Protecting Cryptographic Keys: The Trace and Revoke Approach

Dalit Naor* Moni Naor†

Abstract

The problem of protecting ownership of digital content has become a target for many cryptographic studies in the past decade. These studies provide myriad technology solutions to attack the problem, both from the aspect of rights management and for piracy prevention. In this article we survey a number of relevant techniques, focusing on methods for *protecting cryptographic keys* and in particular for sending encrypted information to a group of users. We present two such methods: the LKH method that is best tailored to scenarios where attentiveness is guaranteed and the Subset-difference scheme that is suited for the stateless case, where users are not necessarily always on-line. The latter is a trace and revoke scheme that allows the detection of keys which are used in a pirate box and, furthermore, can be used to prevent these illegally used keys from decrypting *future* content. We comment on the applications of these technological solutions to the "real" world.

1 Introduction

Digital content is very easy to generate, transfer and reproduce. While this makes it very attractive and brought its proliferation it has also become a major source for violation of ownership of information, whether it is copyright or privacy of individuals and businesses. With the increasing amount of digital content that is readily available, and as a result of constant reduction in prices of storage media the phenomenon of ownership violation is expected to steadily increase.

Ownership protection is fundamentally a social issue; it is the society role to define its usage rules, and to accompany the rules with appropriate means to enforce them, e.g. via legislation. However, technological developments may certainly impact the ground rules, by imposing technical as well as social barriers for the violators.

What can technology offer these days? State of the art methods that may be used to pose limitations on violation of copyright and intellectual property ownership may be categorized into a few families:

- Protecting Content - these methods are aimed at detecting and thus possibly discourage/prevent distribution of content *after* it has been decrypted. These include watermarking and fingerprinting techniques; see Cox et al.[6] for information on watermarking.
- Tamper Resistance, both in hardware and in software. Most violation of content ownership involve some type of hacking and analysis of the protection mechanism. It is believed that better shielding of the process, whether implemented in hardware or in software (which is typically more difficult to protect) will deter violators, as it makes it very hard and thus less attractive to reverse engineer its code and know its details (such as receivers identities and decoding strategy).
- Protecting Cryptographic Keys. Content protection typically utilizes cryptographic operations that base their security on the secrecy of its keys. Key protection techniques like Broadcast encryption and tracing schemes suggest how to send information only to intended users, and how to detect and prevent abuse of keys. These methods are the subject of this paper.

An overall solution to the content ownership problem may apply any combination of techniques above to achieve the goal. For example, the raw content will first be 'watermarked'; it will then be encrypted using some type of a broadcast encryption mechanism; the actual implementation will be shielded with a tamper resistant software or hardware.

2 Background

2.1 Broadcast Encryption

The goal of a key protection-mechanism is to create a *legitimate channel* for the distribution of content and yet disallow abuse of the legitimate channel. Thus, an illegitimate distribution should require the establishment of an *alternative* channel, namely it should not be possible to

*IBM Haifa Research Lab - Tel Aviv site, Haifa University Campus, Mt. Carmel, Haifa, 31905, Israel. Email: dalit@il.ibm.com.

†Incumbent of the Judith Kleeman Professorial Chair, Department of Computer Science and Applied Math, Weizmann Institute of Science, Rehovot 76100, Israel. Email: naor@wisdom.weizmann.ac.il. Research supported in part by a grant from the Israel Science Foundation.

piggyback on the legitimate channel[1]. Alternative channels should be combated using other (e.g. legal) means. To achieve this goal, key protection techniques suggest how to send information only to intended users, and how to detect and prevent abuse of keys.

The problem of a Center transmitting data to a large group of receivers so that only a predefined subset is able to decrypt the data is at the heart of a growing number of applications. Among them are pay-TV applications, multicast communication, secure distribution of copyright-protected material (e.g. music) and digital rights management. The area of Broadcast Encryption was first formally studied (and the term coined) by Fiat and Naor [8] and has received much attention since then. It deals with methods to efficiently broadcast information to a dynamically changing group of users who are allowed to receive the data. Different applications impose different rates for updating the group of legitimate users. It is often convenient to think of it as a Revocation Scheme, which addresses the case where some subset of the users are excluded from receiving the information, due to payments, subscription expiration or since they have abused their rights in the past. In such scenarios it is also desirable to have a Tracing Mechanism, which enables the efficient tracing of leakage, specifically, the source of keys used by illegal devices, such as pirate decoders or clones.

One special case is when the receivers are stateless. In such a scenario, a (legitimate) receiver is not capable of recording the past history of transmissions and change its state accordingly. Instead, its operation must be based on the current transmission and its initial configuration and thus there is a 'fresh' set of revoked users for each transmission. Stateless receivers are important for the case where the receiver is a device that is not constantly on-line, such as a media player (e.g. a CD or DVD player where the "transmission" is the current disc), a satellite receiver (GPS) and perhaps in multicast applications.

An important application that motivates the study of revocation and tracing mechanisms is Copyright Protection [12]. The distribution of copyright protected content for (possibly) disconnected operations involves encryption of the content on media. The medium (such as CD, DVD or a flash memory card) typically contains in its header the encryption of the key K which encrypts the content following the header. Compliant devices, or receivers, store appropriate decryption keys that can be used to decrypt the header and in turn decrypt the content. A *copyright protection mechanism* defines the algorithm which assigns keys to devices and encrypts the content.

[1] For instance in the case of cable TV the pirates should be forced to create their own cable network.

2.1.1 Desiderata

What are the desired properties of a broadcast encryption scheme? A good scheme should be characterized by

- Low Bandwidth - we aim at a small message expansion, namely that the length of the encrypted content should not be much longer than the original message.
- Small amount of storage - we would like the amount of required storage (typically keys) at the user to be small, and as a secondary objective the amount of storage at the server to be manageable as well.
- Attentiveness - do users need to be online 'all the time'? or otherwise is it stateless.
- Resilience - we want the method to be resilient to large coalitions of users who collude and share their resources and keys[2].

2.2 Tracing Traitors

Broadcast Encryption provides means to enforce conditional access by selectively encrypting content to legitimate users only. But what if some users get involved with piracy and leak their keys? This would allow the pirates to construct unauthorized decryption devices and distribute them illegally, possibly at low cost (Figure 1, a).

Tracing traitors schemes offer a way to cope with such piracy. First introduced by Chor et al. [4, 5], their goal is to distribute decryption keys to the users so as to allow the detection of at least one 'identity' of a key that is used in a pirate box or clone using keys of at most t users (Figure 1, b). Furthermore, it is required that no honest user will be implicated. Combined with a revocation method, the trace-and-revoke approach is to design a method that can trace the identity of the user whose key was leaked; in turn, this user's key is revoked from the system for future uses. Such a combination yields a powerful tool to create a network for legitimate distribution of content allowing a compliant user to perform legal operations, yet to render useless the non-compliant users.

While this paper is concerned with tracing leakers who give away their private keys there are methods that attempt to detect illegal users who redistribute the content *after* it is decoded. This requires the assumption that good watermarking techniques with the following properties are available: it is possible to insert one of several types of watermarks into the content so that the adversary cannot create a "clean" version with no watermarks (or a watermark it did not receive). Typically, content is divided into segments that are watermarked separately. This setting with protection against collusions was first investigated by

[2] The methods described in this paper are resilient to any coalition size and may revoke any number of users.

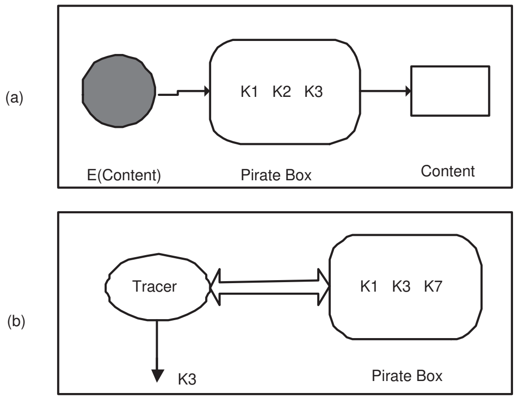

Figure 1: (a) A pirate box (b) The tracing process

Boneh and Shaw [1], and enhanced by introducing time dimension in [9, 17].

There are other approaches to the key leakage problem. The self enforcement approach, suggested by Dwork, Lotspiech and Naor [7], aims at *deterring* users from revealing their personal keys. The idea is to provide a user with personal keys that contain some sensitive information about the user which the user will be reluctant to disclose (for example, the person's credit card number). We should note that tracing traitors and self-enforcement are non-exclusive.

The legal approach, suggested by Pfitzmann [16], requires a method that not only traces the leaker but also yields a proof for the liability of the traitor (the user whose keys are used by an illegal decoder). Hence, leakage can be fought via legal channels by presenting this proof to a third party.

2.3 This paper

Disclaimer: In this paper we make no attempt to survey the entire field, due to the vast amount of relevant research and literature. Instead, we chose to give a taste of a couple, well established and rigorous methods which managed to influence the field and be quite useful in certain applications.

We focus on a particular approach, the 'trace and revoke' approach for protecting cryptographic key, and specifically fill in the details for two algorithms: the LKH (Logical Key Hierarchy) and the Subset-Difference. The LKH method maintains a communication key among a dynamic group of users and is most suitable for applications where the receivers (users) are always on-line (or else a special operation is required). In contrast, the Subset-Difference algorithm is relevant to stateless receivers, where the output of the decryption is based on the current message and the secret information only. Both algorithms are well suited for tracing of coalitions of any size.

In order to evaluate and compare methods we define a few parameters. Let $\mathcal{N}$ be the set of all users, $|\mathcal{N}| = N$, and $\mathcal{R} \subset \mathcal{N}$ be a group of $|\mathcal{R}| = r$ users whose decryption privileges should be revoked. The goal of a revocation algorithm is to allow a center to transmit a message M to all users such that any user $u \in \mathcal{N} \setminus \mathcal{R}$ can decrypt the message correctly, while even a coalition consisting of all members of $\mathcal{R}$ cannot decrypt it. The important parameters are therefore r and N.

A system consists of three parts: (1) A key assignment scheme, which is a method for assigning the receivers secret keys that will allow them to decrypt. (2) The broadcast algorithm - given a message M and the set $\mathcal{R}$ of users to revoke outputs a ciphertext message M' that is broadcast to all receivers. (3) A decryption algorithm - a (non-revoked) user that receives ciphertext M' should produce the original message M using its secret information.

3 The Subset Difference Algorithm

3.1 Framework

Our algorithm defines a collection of subsets of users $S_1, \ldots, S_w$, $S_j \subseteq \mathcal{N}$. Each subset S_j is assigned a long-lived key L_j; each member u of S_j should be able to deduce L_j from its secret information. Given a revoked set $\mathcal{R}$, the remaining users are partitioned into disjoint sets $S_{i_1}, \ldots, S_{i_m}$ that entirely *cover* them (every user in the remaining set is in at least one subset in the cover) and a session key K is encrypted m times with $L_{i_1}, \ldots, L_{i_m}$.

The algorithm uses two encryption schemes: (i) A method F_K to encrypt the message itself. The session key K should be chosen afresh for each message M. F_K should be a fast method and should not expand the plaintext. The simplest implementation is to Xor the message M with a stream cipher generated by K. (ii) A method E_L (and its corresponding D_L) to deliver the session key to the receivers, for which we will employ an encryption scheme. The keys L here are long-lived. The simplest implementation is to make $E_L : \{0,1\}^\ell \mapsto \{0,1\}^\ell$ a block cipher.

Key Assignment : Every receiver u is assigned private information I_u that allows it to *deduce* every key L_i corresponding to the set S_i such that $u \in S_i$. I_u could simply consist of all the keys S_i such that $u \in S_i$, but in the subset-difference algorithm the key assignment is more sophisticated and succinct.

The **Broadcast algorithm** at the Center:

1. Choose a session encryption key K.

2. Given a set $\mathcal{R}$ of revoked receivers, the center finds subsets $S_{i_1}, \ldots, S_{i_m}$ that cover the remaining re-

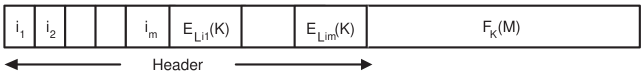

Figure 2: The header and the encrypted message

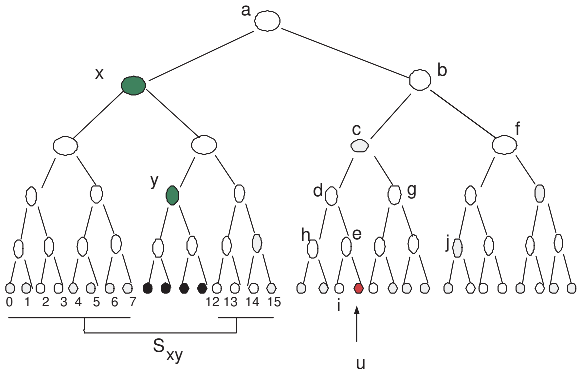

Figure 3: This tree corresponds to $N = 32$. Here, the set $S_{x,y}$ corresponds to the leaves marked as $0, 1, 2, 3, 4, 5, 6, 7, 12, 13, 14, 15$ but *not* the leaves $8, 9, 10, 11$. For a leaf u, the nodes a, b, c, d and e are on the path from the root to u. The nodes hanging off this path are x, f, g, h and i.

ceivers. Let $L_{i_1}, \ldots, L_{i_m}$ be the keys associated with the above subsets.

3. The center encrypts K with keys $L_{i_1}, \ldots, L_{i_m}$ and sends the ciphertext (See Figure 2)

$$\langle [i_1, \ldots, i_m, E_{L_{i_1}}(K), \ldots, E_{L_{i_m}}(K)], F_K(M) \rangle$$

The **Decryption step** at the receiver u, upon receiving a broadcast message $\langle [i_1, i_2, \ldots, i_m, C_1, C_2, \ldots, C_m], M' \rangle$:

1. Find i_j such that $u \in S_{i_j}$ (in case $u \in \mathcal{R}$ the result is **null**).
2. Extract the corresponding key L_{i_j} from I_u.
3. Compute $D_{L_{i_j}}(C_j)$ to obtain K.
4. Using K, extract M from $M' = F_K(M)$.

The subset difference algorithm achieves a very good tradeoff between the message length and the amount of keys that a user needs to store. Specifically, it uses at most $2r - 1$ (or $1.25r$ on average) encryptions to transmit a message that revokes r receivers. It requires every user to store $\frac{1}{2} \cdot \log^2 N + \frac{1}{2} \cdot \log N + 1$ keys. The algorithm has another important property: it is *flexible with respect to r*, namely the storage at the receiver is not a function of r. Other parameters like the efficiency of setting up the scheme and computing the partition (given $\mathcal{R}$) are of lesser importance as they affect the server which is assumed to be much more powerful than the receiver.

As an example, consider a system of $N = 2^{32}$ users (receivers). Every receiver requires to store less than 500 keys (further optimization shows that this number can be reduced by half). A message that revokes r receivers which are picked at random needs to attach a header consisting of $1.25r$ encryptions. If E_L is, say, a DES cipher, then this amounts to a header smaller than $12r$ bytes (or putting it differently, 12 bytes per revocation in the header).

3.1.1 Two Extreme Examples

As an exercise, consider two extreme constructions. In the *all-subsets* example, the collection consists of all possible subsets of users. In the *singletons* example, $\mathcal{S}$ contains only subsets of a single user, namely $S_i = \{u_i\}$. The *all-subsets* construction has an optimal bandwidth: to send a message that revokes the set $\mathcal{R}$ only one encryption is needed - use the key that corresponds to the set $\mathcal{N} \setminus \mathcal{R}$. However, as each user belongs to 2^{N-1} subsets, it needs to store these many keys, which is not feasible. Alternatively, the *singletons* construction has an optimal storage requirement: each user belongs to one set only, hence it needs to store only one key. However, to send a message that revokes a set of r of users, $N - r$ encryptions are needed, one for each user in $\mathcal{N} \setminus \mathcal{R}$.

3.2 The Algorithm Details

The subsets and the partitions are obtained by imagining the receivers as the leaves in a rooted full binary tree with N leaves (assume that N is a power of 2). Such a tree contains $2N - 1$ nodes (leaves plus internal nodes) and for any $1 \leq i \leq 2N - 1$ we assume that v_i is a node in the tree.

The Subsets The collection of subsets $S_1, \ldots, S_w$ defined by this algorithm corresponds to subsets of the form "a group of receivers G_1 minus another group G_2", where $G_2 \subset G_1$. The two groups G_1, G_2 correspond to leaves in two full binary subtrees. Therefore a valid subset S is represented by two nodes in the tree (v_i, v_j) such that v_i is an ancestor of v_j. We denote such subset as $S_{i,j}$. A leaf u is in $S_{i,j}$ iff it is in the subtree rooted at v_i but *not* in the subtree rooted at v_j, or in other words $u \in S_{i,j}$ iff v_i is an ancestor of u but v_j is not. For example, the subset $S_{x,y}$ depicted in Figure 3 contains the leaves marked $0, 1, 2, \ldots, 7, 12, 13, 14, 15$.

The Cover The cover is constructed by an iterative algorithm that gradually shrinks the tree by cutting out sub-

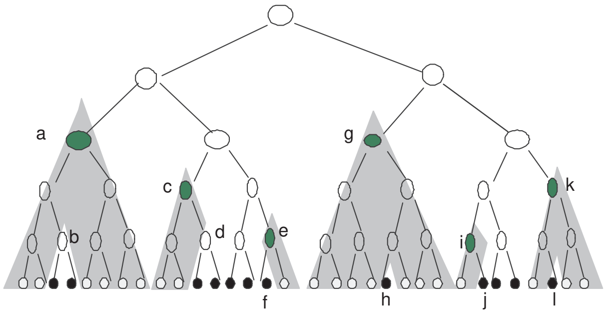

Figure 4: Example for a cover. Here $N = 32$ and $R = 12$. The revoked receivers are marked as black leaves. The cover size is 6 and consists of the sets: $S_{a,b}, S_{c,d}, S_{e,f}, S_{g,h}, S_{i,j}, S_{k,l}$.

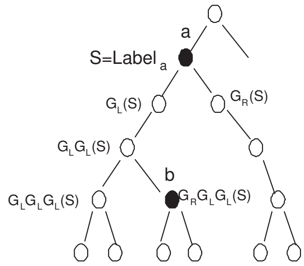

Figure 5: Generation of $\mathrm{Label}_{a,b}$.

trees of the original tree. The details of the algorithm can be found in [14]. Figure 4 shows an example for a cover with $N = 32$ receivers and 12 of them revoked. It can be shown that number of subsets in a cover is at most $2r - 1$ for any group of r revocations. Furthermore, if the set of revoked leaves is *random*, then the average number of subsets in a cover is $1.25r$.

Key Assignment If each receiver needs to store *explicitly* the keys of all the subsets it belongs to, then the storage requirements would expand tremendously: each receiver would need to store of the order of N keys. We therefore devise a key assignment method that requires a receiver to store many fewer keys.

While the total number of subsets to which a user u belongs is of the order of N, these can be grouped into $\log N$ clusters defined by the first subset i (from which another subset is subtracted). The way we proceed with the keys assignment is to choose for each internal node in the full binary tree a random and independent value Label_i. This value should *induce* the keys for all legitimate subsets of the form $S_{i,j}$. For that, we need a (cryptographic) pseudo-random sequence generator G [3] that *triples* the input, i.e. whose output length is *three times* the length of the input; let $G_L(S)$ denote the left third of the output of G on seed S, $G_R(S)$ the right third and $G_M(S)$ the middle third. Our idea is to employ a top-down labeling process similar to the one used by Goldreich et al. [10] in order to construct pseudo-random functions.

Consider now the complete subtree T_i (rooted at v_i). We use the following top-down labeling process: the root is assigned a label Label_i. Given that a parent was labeled S, its two children are labeled $G_L(S)$ and $G_R(S)$ respectively. Let $\mathrm{Label}_{i,j}$ be the label of node v_j derived in the subtree T_i from Label_i. Following such a labeling, the key $L_{i,j}$ assigned to set $S_{i,j}$ is G_M of $\mathrm{Label}_{i,j}$. Note that each label induces three parts: G_L - the label for the left child, G_R - the label for the right child, and G_M the key at the node. The process of generating labels and keys for a particular subtree is depicted in Figure 5.

For such a labeling process, given the label of a node it is possible to compute the labels (and keys) of all its descendants. On the other hand, without receiving the label of an ancestor of a node, its label is pseudo-random. Note that given Label_i, computing $L_{i,j}$ requires at most $\log N$ invocations of G.

What information I_u does a receiver u need to know so that it can derive the keys described above? For every v_i an ancestor of u, look at a complete subtree T_i rooted at v_i and consider the path from v_i to u; let $v_{i_1}, v_{i_2}, \ldots v_{i_k}$ be the nodes just "hanging off" the path, i.e. they are adjacent to the path but not ancestors of u (for example, in Figure 3 the node b is an ancestor of u and the nodes hanging off the path are f, g, h and i). The labels on these nodes provide all the information u needs to derive keys $L_{i,j}$ that u belong to as each j in T_i that is not an ancestor of u is a descendant of one of these nodes. Therefore if u receives the labels of $v_{i_1}, v_{i_2}, \ldots v_{i_k}$ as part of I_u, then invoking G at most $\log N$ times suffices to compute $L_{i,j}$ for any j that is not an ancestor of u. The above is done for every tree whose root is a descendent of u. The total number of labels assigned to u is $\frac{1}{2}\log^2 N + \frac{1}{2}\log N + 1$.

Example In Figure 3 the receiver that corresponds to the leaf u gets the following labels:

$\mathrm{Label}_{a,x}, \mathrm{Label}_{a,f}, \mathrm{Label}_{a,g}, \mathrm{Label}_{a,h}, \mathrm{Label}_{a,i}$
$\mathrm{Label}_{b,f}, \mathrm{Label}_{b,g}, \mathrm{Label}_{b,h}, \mathrm{Label}_{b,i}$
$\mathrm{Label}_{c,g}, \mathrm{Label}_{c,h}, \mathrm{Label}_{a,i}$
$\mathrm{Label}_{d,h}, \mathrm{Label}_{d,i}$
$\mathrm{Label}_{e,i}$

At decryption time, a receiver u first finds the subset $S_{i,j}$ such that $u \in S_{i,j}$, and computes the key corresponding to $L_{i,j}$ which is then used to decrypt K. In the above example, if on a particular message the receiver u needs the key corresponding to $L_{b,j}$, it uses the label $\mathrm{Label}_{b,f}$ and then applies G twice to derive $\mathrm{Label}_{\mathrm{b,j}}$ and another

[3] A function G that stretches its input string (called seed) to a longer output string is a pseudo-random sequence generator if it is not possible to *feasibly* (in polynomial-time) distinguish between the output of G on a randomly chosen seed from a truly random string of similar length.

application of G to derive $L_{b,j}$.

Further work: Recently, Halevy and Shamir [11] have shown a variant of Subset Difference called LSD (Layered Subset Difference), where each subset S_{ij} from the Subset Difference method is represented as a union of a few other subsets, forming a smaller collections altogether. The storage requirements are reduced to $O(\log^{1+\varepsilon} N)$ while the header length is $O(r/\varepsilon)$, providing a full spectrum between the Complete Subtree and Subset Difference methods. A reasonable choice is $\varepsilon = 2$.

4 The LKH Scheme for Rekeying

We now describe the LKH (Logical Key Hierarchy) scheme for maintaining a common group key when members of the group leave and join dynamically, based on [18] and [19]. It is especially relevant in a multicast setting. The scheme is *not* stateless, if a user misses some control message corresponding to a user revocation (e.g., if it temporarily gets disconnected from the network), it needs to ask for all the missed control messages (see [20].)

The goal of the scheme is to maintain a common encryption key for the active group members. We assume that there is a initial set $\mathcal{N}$ of N users and that from time to time an active user leaves and a new value for the group key should be chosen and distributed to the remaining users. We first deal with deletions, and later discuss joins which are generally easier to deal with. The operations are managed by a center who broadcasts all the maintenance messages and is also responsible for choosing the new key. When some user $u \in \mathcal{N}$ is revoked, a new group key K' should be chosen and all non-revoked users in $\mathcal{N}$ should receive it, while no coalition of the revoked users should be able to obtain it. At every point a non revoked user knows the group key K as well as a set of secret 'auxiliary' keys. As in the subset difference scheme, these keys correspond to subsets of which the user is a member. Note that unlike the scheme of Section 3.1 these keys change throughout the lifetime of the system. Also the keys are chosen independently of each other.

The group is initialized as follows. Users are associated with the leaves of a full binary tree tree of height $\log N$. The center associates a key K_i with every node v_i of the tree. Each user u is sent (via a secret channel) the keys associated with all the nodes along the path connecting the leaf u to the root. Notice that the root key K is known to all users and can be used to encrypt group communications.

At the center: In order to remove a user u from the group, the center performs the following operations. For all nodes v_i along the path from u to the root, a new key K'_i is generated. The new keys are distributed to the remaining users as follows: let v_i be a node on the path and v_j be its child on the path and v_ℓ its child that is *not* on the path. Then K'_i is encrypted using K'_j and K_ℓ (the latter did not change), i.e. a pair of encryptions $\langle E_{K'_j}(K'_i), E_{K_\ell}(K'_i)\rangle$. The exception is if v_i is the parent of the leaf u, in which case only a single encryption using the sibling of u is sent. All encryptions are sent to all the users. See Figure 6.

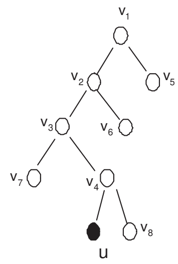

Figure 6: To remove the user u, the center chooses a new set of keys denoted by K'_1, K'_2, K'_3, K'_4 and broadcasts the following: $[E_{K_8}(K'_4), \langle E_{K'_4}(K'_3), E_{K_7}(K'_3)\rangle, \langle E_{K'_3}(K'_2), E_{K_6}(K'_2)\rangle, \langle E_{K'_2}(K'_1), E_{K_5}(K'_1)\rangle]$.

At the user: Each user can maintain the keys of all subsets to which it belongs: consider a (non-revoked) user u' and let v_i be the least common ancestor of leaves u and u' and let v_ℓ be the child of v_i that is an ancestor of u'. Note that of the set of keys that u' maintains the only ones that have been changed are those corresponding to ancestors of v_ℓ. User u' obtains K'_i using $E_{K_\ell}(K'_i)$ and the remaining ones are decrypted along the path to the root using the new keys. On the other hand, user u cannot retrieve any of the newly chosen keys, and all the keys he knew before have been changed, so its secret information becomes useless.

In this scheme each user has to store $\log N + 1$ keys and when removing a single user the center should send a message of $2 \log N - 1$ key encryptions. If more than a single user should be removed the above procedure is repeated for each of the removed users, i.e. a total of $2r \log N - r$ keys, or if they are batched, on the order of $r \log N/r$ keys. This bound was somewhat improved in [2, 3, 15, 13], but unless the storage at the user is extremely high they still require a transmission of $\Omega(\log N)$ encryptions per revoked user.

Joins: When the center wants to add a news user to the set, then first a vacant leaf, i.e. one with no unrevoked user assigned to it should be located. If none exists, then similarly to adding a value to a search tree one of the leaves becomes an internal node with two leaves as children; the previous occupier and the newly joined user are assigned to the leaves. Care of course should be taken to ensure that the tree is balanced. Two new keys are generated for the two new leafs and transmitted *secretly* to the users. As for the other keys, the treatment differs between the case where *backward* secrecy, is required or not, i.e. whether the new user should or should not be able to decrypt previously transmitted messages. If backward secrecy is not required, then the new user also gets all the keys associated with the path from its leaf to the root. If backward secrecy is required, then new keys should be chosen for the path. The value of those keys can then be transmitted as when a user is revoked, by sending $\log N$ encryptions. However, there is a more compact (in terms of communication) way of doing so, by deriving each new key from the old key in a pseudo-random manner, thus saving the broadcast of the $\log N$ encryptions. This works only when a user joins the system and not when revoked, since the old keys are not known to the adversary.

Comparison between the schemes The LKH scheme and the Subset Difference scheme of Section 3.1 are similar in that users are mapped to leaves of a full binary tree and independent labels are assigned to each node in the binary tree. However, there are a number of important differences: the subsets considered in the LKH scheme are only of the form of a full subtree (i.e. no subtraction of two subtrees). Also, as we have seen, these labels are used quite differently - in the LKH scheme some of these labels change at every revocation. In terms of efficiency, if the setting is relatively batch like, i.e. r revocation appear at once, then the LKH scheme requires sending $O(r \log N)$ encryptions, vs. only $O(r)$ in the Subset Difference Scheme.

5 Tracing Traitors

We now discuss how the subset difference revocation mechanism can work in tandem with a tracing mechanism to yield a *trace and revoke* scheme. The goal of a tracing algorithm is to find the identities of those that contributed their keys to an illicit decryption box and revoke them; short of identifying them, we should render the box useless by finding a "pattern" that does not allow decryption using the box, but still allows broadcasting to the legitimate users. Note that this is a slight relaxation of the requirement of explicitly identifying the traitor's identity, however as a mechanism that works in conjunction with the revocation scheme it is a powerful tool to combat piracy.

Suppose that we have found an illegal decryption-box (decoder, or clone) which contains the keys associated with at most t users $u_1, \ldots, u_t$ known as the "traitors". We are interested in "black-box" tracing, i.e. one that does not take the decoder apart but by providing it with an encrypted message and observing its output (the decrypted message) tries to figure out who leaked the keys. We assume that the box has a "reset button", i.e. that its internal state may be retrieved to some initial configuration[4].

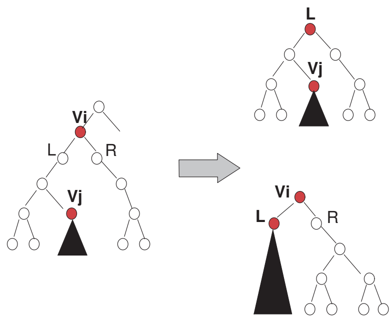

Figure 7: Bifurcating a set $S_{i,j}$, depicted on the left. The black triangle indicates the excluded subtree. L and R are the left and the right children of v_i. The resulting sets $S_{L,j}$ and $S_{i,L}$ are depicted to the right.

A tracing algorithm is evaluated based on (i) the level of performance downgrade it imposes on the revocation scheme (ii) number of queries needed to perform tracing when the device contains the keys associated with t users.

The idea of the tracing algorithm is to take a given cover (composed of few non-overlapping subsets, as described above) and locate a subset that contains a traitor (this process is called *subset tracing*). Once the subset has been traced, it is split into two roughly equal subsets to obtain a new cover. The algorithm is applied recursively until the traced subsets are of size 1 (which implies that their member is a traitor) or the cover renders the box useless (which is also a good outcome).

It turns out that in the Subset Difference collection it is possible to partition any subset $S_{i,j}$ into two subsets in the collection where the ration of their sizes is at most 2. See Figure 7.

The Subset Tracing Procedure: We are given a pirate box and a partition $\mathcal{S} = S_{i_1,j_1}, S_{i_2,j_2}, \ldots S_{i_m,j_m}$ such that the box decodes correctly on the partition. We know that if instead of encrypting with each L_{i_j} the session key K a random (unrelated) key is encrypted, then the box will not decode correctly (from the fact that F_K is a good encryption algorithm). The idea of the procedure is to use a collection of $m+1$ *hybrid* encryptions, where in the ℓth hybrid the first ℓ encryptions are for random keys and the last $m-\ell$ encrypt K. We know that for $\ell = 0$ the box works fine and for $\ell = m$ the box does not work. Hence there must be a point in the middle with a sharp difference between the $\ell - 1$ and ℓ hybrid. It must be the case that set S_{i_ℓ,j_ℓ} contains a traitor.

[4]This excludes a "locking" strategy on the part of the decoder which says that in case it detects that it is under test, it should refuse to decode further. Clearly software-based systems can be simulated and therefore have the reset property.

Analysis: Since a partition can occur only in a subset that has a traitor and contains more than one element, it follows that the number of iterations can be at most $t \log N$. Furthermore, since at each iteration the number of subsets in the partition increases by one, tracing t traitors may result with up to $t \log N$ subsets and hence in messages of length $t \log N$. It is possible to improve the latter parameter so that the maximum length of message needed is only $5 \cdot t$. See details in [14].

6 Discussion

Technology is neither a panacea nor irrelevant: The thesis presented in this paper is that technology has a lot to offer in the area of protection of content, yet it is relevant only when embedded within an overall solution that takes into account social and legal issues as well. These include policy related questions like what usage rules should be imposed, who and when to revoke a user, where lies the liability and more. It is yet to be seen what will drive the use of such technologies in the digital content world. In some cases (such as entertainment) economy is the major driving force; in other cases like the health industry, it will be the legislation process (e.g. HIPPA) that will enforce content owners and distributors to use better technologies for protection of primarily privacy.

This paper has mainly dealt with protection of copyright ownership and usage rules, but what about protecting privacy? One of the features of the trace and revoke approach is that it preserves the privacy of the users. There is no need for any end user identification in the process of either initializing, using, tracing or revoking. The 'punishment' for abusing the system is to make your key useless, i.e. it is dealt within the system.

References

[1] D. Boneh, and J. Shaw, *Collusion Secure Fingerprinting for Digital Data*, IEEE Transactions on Information Theory, Vol 44, No. 5, 1998, pp. 1897–1905.

[2] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor and B. Pinkas, *Multicast Security: A Taxonomy and Some Efficient Constructions,* Proc. of Proceedings IEEE INFOCOM '99, Vol. 2, 1999, pp. 708–716,

[3] R. Canetti, T. Malkin, K. Nissim, *Efficient Communication-Storage Tradeoffs for Multicast Encryption,* Advances in Cryptology - EUROCRYPT'99, Lecture Notes in Computer Science 1592, Springer, 1999, pp. 459–474.

[4] B. Chor, A. Fiat and M. Naor, *Tracing traitors*, Advances in Cryptology - CRYPTO '94, Lecture Notes in Computer Science, Vol. 839, Springer, pp. 257–270, 1994.

[5] B. Chor, A. Fiat, M. Naor and B. Pinkas, *Tracing traitors*, IEEE Transactions on Information Theory, Vol. 46, No. 3, May 2000, pp. 893–910.

[6] I. Cox, J. Bloom and M. Miller, **Digital Watermarking: Principles & Practice**, Morgan Kaufmann, 2001.

[7] C. Dwork, J. Lotspiech and M. Naor, Digital Signets: Self-Enforcing Protection of Digital Information, 28th Symposium on the Theory of Computation (1996), pp. 489– 498.

[8] A. Fiat and M. Naor, *Broadcast Encryption*, Advances in Cryptology - CRYPTO '93, Lecture Notes in Computer Science 773, Springer, 1994, pp. 480—491.

[9] A. Fiat and T. Tassa, *Dynamic Traitor Tracing,* Advances in Cryptology - CRYPTO '99, Lecture Notes in Computer Science, Vol. 1666, 1999, pp. 354–371.

[10] O. Goldreich, S. Goldwasser and S. Micali, *How to Construct Random Functions*, JACM 33(4): 792–807 (1986)

[11] D. Halevy and A. Shamir, *The LSD Broadcast Encryption Scheme,* Advances in Cryptology - CRYPTO '2002, Lecture Notes in Computer Science, Springer, 2002.

[12] J. Lotspiech, S. Nusser and F. Pestoni *Broadcast Encryption's Bright Future,* Computer, August 2002, Vol. 35, No. 8, pp. 57–63.

[13] D. McGrew, A. T. Sherman, Key Establishment in Large Dynamic Groups Using One-Way Function Trees, 1998. Available: `www.csee.umbc.edu/~sherman/`.

[14] D. Naor, M. Naor and J. Lotspiech, *Revocation and Tracing Schemes for Stateless Receivers*, Advances in Cryptology - Crypto '2001, Lecture Notes in Computer Science, Vol. 2139, Springer, 2001, pp. 41–62. Full version: ECCC Report 43, 2002. Available: `http://www.eccc.uni-trier.de/eccc/`

[15] A. Perrig, D. Song and J. D. Tygar, *ELK, a New Protocol for Efficient LArge-Group Key Distribution*, 2001 IEEE Symposium on Research in Security and Privacy, pp. 247–262.

[16] B. Pfitzmann, Trials of Traced Traitors, Information Hiding Workshop, Lecture Notes in Computer Science, Vol. 1174, Springer, 1996, pp. 49–64.

[17] R. Safavi-Naini and Y. Wang, Sequential Traitor Tracing Advances in Cryptology - CRYPTO 2000, Lecture Notes in Computer Science, vol 1880, pp. 316–332, 2000.

[18] D.M. Wallner, E.J. Harder and R.C. Agee, *Key Management for Multicast: Issues and Architectures*, Internet Request for Comments 2627, June, 1999. Available: ftp.ietf.org/rfc/rfc2627.txt

[19] C. K. Wong, M. Gouda and S. Lam, Secure Group Communications Using Key Graphs, Proc. ACM SIGCOMM'98, pp. 68–79, 1998.

[20] C. K. Wong and S. Lam, *Keystone: A Group Key Management Service*, International Conference on Telecommunications, Acapulco, Mexico, May 2000.