Privacy Management for Portable Recording Devices

J. Alex Halderman, Brent Waters, and Edward W. Felten
Department of Computer Science, Princeton University
35 Olden Street, Princeton, NJ 08544, USA
{jhalderm, bwaters, felten}@cs.princeton.edu

ABSTRACT

The growing popularity of inexpensive, portable recording devices, such as cellular phone cameras and compact digital audio recorders, presents a significant new threat to privacy. We propose a set of technologies that can be integrated into recording devices to provide stronger, more accurately targeted privacy protections than other legal and technical measures now under consideration. Our design is based on an informed consent principle, which it supports by the use of novel devices and protocols that automate negotiations over consent and ensure appropriate safeguards on recorded data. We define the protocols needed for this purpose and establish their security. We also describe a working prototype implementation that safeguards audio recorded by laptop PCs in a wireless network.

Categories and Subject Descriptors

K.4.1 [**Computers and Society**]: Public policy issues—*privacy*

General Terms

Algorithms, security, human factors, legal aspects

Keywords

Privacy, recording devices, camera phones

1. INTRODUCTION

Cellular phones capable of recording audio, video, and still photographs are a popular new category of personal recording devices. Unlike conventional camcorders and audio cassette recorders, these new devices are small and cheap, with only minor size and price premiums over regular cell phones. They feature nearly ubiquitous network connectivity, which lets users publish pictures and sounds almost instantly or archive them to high-capacity external disks for long-term storage. Many consumers find these capabilities useful and fun, and sales of camera phones are expected to outpace the combined sales of digital and film cameras this year [9].

Despite these devices' widespread appeal, their rapid adoption has raised serious privacy concerns. Because camera phones are multifunction devices, most users carry them everywhere they go, including places like locker rooms and corporate offices where conventional recording devices would not usually be appropriate. Even in spaces where recording devices are more acceptable, the growing density of portable recorders has increased the likelihood that invasions of privacy will occur. The ability to record pictures and sounds is not new, but the increasing number of recording devices in use—and the rising frequency with which they seem to be misused in sensitive situations—have fueled privacy fears.

Proposed responses to these threats have included legal and technical measures. Municipalities across the country are pondering legal restrictions on the use of camera phones in privacy-sensitive settings [13], while some courts, offices, and private gyms have prohibited their use. A technological proposal called Safe Haven would transmit signals to all camera phones in a particular area to disable their recording features [10].

These approaches share several drawbacks. First, they restrict recordings according to the location of the recording device rather than the complete set of circumstances under which a recording is made. Location is only weakly correlated with the situations in which people feel their privacy is violated. Picture taking in a restroom or locker room may be a clear invasion of privacy, but an audio recording of an office meeting could be either a valuable record or an invasion of privacy depending on the context. A second problem with these approaches is that they address privacy only at the moment of recording, either permitting or forbidding the recording act. In practice, the usefulness or sensitivity of a recording is sometimes only apparent long after it is created and may depend on who is allowed to replay it.

In this paper we present a new approach for addressing privacy concerns in portable recording devices that avoids the weaknesses of the earlier proposals described above. Our system relies on closed devices to encrypt data during recording, and this encryption protects the privacy of the recording even if the data is transferred out of the device. We use a variant of Chaum's Dining Cryptographers protocol [4] to divide up encryption keys so that the privacy stakeholders—the potential subjects of a recording—retain control over whether the recording is decrypted. Any stakeholder can decide that the recording was an invasion of privacy and permanently block its release. This decision can be made at the time of recording, when the recording is released, or any time in between. Stakeholders can decide whether to allow decryption based on the circumstances of the request and the identity of the requester.

1.1 Privacy in Practice

Like any technology, our solution cannot provide tangible benefits unless it is used in practice. Its security depends in part on the participation and trustworthiness of all parties at the time a recording is made (although far less trust is required to ensure strong privacy protection after recording). As such, our scheme is vulnerable to the use of non-compliant recording devices and interference with the recording protocol by dishonest participants. Furthermore, parties to a recording who do not participate in our protocol will usually not be protected by the system. This class of problems has no obvious technical solution; indeed, technology is generally powerless to force its own adoption.

Legal measures, on the other hand, might be one way to overcome these difficulties and realize practical benefits from our system. In many states, existing telecommunications privacy laws prohibit the recording of telephone conversations without the consent of every participant, and other forms of recording in private spaces are sometimes similarly restricted. New legislation might not be necessary if courts recognize our privacy technology as one means of obtaining consent for the purposes of these laws (although in our system consent is obtained prior to playback rather than prior to recording). Although our system could still be circumvented by dishonest or non-compliant parties, they would face potential legal sanctions for invasion of privacy. Law-abiding recorders would also have an extra incentive to adopt our scheme: it could serve as an automatic tool for negotiating the required consent, in addition to its direct privacy protection benefits.

The law may be one useful tool for moving our privacy management system into practice, but there are many others, including economic and social forces. For instance, device manufacturers might be motivated to adopt privacy protection technologies if it meant governments would forgo regulating devices like camera phones. Customer choice could be another important driving factor. We envision a "privacy friendly" branding initiative that would allow recording subjects to identify devices that respect privacy. Since potential subjects would be more comfortable being recorded by devices that complied with privacy protection, customers would choose devices that supported it over ones that did not. As more people employ these technologies, a "network effect" will induce others to do so as well, so that their privacy will be protected too. Over time, social norms governing when it is appropriate to make recordings may evolve to require the use of technical mechanisms like ours. Specialized communities, like particular organizations or professions, might adopt their own rules that require the use of technological protections. For instance, the local policy of a doctors' organization might require the use of our scheme to safeguard patient confidentiality during medical consultations.

If the public remains concerned about invasive recordings, legal and social changes like these are inevitable. We believe our scheme will play a part in the evolutionary process that produces these changes, but it will not require any radical new measures in order to succeed.

1.2 Outline of the Paper

The remainder of this paper is structured as follows. Section 2 discusses the principles we enforce to protect privacy. Section 3 gives an overview of our architecture. Section 4 presents the secure-XOR protocol on which we base our system and discusses its security. Section 5 formalizes the requirements for our higher-level cryptographic protocols, presents their design, and establishes their security. Section 6 describes a prototype implementation of our system. Section 7 discusses the feasibility of alternative designs. Section 8 presents some related work, and Section 9 concludes.

2. PRIVACY PRINCIPLES

In designing a privacy protection architecture, we have to manage the inherent tension between our need for useful recording devices and our desire to respect privacy. A maximally useful device would be able to record any subject it was capable of sensing, yet absolute privacy protection would require that we record nothing at all.

To demonstrate the level of privacy achievable by technical means we have designed a system that errs on the side of strong privacy protections. Ultimately, the balance between usefulness and privacy will be determined by social expectations, and the strict requirements on which we base our system can be moderated (with appropriate adjustments to our protocols) to accommodate future norms.

To this end, our system enforces two principles:

- *Unanimous consent.* No event will be recorded without the consent of all persons present. No recording will be released to anyone without the consent of all persons who were present when the recording was made.
- *Confidentiality of policy.* To the extent possible, a person's decision to grant or withhold consent will not be revealed to anyone else. (For example, if a recording is released, the participants will be able to infer that they all consented. But if a recording is not released because consent was withheld, then ideally nobody should be able to tell how many people withheld consent or who they were.)

Our system applies cryptographic protocols that are designed to obey these principles. The recording, if made, is encrypted under a key that is managed such that unanimous consent is required to decrypt and such that consent can be granted or withheld confidentially. These protocols are based on a variant of Chaum's Dining Cryptographers anonymous message broadcast protocol [4]. We describe them in detail later in this paper.

Although anonymity is not one of our principles, ensuring the anonymity of the participants in a recording may be as important as protecting their privacy for some applications. Our protocol requires that the participants exchange keys and that they can be contacted by some mechanism after a recording is made, but it can be compatible with anonymous operation when desired. The details of providing anonymity will be application specific, but we expect that Incomparable Public Keys [17] and existing anonymous messaging systems [3, 16] would provide the basis for such an implementation.

Our requirement of consent implicitly assumes that everyone who is present when a recording is made has a privacy interest in that recording. This assumption merits a brief discussion.

In principle, a thoughtful human observer could make a case-by-case, context-based judgment to determine whose privacy interests need to be protected with respect to any particular recording. Yet in practice, such a manual process would be not only inefficient but unwise, as we do not want to trust any person to this degree. Our only option, then, is to design our technology to make certain assumptions about whose interests are at stake.

Some cases are easy. For example, an audio recording of two people conversing obviously implicates the privacy interests of both participants. We generalize this by assuming that whenever a recording is made, every person who is present at the time of the recording has a privacy interest in it. We include even people who, though present, are not directly recorded, such as participants in a group conversation who never speak. We do this on the conservative assumption that speech addressed to a person may implicate his privacy interests whether or not he speaks.

Sometimes conversations can involve private information about non-participants. For example, if two doctors talk to each other about the treatment of a patient, then the patient clearly has a privacy interest in the doctors' conversation. It is hard to see how any technology could distinguish this from a case where the doctors were discussing their own personal lives (involving, let us assume, only their own privacy interests). Accordingly, we give the two doctors joint discretion in deciding how to handle any recording of the patient-care discussion, and we rely on the doctors to honor their ethical responsibility not to allow the disclosure of the patient's information to others (just as we would rely on them to maintain the confidentiality of traditional written medical records).

Based on all these considerations, we believe that our approach of giving each person present a veto over creating recordings, and over the subsequent release of such recordings, is the best available course.

Our confidentiality of policy requirement also warrants justification. This protection is necessary for our system to approximate the privacy afforded when no recording devices are present. Often the mere knowledge that a recording exists and is being concealed can threaten privacy interests. If our system did not protect the confidentiality of policy decisions, a person's refusal to consent might lead to speculation that she had something to hide, or even to attempts to coerce her into changing her policy. This requirement may not be necessary in some privacy applications, but it comes at no additional cost, since we achieve it as a side effect of the mechanism we use to satisfy the consent requirement.

2.1 Law Enforcement Access

There may be times when law enforcement agencies have a legitimate need to access recordings against the wishes of the privacy stakeholders. Of particular concern is the ability of criminals to create "evidence free zones" by denying any permission to record their activities. For instance, witnesses to an assault might capture evidence on their portable recording devices, but our protocol would give the criminal the ability to confidentially deny permission to replay the recordings in court. On the other hand, any system for overriding privacy policies would have significant potential for misuse. Building such a system would also complicate the implementation of the privacy protection scheme and might introduce new vulnerabilities. We expect there will be much debate over whether (or under what circumstances) law enforcement access should be permitted. Pending the outcome of this discussion, our core protocol omits any override capability.

3. ARCHITECTURE

Our architecture assumes that each party who wants to create a recording carries a portable recording device running our system, and that all other parties who desire privacy protection carry similar devices with or without recording abilities. These devices must be capable of short-range wireless communication and have sufficient processor power to carry out cryptographic operations. Figure 1 depicts two such devices as we envision them, though our system scales to more than two devices.

To execute our protocol, the devices must be able to discover and exchange short messages with compatible devices within the recording area. The owners of these nearby devices are understood to have potential privacy stakes in any recordings. The set of in-range devices could be discovered by using the same medium as the intended recording (i.e., a method based on sound for audio recordings or based on light for photographs), or approximated by short range wireless radio such as Bluetooth. Following discovery, the devices may communicate over higher-bandwidth channels for subsequent protocol steps.

Our protocol requires that the devices exchange public signing and encryption keys. Typically, the use of public keys would require some type of Public Key Infrastructure. However, we are able to use a much simpler mechanism by making two critical observations. We first observe that if the devices communicate by a short range wireless link, then any public key that was broadcast as part of the protocol must have come from a device that was within the range (i.e., a device that has a privacy stake in the protocol). The second observation is that due to our principle of unanimous consent, a device gains no additional power by presenting multiple identities (broadcasting multiple public keys). Therefore, we are able to accept any public key that is broadcasted in a discovery phase as a valid public key and no additional authentication needs to take place. This concept is similar to that of using *Location Limited Channels* to authenticate public keys [1].

Even though it is acceptable for a device to give multiple public keys, it is important that every device be able to have at least one public key recognized by other members of the group. Therefore, devices should be able to detect interference and have a retransmission mechanism to protect against malicious jamming. If a device detects that it is being jammed, it might be appropriate for the device to warn the user of such an event.

In the following example we illustrate a typical two-party recording session. Suppose Alice and Bob are together in a room, and that each has a device as we just described. Bob tries to initiate a recording. His device surveys the immediate surroundings and discovers Alice's device. Alice and Bob will be privacy stakeholders in the recording.

The two devices exchange public keys and negotiate a symmetric encryption key using the protocol discussed in Section 5. Since Alice and Bob have privacy interests in the

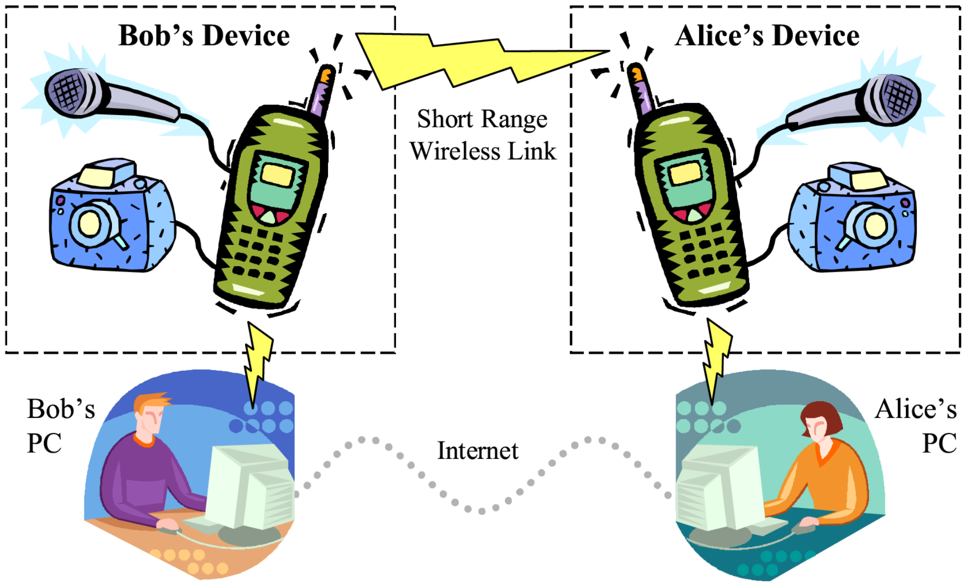

Figure 1: Our conceptual view of two recording devices.

recording, their devices retain "key shares" that can later be used to reconstruct the key in accordance with the principles of mutual consent and confidentiality of policy given in Section 2. Bob's device encrypts the recording and discards the key and plaintext. It stores the encrypted data along with information that will let him contact Alice in the future to request permission for decryption. Nobody can decrypt the recording without obtaining permission from Alice and from Bob, and at any time Alice or Bob can render the recording permanently unrecoverable by discarding his or her key share or replacing it by a random value.

We rely on the recording devices to faithfully execute our protocol, but our design allows the encrypted recordings to be safely transferred out of the recording devices and stored or decrypted using untrusted hardware. Bob could use this feature to transfer recordings from the limited storage of his portable device to his PC without having to ask Alice for permission. Bob could also email the recording to Carol, and Alice could grant Carol permission to decrypt it but not allow Bob to do so.

Now we illustrate how a protected file would be decrypted. Say Carol receives a copy of Alice and Bob's protected recording and wants to decrypt it. She runs software that executes our protocol (either on a portable device or a PC). The software examines the file header and sees that Alice and Bob have privacy stakes in the recording, so it contacts each of their recording devices (or similar software running on their PCs) and requests permission for decryption. Alice and Bob each decide whether to release the data to Carol. Their software can be programmed to determine this policy automatically using a set of rules (for example, they could specify that any file older than one year should be released), or it can prompt them to make a manual decision for every request. If Alice and Bob both grant permission, Carol's system can decrypt the file. If either denies permission, our confidentiality of policy requirement ensures that Carol cannot tell whether Alice, Bob, or both refused her request.

Our system supports recording and playback using two primary operations. The *Create* operation negotiates a key to encrypt and store a new recording. The *Decrypt* operation allows a playback device to recover the encryption key, but only in accordance with our privacy principles. Both Create and Decrypt are built on simpler protocols that implement a secure distributed XOR computation. We will first describe the distributed XOR protocol and establish its security, then we will discuss the two higher-level protocols.

4. SECURE XOR COMPUTATION

In our protocol each recording is encrypted with a secret key that is formed by XORing the private keyshare held by each owner [7]. Since we require unanimous consent of the owners to decrypt a protected recording, we allow any owner to veto the decryption by substituting a random value for her keyshare during the XOR computation. To preserve confidentiality of policy decisions, we must be able to securely compute the XOR of all the participants' keyshares so that a requester learns nothing more than the XOR of all these inputs. We achieve this by implementing a variant of Chaum's Dining Cryptographers protocol for anonymous message broadcast [4].

4.1 Protocol

Let there be n parties $P_1, \ldots, P_n$ who know secret values $k_1, \ldots, k_n$, respectively. Executing a Secure XOR computation will allow a third party Q to calculate $K = k_1 \oplus k_2 \oplus \cdots \oplus k_n$ without learning any of the arguments $k_1, \ldots, k_n$. We assume each party has a private signing key and that the corresponding public key is known to the other participants. The computation is performed using the following sequences of messages (which we assume are passed through a confidential channel):

1. $\forall_{i \in \{1,\ldots,n\}}\ Q \rightarrow P_i$:
$$\mathrm{ID}_{\mathrm{Req}} = \mathrm{S}_Q(\mathrm{ID}_Q, \mathrm{ID}_{P_1}, \ldots, \mathrm{ID}_{P_n}, \mathrm{Nonce}, \mathrm{Recording\ ID}, \mathrm{Time}, \mathrm{Description})$$

Party Q sends a signed request to initiate the secure computation to each of the parties $P_1, \ldots, P_n$ who will provide terms to be XORed. The request includes values representing Q's identity, (ID_Q) and the identities of the other n parties, ($\mathrm{ID}_{P_1}, \ldots, \mathrm{ID}_{P_n}$). A nonce is chosen randomly for the particular request, and the ID of the desired recording is given so the parties know which inputs to provide. The current time is given along with the type of operation (either Create or Decrypt for our higher level protocols).

Each participant will maintain a clock and a list of recently used request nonces. If a request has either a nonce on the list or has an inaccurate time stamp that is outside a tolerable range, the participant will refuse to continue with the computation. Request nonces with old enough timestamps may be purged safely from the list.

2. $\forall_{i \in \{1,\ldots,n\}}\ P_i \rightarrow Q:$
   $\mathrm{S}_{P_i}(\mathrm{ID}_{P_i}, \mathrm{ID}_{\mathrm{Req}}, C_{i,1}, \ldots, C_{i,n})$

   Upon receiving the request, each P_i generates n random values $B_{i,1}, \ldots, B_{i,n}$, each of length equal to the input size of the keyshares. Each P_i then signs each value and encrypts it to one of the other parties to form a set of encrypted *blinding factors*. We write the encrypted blinding factor generated by P_a and designated for P_b as $C_{a,b} = \mathrm{E}_{P_b}(\mathrm{S}_{P_a}(B_{a,b}, \mathrm{ID}_{\mathrm{Req}}))$. (The encryption must be non-malleable [6]). P_i signs a list of all n blinding factors he has generated along with his ID and the request ID, and he sends all this back to Q. P_i retains the values $C_{i,1}, \ldots, C_{i,n}$ until the protocol execution is complete.

3. $\forall_{i \in \{1,\ldots,n\}}\ Q \rightarrow P_i:$
   $\mathrm{S}_Q(\mathrm{ID}_{\mathrm{Req}}, C_{1,i}, \ldots, C_{n,i})$

   After Q receives responses from all n other parties, he collects all the blinding factors designated for each party P_i (i.e., he takes the ith blinding factor from each list), signs the blinding factors in addition to the request ID, and passes all this to P_i.

4. $\forall_{i \in \{1,\ldots,n\}}\ P_i \rightarrow Q:$
   $$X_i = \mathrm{S}_{P_i}(\mathrm{ID}_{\mathrm{Req}}, k_i \oplus C_{1,i} \oplus \cdots \oplus C_{n,i} \oplus C_{i,1} \oplus \cdots \oplus C_{i,n})$$

   In response to Q's message, P_i decrypts the blinding factors $C_{1,i}, \ldots, C_{n,i}$, verifies the signatures and request IDs, and extracts the values $B_{1,i}, \ldots, B_{n,i}$. He then computes the XOR of all these values, the values $B_{i,1}, \ldots, B_{i,n}$ he created earlier, and his secret argument k_i. He signs the result and the request ID and returns them to Q.

5. After receiving all n responses, Q can compute
   $K = X_1 \oplus X_2 \oplus \cdots \oplus X_n = k_1 \oplus k_2 \oplus \cdots \oplus k_n$.

   This relation holds because each random value $B_{i,j}$ has been XORed into the values $X_1, \ldots, X_n$ exactly twice—once by P_i and once by P_j. As a result, the values cancel and what remains is the XOR of $k_1, \ldots, k_n$.

The secure XOR computation guarantees two properties. First, if the requester Q is honest, then Q learns the result of the XOR operation and nothing else, and no other party learns anything. Second, if Q is part of a malicious collaboration, then the collaboration learns the XOR of the inputs of all of the honest participants P_i, and nobody learns anything else.

We compare our protocol to Chaum's anonymous broadcast protocol [4]. In Chaum's protocol each party shares a keystream with every other party, and each party XORs his message with the keystreams that he shares with every other party. The result is the XOR of all parties' outputs, which equals the XOR of all messages. Every shared keystream is factored in twice and cancels out in the final result. The anonymity of the broadcasts derives from the fact that each keystream is known only to the two parties who share it.

In our protocol the keystream between parties P_i and P_j is $B_{i,j} \oplus B_{j,i}$ and the messages are the size of keyshares. In our method each owner prepares n blinding factors that are signed and then encrypted. The requesting party collects all of them and redistributes the factors. Each party will then need to decrypt and verify n messages and send the final output back to the requester. We chose to put the communication burden on the requester. He must contact all the parties and collect all the messages. Additionally, only the requester learns the result in our protocol.

4.2 Security of Secure XOR

Chaum [4] showed that if the keystreams between parties were kept secret then any set of collaborating adversarial participants could only learn the XOR of the output from the collection of honest parties. (The keystreams between two participants where one is part of the collaboration is assumed to be known to the collaboration.)

The security proof of our algorithm follows Chaum's security proof closely. In our protocol, $B_{i,j} \oplus B_{j,i}$ serves as the keystream between participants P_i and P_j. For each request every honest participant insists on receiving blinding factors from every other participant. The blinding factors are signed and encrypted with a non-malleable cryptosystem and tied to a specific request. Since a request with a specific timestamp and request ID can only be used once, replay attacks will not work. Also, since the cryptosystem is non-malleable, a dishonest participant cannot create a blinding factor that is related to one between any two honest parties. Therefore, the blinding factors between any two honest parties form a Chaumian keystream and the collaboration can only learn the XOR of all the honest parties' input.

5. RECORDING STORAGE PROTOCOL

In this section we describe our Private Recording Storage Protocol (PRSP), a system that manages the privacy of recordings according to the principles of unanimous consent and confidentiality of policy. We begin by describing the types of participants and operations used by the protocol. Then we give formal definitions of security for the privacy requirements. Finally, we describe our protocol in detail and show that it is sound.

Operations in PRSP consist of interactions among three classes of parties:

- *Recording Creators.* Creators initiate the protected storage of recorded data in their possession by inviting one or more parties to become owners of the recording and then encrypting the recording with a key derived from shares provided by all the owners.

- *Recording Owners.* Owners have some privacy stake in a particular recording protected by PRSP, and they retain shares in the decryption keys. Successful decryption requires the consent of every owner for a particular recording. Recording creators may also be owners.

- *Recording Requesters.* A requester is a party who wants to decrypt a recording stored using PRSP. This requires obtaining the encrypted data and reconstructing the decryption key (with the consent of all owners). Recording creators and owners may also be requesters.

We assume that every party has public signing and encryption keys that are known to all other parties.

Parties engage in two basic operations: *Create* and *Decrypt.* A party initiates a Create operation before beginning a new recording. The operation involves participating in a secure XOR computation with the owners to derive a key then encrypting the recording and discarding the plaintext and the key.

A requester must initiate a Decrypt operation to view a protected recording. This operation entails another secure XOR computation with the owners. If every owners consents to the recording's release, the decryption will result in the original key and the requester can decrypt the recording.

5.1 Security Properties

Our goal in designing PRSP is to provide privacy protections in accordance with the two principles set forth in Section 2: unanimous consent and confidentiality of policy. Here we provide a formal definition of security for each property.

5.1.1 Unanimous Consent Requirement

Our first requirement is that an adversary cannot read a protected recording if she has been denied permission by any of the recording's owners. The following definition of security formalizes this requirement.

Let there be a non-empty set L of legitimate parties and an arbitrary-size set A of parties completely under the control of the adversary. The parties engage in a four-stage experiment, as follows:

1. *First Probe Phase.* The adversary performs arbitrarily many Create and Decrypt operations. For each Create, the adversary chooses the creator and owners from $L \cup A$ and provides the data to be protected. For each Decrypt, the adversary specifies the policy (grant or deny decryption) for every owner.

2. *Target Phase.* The adversary creates two data recordings of equal length. The adversary chooses a creator $C \in L$ and a set of owners $O \subseteq (L \cup A)$ with at least one owner from L. The creator flips a two-way coin outside the adversary's view and uses the result to select one of the data recordings provided by the adversary. The creator then performs the Create operation to protect that recording with owners O.

3. *Second Probe Phase.* This phase is similar to the first probe phase, but the adversary can now request decryption of the recording protected in the target phase with the following restriction: for each Decrypt call on that recording, the adversary must specify at least one party from L who will deny the decryption.

4. *Guess.* The adversary guesses the outcome of the coin flip based on information she gained in the preceding phases.

Our protocol satisfies the unanimous consent requirement if the adversary has at most a negligible advantage in guessing the result of the coin flip, that is, if the adversary cannot guess the result of the coin flip with probability non-negligibly better than $1/2$.

In this definition the adversary attempts to distinguish which of two recordings was actually encrypted. We make two remarks about our definition. The first is that the adversary is allowed to view any number of other recordings in the probe phases. These recordings could have the same set of owners as those in the target phase. Therefore, the actions an owner takes in allowing decryption of one recording should not give the adversary help in decrypting another recording. The second remark is that the adversary is allowed to attempt decryption of the target recording multiple times. Each time she will be denied by at least one legitimate owner, but it is possible that no legitimate owner denies decryption every time. Therefore allowing decryption in one instance to a particular requester should not allow decryption for other instances.

5.1.2 Confidentiality of Policy Requirement

Our second requirement is that if there are at least two parties acting outside the adversary's control and a decryption request is denied, then the adversary cannot determine which subset of these legitimate owners denied permission. This condition is independent of whether the requester has access to the original material and the owners' decisions on other access attempts. We model this property with the following experiment:

Let there be a set L of legitimate parties (where $|L| \geq 2$) and an arbitrary-size set A of parties completely under the control of the adversary. The parties engage in a four-stage experiment, as follows:

1. *First Probe Phase.* The adversary performs an arbitrary number of Create and Decrypt operations. For each Create, the adversary chooses the creator and owners from $L \cup A$ and provides the data to be protected. For each Decrypt, the adversary specifies the policy (grant or deny decryption) for every owner.

2. *Target Phase.* The adversary chooses a recording protected during the first probe phase that is owned by at least two parties from L and has been created by a party from L. Then he specifies $m \geq 2$ unique ways for the legitimate owners to deny decryption permission (that is, assignments of grant and deny policy decisions where at least one legitimate owner chooses to deny). An m-way coin is flipped outside the adversary's view to select which of the policy scenarios the participants will act out. The adversary then attempts a decryption request, which is denied in the way determined by the coin flip.

3. *Second Probe Phase.* The adversary makes an arbitrary number of calls to the Create and Decrypt operations (including the recording from the target phase) as in stage 1. She also has the power to attempt decryption on the target phase recording and have the parties deny him as they did in the target phase.

4. *Guess.* The adversary guesses the outcome of the coin flip based on information he gained in the preceding phases.

Our protocol satisfies the decrypt policy confidentiality requirement if the adversary does not have more than a negligible advantage in guessing the outcome of the m-way coin flip.

5.2 Protocol Description

PRSP uses the secure XOR computation to perform the Create and Decrypt operations as follows.

5.2.1 Create Operation

To create a recording containing plaintext T under PRSP, the creator sends an invitation request to the set of potential owners asking them to claim ownership in the recording. Parties who respond positively to this request participate in a secure XOR computation with the creator over an encrypted channel. The owners $P_1, \ldots, P_n$ generate random key components $k_1, \ldots, k_n$ and allow the creator Q to calculate $K = k_1 \oplus \cdots \oplus k_n$ using the secure XOR mechanism described above. When invoking a Create operation the creator will choose a random number to act as the recording ID. For every recording the owner will store her keyshare and the recording's unique ID. If a Create operation is attempted with an ID that has already been used then the owner will refuse to participate.

The creator encrypts the recording with key K and stores it along with the recording's unique ID, and the IDs of the owners: $(\mathrm{E}_K(T), \mathrm{ID}_T, \mathrm{ID}_{P_1}, \ldots, \mathrm{ID}_{P_n})$. The creator Q discards T and K.

5.2.2 Decrypt Operation

When a requester S seeks to decrypt a PRSP-protected recording, she initiates a secure XOR transaction with the recording's owners that includes the requester's ID and the recording's unique ID. Based on this information, each owner independently decides whether to grant the requester permission to decrypt the recording. If an owner chooses to grant permission, she submits her keyshare into the XOR computation. Otherwise she submits a random bitstring.

If every owner grants permission, the result of the secure XOR computation will be the correct decryption key K. However, if one or more of the owners declines permission, the computation results in a random value and the requester is unable to decrypt the recording's contents.

5.3 Security of Private Recording Storage

Here we demonstrate how our protocol meets the security definitions given in Section 5.1.

First, we consider the unanimous consent requirement. We notice that probing recordings other than the target recording accomplishes nothing because recordings are associated with unique IDs and each owner uses an independent keyshare for each recording. The creation of the target will not leak any information since only the creator, a legitimate user, will have to know which plaintext was chosen and learn the results of the first XOR computation. The creator will discard this information, in accordance with the protocol. The adversary will not be able to choose the key used for decryption, because the creator and at least one owner is legitimate. The legitimate owner will submit a random keyshare resulting in the key being randomly created.

Now suppose that the adversary requests a decryption of the target recording. At least one of the legitimate owners will deny decryption by providing random input to the XOR computation, so the XOR of the shares of the collection of honest parties will be random. Due to the security of the XOR computation the adversary will only learn the XOR of all the honest parties' inputs (which is random) and will therefore gain no information from any probe.

Next, we examine the confidentiality of policy requirement. In this experiment the adversary may control recording creators so we must assume that she knows the contents of any protected recording. However, the policy choice of each owner remains confidential.

Probing recordings other than the target recording does the adversary no good since there is a different keyshare for the target recording. In the target phase the adversary picks $m \geq 2$ ways that she can be denied by legitimate owners. A coin flip will then choose in which of these ways she is denied. Regardless of the coin flip, at least one legitimate owner will provide a random input to the XOR computation, ensuring that the collective XOR output from the legitimate owners will be a random value. By the security of the XOR computation the adversary cannot learn anything from this. Each time the adversary is denied, she will get a new random value unrelated to any of the earlier ones. Therefore, the adversary can learn nothing from the probes and will have no advantage in guessing the set of owners denying decryption.

6. IMPLEMENTATION

We have constructed a working prototype of our privacy management architecture that protects audio recorded by laptop PCs connected via wireless Ethernet. Our implementation consists of a suite of software programs as depicted in Figure 2. They are written in C++ and Python using the Cryptlib toolkit [8] for cryptographic functions. Currently, our prototype runs on the Linux operating system, and a Windows version is in progress.

We simulate portable recording devices with laptop PCs equipped with microphones and wireless Ethernet. Since the laptops do not have the necessary hardware to discover nearby devices and exchange public keys using broadcasts, we simulate these functions with a Room Server application that runs at a fixed network location. Device owners manually input their locations using our client software, which transmits this information, the device's IP address, and its public keys to the Room Server. The Room Server broadcasts these items to all devices reporting the same location. The owners of these devices are designated as privacy stakeholders for any recording made in that room.

Each laptop runs two applications, a Recorder and an Agent. The Recorder allows the user to begin audio recording, at which point it executes the PRSP Create operation to generate an encryption key. All nearby devices (as reported by the Room Server) participate in this operation and receive shares of the key. Each device also runs an Agent program that handles the privacy stakeholder's side of the protocol. The Agents listen for incoming Create requests, participate in key generation, and store keyshares for each recording. The user who initiates the recording process also runs an Agent and receives an equal share of the key.

In practice, the portable recording devices (represented by the laptops in our implementation) may not be reachable when decryption requests are made. Our solution is

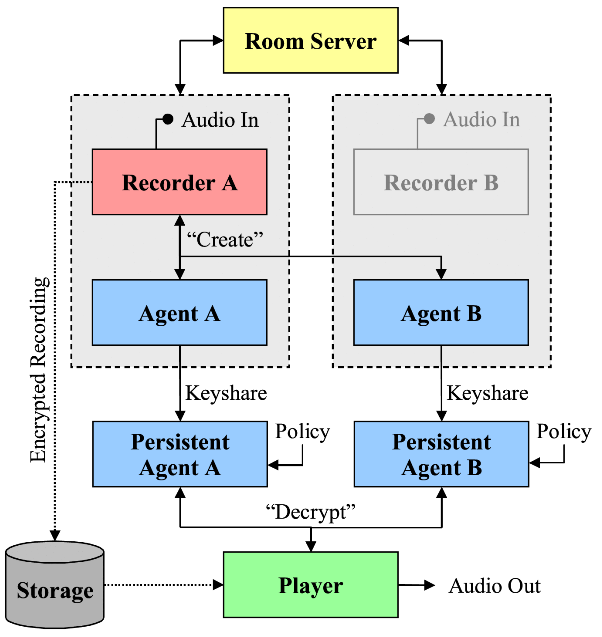

Figure 2: Major components of our implementation for recording and playback. The dashed boxes represent two portable recording devices. In this illustration, Party A is making a recording where Party A and Party B have privacy interests.

to allow each user to maintain a second Agent located at a fixed IP address with high connectivity. This persistent Agent periodically receives all the keyshares generated by the Agent in the portable device.

When a recording is complete, the Recorder compresses the audio data and encrypts it with the generated key. The Recorder stores the ciphertext in a file along with each stakeholder's public keys and the address of each persistent Agent. The user who made the recording can distribute or relocate this encrypted data file without any threat to the stakeholders' privacy.

Anyone who wants to decrypt the recording uses a Player program (suitable for use as a web browser "helper" application) that contacts the stakeholders' persistent Agents and executes the PRSP Decrypt operation. Each stakeholder sets the policy for her Agent; she can choose to grant such requests or to deny them. If all stakeholders grant the request, the Player calculates the correct decryption key and releases the plaintext audio. If any of the stakeholders denies the request, the Player calculates an incorrect key and reports an error.

7. ALTERNATIVE ACCESS RULES

We have defined and implemented a system that grants access to a recording if all of the stakeholders wish to participate in its decryption. In the literature there are many other types of secret sharing schemes that might be of interest to implement, such as k-out-of-n threshold schemes and general access schemes [11]. Implementing these types of schemes could be useful for privacy protection. For instance, one might apply the rule that if a majority of stakeholders wish to allow a recording to be decrypted then it may be decrypted. Another example is when an owner accidentally loses her share—it would be desirable to have a scheme that allowed the rest of the group to retrieve the data. These schemes might be of interest for various reasons, but we found that one of our primary design goals, confidentiality of policy decisions, conflicts with many of them.

Suppose we wished to implement a k-out-of-n threshold scheme while maintaining the confidentiality of decisions. In a k-out-of-n threshold scheme, if at least k out of n owners wish to allow decryption then the recording can be decrypted. However, due to our confidentiality of policy requirement, all owners must participate in any secure decryption.

Consider what would happen if we did not require all owners to participate. Suppose there were $k-1$ collaborators who attempted to decrypt a recording with one legitimate user. If the collaborators all allow decryption, then decryption will occur only if the one, isolated legitimate user agrees to it. This clearly violates our confidentiality of policy requirement.

Thus we must require that all users participate in decryption, but this means that one user can deny decryption by refusing to participate. The scheme therefore can not be k out of n.

Generalizing from this example, advanced schemes that require confidentiality of policy seem to be inherently difficult. Yet there is a slight relaxation of one threshold scheme that might work. We could use a secure election [5] to vote on what the outcome should be and then have the participants execute our protocol according to the outcome. However, this does require that one who votes "no" would still allow decryption if the results of the election are "yes."

8. RELATED WORK

The idea of sharing a secret among several parties has been known for some time [7]. Blakley [2] and Shamir [15] each independently invented threshold sharing schemes. Ito, Saito, and Nishizeki first studied secret sharing for general access structures [11].

David Chaum developed DCNet [4] for anonymous broadcast of messages in a group. However, we believe that the way we use it for secret sharing is novel. Additionally, Chaum did not specify in his paper how the participants share a keystream. We develop a correct, efficient method that only requires a pair of participants to asynchronously send messages to each other.

Balfanz et al. [1] describe the use of Location Limited Channels for public key authentication. In their scheme, a user is able to authenticate the public key belonging to some other device if it has some type of location limited communication channel with the other device, such as an infrared connection. They allow eavesdropping on a channel but require that an attacker must not be able to (undetectably) communicate on the channel. Our broadcast key distribution scheme is similar in that we require that any legitimate device should be able to transmit its public key. However, in our protocol a recording device seeks to gather public keys of all devices within a short range and thus will typically use a less limited communication medium for this purpose.

There has been a recent focus on the need to implement technical solutions to protect individuals' privacy in a world where computing is becoming pervasive. Recent work [12, 18, 14] has examined measures that can be taken to protect consumers from the emerging privacy threat presented by cheap RFID tags while maintaining the useful functionality of these devices.

9. CONCLUSION

The privacy management system we have proposed provides significant protection for the privacy rights of recorded parties while retaining many benefits of portable recording devices. People are recorded only when they consent, a recording is released later only with the consent of all parties present in the recording, and decisions about consent are kept confidential, to the extent possible.

This approach to privacy management offers a more effective solution to the privacy concerns associated with devices like camera phones than current alternatives—legal prohibitions and local area lockouts—which would enjoin all recording in specific places rather than only in situations where privacy is a concern. Since our approach is based on encryption instead of sealed storage, users can archive or distribute protected recordings the same way they do unprotected ones while still allowing the privacy stakeholders to decide whether and to whom the data will be released.

Our prototype demonstrates the feasibility of this approach in the context of a mobile computing platform. More widespread adoption in the context of camera phones requires only that the small number of cellular phone manufacturers employ these mechanisms. Growing public anxiety over the threat to privacy posed by densely distributed portable recording devices—and the threat of legislative responses if no proactive steps are taken—may induce manufacturers to embrace a design like ours.

10. REFERENCES

[1] D. Balfanz, D. Smetters, P. Stewart, and H. Wong. Talking to strangers: Authentication in adhoc wireless networks. In *Symposium on Network and Distributed Systems Security (NDSS '02), San Diego, California*, February 2002.

[2] G.R. Blakley. Safeguarding cryptographic keys. In *Proc. AFIPS 1979 National Computer Conference*, volume 48, pages 313–317, 1979.

[3] David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 4(2), February 1981.

[4] David Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *J. Cryptol.*, 1(1):65–75, 1988.

[5] Richard A. DeMillo, Nancy A. Lynch, and Michael J. Merritt. Cryptographic protocols. In *Proceedings of the fourteenth annual ACM symposium on Theory of computing*, pages 383–400, 1982.

[6] Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography. In *Proc. of the STOC'91(23rd)*, pages 542–552, 1991.

[7] H. Feistel. Cryptographic coding for data-bank privacy. RC 2827, 1970.

[8] Peter Gutmann. The design of a cryptographic security architecture. In *The Usenix Security Symposium*, 1999.

[9] Tony Henning. The camera-phone phenomenon. *The Future Image Report*, March 2003.

[10] Iceberg Systems, Limited. Safe Haven. www.icebergsystems.net.

[11] M. Ito, A. Saito, and T. Nishizeki. Secret sharing scheme realizing general access structure. In *Proceedings IEEE Globecom*, pages 99–102, 1987.

[12] A. Juels, R. Rivest, and M. Szydlo. The blocker tag: Selective blocking of RFID tags for consumer privacy, 2003.

[13] Jo Napolitano. Hold it right there, and drop that camera. *The New York Times*, December 11, 2003.

[14] Sanjay E. Sarma, Stephen A. Weis, and Daniel W. Engels. RFID systems and security and privacy implications. In *Workshop on Cryptographic Hardware and Embedded Systems*, pages 454–470. Lecture Notes in Computer Science, 2002.

[15] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

[16] P. F. Syverson, D. M. Goldschlag, and M. G. Reed. Anonymous connections and onion routing. In *IEEE Symposium on Security and Privacy*, pages 44–54, Oakland, California, 4–7 1997.

[17] Brent R. Waters, Edward W. Felten, and Amit Sahai. Receiver anonymity via incomparable public keys. In *Proceedings of the 10th ACM conference on Computer and communication security*, pages 112–121. ACM Press, 2003.

[18] Stephen A. Weis, Sanjay E. Sarma, Ronald L. Rivest, and Daniel W. Engels. Security and privacy aspects of low-cost radio frequency identification systems. In *1st Annual Conference on Security in Pervasive Computing*, 2003.