reveal no information about who have access and are totally unlinkable. Protecting the system is another issue that is orthogonal to the scheme we propose. Assuming the system is well protected with techniques such as firewall and the communication between tag readers and the server is secure, the unlinkability attack is impossible by scrutinizing the data.

6 Performance Evaluation

We have implemented a prototype system on the smart room testbed to carry out our protection scheme. Our smart room has four cameras as well as one tag reader. Images are typically captured at several images per second.

Our prototype system consists of the following four pieces of software:

1. An FTP server that receives image data from the four video cameras and saves them to disk.
2. A data encryption process that constantly checks for new data on disk and encrypts it.
3. A user data server that serves the encrypted data to users upon request.
4. A proof-of-concept client program that continuously requests the most up-to-date data from the user data server, tries to decrypt them on the fly, and displays the recovered images on the screen (if the decryption is successful); thus if the user has legitimate access to the data, he should be able to see a smooth video replay of what's being captured in the room.

Conceptually the first two programs should be merged into one; we did not implement this due to time constraint. Instead, we simulated the effects of one single integrated data receiving/encryption server by deleting the plaintext files after we have generated the encrypted version. Nevertheless, we believe that an integrated server is more in line with our security principles and will be implemented in future versions of our system. We used Crypto++ Library^{TM} 5.0 [29], an open source crypto library written in C++, for our cryptographic functions.

The encryption server is the crucial part of the system and will be discussed in detail below. Logically it consists of two parallel modules: 1) session control and 2) data encryption. The session control module, a.k.a. the session manager, monitors the users (and their public keys) in the current session and generates session keys. It is notified of user arrivals and departures by the RFID tag reader. Whenever the session changes, the session manager destroys the old session and its associated state, and creates a new session and chooses a new session key.

The data encryption module monitors the arrival of new image data from the cameras, and encrypts those data with the current session key. We chose Triple-DES with 192-bit keys as our data encryption algorithm. In addition, the module encrypts the session key with the public key of each user in the current session. We used RSA public key encryption algorithm for this purpose. The encrypted keys are hidden in the fixed-size key set as described in Section 5 and stored together with data.

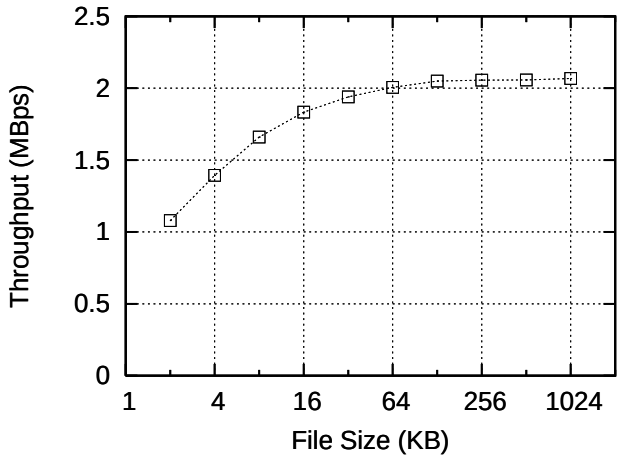

Fig. 3. System throughput

We have run several tests on our system. One involves multiple users simultaneously requesting the latest video captures while they are moving in and out of the room randomly. Our system can detect the changes of their presence in the room on time and reflect them with the changes of access rights. Thus the users could see the images while they are in the room but are unable to decrypt them when they are not. To determine the throughput of our system, we feed the system with a stream of files of fixed sizes arriving at high speed and measure the time it takes to process them. The experiments were run on a PIII 900MHz machine running Linux 2.4.18 Kernel. The code was compiled with gcc3.2. We did not account for the cost of generating and encrypting session keys in our experiments because these operations are performed only once per session and are dominated by the cost of encrypting the data files. Figure 3 shows the system throughput with different image file size. As our processing time includes two disk I/Os (one read and one write) as well as the encryption, the throughput improves as the file size increases. With a 1MB file size, the system can achieve a throughput of 2.07MBps while with 8KB file size the throughput is about 1.66MBps. Assuming a capture rate of 20 files per second, our system can support up to 10 cameras. Although these are more than enough for our current needs, there is

plenty of room for improvement. First, the encryption algorithm we use, Triple-DES, is quite strong and expensive. Other symmetric encryption algorithms exist that can yield much higher throughput. As documented in the Crypto++ 5.0 Benchmarks (http://www.eskimo.com/~weidai/benchmarks.html), DES yields almost 3 times throughput as Triple-DES, Blowfish 4 times, Twofish 6 times, and ARC4 more than 13 times. In cases where the strong property of Triple-DES is not required, these algorithms can be used for higher throughput. Second, we expect the throughput to go up with an integrated receiving/encryption server, as we can eliminate one disk I/O from the critical path. mnvThird, the hardware we are using is only mediocre or even archaic according to today's standard. There are plenty of high end systems available in the market with reasonable cost. And finally code optimization has yet to be done.

7 Towards Trustworthy Environments

Our scheme provides a flexible protection method for user data in ubiquitous computing environments and can be used to complement other architectures to achieve higher level of security. However, the scheme as described so far suffers from a significant weakness: the security of the scheme is conditioned upon the assumption that the server is honest and the data are safe only after the server performs the encryption scheme *faithfully*. Ubiquitous computing systems, ours and others, suffer from the trust problems we described at the beginning of the paper. In future, users will move through a procession of sensor-equipped spaces, all of which can record their actions. Some of the owners of those spaces will be incentivized to collect and use that data. But the user about whom the data is recorded, who we argued is the rightful owner of the data, may not wish this to happen (recall Langheinrich's principle 1 against covert monitoring). The law will almost certainly have a say in this state of affairs. But laws have no potency unless they can be enforced, which means unauthorized monitoring must be detectable at reasonable cost. Bits flowing around the Internet can be extremely difficult to trace, so this is certainly a challenge. It creates a technical problem: How do we monitor and verify that a smart room or space is compliant with stated privacy policies and transmitting only the data it is supposed to?

We are approaching this problem from the framework of trusted computing. That is, we assume that most of the hardware in the environment is untrusted, but that a small and inexpensive trusted device (a tamper-proof device like a smart card, but more powerful) is incorporated into the infrastructure in a way that it can verify that the system is satisfying particular constraints. This device would need to be inserted and inspected as part of certification of the space by a third party or government, much like a GFCI receptacle (ground-fault detectors required near water). In our case, the constraints are that the system should send only messages encrypted with keys of users who are in the room, and that the system should not leak information in the messages encrypted with authorized keys. The first guarantee can be given with ZKP (Zero-Knowledge Proof) techniques [30]. The second (leakage) is quite subtle and is the subject of

our ongoing work. In both cases though, the system should obey a key principle, described next.

7.1 Data Transparency Principle

Paradoxically, the first step in making an ubicomp environment trustworthy is to make sure all the data flowing out of it is encrypted, but can be seen by other users or by the inspection subsystem (which is trusted component within the infrastructure).

Data Transparency: Encrypted data recorded or transmitted by a ubicomp system should be easily observable. Where possible, the data itself should demonstrate compliance with stated principles.

The information protections provided by well-implemented cryptography are much more reliable than access restrictions using the operating system or network routing. Once data *are* encrypted, ZKP techniques allow us to prove things about it without disclosing any new information about the data. In particular, we can show that it is encrypted with particular keys. And we can attempt to show the absence of leakage.

7.2 Verification Mechanisms

How to verify the environment's compliance with privacy policies is the subject of our current research. We already noted the value of ZKPs for this step. ZKPs allow an agent A to prove to an agent B that A has information that would be very hard for B to compute, without disclosing that information ([30]). More concretely in [1], ZKPs are used to prove that private user data was validly generated without disclosing the data. In our case ZKP can be used to prove that access rights of legitimate users are indeed embedded in the data [3]. The challenge with the leakage property is that there are many ways to leak information. Each leakage mechanism defines a property of the encrypted data. So proving non-leakage appears to be proving that the data does *not* have an open set of properties. This is not feasible with ZKP. To this problem we take another approach. Namely the non-leakage property is enforced by introducing verifier devices from a third party or the government and forcing the system to first fix its data by bit-commitment, and then encrypt using a security parameter created by the (trusted) verifier. That means that the system has no way to anticipate what the encrypted data will look like, and it will appear highly random.

It is unlikely that any scheme can prevent all types of cheating. We believe ours could certainly raise the technical bar for cheating, and possibly impose strong constraints on the rate of information leakage. This should be enough to reduce the frequency of non-compliance significantly.

[3] Please note that we are using ZKP to reason about the properties of the *key set*, not the actual data, which can be anything from video to audio and it may not be possible to apply ZKP.

8 Conclusion

As many researchers in ubiquitous computing have noted, in order for ubiquitous computing to be really beneficial and socially acceptable, user privacy has to be considered carefully at early stage of system design. In this paper, we argue that the essence of preserving user privacy is protecting user data and propose two design principles and an enforcing protection scheme. The "data discretion" principle stipulates that access to information stored in a system should only be granted to individuals who would have access to the data in the "real-world". Explicit notion of ownership should be established as the information is generated to determine access right. The "data transparency" principle states that, rather than trying to enhance privacy by hiding the existence of information or communication, a system should rely on well-implemented cryptography for data protection and make the recording and transmitting of encrypted data observable. Only when the usage of data is made open can the system perform effective monitoring to enforce compliance with privacy policies. We consider this to be a very important step towards building trustworthy environment.

Acknowledgements The authors would like to thank Alessandro Acquisti and the anonymous reviewers for their valuable comments. This work was supported in part by National Science Foundation award #EIA-0122599 (Title: "ITR/SI: Societal Scale Information Systems: Technologies, Design, and Applications").

References

1. Canny, J.: Collaborative filtering with privacy. In: IEEE Symposium on Security and Privacy, Oakland, CA (2002) 45–57
2. Langheinrich, M.: Privacy by design – principles of privacy-aware ubiquitous systems. In Abowd, G., Brumitt, B., Shafer, S., eds.: Proceedings of Ubicomp 2001. Volume 2201 of Lecture Notes in Computer Science., Springer (2001) 273–291
3. Abowd, G.D., Mynatt, E.D.: Charting past, present, and future research in ubiquitous computing. ACM Trans. on Computer-Human Interaction **7** (2000) 29–58
4. Al-Muhtadi, J., Campbell, R., Kapadia, A., Mickunas, D., Yi, S.: Routing through the mist: Privacy preserving communication in ubiquitous computing environments. In: International Conference of Distributed Computing Systems (ICDCS 2002), Vienna, Austria (2002)
5. Cranor, L., Langheinrich, M., Marchiori, M., Reagle, J.: The platform for privacy preferences 1.0 (p3p1.0) specification. W3C Recommendation (2002)
6. Anonymizer Inc.: Anonymizer. http://www.anonymizer.com (2003)
7. Bellotti, V., Sellen, A.: Design for Privacy in Ubiquitous Computing Environments. In: Proceedings of the Third European Conference on Computer Supported Cooperative Work (ECSCW'93), Kluwer (1993) 77–92
8. Spreitzer, M., Theimer, M.: Providing location information in a ubiquitous computing environment. In: Proceedings of the 14th ACM Symposium on Operating Systems Principles), ACM Press (1993) 270–283
9. Gruteser, M., Schelle, G., Jain, A., Han, R., Grunwald, D.: Privacy-aware location sensor networks. In: Proceedings of the 9th Workshop on Hot Topics in Operating Systems (HotOS IX), Lihue, Hawaii, USA (2003)

10. Gruteser, M., Grunwald, D.: Anonymous usage of location-based services through spatial and temporal cloaking. In: Proceedings of the First International Conference on Mobile Systems, Applications, and Services (MobiSys 2003), San Francisco, CA, USA (2003)
11. Hengartner, U., Steenkiste, P.: Protecting access to people location information. In: Proceedings of First International Conference on Security in Pervasive Computing (SPC 2003), Boppard, Germany (2003)
12. Gribble, S.D., Welsh, M., von Behren, J.R., Brewer, E.A., Culler, D.E., Borisov, N., Czerwinski, S.E., Gummadi, R., Hill, J.R., Joseph, A.D., Katz, R.H., Mao, Z.M., Ross, S., Zhao, B.Y.: The ninja architecture for robust internet-scale systems and services. Computer Networks **35** (2001) 473–497
13. Kagal, L., Undercoffer, J., Perich, F., Joshi, A., Finin, T.: A security architecture based on trust management for pervasive computing systems. In: Proceedings of Grace Hopper Celebration of Women in Computing 2002. (2002)
14. Kagal, L., Cost, S., Finin, T., Peng, Y.: A framework for distributed trust management. In: Proceedings of IJCAI-01 Workshop on Autonomy, Delegation and Control. (2001)
15. Hengartner, U., Steenkiste, P.: Access control to information in pervasive computing environments. In: Proceedings of the 9th Workshop on Hot Topics in Operating Systems (HotOS IX), Lihue, Hawaii, USA (2003)
16. Ferraiolo, D., Kuhn, R.: Role based access control. In: Proceedings of the 15th National Computer Security Conference. (1992)
17. Lupu, E., Sloman, M.: A policy-based role object model. In: Proceedings of the 1st IEEE Enterprise Distributed Object Computing Workshop (EDOC'97), Gold Coast, Australia (1997) 36–47
18. Sandhu, R.S., Coyne, E.J., Feinstein, H.L., Youman, C.E.: Role based access control models. IEEE Computer **29** (1996) 38–47
19. Sampemane, G., Naldurg, P., Campbell, R.H.: Access control for active spaces. In: Proceedings of the 18th Annual Computer Security Applications Conference, Las Vegas, Nevada (2002)
20. Gelernter, D.H.: Mirror Worlds: Or the Day Software Puts the Universe in a Shoebox: How It Will Happen and What It Will Mean. Oxford University Press (1992)
21. Mann, S.: Smart clothing, turning the tables. In: ACM Multimedia Conf. (1996)
22. Rhodes, B.: The remembrance agent: A continuously running automated information retrieval system. In: The Proceedings of The First International Conference on The Practical Application of Intelligent Agents and Multi Agent Technology (PAAM '96), London, UK (1996) 487–495
23. Goldberg, D., Nichols, D., Oki, B., Terry, D.: Using collaborative filtering to weave an information tapestry. Comm. ACM **35** (1992) 51–60
24. Canny, J.: Collaborative filtering with privacy via factor analysis. In: Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Tampere, Finland, ACM Press (2002) 238–245
25. Menezes, A.J., Oorschot, P.C.V., Vanstone, S.A.: Handbook of Applied Cryptography. CRC Press Series on Discrete Mathematics and Its Applications. CRC Press (1996)
26. Pedersen, T.: A threshold cryptosystem without a trusted party. In: Proceedings of EUROCRYPT '91. Volume 547 of Springer-Verlag LNCS., Springer (1991) 522–526
27. Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Secure distributed key generation for discrete-log based cryptosystems. LNCS **1592** (1999) 295–310

28. Fouque, P.A., Stern, J.: One round threshold discrete-log key generation without private channels. Public Key Cryptography (2001) 300–316
29. Wei, D.: Crypto++ Library™ 5.0 (2002)
30. Goldreich, O., Oren, Y.: Definitions and properties of zero-knowledge proof systems. Journal of Cryptology **7** (1994) 1–32