Privacy-Preserving DRM

Radia Perlman
Sun Microsystems
radia.perlman@sun.com

Charlie Kaufman
Microsoft
charliek@microsoft.com

Ray Perlner
NIST
ray.perlner@nist.gov

ABSTRACT

This paper describes and contrasts two families of schemes that enable a user to purchase digital content without revealing to anyone what item he has purchased. One of the basic schemes is based on anonymous cash, and the other on blind decryption. In addition to the basic schemes, we present and compare enhancements to the schemes for supporting additional features such as variable costs, enforcement of access restrictions (such as "over age 21"), and the ability of a user to monitor and prevent covert privacy-leaking between a content-provider-provided box and the content provider. As we will show, the different variants have different properties in terms of amount of privacy leaking, efficiency, and ability for the content provider to prevent sharing of encryption keys or authorization credentials.

Categories and Subject Descriptors

C.2.0 [**Computer Networks**]: General – *Security and protection.*
K.4.1 [**Computers and Society**]: Public Policy Issues – *privacy.*
E.3 [**Data**]: Encryption

General Terms

Algorithms, Design, Economics, Security, Human Factors.

Keywords

Algorithms, Protocols, Blindable Parameterizable Public key, pricay, DRM.

1. INTRODUCTION

Most work in the field of Digital Rights Management (DRM) focuses on the problem of preventing its circumvention. This paper looks at a different problem: how to charge for the use of data while allowing the user to maintain her privacy (in the sense of not revealing to the content provider what data was purchased by which user). In some scenarios, privacy is of greater concern to the user than the payment required. This paper presents and contrasts two basic approaches, plus variants, of systems in which content is distributed in encrypted form, and the user pays to receive a decryption key. The first is based on Chaum's anonymous cash [5]. The second is based on blind decryption [15].

In addition to the basic schemes, we provide various methods of enhancing these schemes for functionality such as different costs for different content, ability of a third party to create content to be distributed by the content provider, and enforcement of authorization policies. Additionally, we examine the scenario where for DRM enforcement reasons, there is a sealed box provided by the content provider on the user's premises, communicating with the content provider to acquire keys, and doing the actual decryption. We examine the problem of whether the user can detect or prevent the sealed box from covertly telling the content provider what content the user is decrypting. We show that it is impossible, if the user is only passively monitoring the channel, for the user to know whether the box is indeed leaking information. We then show a mechanism in which the user can cooperate with the box in forming the message to be sent to the content provider, and be assured there is no collusion going on, without impacting the ability of the content provider to enforce DRM.

Although the focus of this paper is not the crypto, we do introduce a new variant of asymmetric keys; the ability to have a family of blindable keys, parameterized by an arbitrary string, which we will use to encode information such as authorization policies or monetary units. This functionality can be provided by a somewhat unusual use of IBE (identity based encryption), but we also introduce two alternative algorithms, which lack some of the properties of IBE that are not needed for our application.

We will assume that there are enough items of content distributed by the content provider that the mere fact that a user is doing business with the content provider, and the amount of money the user spends with the content provider, is not a privacy issue. However, as we will show, privacy leaking is not absolute, and some of the solution variants have different tradeoffs.

Encrypted content must be accessed anonymously, though that is not the focus of the paper. Encrypted content might, for instance, be broadcast video, or content posted on the Internet. If the content is broadcast, say from a satellite or via cable TV, there is no problem with accessing the encrypted content anonymously. If the encrypted content is downloaded from the Internet, some sort of anonymization technique would be required, e.g., [10], [11], [16].

In addition to the encrypted content for an item, there will be associated metadata. An example of metadata might be the decryption key for the content, encrypted with the

public key of the content provider, or perhaps also an authorization policy for accessing that item.

Another aspect of DRM, also not the primary focus of this paper, is how to prevent a user from copying content and sharing it with others. There hasn't been a foolproof technical solution, especially since the analog output of video and audio has to be available. For instance, it is not uncommon for people to carry a camcorder into a theater, record the movie as it is played, and then sell copies later. Various proposed solutions for enforcing DRM include threats of prosecution if caught illegally copying and distributing, watermarking to discover which copy leaked [1], [7], [4], [9], and various software and hardware techniques to prevent copying [14], [12]. Even though there might never be a foolproof technical solution, it is common today for digital content to be distributed with some degree of copy protection, even in software-only systems. This is evidence that content providers believe that copy protection deters a sufficient amount of copying that the complexity (and customer annoyance) of the DRM is of positive value (to the content provider).

So this paper is not about how to make DRM itself more secure; it is instead focused on enhancing DRM with additional functionality.

We consider issues such as avoiding timing clues, enforcing authorization policies (such as "over 18", "citizen of US", or "citizen of any country except Monaco or Grenada") that might restrict access to some content; the comparative implications on our scheme variants when authorization policy might be very complex; the ability to do per-item accounting; and the ability of the user, when communicating with the content provider through a sealed box, to be assured that the box is actually preserving the user's privacy.

DRM enforcement commonly involves using a sealed box (e.g., the box that a video satellite provider installs at the user's house with a subscription to their service). We assume in such deployments:

- The box's only means of communication with anything is through a channel that can be monitored by the user.
- The user can modify messages to/from the box (the user can place a box between the box and its only means of communication to anything else).
- The user cannot examine the logic inside the box to determine whether the algorithms inside the box are indeed designed so as not to divulge the user's identity.

This fairly common deployment scenario leads to interesting functional differences between the schemes.

First we present the basics of the two schemes.

2. First: Basic anonymous-cash-based DRM

2.1 The concept of anonymous cash

Chaum [5] introduced the concept of anonymous cash. The basic idea is that a data structure with a particular syntax, signed with the private key of the bank, is worth a fixed amount of cash. The data structure includes a random number large enough to assure that independently chosen values will be unique. The anonymity comes from the construct of blind signatures, where Alice can get the bank to sign something without the bank knowing what it is signing.

Alice chooses a random number R, hashes it, and formats it according to the rules of valid currency, "blinds" it, and presents the blinded result to the bank, which signs the result with its private key. Then Alice applies the blinding function's inverse function ("unblind") to obtain a value we will refer to as "the bank's signature on R".

The bank will not know the values of R that Alice has purchased, so when R is "spent" the purchase cannot be traced to Alice, though the bank will know how many tokens Alice has purchased. Merchants accepting the anonymous cash can verify it is valid by checking the bank's signature. The only problem is assuring that Alice doesn't spend the same valid unit of anonymous cash more than once. If there is only one place accepting the anonymous cash (in this case the content provider), then double spending can be prevented by having the content provider remember all the R's that have been spent. Alternately, if the bank that is issuing the anonymous cash is online, then the cash can be spent with multiple merchants, provided that the bank remembers all the R values used, and is consulted by each merchant on each transaction before the anonymous cash is accepted.

Chaum, Fiat, and Naor extended the notion of electronic cash to allow for an off-line bank [6]. In this scheme, Alice might successfully spend digital cash multiple times, but once the bank collects the transactions (the spent cash), the culprit's identity will be revealed.

The latter anonymous cash scheme is more complex and expensive and our application does not require the off-line assumption, so we will use the simple notion of random R's that have been blindly signed in advance, to indicate that the holder of the signed R is allowed to trade that R for a unit of merchandise.

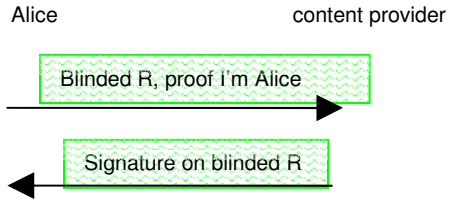
2.2 Using anonymous cash for DRM

In our application there is no reason for there to be a third party (the bank) providing general purpose tokens that can be spent with multiple merchants. Alice can directly purchase tokens from the content provider.

2.2.1 *Obtaining cash*

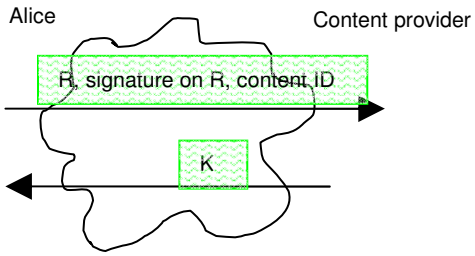
This will be done non-anonymously, in a conversation that must be authenticated and encrypted. The shaded text box indicates encryption.

Alice must pay for the cash through some mechanism such as a credit card, or having an account with the content provider, and having her account debited when she obtains cash.



2.2.2 *Purchasing content*

To purchase content, Alice presents the anonymous cash, together with the metadata for the content she wishes access to, and the content provider returns the content key. This interaction must be both **anonymous** (because the content provider will know what content is being requested and must not know who is requesting it), and **encrypted** (since otherwise an eavesdropper could steal the cash or the content key). The cloud in the diagram indicates an anonymization infrastructure. Note that an anonymization infrastructure is very expensive, in terms of computation and bandwidth [10].



Since the transaction where Alice is requesting a content key must be anonymous and encrypted, the metadata for an item could simply be the item's ID, and the content provider would keep a table of (content ID, content key) pairs. (In contrast, as we will see in section 3, in the blind decryption scheme, the metadata for an item must be {K}P, i.e., the content key encrypted with the content provider's public key.)

However, it might be preferable, even in the anonymous cash scheme, for the metadata to be {K}P rather than simply a "content ID" if:

- the content is to be prepared by a 3rd party; otherwise, it would be necessary for the 3rd party to securely relay the content key for that content to the content provider.
- it were inconvenient for the content provider to securely keep a large table of (content ID, key) pairs.

3. Second scheme: Blind decryption

In this second scheme, we use blind decryption instead of blind signatures. Blind decryption is similar in spirit to blind signatures, but there are more algorithms that work for blind decryption than blind signatures because blind decryption does not require a "public" key. Blind decryption works with various schemes including RSA keys (as with blind signatures), Diffie-Hellman keys, and IBE (identity based encryption).

3.1 Mechanics of Blind Decryption

3.1.1 *RSA Keys*

With RSA keys, blind decryption is a simple variant of blind signatures. If the content provider's public RSA key is (e,n), with the private key being (d,n), then the encrypted data key K will consist of K^e mod n.

To obtain K, Alice blinds K^e mod n by choosing a random number R, "encrypting" R with the content provider's public key, to obtain R^e mod n, multiplies the two quantities together to obtain (K^e * R^e mod n), and presents the result to the content provider, which uses its private key by raising to d mod n, resulting in K*R mod n, which it returns. Alice divides by R mod n to obtain K.

3.1.2 *Diffie-Hellman Keys*

Blind decryption can work with Diffie-Hellman keys, with any Diffie-Hellman group, including elliptic curves. We will call the operations "multiplication" and "exponentiation" although in the literature, elliptic curve operations are usually called "addition" and "multiplication". But we find the description with multiplication and exponentiation more clear for people who are familiar with Diffie-Hellman but not with elliptic curves. That way the formulae work with both mod p Diffie-Hellman and with elliptic curves. Note: the Diffie-Hellman blind decryption we are presenting is a simplification of one presented in [15], and it works for blind decryption, but would not work as a blind signature scheme. Also, for brevity, assume the operations are being done mod p (rather than having us say "mod p" each time).

Assume the content provider's public Diffie-Hellman key is g^x, and the private key is x.

A content key K is of the form g^{xy} . If the encryption algorithm requires a particular form factor for the key, such as being 128 bits, then some function would be performed on g^{xy} to convert it to the right form factor, such as a cryptographic hash.

The metadata associated with the item that is encrypted with key g^{xy} includes g^y.

In other words, g^{xy} (or more likely a cryptographic hash of g^{xy}) is used as a symmetric encryption key (for any symmetric key algorithm such as AES) to encrypt the content, and the metadata includes g^y. To decrypt the content, Alice must obtain g^{xy}. If blinding were not necessary, Alice could send the content provider g^y and have the content provider apply its private key (i.e., exponentiate by x), and return g^{xy} mod p. But we need this operation to be blinded.

Each item of content distributed by a particular content provider is encrypted with a different key (a different y was chosen), but they all use the same secret x. The value y is independently and randomly chosen for each item.

To blind g^y mod p so that the content provider cannot know which key Alice is purchasing, Alice chooses a value z and computes z^{-1} mod q, where q is the order of the cyclic group generated by g. For mod p groups, q is a large factor of p-1. She raises g^y to z to obtain g^{yz}, and sends that to the content provider.

The content provider raises this to its private key (x) and returns to Alice: g^{xyz}.

Alice unblinds g^{xyz} by exponentiating by z^{-1} to obtain the content decryption key g^{xy}.

3.1.3 IBE (Identity-based encryption)

The Boneh-Franklin (BF) scheme used in IBE [2] can also be used by our scheme for blind decryption, although we will be using it in a different way. In IBE, as traditionally used, there is a master key generator, anyone knowing the domain parameters can generate a public key from a string, and the master key generator calculates the corresponding private key (using the domain secret), and gives the private key to the rightful owner of the public key.

However, in our schemes, there is only one "rightful public key owner" -- the content provider. In the way we use the BF math, the content provider will act as the master key generator, in the sense of knowing the domain secret, but it will not give private keys to anyone (other than calculating its own private key). Clients will never know any private keys; they will only know the domain parameters in order to obtain the content provider public key.

In "normal" IBE, there would be a family of public keys, parameterized with a string "ID". At this point in the paper, we only need a single public key (the content provider's public key), so we can assume that "ID" is a constant. Later in the paper (section 6.3.3) we will want to use a string to create a family of keys, but they will all still be public keys belonging to the content provider.

To create a blindable public key, we will modify a simplified version of Boneh-Franklin IBE. Recall that the BF scheme uses a bilinear map ê(P,Q), (usually a twisted Weil or Tate pairing) which maps two order q elliptic curve points to an order q finite field element, and has the property that $\hat{e}(P^a, Q^b) = \hat{e}(P,Q)^{ab}$, for points P, Q and integers a, b.) Recall also that the security of BF relies upon the Bilinear Diffie-Hellman assumption that given P, P^r, P^s, P^t, it is difficult to find $\hat{e}(P,P)^{rst}$.

In the case of the basic IBE scheme, a trusted server, called the private key generator, chooses a secret integer, s, and an elliptic curve point P, and it publishes as system parameters P^s, P, and a specification of the group that P lives in. The private key generator can generate a private key corresponding to any public key, "ID", by using a special hash function H to map "ID" to an element of the group generated by P. We will write H("ID") as P^t, despite the fact that no party, including the key generator, will be able to compute t. This notation (P^t) is simply used here to make the bilinear Diffie-Hellman problem embedded in the scheme more transparent. The private key corresponding to "ID" is $H("ID")^s$, which may also be written as P^{ts}. To obtain a shared secret key with the holder of the public key, "ID", an encryptor chooses a random number r, and transmits P^r. The shared secret is then $\hat{e}(P,P)^{rst}$ which is calculated as $\hat{e}(P^s, H("ID"))^r = \hat{e}(P^s, P^t)^r$ by the encryptor, and $\hat{e}(P^r, H("ID")^s) = \hat{e}(P^r, P^{ts})$ by the holder of the public key "ID".

Blinding may be added as follows: suppose a message is encrypted with $\hat{e}(P^r, H("ID")^s)$, and you know P^r and "ID". Now suppose you want to decrypt the message with the help of the "ID" holder, but you don't want him to find out which value of P^r was used – since that would unambiguously identify the message you are trying to decrypt. You can do this by choosing a random blinding factor, b, and sending P^{rb} to the holder of "ID". He will send back $\hat{e}(P^{rb}, H("ID")^s) = \hat{e}(P^{rb}, P^{ts}) = \hat{e}(P,P)^{brst}$. You can now get $\hat{e}(P,P)^{rst}$, by raising $\hat{e}(P,P)^{brst}$ to the b^{-1}(mod q).

3.2 Purchasing Content with Blind Decryption

In the anonymous cash scheme, when Alice is purchasing a content key, she must do it anonymously, and the conversation must be encrypted. In our blind decryption scheme, it is not necessary for the conversation to be anonymous or encrypted, but it does need to be integrity-protected (signed by Alice).

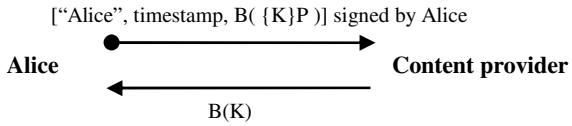
There is no need for an anonymizing network. The content provider will know which user (Alice) is accessing an item, and it can debit her account at that time, but it will not know *which* item Alice is accessing.

The protocol for requesting decryption is for Alice to send the content provider a message containing Alice's identity (so her account can be charged for the decryption), along with an encrypted blob (consisting of the blinded encrypted key) that the content provider will "decrypt" with its private key. ("Decrypt" is in quotes because the result will still be

encrypted with the blinding function). This message must be signed by Alice, e.g., with a MAC with a secret key Alice shares with the content provider, or signed with her private key, because her account will be debited for the cost of the decryption and we must assure that a third party cannot request a decryption be charged to Alice. It also must be resilient against replays, so an eavesdropper cannot cause Alice to be charged multiple times for the same decryption.

A simple method of avoiding replays without adding messages is for Alice to include a timestamp or sequence number, and have the content provider store the timestamp (or sequence number) of the previous decryption request from Alice, and ensure that the timestamps from Alice are monotonically increasing.

Alice will not be anonymous in this scheme. She will authenticate to the content provider, and her account will be debited for each decryption of a content key she requests. The content provider will know that Alice has purchased *some* content, but not *which* content.



Using blind decryption to obtain a specific encryption key

4. Comparison of the basic schemes

4.1 Efficiency

The blind decryption scheme is dramatically more efficient because it does not need an anonymization infrastructure.

Also, in the anonymous cash scheme there needs to be two conversations; a (nonanymous) conversation to purchase tokens, followed by an anonymous, encrypted conversation to request (and pay for) a content key. In contrast, with blind decryption, there need only be a single interaction; debiting Alice's account, and having Alice request a content key is done in the same (nonanonymous) two-message exchange.

Also, with the blind decryption scheme, the content provider only requires a single private key operation (to blindly decrypt {K}P). The anonymous cash scheme requires one private key operation for the content provider to blindly sign each token, as well as a private key operation to establish the server-side-authenticated encrypted channel required for content key requests. Additionally, the anonymous cash scheme is likely to require an additional private key operation to set up the encrypted conversation in which Alice purchases tokens, although it could be done with a long-term shared secret key between Alice and the content provider, and many tokens can be purchased in the same conversation. Additionally, although we showed the protocol where the metadata is the content ID, and retrieving the content key is a table lookup, that scheme requires the content provider keeping a large database (keys for all the content items), so it is likely to be preferable for the metadata to be {K}P, in which case the anonymous cash scheme would require at least 3 private key operations for the content provider, vs 1 for the blind decryption scheme.

The main expense of the anonymous cash scheme (compared to the blind decryption scheme) is the cost of the anonymization infrastructure, both in bandwidth and computation, placing computational burdens not just on Alice and the content provider, but also on the relay nodes. Although obtaining the encrypted content (in either scheme) might in some cases require an anonymization network, there are scenarios (such as acquiring content through broadcast video) in which the blind decryption scheme would not need such a channel. However, the anonymous cash scheme will *always* require the existence of an anonymization infrastructure (though in most descriptions of anonymous cash in the literature, this important detail is omitted).

4.2 Per-item accounting

The anonymous cash scheme allows the content provider to know how many people have purchased each item of content, (although it does not know specifically which people have purchased which content). In contrast, the blind decryption scheme does not allow this.

It might be important in some applications for the content provider to know how many people have purchased each item, in order to determine the royalty amount for each content contributor. However, many schemes deployed today (e.g., premium TV channels that show many movies) do not have any mechanism for the content provider to know how many people have watched specific movies. Payment to receive a premium channel is a flat rate regardless of how much or which content is accessed within that channel. So in many applications this per-item accounting is not required.

5. Some additional features (both schemes)

5.1 Variable Charging

It is possible that some content might cost more than other content. With the anonymous cash scheme, it is simple to charge different amounts for different content, since the content provider knows which key is being requested. So, the content provider could require n tokens to purchase an item worth n units.

This straightforward approach doesn't work in the blind decryption scheme, since the content provider does *not* know which key it is decrypting.

5.1.1 Multiple Keys

In blind decryption, a piece of content that costs n units of money could require n encryption keys and n decryption requests. So for instance, the metadata for an item costing n units could contain, for i = 1 through n, $\{K_i\}P$. Alice would need to decrypt each one of the K_i, and then perhaps $\oplus$ them or hash them together to obtain the content key.

Note that requiring n decryptions or requiring n blindly signed tokens to purchase an item worth n units, puts a burden of n-1 additional private key operations on the content provider, in either scheme (either it has to blindly sign n tokens, or do n blind decryptions).

5.1.2 Multiple-value tokens and multiple-value public keys

Instead of making an item worth n units require n private key operations, we can make it require, say, $\log_2$ n operations, using either anonymous cash or blind decryption, by having the content provider have different public key pairs for different denominations of money.

For instance, with the anonymous cash scheme the content provider could have a public key, P1, worth 1 unit; P2 worth 10 units; P3 worth 100 units. When Alice purchases anonymous cash, she can specify the denomination that she'd like. If she specifies she wants a 100-unit token, the content provider would debit her account 100 units of money, and blindly sign with public key P3. To purchase something worth 14 units, she could present 14 single tokens, or a 10-unit token plus 4 singles.

This savings can also be done with blind decryption. Suppose there was an item worth 14 units. (Assuming the denominations of the content provider's public key are 1, 10, and 100), the metadata associated with the 14-unit item would contain 5 wrapped keys; ((unit=10, {K1}P2), (unit=1, {K2}P1), (unit=1, {K3}P1), (unit=1, {K4}P1), (unit=1, {K5}P1)). Alice would need to do 5 blind decryptions, each time specifying the unit, e.g.,

> ["Alice", timestamp, B({K}P2) **unit=10**] signed by Alice

And the 5 keys would be cryptographically combined to form the content key.

Note that if the metadata gives Alice the choice of unwrapping 14 single-unit keys, or 5 variable-unit keys (e.g., a ten and 4 ones), then these keys could not be simply be hashed together to form the content key. Either the function would have to be $\oplus$ (where it is easy to make two different sets yield the same answer), or if a hash was used, you'd wind up with two different quantities, say K1, and K2. The real content key C could be stored in the metadata as {C}K1 and {C}K2, so that C would be retrievable whether Alice had computed K1 or K2.

5.1.3 Issue: Privacy and large-unit tokens or decryptions

If all G-rated content cost 1 unit, and all X-rated content cost 10 units, the variable charging could leak information. In the anonymous cash scheme, Alice could buy anything she wants with (lots of) unit tokens, and the content provider would not know who was purchasing the expensive content. Or even the fact that she has purchased a large denomination note does not mean she is intending to buy a single expensive item, since she could pay for multiple single-unit purchases in the same transaction with a single large-denomination note.

With the blind decryption scheme, Alice is not anonymous, and has to unwrap the content in the same denominations that it was wrapped. To help protect privacy:

- Alice could spread decryptions over time, so the content provider wouldn't be able to tell the exact amount of any item (e.g., for a 14-unit item, she could request decryption of the 10-unit key at a different time from requesting the 4 single-unit keys).
- The content provider could provide metadata for a 14-unit item that would allow retrieving the item using n single-unit decryptions, rather than the smaller number of decryptions possible using larger denomination keys. Both types of metadata could be provided, giving Alice the choice. So, the content key could be the $\oplus$ of 14 single-unit decryptions in the metadata, or the $\oplus$ of a ten-unit decryption plus four single-units.

To avoid having users opt for unwrapping content using single unit keys (putting a computational burden on the content provider), the content provider could provide a lot of content (rather than just X-rated content) that is worth more than one unit, for instance a package of all the Disney movies together, or entire seasons of "Little House on the Prairie". Or, the content provider could provide a discount for using the larger-unit keys (the metadata for a 14-unit item could give Alice the choice of unwrapping 14 single-unit keys, or, say, a 10-unit key and two single-unit keys, so that the item would cost only 12 units if she uses the larger denomination key. Or in the case of purchasing anonymous cash, the content provider might provide discounts for large-value tokens, e.g., charging 9 units to obtain a 10-unit token.

5.2 Timing Issues

There might be a piece of popular content that many users may attempt to access at the time that it is broadcast for the first time. The fact that someone is asking to access something at just that time would be a clue that the user is likely accessing that particular piece of content.

To mitigate this issue, the content provider should provide the metadata for content well in advance of the broadcast. Even if the data for the content does not exist, there is no reason why the key with which that content will be encrypted could not be chosen well in advance, and posted. Then users can collect the metadata for that content and request decryption of the key(s) well in advance of the existence of the content.

6. Authorization Categories

In some cases it is not sufficient to pay for content; one must be authorized to purchase that particular content. For example, X-rated content might only be legally purchasable by someone over age 21. Or some other content might only be legal to sell to citizens of some countries. The system must allow anonymous purchase, but only to qualified individuals.

In this section we discuss three different methods of providing for authorization, and if/how each of the two basic schemes can be modified with each of these:

- Authorization secrets used as credentials
- Authorization secrets used as content key components
- Content provider keys parameterized by authorization policy

The various approaches have different tradeoffs in terms of amount of privacy information leaked, efficiency, functionality, and ability to prevent credential sharing.

The authorization policy for an item must appear in the metadata in cleartext, so that Alice can tell what types of authorization she must obtain in order to purchase the item. We will use the term "ACL" to mean the authorization policy associated with an item, and assume it can consist of any Boolean combinations of groups, roles, identities, attributes, etc.

An obvious concern is that any sort of authorization secret could be copied, and sent to non-authorized users. However, this is not a special concern with authorization, since this is also true of the content keys. The entire system depends on some sort of DRM enforcement to hinder sharing of content keys as well as authorization secrets. One mechanism, which we will explore in greater depth in section 7, is to use a sealed box like the one that comes with a subscription to satellite TV or cable. But software-only DRM schemes are prevalent today, even though they aren't 100% effective. So, they must be sufficiently effective at deterring sharing to satisfy the content providers.

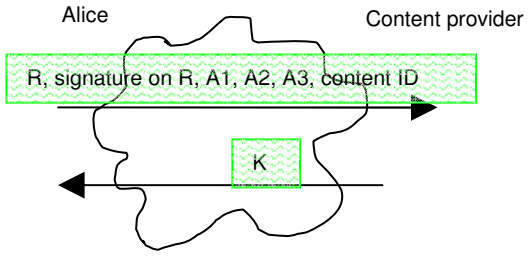
Assume for each authorization category (e.g., over 21, or citizen of country X), there is a server that can determine whether someone is a member of the relevant group or has the relevant attribute. If Alice can prove to that server that she has attribute Z, that server presents her with a secret, S_Z.

To prevent an eavesdropper from stealing the secret, the conversation in which Alice obtains S_Z must be encrypted. To prevent Alice from sharing S_Z with unauthorized users, some sort of DRM scheme must be in place.

To lessen the threat of authorized users sharing authorization secrets with others, given that a DRM scheme is likely not to be 100% effective, the authorization secret can, in some of our schemes, be changed periodically, and then authorized users will need to get the new value when their old value becomes invalid. In one of our schemes (public keys parameterized by ACL), there are no authorization secrets to share.

6.1 Authorization secrets as credentials

This scheme only works with the anonymous cash scheme. When Alice is anonymously requesting a decryption, she presents all the authorization secrets, A1, A2, A3 that prove she satisfies the ACL for the requested item, along with anonymous cash. It will be known which authorization secrets Alice has ever obtained, but not whether she ever uses them to purchase ACL-restricted content. For maximum privacy, it might be best for Alice to automatically request all authorization keys for which she is eligible so as not to leak any hints about what kinds of content she might be seeking. An authorization secret would only need to be obtained once (per user), and that would enable that user to access any content that requires that authorization.



Content provider looks up ACL associated with "content ID", and verifies that A1, A2, and A3 are sufficient credentials to satisfy the ACL

It is straightforward to accommodate complicated authorization policy, e.g., of legal age in the country of residence. Since the ACL is part of the metadata, the client can calculate what credential secrets need to be sent to satisfy the policy. The content provider can know what the policy for that content is, in one of two ways:

- The content provider stores, for each item of content, (content ID, key, ACL)

- To save the content provider from keeping such a large table, the metadata for the content would be [{K}P, ACL] signed by content provider.

However, there is a potential for privacy leaking. If there is a group with a very small number of members, and someone requests access to something requiring being a member of that group, there is no way to avoid leaking that someone from that group accessed that item. Even if all groups were large, it could be that the intersection of several groups could be very small. If access to the item is the AND of a bunch of groups, it is unavoidable (with this scheme) to divulge that someone who is in the intersection of all the groups has accessed the item.

The issue is with the OR of several groups. Suppose the ACL says that you must be accredited as fluent in at least 3 languages, and Alice happens to know Bulgarian, Bengali, and Navajo. When the anonymous requester presents those three credentials, it will narrow the potential requesters to a very small set, even though each of the groups is large, and even though the ACL would usually allow for satisfaction while still being part of a very large potential set (e.g., with English, French, Spanish).

One feature of this scheme (as opposed to the one we will present in the following section), is that it is relatively easy to periodically change the authorization secrets, to mitigate against some stealing of credentials. When an authorization secret has changed, the user will have to obtain the new secret.

6.2 Using authorization keys to encrypt content

This variant works with either anonymous cash or blind decryption. We assume that Alice obtains a (symmetric) encryption key for each authorization category that she qualifies for. As with section 6.1, it will be known which authorization secrets Alice obtained, but not whether she ever purchases content requiring them.

This scheme can handle any Boolean combination of authorization categories. To access an item that requires, say, authorizations X and Y, Alice would need to have obtained authorization secret keys K_X and K_Y, in addition to the K wrapped inside the metadata. So, the metadata might consist of: ({K}P, $\{K_1\}K_X$, $\{K_2\}K_Y$). The decryption key for the content could be, for instance, $h(K,K_1,K_2)$. Alice unwraps {K}P with the help of the content provider, but is able to unwrap K_1 and K_2 because she knows K_X and K_Y.

The OR operation would require organizing the metadata to give the client the choice as to what to unwrap. For example, if the ACL was "citizen of US OR citizen of Canada", the metadata might contain (("citizen of US", $\{\{K\}P\}K_{US}$) , ("citizen of Canada", $\{\{K\}P\}K_{CANADA}$)).

If there were an ACL such as "citizen of any country other than Monaco" this would require a large amount of metadata, since that would be the OR of hundreds of countries. In contrast, the authorization claim secrets scheme (6.1) only requires that Alice present the single authorization claim secret for some country other than Monaco (we won't worry about whether someone who is a dual citizen is allowed to see content in this case).

In this scheme (using the authorization secret as a decryption key), it is not as easy to periodically change an authorization secret as it would be in scheme 6.1. It could be done, but it would involve preparing new metadata for all affected content.

6.3 Authorization category-specific public keys

In this scheme, the content provider has different public keys, one for each authorization group. In the blind decryption scheme, this would mean that an encryption key for an item would be wrapped with a category-specific public key. In the anonymous cash scheme, it would mean that the cash token would be signed with a category-specific public key. In other words, in the blind decryption request, Alice would specify "blindly unwrap this using your 'US-citizen' key", and in the anonymous cash purchasing request, Alice would specify "blindly sign this using your 'US-citizen' key".

These could be completely independent keys, or they could be generated cryptographically through any of the schemes that we will present in section 6.5.

6.3.1 Boolean combinations with blind decryption

Boolean combinations of authorization categories can be handled, with blind decryption, the same way as in scheme 6.2. In other words, an item requiring authorizations A_1 AND A_2 could be encrypted with $h(K_1, K_2)$ and include as metadata (A_1: $\{K_1\}P_{A1}$) and (A_2: $\{K_2\}P_{A2}$). Alice would have to unwrap both keys to read the item. The keys would have to be half the price of the intended cost of the item. The metadata for A_1 OR A_2 would be similar, but just have a single K, such that unwrapping either quantity will work, as in: ((A_1: $\{K\}P_{A1}$) OR (A_2: $\{K\}P_{A2}$)), and either of those unwrappings would be the actual cost of the item.

6.3.2 Boolean combinations with anonymous cash

With anonymous cash, (assuming the metadata is just the content ID), it works somewhat like scheme 6.1, in that a cash token signed with an authorization-specific key works both as a unit of currency and as proof of authorization. If Alice has to prove A_1 OR A_2, she merely presents either a token signed with the A_1-specific public key, or a token signed with the A_2-specific public key. If Alice has to prove A_1 AND A_2, during the anonymous content request, she could present two (half-price) tokens, one signed with A_1 and one signed with A_2.

6.3.3 ACL-specific keys

An alternative for Boolean combinations is to have a public key which is specific to the entire ACL, e.g., a specific

public key for "(paid up member of ACM OR IEEE) AND citizen of US". In other words, in the blind decryption scheme, the metadata would consist of $\{K\}P_{ACL\text{-}string}$. In the anonymous cash scheme, the client would request a cash token signed with the ACL-specific key $P_{ACL\text{-}string}$.

That approach has the disadvantages of

- requiring a lot of content provider keys (but in section 6.5 we will explain how that can be practical), and
- leaking privacy, because although there might be a lot of items of content requiring each of the component authorization categories, there might be very few (or even just a single one) with the specific combination of those categories in the ACL.

6.4 Comparison of 6.3 with 6.1 and 6.2

With authorization-specific content keys, Alice cannot cheat by stealing authorization secrets, since when she requests cash tokens, or requests blind decryption, she is not anonymous, and the content provider checks her authorizations by looking them up in her profile. However, it has a serious privacy disadvantage relative to the other two schemes, that the content provider will know how many decryptions Alice is asking for, for each ACL.

6.5 Blindable Parameterizable Keys

In this section, we present a new cryptographic tool; blindable parameterizable keys, and give several ways of accomplishing this. Armed with such functions, the content provider can have a family of keys, parameterized by the ACL.

6.5.1 Using Identity Based Encryption

The notion of keys parameterized by a string sounds a lot like IBE [17] [2], and indeed the same math can be used for parameterizable blind decryption (but not blind signatures), but we are using IBE in a different way.

We described in section 3.1.3 how to use IBE for blind decryption, but in section 3.1.3 we were not parameterizing the single content provider public key. To make the scheme work with a different public key for every ACL string, we make it more like IBE in the sense that the public key used *is* derived from the ACL string. The rest of the system still works as it did in section 3.1.3 – the content provider knows the domain secret, and can convert any public key into a private key, and the clients never need to know any private keys; just the domain parameters.

6.5.2 Parameterized Diffie-Hellman

Parameterization can be done with our Diffie-Hellman variant of blind decryption. Alice would only need to know "g", and "p". The content provider would only need to know a single secret "x". The metadata for content for "over 21", would consist of (g^y mod p, "over 21"). The content key for that data would be calculated by calculating S=h(x, "over 21"), and then raising the metadata to S to obtain the content key g^{yS} mod p.

Alice blinds g^y mod p by choosing a random z, calculating the inverse exponent z^{-1} for mod p exponentiation, and presents that along with the string "over 21". The content provider uses the string "over 21" to calculate S, and returns g^{yzS} mod p. Alice exponentiates, mod p, by z^{-1} to obtain g^{yS} mod p, the content key.

6.5.3 Parameterizable RSA

Note that the schemes we present in sections 6.5.1 and 6.5.2 work for blind decryption but not blind signatures, so neither of them would work for anonymous cash. A scheme that might work as a blindable parameterizable public key scheme is RSA, where the content provider's public key, instead of being (e,n), is simply the modulus n. The public exponent for a given ACL would be the hash of that ACL string.

RSA is clearly not secure if multiple users use the same modulus, since knowledge of a key pair allows you to factor the modulus [3], but we are not proposing that. Instead we are proposing a single user using modulus n, but using a family of exponent pairs parameterized with a string.

It is a good idea for all the public exponents to be relatively prime, (so that Alice can't convert something raised to e to something raised to an exponent that she is authorized for). With exponents being hashes, this threat is unlikely to ever happen in practice, but it is possible (with some computational cost) to make all the exponents prime by not simply hashing the ACL string, but instead, hashing the ACL string, padding with some number, e.g., 32, of zero bits, and then finding the first prime greater than that.

6.6 Multiple Users on the same system

It is common to have multiple users in the same household sharing the same system. They might have different authorizations. For instance, the parents might be over 21 and the children might not be.

In such cases, there must be some ability to maintain multiple distinct accounts, and have some sort of log-in so that the system knows on which user's behalf it is acting. The system should keep a database on behalf of each user, of items such as authorization secrets, content keys, and anonymous cash tokens.

When anyone in the household purchases a content key, it would be a matter of policy whether that key would also be made available to all the household accounts that would be authorized to view that content, or whether each account would need to purchase the content separately. It might be a privacy concern, for instance, for household members to see which items have already been purchased by some other household member.

6.7 Revocation Issues

An authorization secret could be stolen, or Alice might no longer be authorized in some category (say, her membership in an organization has lapsed). If communication to the content provider is done with a sealed box, or with reasonably trusted DRM software, then the content provider could keep the authorization secrets in the client up to date. For instance, if "current member of ACM" is required for some types of content, the content provider could communicate with ACM periodically to get its list of members, and install the "ACM" authorization secret into the boxes (or software) of all the authorized users, and remove the secret from boxes (DRM software) of users who were, but are no longer, members.

Given that even with DRM, authorization secrets might be stolen by determined attackers, it is an advantage of scheme 6.1 that the secrets can be changed periodically.

In contrast, with multiple content provider public keys (6.3), revocation is very simple. All that is required is that the content provider keep track of all of Alice's authorizations. If, for instance, her membership in an organization lapses, that organization would inform the content provider, which would remove membership in that organization from Alice's profile, and no longer allow Alice to decrypt anything requiring that authorization. With anonymous cash-based authorization-specific content provider schemes, once Alice has obtained authorization-specific cash tokens it will not be possible to take them back (unless enforced through the DRM software/hardware).

7. DRM-Enforcement Sealed Box

This section considers the implications on the design in the common deployment scenario where the content provider provides a sealed box, and communication between the "user" and the content provider is actually done between the box and the content provider. We assume that the user can communicate with the box, to tell it which content the user would like to access.

We assume the box is reasonably difficult to tamper with, and an additional hindrance would be that tampering with it would be illegal. A plausible deployment of such a "box" might be a smart card or other sealed module that installs into the user's PC.

7.1 Hindering Copying of Authorization Keys

In many of the variants we have presented, a user collects content keys and authorization keys. So, an obvious implication is that one person can obtain a key to decrypt a piece of content, or an authorization key for "over 21", and widely distribute it.

However, each box will be known to the content provider. Either the content provider will know a public key for each box, or will have a shared secret key with each box. Communication is between the server and the box, and any information that must be kept from the user (such as an authorization key) can either be done through an encrypted channel (such as SSL) between the box and the server, or can be returned to the box encrypted with a key known only to that box. Content and authorization keys, as well as the private key for a particular box will be stored inside the box, and the box would be designed to make it be very difficult to extract keys from the box.

If a determined user does extract keys from a box, all is not lost. It still would be difficult to insert such keys into other boxes. In other words, assuming a reasonably competent job of engineering the boxes to be tamper-resistant, it would not only take a great deal of ingenuity and lack of fear of prosecution to extract the keys from one box, but it would take an equal amount of tampering to insert keys *into* a box, since an untampered-with box would only accept such keys during communication with the content provider.

If the identity key for a particular box were compromised, that might enable simulating an entire box in software (and therefore it would not take much effort to deploy clones), but the compromise of that one box would become known to the content provider quickly (as, for instance, the owner of that box would be charged for all content requested by any clone), and the content provider would revoke the key for that box. Although the content keys and authorization keys known to that compromised box might still be publicly known, it would still be difficult to install these keys into existing boxes.

7.2 Monitoring Privacy Preservation

The box is provided by the content provider, so even if in theory the protocol is intended to enable preserving the user's privacy, the content provider might be motivated to cheat.

Communication is between the box and the content provider, but as we said in the introduction, the user can monitor what is transmitted.

In the anonymous cash scheme, when decryptions are requested, this must be done over an encrypted channel, with a key between the box and the content provider. The user cannot tell what the box is saying. The box could easily be (intentionally) leaking its identity when it asks for a decryption of a particular piece of content.

In the blind decryption scheme, it is also possible for the box to cheat in a way that the user cannot detect through passive monitoring. When the box asks for decryption of a piece of content, the communication is not encrypted, so the user can indeed verify that what the box transmits is "["Alice", timestamp, B({K}P1)] signed by Alice". However, there are several ways for the box to cheat in a way that would be undetectable by Alice, even though Alice can see what it is transmitting.

First we will explain how the box can cheat, and then explain in section 7.3, with a protocol between Alice and the box, how we can allow Alice to enforce privacy protection without interfering with the (legitimate) DRM-enforcing protocol between the box and the content provider.

7.2.1 Cheating with a weak blinding function

There is no way for the user Alice to know whether the box is truly choosing a random number for the blinding function, or whether it is sneakily identifying the content Alice is purchasing, by using a blinding function predictable to the content provider.

An example method for the box to cheat and let the content provider know which item Alice is requesting, without Alice being able to detect that it is cheating, is as follows:

The random number it could use for the blinding could be a hash of the secret the box shares with the content provider, and the time. The granularity of time units must be small enough so that consecutive decryption requests would have different blinding quantities, but large enough so that it is not expensive for the content provider to do a brute force search on all possible blinding functions derived that way until it obtains a K with recognizable formatting. Recognizable format, for instance, might be where K in {K}P was padded with specific structure, e. g., according to the PKCS #1 standard [13].

7.2.2 Cheating by using the integrity check

If the integrity check between the box and the content provider is a shared secret key, the key will not be known to Alice, because the content provider does not want Alice to be able to ask for content keys.

In this case, the box can leak, say, the ID of the content that Alice is requesting, by, say, adding the ID of the content to the integrity check. For example, if the proper integrity check for the message

"Alice", timestamp, B({K}P1)

using the shared secret K is "X", and the ID of the content being requested by Alice is n, then instead of sending X as the integrity check, the box could send X+n. To retrieve "n", the content provider computes the correct integrity check for the message (X), and subtracts it from the integrity check as sent by the box.

There really is no way to fix this, so the integrity check must be a public key-based signature, where Alice must have access to the box's public key so she can verify that the box is providing valid signatures.

However, there is still a problem. In many public key signature schemes, e.g., ElGamal, there is a per-message random number x, where g^x mod p is part of the signature. The box could choose an x that leaks the ID of the content being requested. For example, the box could try lots of x's, until it finds one for which the lower bits of g^x mod p reveals the ID of the content. If it were exactly the ID of the content, Alice would be able to detect this; however, there are ways for the box to do this undetectably to Alice. For example, if the box shares a secret S with the content provider, and if both the box and the content provider remember the timestamp T of the last request to the content provider, the box could compute T encrypted with S, take the bottom n bits of {T}S (where "n" is the number of bits in a content ID), $\oplus$ the result with the content ID to obtain the quantity Q, and find an x such that the bottom n bits of g^x mod p is Q.

Thus there really is no way for Alice to passively monitor the channel and be reassured that the box is indeed preserving her privacy, in either the anonymous cash scheme or the blind signature scheme.

However, in the next section we will provide a mechanism for Alice to interact with the box and be assured that the box is not colluding with the content provider.

The only way this can work, as we will show in the next section, is with the blind decryption scheme, and with public key signatures. We will show how Alice can protect against both methods of the box cheating (weak blinding function, and leaky integrity check).

7.2.3 Cheating by using the timestamp, or timing

If the timestamp has sufficient granularity, it would be possible for the box to leak information in the low order bits of the timestamp. Also, it might be possible for the box to covertly signal information to the content provider based on when it sends requests. Both of these threats are easily countered, as explained in the next section.

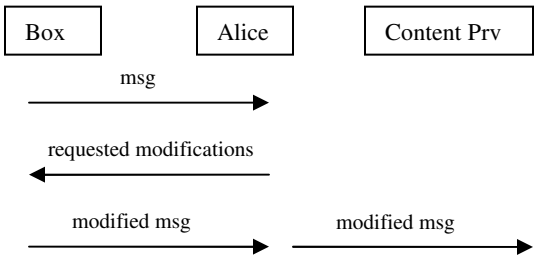
7.3 User-enforced Privacy Protection

With the anonymous cash approach, the user has no recourse other than trusting that the content provider's box is indeed protecting the user's privacy, because the conversation between the box and the content provider must be encrypted. The DRM system will not allow Alice to monitor the conversation (e.g., by letting the encryption be between Alice and the content provider rather than the box and the content provider) because she is not allowed to see the content key.

However, it is possible, with the blind decryption schemes, to have a protocol between Alice and the box in which Alice can be assured that her privacy is being protected.

The basics of the protocol are that the box emits a message it would like to send to the content provider. Because Alice sits between the box and the rest of the world, Alice can choose either to send this message on to the content provider, or to intercept the message. If she intercepts the message, she can send it back to the box, together with instructions for modifying the request. The box then modifies the message it would have sent, using Alice's

instructions. Alice will be able to verify that the box incorporated Alice's R into the message the box sends to the content provider.



7.3.1 Foiling weak blinding

As we discussed in section 7.2, with the blind decryption scheme, the box could choose blinding functions that are predictable by the content provider, and thereby allow the content provider to discover which content Alice was accessing. This is unavoidable if Alice is merely passively monitoring the channel.

However, there is a way (with the blind decryption scheme) for Alice to enforce that there be no such convert channel between the box and content provider. The simplest solution (which doesn't quite work, but we will fix it) is to have Alice insert an extra level of blinding in the message to the content provider, and reverse her level of blinding before passing the result back to the box.

In other words, what we'd like is that the box would transmit

- "Alice", timestamp, B({K}P)

to the content provider, but the message would be intercepted by Alice, who would add an extra level of blinding, say with function B2, and forward to the content provider:

- "Alice", timestamp, B2(B({K}P))

The returned message from the content provider will be

- B2(B(K))

Alice would then unblind with B2's inverse, and forward B(K) to the box.

But this would not work. The problem is, the message between the box and the content provider needs to be integrity protected; otherwise, anyone could ask for decryptions, and Alice's account would be debited. Even Alice is not trusted (by the content provider) to generate messages, since the content provider wants to keep decrypted content keys inside the closed system (only accessible by the boxes provided by the content provider).

Since the message from the box to the content provider is integrity protected, Alice cannot modify it without invalidating the message.

So, the solution is for Alice to interact with the box in order to influence what it uses for blinding.

The constraints are:

- The box cannot trust Alice to do the complete blinding (because Alice is not allowed to see the content key).
- The signed message to the content provider must be generated by the box (since only it is trusted by the content provider to sign messages).
- Alice needs to be able to verify that the box is not attempting to leak information, and that it really is applying the extra level of blinding she requests.

So the protocol is to allow Alice to ask the box to apply an extra level of blinding, with a key that she chooses and specifies to the box. She will be able to verify that her level of blinding has been applied, because she can compare the box's output before and after her blinding function has been applied. The box will be able to unblind with both functions; the blinding function it chose, and the one that Alice chose. The content provider will act as it did before, though if it were attempting to collude with the box, it will notice that the box is no longer colluding with a weak blinding function (since the content provider will not be able to unblind the message from the box to discover what key Alice is attempting to access). If there was no collusion attempt going on between the box and the content provider, the double blinding will be undetectable by the content provider.

7.3.1.1 Using RSA keys

The box originally chooses the blinding function R1, and emits the signed message: "Alice", timestamp, $R1^e * K^e \bmod n$. Alice intercepts this message, chooses a random R2, and returns the message to the box saying "please add an extra level of blinding using R2."

The box then transmits the signed message:

"Alice", timestamp, $R2^e * R1^e * K^e \bmod n$

Alice examines this by dividing by $R2^e \bmod n$, to ensure that the result is what the box originally transmitted ($R1^e * K^e \bmod n$).

If the answer is correct, she forwards the now doubly blinded message to the content provider

The content provider applies its private key and returns

$$R2 * R1 * K \bmod n$$

Alice lets the message go to the box, which knows both R1 and R2, and can therefore extract K.

This protocol will work, in the sense that the key will be properly extracted for the content that Alice requested, and also, that Alice is assured that the box has not leaked to the content provider the identity of the content she has requested.

If the content provider had been attempting to collude with the box by having it use a predictable blinding function, the content provider will notice that it is unable to unblind what it received.

7.3.1.2 Using Diffie-Hellman keys

If instead the content provider had a public Diffie-Hellman key, say g^x mod p, then the protocol to extract the encryption key for a piece of content from the metadata for that content, say g^y mod p, would be:

- The box would choose a blinding number z1, exponentiate by z1 mod p, and transmit the signed message:
  - "Alice", g^{y*z1} mod p
- Alice would intercept this, choose her own blinding number z2, and say to the box
  - Add blinding using z2
- The box would then transmit the signed message:
  - "Alice", $g^{y*z1*z2}$ mod p
- Alice raises $g^{y*z1*z2}$ mod p to her number's inverse exponent and verifies that the result is the original one transmitted by the box, i.e., g^{y*z1} mod p
- Alice lets the message go through to the content provider, and allows the return message to go through to the box.

7.3.2 Foiling Leaky Signatures

The other method for the box to cheat and collude with the content provider is by leaking information in the integrity check. If the integrity check is a secret key shared between the box and the content provider, there is nothing Alice can do.

However, if the integrity check is based on the box's public key, then Alice can ensure there is no cheating, as long as she has access to the box's public key (and she monitors that signatures that the box emits are correct).

With RSA keys, and with PKCS #1 v1 padding, there is no problem.

With signatures involving a per-message random number, such as ElGamal, it is possible (as we showed in section 7.2.2) for the box to leak information.

As with double blinding, Alice can enforce that the box is not choosing a bad random number x, by allowing Alice to contribute to the random number. As with double blinding, the box first presents to Alice the message it would like to send, including g^x mod p. Alice then chooses her own random number y, and tells the box to include "y" in its signature. Then she tests whether the box modifies g^x mod p to instead be g^{xy} mod p, and still sends a valid signature.

7.3.3 Foiling Other attacks

7.3.3.1 Timestamp

The box could, in theory, leak some information in the least significant bits of the timestamp, assuming the timestamp had sufficient granularity that it could do that while still having a timestamp that was plausible to Alice. If it was using a sequence number, then it could not embed information, since the sequence number would be constrained to be one bigger than the last request.

In some cases Alice might not be keeping sufficient state to be able to monitor the sequence numbers, and therefore it might be more convenient to use a timestamp.

When she is making the request to modify the message, she can also request a specific timestamp, close enough to the actual time so it would still be a valid timestamp, but without the box being able to control the low order bits.

7.3.3.2 Timing

To foil the box leaking information by when it sends requests, Alice can delay a message between the box and the content provider by some amount of time before passing it on.

7.3.3.3 Box-initiated encrypted communication

There are times when the content provider needs to transmit encrypted information to the box; e.g., authorization secrets. If this were done by establishing an encrypted channel between the box and the content provider, then the box can transmit any information it wants without Alice being able to monitor it. For example, it could inform the content provider which items Alice has recently purchased.

There is no reason for the box to be sending encrypted information to the content provider (other than the blinded content key, which we discussed in section 7.3.1.) But the content provider does need to send encrypted authorization secrets to the box.

Rather than doing this by establishing an encrypted channel, authorization secrets can be encrypted by the content provider with the box's public key, or with a shared secret key between the content provider and the box. As long as the information from the box to the content provider is encrypted, there is no way for the box to leak information to the content provider.

8. Conclusions

We have examined two families of privacy-preserving DRM schemes, one based on anonymous cash, and the other based on blind decryption.

The blind decryption scheme is less expensive, because purchase of decryptions, and decryption requests, can occur in the same message. In contrast, the anonymous cash scheme requires a (non-anonymous) communication to purchase tokens, and a separate anonymous communication for purchasing decryptions. Also, the anonymous cash scheme requires an anonymization network.

We provided a way (in either scheme) to provide differential costs of items using multiple denomination content provider public keys.

The anonymous cash scheme allows the content provider to do accounting of how many accesses there are for each item of content, which might be important if royalties to the copyright owners of individual items of content are based on number of accesses. The blind decryption scheme does not support this.

We examined several variants for supporting additional authorization. We concluded that authorization encryption keys worked equally well with anonymous cash or blind decryption, and leaked the least privacy information. The authorization claim secret scheme had the advantage that authorization keys could be changed inexpensively. The multiple content provider public key scheme has the privacy disadvantage that it knows the authorization policy of the content that Alice is decrypting. However, it does have the advantage that there are no authorization secrets to steal from authorized users, and revocation of a user's authorization in a category is trivial.

To make it practical to have many content provider public keys, e.g., based on potentially complex authorization categories, we provided a scheme, inspired by IBE, wherein the content provider's Diffie-Hellman key is derived from the authorization string. This is not an IBE scheme because Alice never finds out (or needs to find out), the particular content provider public key. All she needs is the Diffie-Hellman parameters (g and p), and the string, (say "citizen of US AND over 21").

The most likely deployment scenario for this type of application is where communication is not directly between the content provider and an open computer controlled by the user, but rather by a sealed box approved by the content provider and provided by the content provider to sit in the user's house.

We examined the implications of this design. In particular, we concluded there is no way in any of the schemes, if the user can only passively monitor all communication to and from the box, to see if the box is indeed performing the privacy protection protocol properly, rather than covertly leaking to the content provider what the user is accessing.

We concluded that only in the blind decryption scenario would it be possible to enhance the system with a protocol between the user (a computer controlled by the user) and the box, so that the box can continue to enforce the legitimate interests of the content provider, but the user can enforce that the box not covertly leak privacy-compromising information to the content provider. We discussed several ways in which the box could covertly pass information to the content provider that would be undetectable to Alice, if she were only passively monitoring the communication, and we presented methods for Alice to be assured no such covert channel is going on, by allowing Alice to influence the messages between the box and the content provider.

9. Acknowledgements

We would like to thank John Kelsey, Dave Molnar and Hilarie Orman for their helpful comments and advice.

10. REFERENCES

[1] Bender, W., Gruhl, D., Norimoto, N., "Techniques for data hiding", Proc. of SPIE, 1995.

[2] Boneh, D., Franklin, M., "Identity-Based Encryption from the Weil Pairing" *Advances in Cryptology - Proceedings of CRYPTO 2001.*

[3] Boneh, D., "Twenty years of attacks on the RSA cryptosystem", Notices of the AMS, 1999.

[4] Boneh, D., Shaw, J., "Collusion-secure fingerprinting for digital data", CRYPTO 1995.

[5] Chaum, D., "Blind signatures for Untraceable payments", Advances in Cryptology - proceedings of Crypto 82, 1983.

[6] Chaum, D., Fiat, A., and Naor, M. "Untraceable electronic cash". Proceedings on Advances in Cryptology (Santa Barbara, California, United States). 1990.

[7] Cox., I., Miller, M., Bloom, J., "Digital Watermarking", Morgan Kaufmann Publishers Inc., 2001.

[8] Cox, I., Kilian, J., Leighton, T., Shamoon, T., "Secure Spread Spectrum Watermarking for Multimedia", IEEE Transactions on Image Processing, 1997.

[9] Craver, S., Memon, N., Yeo, B. L., and Yeung, M. M. "Resolving rightful ownerships with invisible watermarking techniques: Limitations, attacks and implications". IEEE Journal. Selec. Areas Comm. 1998.

[10] Dingledine, R., Mathewson, N., Syverson, P. "Tor: The Second-Generation Onion Router". Usenix Security Symposium, 2004.

[11] M. J. Freedman and R. Morris. Tarzan: A peer-to-peer anonymizing network layer. In 9th ACM Conference on Computer and communication Security, 2002.

[12] Iannella, R., "Digital Rights Management (DRM) Architectures, D-Lib Magazine, June 2001.

[13] Jonsson, J., Kaliski, B., "Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1", RFC 3447, February 2003.

[14] Nair, S., Tanenbaum, A., Gheorghe, G., Crispo, B., "Enforcing DRM policies across applications",

Proceedings of the 8th ACM workshop on Digital rights management, 2008.

[15] Perlman, R., “The Ephemerizer: Making Data Disappear”, Journal of Information System Security, 2005.

[16] Saint-Jean, F., Johnson, A., Boneh, D., Feigenbaum, J., "Private Web Search", Proceedings of the 2007 ACM workshop on Privacy in electronic society", 2007.

[17] Shamir, A., “Identity-Based Cryptosystems and Signature Schemes”, *Advances in Cryptology: Proceedings of CRYPTO 84.*

[18] Shamir, A., “How to Share a Secret”, Communications of the ACM, v 22 n 11, p 612-613, Nov 1979.