Applying Cryptographic Techniques To Problems In Media Space Security

Ian E. Smith, Scott E. Hudson, Elizabeth D. Mynatt, and John R. Selbie

Graphics, Visualization & Usability Center and College of Computing
Georgia Institute of Technology, Atlanta, GA 30332-0280
+1 (404) 894-6266
{iansmith, hudson, beth, selbie}@cc.gatech.edu

ABSTRACT

Media spaces integrate audio, video, and computing systems for the purpose of remote collaboration and awareness, frequently between people engaged in a cooperative task. Technological advances have made these systems feasible using desktop computers and broadband, digital networks. Using a media space over a shared network requires that numerous security and privacy issues be addressed. One advantage of digital media spaces is that properties of the media space can be manipulated so that users feel more comfortable with the technology. This paper details cryptographic techniques which can be used to create a secure and usable media space. This paper also explains the properties of a user interface which would enable users to ensure their level of privacy. This work also details two interface designs which provide users with sophisticated, flexible control of their media space without requiring a detailed understanding of the underlying cryptographic mechanisms.

KEYWORDS

Media spaces, multimedia, computer supported collaborative work, cryptography, user interfaces, security, privacy.

INTRODUCTION

Media spaces are audio-visual and computing environments which are designed and built to support remote collaboration and awareness between a number of participants [2]. Typically, users in separate physical locations are *connected* via permanent audio and video channels. There has been significant work done over the past few years on "media spaces" at several research centers including Xerox PARC [20], EuroPARC [12] [3] [6], the University of Toronto [19][14], NTT [13] and Bellcore [9]. In all these cases, the users of the systems wished to collaborate and interact with each other (although to varying degrees) and usually the users are in the context of a work-related activity. Most of these systems have been built by using a combination of existing digital data networks for the computing facilities and a switched analog network for the transmission of audio and video data (a notable exception is the Montage system [21]).

Issues regarding security and privacy have an important impact on the usefulness of these systems. People naturally feel uncomfortable working in front of a video camera and a potentially open microphone. Without proper support, this experience can be like working in front of a giant one-way mirror -- one has little knowledge or control of who is watching or listening. One never knows when the boss or some particular co-worker might be listening, and hence the conventional perceptions and properties of private versus public spaces that we normally take for granted can be invisibly altered.

To make media spaces practical in a wide range of settings, security and privacy concerns must be addressed. Consequently, there has been an ongoing discussion of these issues throughout much of the previous work in media spaces. For example, Gaver, et. al [12] describe four dimensions along which users need or desire media space security. User's want:

- *Control* over who can see and hear them at a given time.
- *Knowledge* about when someone is seeing or hearing them.
- Information about the *intention* behind a connection.
- To avoid connections being an *intrusion* on their work.

Other authors have outlined similar requirements with respect to the security and privacy concerns in a media space [14]. Gaver also noted that the first and last of these axes are related; by giving users control of who can communicate with them, they can manage their level of intrusion [10].

With respect to the axis of "control", most systems have used some type of "block out" mechanism which is designed to prohibit some subset of the other users from accessing the audio and video resources in their work-area. Two of these systems combined this "blocking out" with a metaphor of a door [14] [18], thus allowing people to express their preferred amount of interaction in a natural way.

When trying to keep users aware of which other users of the system were presently looking at them through their cameras or listening to them through their microphones, all the systems provided some type of visual feedback about the current state of the system. The RAVE system at EuroPARC also provides audio feedback about the current system state [11].

When considering security in the media space, it is useful think about how and why an abuse, called an *attack*, might be perpetrated against a user. The obvious reason is the Orwellian monitoring of employees by their superiors, in which the superior clandestinely monitors the media space data to determine information about the actions of his or her subordinates. However, other, more subtle, attacks can be concocted by unscrupulous persons, and we can look to other computer-based for examples of these types of attacks. For example, in the past, people have constructed programs to save a copy of all files printed on a particular printer in some secret place, without the person who printed the files permission. This action allows the person who knows about the extra copy of the files to monitor others potentially sensitive information and use it to his or her advantage. An analogy to this in the media space setting would be an attack in which all media space data originating from a particular person is archived somewhere for later review. With

COOCS 95 Milpitas CA USA © 1995 ACM 0-89791-706-5/9 5/08..$3.50

today's digital editing capabilities for audio, video, and data an *attacker* could potentially edit the content to construct an audio and/or videotape which had "real" data spliced together in some new way so as to create problems for the original speaker. Another analogy can be drawn between existing attacks on other computer-based communication and the media space case. A frequent problem in many settings is the illegal access of electronic mail by either system administrators or other users. The interception of electronic mail traffic flowing between two other users can frequently convey significant (albeit unethical and illegal) business and/or personal advantages. In the case of the media space, if one were able to quietly observe audio communications between users, similar advantages could be gained.

Returning to Gaver's dimensions, it should be noted that his third and four dimensions relate *only* to users of the system participating in the normal way, users who are "playing by the rules." Surely a wise attacker would not want to be an intrusion on your work area or inform you of his or her intentions! Since our work focuses on efforts to secure a media space from such attackers, we will not consider the dimensions of intention and intrusion further in this work.

With respect to the other two of Gaver's dimensions, a serious problem with existing systems is the underlying assumption that all the users who could potentially gain access to the media space's data are users of the system and, therefore, are bound by its security measures. This assumption is most clear in Gaver's second dimension of knowledge; there is the assumption that in these systems it is *possible* to know who could be seeing or listening to you at a given time. Further, if we consider for a moment that the system might have less than complete knowledge about the potential recipients of your media space data, the first of his dimensions is seriously compromised. In short, why would a person attempting to use the media space for illicit purposes limit him or herself to using the "normal" media space software?

Exacerbating our security problem is the current push towards desktop digital video and shared, wideband digital networks (such as FastEthernet and Asynchronous Transfer Mode) as the communications mechanism for media spaces and related multimedia systems. The shared, digital network allows potential interlopers an easy way to access the audio, video, and other data from the convenience of their desk! This breach is possible because they are connected to the network the same way that legitimate users are. To make matters worse, if more than two people are in the media space concurrently (a likely occurrence), the digitized audio and video signals will have to be broadcast or multicast[1] over the shared network. Such a broadcast or multicast *insures* that a potential attacker gets a copy of the data at his or her desk!

Cryptographic techniques provide a technological infrastructure which can deliver fairly strong guarantees of privacy and control even under these conditions. Our aim is both to prevent malicious users from gaining unauthorized access, and to provide the cues needed to help the user avoid mistakes and misperceptions in more ordinary usage. Because very few potential users will know, or want to know much about cryptography, we show that natural, accessible, and understandable abstractions can be built on top of the underlying cryptographic mechanisms.

In addition to allowing users control of the underlying cryptographic algorithms without concern about the particulars of their implementation, our goal is to design an interface which can effectively display the state their connection to the media space. This system must make every effort to convey the state of the system so the user can avoid accidental transmissions, since a mistake in this setting could result in unpleasant personal and/or professional consequences.

Our specific approach in solving the problems described above hinges on the application of cryptographic techniques allowing media space users to communicate efficiently and privately over a shared data network. For this paper, we assume that media space participants wish to broadcast (potentially large amounts of) audio, video and other types of data over a digital network which may contain attackers who are clandestinely eavesdropping on the communication. Further, we assume that any attacker listening in on these communications can record the data gleaned for future reference and is in the possession of "reasonable" amounts of computing power.

Our discussions are broken into four major sections. In the first section, we outline our approach to using cryptography to support a media space and provide the reader with the necessary cryptographic background. Next, we outline our user interface approach to giving users control over their media space privacy. In this section, we explain our efforts to create a usable and understandable system which can use the underlying cryptography effectively. In the following section, we describe the current status of our system and provide some implementation details. We conclude with some pointers for future work.

USING CRYPTOGRAPHY IN THE MEDIA SPACE

Cryptography Background

We now briefly introduce the cryptographic techniques necessary for our discussion of media space security problems. In any cryptographic system, users wish to communicate some piece of data between themselves (called the *plaintext*) and to insure that persons other than the intended recipient(s) are not able to have access to the data. This operation is accomplished by encrypting a copy of the plaintext with a *key* which renders the plaintext into an unusable form called the *ciphertext*. The ciphertext is transmitted to the recipient(s) and they decrypt the message with some key, recreating the plaintext. In a "safe" cryptographic system, without access to the proper keys, the ciphertext is useless and thus the ciphertext may be shown openly (broadcast) to anyone, as only the person(s) with the keys will be able to return it to the plaintext form.

In a *symmetric* (sometimes called *traditional*) cryptographic system, the encryption and decryption keys are the same; this symmetry implies that the sender and receiver must have communicated the key between themselves (hopefully securely!) at some point in the past. This type of system raises a serious problem: how do I distribute a key to someone else, if I can not be sure that the communication channel is secure? (If you had access to a secure channel you could simply send the plaintext over the channel and avoid using cryptography altogether.) The most commonly used and widely studied symmetric key system is the Digital Encryption Standard of the U.S. government (DES) [15]. DES serves as the primary encryption technique used in our system.

1. A *multicast* is an efficient and controlled form of a broadcast transmission. A multicast transmission originates at a single node in the network and will arrive at many nodes in the network, although usually not all. In many cases (such as Internet Protocol multicast-ing) the sender does not know the identity of all the recipients. [7] [4]

The problem of distributing keys safely was solved in 1976 [5], with the invention of a public key cryptographic systems. In a public key system, each user is given two (related) keys: one called the "public" key and one called the "private" key. Data encrypted with one can be decrypted only with the other. This strategy effectively eliminates the key distribution problem above, as users can simply make their public key widely known and ensure that their private key is kept secret. For example, anyone wishing to communicate with a user Alice need only know Alice's public key, since they can encrypt data *destined for* Alice by using the public key system and Alice's public key. At that point the ciphertext can only be decrypted by Alice, since only she knows the private key that can decrypt data encrypted with her public key.

An interesting side effect of public key encryption systems is that they can be used "in reverse" to perform authentication. An authentication insures that a given transmission has not been tampered with, and that its origin can be verified. This process is usually known as a digital signature. Consider Alice from before: She encrypts a piece of text that says "Mary had a little lamb" with her private key and then sends the ciphertext over an open network to Bob claiming that the resulting ciphertext is "Mary had a little lamb." Bob can authenticate the received data: He tries to decrypt it with Alice's public key. If the result is what Alice claims, Bob can be sure that she sent it and that it was not altered in transit, as only the matching private key could have generated the data that he successfully decrypted using Alice's public key. The most common (and in the United states the only easily available) public key system is the RSA system, named for its inventors Rivest, Shamir, and Adelman [17].

Properties Of A Media Space Cryptosystem

As with many cryptographic systems, the crux of the problems associated with media space security can be boiled down to the problem of distributing the keys necessary for the cryptographic algorithms to provide the desired security. Our key distribution problems are made considerably worse by the fact that to make our media space usable it must have certain desirable properties:

1. *Security:* We would like to avoid situations where there is any possibility of potentially sensitive or embarrassing information being transmitted to parties (either by mistake or abuse) who should not have access to it.
2. *Separate Channels:* We would like to create several distinct communication *channels*. Each channel corresponds to a media type that the user may want to control separately. Example channels are audio, video (full motion), video stills, whiteboard marks, etc. Note that although video stills are video in a strict sense, users would like to control these differently than conventional full-motion video [6][3] and thus we separate these into a different channel (even though the data will in many cases come from the same source).
3. *Temporal Separation*: We would like our system to have the property of *temporal separation*. This is to say that if you change the privacy attributes of your transmission at any time, access to all other previous points in the transmission is still independently controlled, and the data or keys exchanged at any one of these points in time may not be used to access data from other points in time.
4. *Flexible Broadcast*: We would like users to be able to communicate with any other user and/or any *number* of other users without previous planning, supporting the serendipity that is crucial for certain types collaboration [8] [2].
5. *Prompt Termination*: We would like to mimic the semantics users are already familiar with from everyday life with respect to communication channels being closed or "hung up" in a prompt and predictable fashion.

Discussion Of Media Space Security Properties

Although our first goal may initially seem redundant, there are some subtleties present in it. In a system such as ours, the system is going to use cryptography when necessary to insure that information is not accessible to a potential attacker, but there is room for human error. Besides using technology to avoid information leakage, the system should make every effort to help the user avoid situations with unexpected or unfortunate consequences.

An example of this kind of situation is present in everyday life with the "mute" buttons on many telephones. The mute button is designed to allow the user to keep the party on the remote end from hearing conversations at the local site. Interestingly, if a user does not fully depress the mute button or presses another button by mistake, the person on the remote end may overhear information they were not (specifically) intended to hear. Further, the mute button on most telephones provides no feedback if it is working properly, but worse yet provides no simple way for a user to evaluate its state (you can hardly ask the person on the other end!).

From a cryptographic standpoint the second of our objectives (separate channels) is clear: each media type should have a different key associated with it (or, as we will see later, a set of keys). The third objective, temporal separation, is to prevent a *reuse attack*. In such an attack, an attacker might legitimately obtain a key for a media type from a target user and then use this key at a later point in time, when the target is not intending to communicate with the attacker. Similarly, an attacker could record encrypted data and then legitimately obtain a key and use the key to go backwards in time accessing data that the target user assumed was secure.

Our fourth objective of support for flexible broadcast has two important implications. First, because users may need to communicate with relatively large numbers of other participants, efficiency considerations dictate the use of broadcast or multicast. If a user were to send a separate stream of data to each participant, the underlying network would experience a load increase proportional to the square of the number of participants. Given the large volumes of data involved in transmitting audio and video, ignoring this fact could have a grave impact on many networks. However, since data is broadcast, it is likely to be delivered not only to intended users, but also to potential interlopers.

Second, connection patterns between users may be complex and dynamic. Consider a negotiating session between two rival corporations taking place over a insecure channel. It is crucial at certain times for the rivals to communicate intra-corporation and at others inter-corporation. Such communication patterns need to be under interactive control of the participants involved rather than being fixed in nature. This flexibility makes use of centralized (trusted) authorities for key distribution problematic.

Although our final objective of prompt termination has obvious benefits both from learning and ease of use standpoints, it also has a serious impact on the cryptographic system. Users of our system are accustomed to analog systems such as telephones, VCRs, and televisions which have electrical properties that can be difficult to implement in the digital world. Consider the act of hanging up the telephone: The electrical circuit needed for data to be transmitted to the remote site during a conversation is actually broken at the time the receiver is placed in the cradle. In the case of our system, a user Bob may wish to "hang-up" on a user Chris, and Bob's expectation is that at this point the "connection" (which never existed in the same sense as in the analog case) is broken. However, it is likely that after the "hang up," data is going to be continued to be broadcast from Bob's workstation as Bob may be engaged with

other users besides Chris. We must produce a system in which Bob's data cannot be accessed by Chris after Chris has been "hung up on."

Our Solution

A basic approach to addressing the problems given above is to create a separate key for each channel of information being transmitted by a user. We call this key the *channel key*. All information going out on that channel is then encrypted with that key, using DES encryption. In addition, the user distributes the key to exactly those participant that are allowed to receive the transmission, using the RSA public key system. As the encrypted data arrives, participants with the key can then decrypt it to reconstruct the original information. Other users may still receive the encrypted data, but without the channel key are unable to make any use of it. In this way the encrypted information can be safely broadcast in an unlimited fashion, but a user can still limit access to the actual data by only giving the key to authorized receivers. As we will see later, this process of granting access to information will, in a cryptographic sense, boil down to distributing a key.

Although this basic scheme meets criteria 1 (security), 2 (separate channels), and 4 (flexible broadcast), it does not support temporal separation. This limitation is because once a participant is given the channel key, they may not only use it to decrypt the current transmissions, but may also use it to decrypt transmissions on that channel that they might had recorded in the past. Further, there is no way to take the channel key away from a participant, so they will also be able to access all future transmissions on that channel.

In order to achieve temporal separation, it is necessary to change the channel key any time a new participant is granted access to a channel (so that they cannot retroactively access information transmitted earlier) or an existing participant has access removed (so they cannot continue to receive the information). Changing the channel key implies that the key must again be distributed to each participant via RSA public encryption. If there is no disruption in information flow, this action must be completed before the key is actually put into use. Unfortunately, if there are a large numbers of participants, and each needs to be sent the key separately, this process can cause a delay. Further, this delay directly conflicts with our final criteria of prompt termination, since termination of the "connection" does not occur until after the new key has been distributed to every authorized participant.

A Technique For Organization And Optimization: Groups

Our solution to the problems presented above was to take advantage of users' natural tendency to organize their world into manageable units. We provide a notion of *groups* which is a user definable set of people with some semantic relation. In addition to being a useful abstraction for the end user, this notion allows us to perform an important optimization in the cryptographic system, which makes it much more efficient to change the access granted to members of a group, and to add or remove groups.

This optimization involves providing a *group key* to each member of a group. Whenever the channel key needs to be changed -- for example when we hang up on a group or individual and need to ensure that they can no longer listen in -- rather than communicating the new channel key individually to each user retaining access, we can instead safely broadcast the new channel key encrypted with the group key. This strategy allows us to change channel keys quickly -- typically broadcasting only one message with copies of the new channel key encrypted with each group key -- and hence meeting our criteria 5 of prompt termination, while still maintaining the temporal separation of our criteria 3.

Only when membership of an existing group changes do we need to distribute a new group key to each member of a group, so that old members of the group cannot receive information broadcast in the future, and new members cannot retroactively decrypt recorded information from their past membership.

USER INTERFACE

The chief difficulty in designing a user interface for a system such as this is *visibility*. The system's state is complex, and needs to represented in a way that not only is understandable, but makes certain important aspects stand out. It must achieve the goal of making important information visible to the user while not becoming a bother or consuming enormous amounts of screen real estate.

A difficulty here is that the user has two different (although related) types of activities. One type of activity involves actions that affect particular users or groups of users. Some examples of user goals which correspond to these activities are:

- Make a two-way audio connection to Bob and Sue.
- Make sure I can see what is going on Joe's office; he's due back soon and I need to talk to him as soon as he gets in.
- Connect me with all my coworkers so I can stay aware of how things are going on the project.

These types of activities are all *user-centric* and generally require the system to perform some type of action on behalf of the user.

The other type of action is *media-centric* and requires that system provide information to the user about the user's resources. Examples of user goals which correspond to these activities are:

- Tell me who can hear what is going on in my office?
- Can I be seen?
- Why am not receiving anything from Bob and Alice?

Designing for both of these goals presents a fundamental difficulty: How do we organize the interface so that the user can control actions at the level of other users (or groups) and yet monitor the state of their resources at the level of media? The natural design choice here is, of course, to do both.

Our First Interface Design

Our first interface design is shown in Figure 1. We designed this interface to be a multi-viewed, direct manipulation interface. The two larger display areas- one in the upper right of the display and the other occupying the bottom portion- have been dubbed the *playing field* and the *media view*. The two vertical columns in the upper left are the *participant list* and *group list* from which the user selects other users and groups to manipulate. The small group of four buttons in the middle of the left hand part of the interface control the media space's access to audio resource; this is necessary on workstations which do not allow multiple applications to share access to audio devices.

The row of icons across the top represent different media types and a button to request help (on the far right). The media icons are (from left to right) audio transmission, audio reception, video transmission, video reception, still video frame transmission, and whiteboard marks transmission. The reason for both the audio transmission and audio reception icons is that local user controls only the resources of the local workstation not any remote resources, implying that it is not possible to create a two-way connection. Thus, one uses audio transmission to indicate that you are sending audio data to some remote workstation, and audio

reception to indicate that you are willing to play out (display) audio received from some remote workstation. When displaying remote media, the local user indicates that he or she is willing to display data from a remote location, but this will actually cause a display only if the remote user transmits to the local user.

If a local user wishes to perform a user-centric action, the user selects the remote user's icon from the user list and uses the mouse to drag it onto the playing field. The user's icon then displays six boxes (one for each supported media type) which can accept media icons. These boxes have been dubbed *slots* by users. To give a *capability* to the remote participant, the local user drags the correct media icon from the array of media icons at the top of the display. When the icon nears the remote participant's slots it "snaps" into place, removing the need for the user to place the icon exactly [1].

The lower region of the screen, the media view, is simply a another display of the upper area, but sorted by media instead of by user. Next to each media is a row of pictures of users who currently have this capability. The media view allows the user's media-centric questions to be easily answered with one glance at the appropriate row of icons.

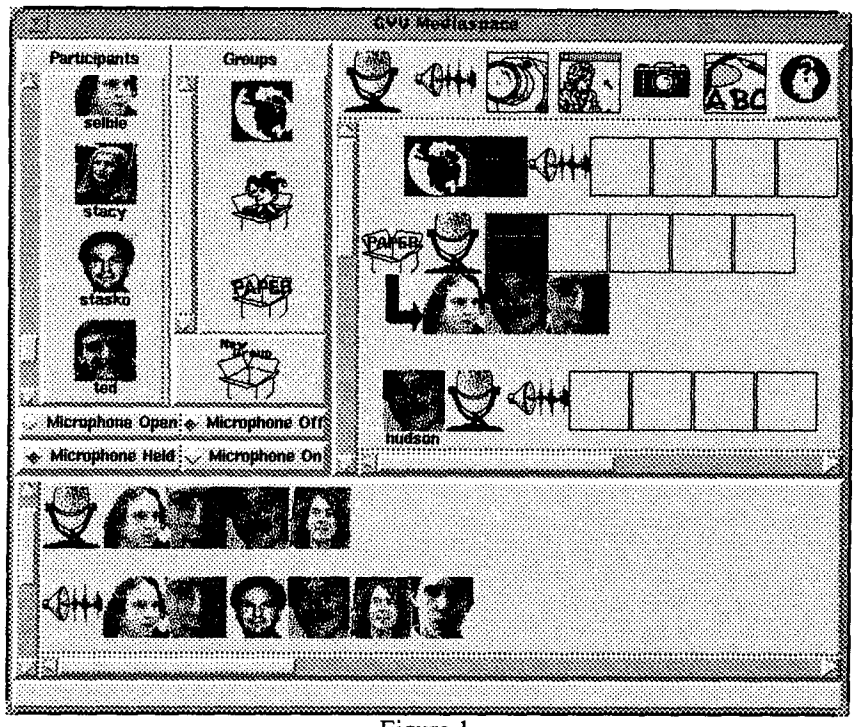

Figure 1

As we mentioned before, the ability to create groups is important to our design, and this action is accomplished by using the mouse to drag a group from the group list onto the playing field. When groups are initially created, they have no members. A group also has a set of slots beside it, like the display for a user, and they function similarly, although operating on the entire group instead of one person. A groups membership is indicated below the group icon, as in the Paper group in figure 1. To add members to a group a user icon is dragged from the user list to the group icon (or near it) and it again snaps into place at the end of the list of users.

The ability to create groups *and* users, however, has a side effect that creates a visibility problem. What if the user's commands with respect to a group and a member of the group are in conflict? For example, if a media space user Alice specifies that group X is not allowed hear her, but says that Bob (a member of group X) can hear her, what should the system do? What should be displayed? It is useful to consider the ramifications that could result if one looked at the display for the media space, decided that it was ok to speak about some topic, and then did so without realizing that somewhere else on the display (or worse, not displayed at all) was information that might have affected this decision.

In this design, we chose to allow the user to specify things of this type, and have the system display the "conflict" in the group in which Bob is a member. Figure 1 displays this type of situation, with the group named "World" and the group named "Paper." (The World group is a system-provided group which allows you to manipulate all users of the system as a unit; these are effectively defaults.) The "grayed-out" versions of the icons indicate that a user who is a member the designated group has this capability but he or she does not have it because of the group. For example, in Figure 1, our system's user has given the capability to hear him or herself to user "Hudson" specifically and consequently the microphone is displayed next to his icon. Hudson is a member of the World group (since everyone is) and Hudson has been placed in the Paper group. In both cases, that entire group does not have the capability to hear the local user, but one of its members does.

Our Second Design

Our first design was evaluated by some informal testing, both in our research group and from outside our group. The results seemed to indicate the interface had three main problems:

1. The interface takes up too much screen real estate to be displayed all the time.
2. The display of the group permissions (especially the grayed out state) was cumbersome and difficult for users to understand.
3. The information displayed was not exactly what was needed; some additional dynamic information was needed and too much detail was presented in some cases.

Given this input, we have created a second design which attempts to address these issues. This design at this point is only on paper and is not yet implemented, it is presented here to show our current efforts to rectify the problems in the previous design.

Our second design calls for a set of small windows to be displayed on the users screen, each independent from others; an example window is in Figure 2 and measures 180 pixels wide by 140 pixels tall. Note that Figure 2's size is proportionally correct with respect to Figure 1.

The central portion of the window is a video image from the remote user. This may be full-motion video or periodic still images, depending on which of these the remote user has allowed the local user to view. In the lower left corner of the display, overlapping with the video, is a small picture of the face of the person one

would expect to find in the video image. This is critical information if you have many of these media space windows, and people may move between them; i.e. it is important to know if Alice is in her office or in a coworker's office at a glance. As a further reminder of this information, the name of the person who is expected in the video image is displayed along the bottom of the window.

Over the upper left, upper right, and lower right corner are three small *cover icons* which allow the user to manipulate the capabilities he or she grants to the remote user that is pictured in the video (and shown with the face). Each of these icons is a toggle switch and the icon is displayed only when the capability has been given to the user. The cover icon is toggled by clicking on it with the mouse. These icons include (clockwise, from top left) allowing the remote user to see the local user, allowing the remote user to hear the local user, and allowing the remote user to be heard by the local user. Clicking directly on the video image implies allowing both the remote user to hear and be heard, enabling the idea of just clicking on the person you want to speak to. This is not strictly true, as the remote user will only hear the local user if such permission has been granted, but this is a common enough occurrence to merit the use this interaction technique.

Along the right edge of the window is a small *level meter* which is active whenever the local user is speaking and the remote user can hear the local user. The "level" of the level meter is a black line which moves up and down proportionally to the intensity of the users speech input, providing visual feedback that the system is functioning as well as alerting the user (with its motion) that he or she can be heard by others. Note that due to the fact that more than one remote user may have the ability to hear the local user at a time, many level meters may be active at once. This addresses one of the difficulties present in our first design: Our first design did not provide dynamic feedback that the audio device was currently in use.

Along the top of the display is a set of rectangles, which can be filled in with varying amounts of intensity (indicated by darker shades of grey to indicate higher intensity). These are an *activity indicator* for the system. These indicate activity in the system (from left to right) in the last minute, last 10 minutes, last hour, last 8 hours, last day and last 3 days. Activity can be either defined as speaking into the microphone to another user, or amount of change in the video scene. This is an important tool for helping the user determine the meaning of video images presented by the display. E.g. If the video shows that Bob is out of his office, did he just leave for a meeting or is he on vacation? At this point, it is unclear exactly how the cryptographic system should interact with this activity information. We currently are proposing that this information be "piggybacked" onto the video capability, i.e. if you can see the video of a particular user you can access their history information, but only from the point in time at which the video capability was granted.

In a previous section, we explained how groups were part of our solution to the cryptographic problems presented. Groups in our second design are represented very similarly to users; groups are given their own window with a similar set of cover icons in the same spatial locations as their counterparts on the user window. Manipulation of the cover icons works identically for the group window, but applies to all members of the groups instead of an individual user. A pop-up menu is accessible which controls the various functions necessary for groups (adding members, deleting members, etc.). This menu also lists the group's current membership.

Let us return to the issue of how conflicts are resolved between permissions granted to users and to groups. A criticism of our first design was the complexity of the display with respect to these conflicts, and we attempted to simplify this issue in our second design. We employed a *conservative-most recent change* permission strategy in this design. We chose this strategy in an effort to reduce the complexity of the group display as well as conserve screen space.

Let us first explain the "conservative" part of the second design. Unlike the previous design, which attempted to show the system's complete state, our second design does not allow states to be reached which require a "three-stated" display, i.e. if the user looks at a media (cover) icon on a group window, it is either displayed or not. Further, when our new design must resolve conflicts to avoid needing the three-stated display, it choose to err on the side of displaying information which is the worst possible situation with respect to data leaving the workstation. In other words, when in doubt, it will choose to show displays that give the user the *most* dangerous situation about data leaving the workstation. This can create "false alarms" about the security of their data, but can never *fail* to alarm the user. This is less flexible that our original design, but it significantly simplifies the display.

When the user manipulates the capabilities of groups, the capability is either immediately granted or removed from *all* members of the group, without considering whether or not they already had this permission. This is the origin of the "most recent change" part of our strategy. If at some later point, an individual user's capabilities are changed and he or she is a member of group, that group's cover icon for the relevant media is shown as permitted if *any* member of the group has the capability. E.g. If the group G contains 2 users Alice and Bob and the group is changed to not have the capability to hear the local workstation but Alice is individually changed to override this, the group display for G would indicate that the capability is present.

Having addressed users and groups with our new design, we turn to the user's media-centric questions. We chose to basically preserve our first design's display, with the small change that each media is now assigned its own window, which is the same size as the user and group windows. This window always has an icon present in it indicating the media type being displayed in the window and is titled appropriately. All the faces of users who possess the capability for the media type of this window will be shown.

All the windows- user, group, and media- have a pop-up menus associated with them. These are all accessed in the same manner; some functions are available from all the menus (such as creating new windows for particular media types) and some functionality is special to the type of window being interacted with. These menus are necessary because many (less-used) functions cannot be directly accessed through the windows.

Ultimately, the trade-off shown clearly by this pop-up menu (ease-of-access and visibility versus screen space) is a trade-off we were faced with throughout our second design effort. Many of the choices in the original design were made under the assumption that is was best to present *all* relevant security information. Our informal evaluations showed otherwise, and we have created our new design with the idea that the smaller displays with convenient access to common functions (and thus less convenient access to uncommon functions) is more in line with users needs.

Figure 2

IMPLEMENTATION

We have currently implemented the first of our two designs and are actively working on the second. The first design used a combination of C code for an interface to the cryptographic processing, hardware interfaces, and networking support and Tcl [16] code for the user interface (via the Tk toolkit). The cryptographic functions proper were supplied by RSA Laboratories (the RSAREF toolkit) and Eric Young (DES implementation). The current system contains about 5100 lines of C code and 3500 lines of Tcl code. The current implementation runs on Silicon Graphics and Sun workstations.

Although the user interface designs are set up to control many types of media, the current implementation can only work with audio input and output. This limitation is due to the restricted number of workstations present in our network with digital video input capabilities.

CONCLUSION AND FUTURE WORK

We have presented a discussion of the problems that can arise from using a media space on a shared digital network and some properties that a media space should have if it is to give users confidence that their privacy is being respected. We have also explained our approach to designing a system which can meet the constraints outlined as well as possible, given that quick termination of media space conversations is a trade-off with respect to temporal separation.

In the future, we plan to collect systematic data about the use of the system via instrumentation of the underlying software as well as user observation. We are particularly interested in measuring how well our group mechanism performs at saving time in actual use. There are also questions remaining about the effectiveness of our interface(s) at conveying the system state and allowing the user to effectively manipulate the systems capabilities.

Our next major implementation issue is the addition of video capabilities to the system, especially the ability to capture and transmit still video images, ala Polyscope [3] or Portholes [6]. Our preliminary investigations concerning the performance of the system indicates that the data throughput rate required for full motion may present new issues in the cryptography area as well.

REFERENCES

[1] Bier E., Stone M. Snap-Dragging. *Computer Graphics*, August 1986, p. 233-240.

[2] Bly S., Harrison S. , Irwin S. Media Space. In *Communications Of The ACM*, 36 (1), January 1993. p 28-47.

[3] Borning A., Travers M., Two Approaches To Casual Interaction Over Computer and Video Networks. In *Proceedings of Conference On Computer Human Interaction (CHI) '91, p. 13-19.*

[4] Deering S. Host Extensions for IP Multicasting. RFC 1112, August '89.

[5] Diffie W., Hellman M. New Directions In Cryptography. *IEEE Transactions On Information Theory.* 22: 644-654, '76.

[6] Dourish P., Bly S. Supporting Awareness In A Distributed Workgroup. In *Proceedings Of Conference On Computer Human Interaction (CHI) '92. p. 541-547.*

[7] Eriksson H. MBONE: The Multicast Backbone. *Communications of The ACM*, 37(8), August '94. p. 54-60.

[8] Fish R., Kraut R. Chalfonte B. The VideoWindow System In Informal Communication. In *Conference On Computer Supported Cooperative Work (CSCW)*, '90. p. 1-11.

[9] Fish R., Kraut R., Root R, Rice R. In *Communications Of The ACM*, 36(1), January 1993. p. 48,61.

[10] Gaver W. The Affordances Of Media Spaces For Collaboration. In *Proceedings Of Conference On Computer Supported Cooperative Work '92. p. 17-24.*

[11] Gaver W. Sound Support For Collaboration. In *Proceedings Of European Conference On Computer Supported Cooperative Work '91.*

[12] Gaver W., Moran T., MacLean A., Lovstrand L., Dourish P., Carter K., Buxton W. Realizing A Video Environment: EuroPARC's RAVE System. In *Proceedings of Conference on Computer Human Interaction (CHI) '92, p. 27-35.*

[13] Ishii H. TeamWorkstation: Towards a Seamless Shared Workspace. In *Proceedings Of Conference On Computer Supported Cooperative Work '90. p. 13-26.*

[14] Mantei M., Baecker R., Sellen A., Buxton W., Milligan T., Experiences In The Use Of A Media Space. In *Proceedings Of Conference On Computer Human Interaction (CHI) '91. p. 203-208.*

[15] National Institute Of Standards and Technology (NIST). FIPS publication 46-1: Data Encryption Standard. January 22, 1988. Originally issued by Nationa Bureau of Standards.

[16] Ousterhout J. Tcl And The Tk Toolkit. Addison Wesley, Reading Massachusetts.

[17] Rivest R., Shamir A., Adelman L. A method for obtaining digital signatures and public key cryptosystems. *Communications Of The ACM*, 21 (2), Febuary '78.

[18] Root R. Design Of A Multi-Media Vehicle For Social Browsing. In *Proceedings Of Conference On Computer Supported Cooperative Work '88. p. 25-38.*

[19] Sellen A. Speech Patterns In Video-Mediated Speech. In *Proceedings Of Conference On Human Computer Interaction (CHI) '92. p. 49-59.*

[20] Stults R. (1988) Experimental Uses of Video to Support Design Activities. Xerox PARC technical report SSL-89-19, December 1988.

[21] Tang J., Issacs E. Personal Communication.