A DRM Based on Renewable Broadcast Encryption

Mahalingam Ramkumar[a] and Nasir Memon[b]

[a]Department of Computer Science and Engineering
Mississippi State University, Mississippi State, MS, USA.
[b]Department of Computer and Information Science
Polytechnic University, Brooklyn, NY, USA.

ABSTRACT

We propose an architecture for digital rights management based on a renewable, random key pre-distribution (KPD) scheme, HARPS (hashed random preloaded subsets). The proposed architecture caters for broadcast encryption by a trusted authority (TA) and by "parent" devices (devices used by vendors who manufacture compliant devices) for periodic revocation of devices. The KPD also facilitates broadcast encryption by peer devices, which permits peers to distribute content, and efficiently control access to the content encryption secret using *subscription* secrets. The underlying KPD also caters for broadcast authentication and mutual authentication of any two devices, irrespective of the vendors manufacturing the device, and thus provides a comprehensive solution for securing interactions between devices taking part in a DRM system.

Keywords: Broadcast Encryption, DRM, Key Pre-distribution

1. INTRODUCTION

In the rapidly evolving world of highly interconnected devices, the problem of ensuring fairness in transactions between creators and consumers of digital content becomes increasingly complex. The end-users wish to use all available means that technology provides, to improve their quality of content consumption. The content creators, however, unsure of how novel methodologies for consuming content could affect their profits (possibly because some such methods may aid mass piracy of content), are reluctant to follow suit (and are ultimately "forced" to cater for the demands of the consumer). Starting with the introduction of phonographic records in the market, and as recently as the advent of peer-to-peer file sharing systems, content distributors have used all possible means, and fought many battles in court, to inhibit use of technology for distribution of content.

In the very near future, however, sheer economics would dictate that content distribution occurs predominantly over the Internet, either through direct downloads from servers or through file-sharing among co-operative peer-to-peer network hosts. Every user will have physical possession, at all times, of a portable "trust" module, not unlike smart cards, that may be used for end-user authentication. The portable authentication module may be plugged into general purpose communication devices (perhaps using an ubiquitous interface like USB) like PDAs, laptop computers, mobile phones or even desk-top computers for authentication of exchanges.

Purchasing content, involving a financial commitment by the end-user in return for flexible and well-defined rights for content consumption will also occur over the Internet. The buyer may be able to specify the device onto which the content has to be "pushed," and thus, in a few hours or even minutes, may have access to the content at the desired location. The rules for content consumption may cater for number of permitted views by the consumer, limit the devices (either by selective inclusion or by selective exclusion) that the consumer may employ to render the content, and facilitate restrictions on the duration after which the rights terminate.

In this paper we provide an architecture for digital rights managements (DRM) for the scenario described above. Central to the ability of a DRM to achieve its goals is the ability to develop "trust relationships" between various *components* of the system. From a very broad perspective, a DRM for digital media distribution, consists of three primary entities

Further author information: (Send all correspondence to M. Ramkumar)
E-mail: ramkumar@cse.msstate.edu, Telephone: 1 662 325 8435
N. Memon: E-mail: memon@poly.edu, Telephone: 1 713 260 3970.

1. the content creator,
2. the content, and
3. the consumer.

If the content creator could simply trust the consumer to abide by the conditions under which he/she was provided access to the content, the DRM has achieved it goals.

From a less broader perspective more independent entities of the DRM come to light. For instance the content perhaps resides in a camera initially, which is then transferred to other devices where it may undergo further processing. Processed content may be copied into several storage media for distribution, or on to a media server connected to the Internet. The content creator may transfer some rights to one or more distributors. At the site of the consumer, the consumer employs a set-top box (STB) for rendering the content. Obviously, we could zoom in further and identify even more components of the system, all of which need to work together to ensure proper functioning of the DRM.

From a DRM perspective, two entities trust each other if there exists some means of convincing each other that they "play by the rules," or are "compliant" to pre-imposed rules,[1,2] From a cryptographic perspective, two nodes can trust each other if they can establish an *authenticated shared secret*. This is usually facilitated by a key distribution scheme (KDS), which provides each node with one or more KDS secrets. The KDS secrets are then used to establish or discover *shared* secrets. The KDS secrets provided to a node could however, be used as a *hook* for compliance. In other words, only nodes (or devices) that have been *checked* for compliance would be provided with the necessary secrets. Thereafter, the ability of any two devices to establish shared secrets, indirectly provides a means for *verification of compliance*.

Note that the established trust between two devices rests on the assumption that KDS secrets cannot be exposed from, or transferred from one device to another. By exposing secrets buried in a device, an attacker may be able to transfer such secrets to a non-compliant device, which will however end up being trusted by other devices. There is thus a need for some mechanism for read-proofing and / or tamper-resistance[3] of the devices with secrets. Nevertheless, history has shown that any form of tamper-resistance can perhaps be compromised, given an attacker with unlimited motivation, time, and resources. So long-lived security infrastructures should provide for some mechanism for periodic renewal of the stored KDS secrets.

1.1. Building Trust

There are two fundamental approaches, or key distribution schemes, for establishing trust relationships. A first approach involves a *centralized* client-server model (eg., Kerberos or any variant of the Needham-Schroeder symmetric-key[4] protocol). In such an approach, each entity involved in the DRM first establishes a trust relationship with the trusted server (or shares a secret with the central server). This trust can then be leveraged to develop trust relationships between other entities by *involving the trusted server in the mediation process*. In most cases this approach is rendered impractical due to the requirement of each component to have access to the trusted server at all times, and more importantly, for reasons of privacy. A second, distributed approach is that of a mechanism to establish trust between components of the system in an *ad hoc* manner. While a common approach for ad-hoc establishment of trust is the use of a public key infrastructure (PKI), it may be impractical for low-cost consumer electronic devices to include hardware capable of performing asymmetric cryptography.

1.2. Key Pre-distribution

Another option is key pre-distribution (KPD).[5] A KPD consists of a trusted authority (TA), and N nodes with unique IDs. The TA chooses a set of secrets $\mathcal{R}$. A certain number (say k) of secrets (usually a function of the TA's secrets $\mathcal{R}$ and the unique ID of the node) are *pre-loaded* in each node. The k preloaded secrets in each node are collectively referred to as the node's *key-ring*. Thereafter, any two nodes A and B can discover a shared secret K_{AB} *independently* (without further involvement of the TA). Also, no *other* node can calculate K_{AB}.

The primary distinction between conventional key distribution schemes (like Kerberos, PKI) and KPD, is that in the former, compromise of keys from devices (say A and B) does not affect the security of interaction between other devices* (say interaction between C and D). However, for the latter, compromise of keys in A and B may directly affect the security

*They will still be affected indirectly as they might end up trusting compromised devices A and B.

of interactions between C and D. This is due to the fact that the preloaded keys in different devices are *not independent* - they are all derived from the same set of TA's secrets $\mathcal{R}$.

There is thus a concept of n-*secure* KPDs. A n-secure KPD can resist coalitions of up to n nodes. Or no coalition of up to n nodes, pooling their keys together, can discover the shared secret between any two nodes (neither of which is a part of the coalition). This trade-off in security (the need to limit sizes of attacker coalitions) is the price paid for the ability of KPDs to establish ad hoc security associations *without* asymmetric cryptography. However, the accompanying guarantee of tamper resistance (which is mandatory in any case for DRM applications) serves as a deterrent for establishing large attacker coalitions. Note that mere physical possession of a device does not warrant inclusion of a device in an "attacker coalition." The attacker actually needs to expose secrets by tampering with "tamper-resistant" and "read-proof" devices.

1.3. Proposed DRM Architecture

The proposed architecture for digital rights management system is based on a recently proposed random key pre-distribution scheme, HARPS.[6] The solution is an extension of the DRM using HARPS proposed earlier by us,[7] in which the ability of HARPS to perform broadcast encryption was *not* considered. This ability[8] significantly improves the usefulness of HARPS as an enabler for DRM.

Recently proposed solutions for digital rights management[2] favor the use of broadcast encryption[9] -[12] over PKI. The proposed KPD scheme caters for broadcast encryption, besides many other useful security primitives[†] like

1. discovery of shared secrets which could enable authenticated and secure exchanges between devices (even if they are manufactured by different vendors),
2. peer-to-peer broadcast authentication,
3. peer-to-peer broadcast encryption, and
4. "seamless" and safe renewal of secrets for long-lived security of the deployment.

Section II is a brief discussion of how broadcast encryption is used in current DRM systems, and the limitations of extending such an approach for a full-blown DRM system. In Section III we briefly review HARPS and discuss many of its desirable properties. In Section IV we outline the architecture of the proposed DRM system.

2. DRM BASED ON BROADCAST ENCRYPTION

Broadcast encryption provides a mechanism for a set of g devices (from a "universe" of N devices) to arrive at a shared secret that r *revoked* nodes (either individually or together) cannot discover, where $g + r = N$. Typically, the approach involves encryption of the broadcast secret using *multiple* secrets. The encrypting secrets are chosen is such a way that every one of the g nodes would be able to decrypt at least one instance of the encrypted broadcast secret, while the revoked nodes cannot decrypt *any* of them. This results in a secret that is known only to the g non-revoked nodes.

Let $\mathbb{R}$ represent the universe of all secrets (completely known only to the TA who deploys the system in the first place). Each device is preloaded with a subset of those secrets. Let $\mathbb{S}_i$ denote the secrets preloaded in device i. Let the union of all secrets in the r revoked devices be $\mathbb{A}$. The source of the broadcast (say the TA) chooses a random broadcast secret and encrypts it with a chosen subset of secrets in $\mathbb{B} = \mathbb{R} \setminus \mathbb{A}$.

Apart from obvious applications in multicast communications scenarios, a well known practical application of broadcast encryption is in the CPRM (content protection for recordable media) scheme for DVD content. While CPRM uses a relatively naive broadcast encryption scheme, many newer schemes have been developed since the introduction of CPRM, most of them employing more sophisticated tree-based schemes.

An obvious measure for the efficiency of broadcast encryption is the bandwidth needed for transmitting the encrypted secret - which directly corresponds to the *number* of independent encryptions used for broadcasting the secret. The most efficient of broadcast encryption scheme thus far[12] needs an average of $1.33r$ encryptions (or 1.33 encryptions per revoked device).

[†]Most of them, not possible with other broadcast encryption solutions.

At first, *all* N devices share a secret R_0 - there are no revoked devices. Periodically, the TA broadcasts revocation messages which revokes a set of devices. In general the i^{th}revocation, which revokes r_i devices, is a broadcast message consisting of multiple encrypted versions of the revocation secret R_i. At the end of the i^{th}revocation, the $N - \sum_{j=0}^{i} r_j$ non-revoked devices share a secret $R_I = R_0 \oplus R_1 \oplus \cdots \oplus R_i$ which none of the revoked nodes have access to.

In the example of encrypting DVD content, the content that is produced after the i^{th}revocation, is encrypted (directly or indirectly[‡]) with the revocation secret R_I. Each device (DVD player) has a set of secrets which enables the device to decrypt revocation messages. Note that the revocation messages themselves could be distributed along with the content in the DVD!

While broadcast encryption provides an efficient way to revoke devices, the tricky part is - how does one find out what devices need to be revoked? Let us imagine a (fictitious) DRM based on broadcast encryption (say for DVDs). The universe of N compliant devices in this case consists of DVD players, and devices used for encrypting content[§]. At a given point in time, let us say some, of the universe of N devices have been revoked, and R_I is the current broadcast secret shared by the non-revoked devices. A content introduced at this time for circulation would be encrypted with some key K_C, and the content encryption key encrypted with the current revocation key $K = E_{R_I}(K_C)$, would also be distributed along with the content. The main purpose of the DRM is to ensure that non-compliant devices and revoked devices do not have the ability to decrypt the content. Only non-revoked compliant devices, that have access to R_I, can decrypt K to obtain K_C, and thus decrypt the content.

Now the job of the DVD player is to decrypt to compressed content, decompress it, perform a conversion from digital to analog, and send the analog video signal to a monitor. An attacker, with access to the internals of the DVD player could

- intercept the decrypted (and compressed) signal before decompression, or
- intercept the content after decompression, before the video is converted to an analog signal.
- open up a DVD player and determine the current revocation secret.
- expose some or all preloaded secrets (the secrets used for decrypting the revocation secrets).

Either of the first two alternatives provides the attacker with access to "clear" content, which he could redistribute. With the third attack, he would be able to decrypt any content that has been distributed until that point in time - he does not need the DVD player anymore. The fourth attack permits decryption of future revocation secrets too. Unfortunately, if any of these attacks is successful, there is no way for the TA to determine that an attacker has "broken" the system.

Another motivation of an attacker may be to manufacture compliant DVD players illegally (say without paying a required licensing fee). For this purpose, the attacker may need to tamper with and expose secrets in a few legitimate devices. If the TA happens to discover an illegitimate DVD player, he may be able to use "traitor tracing" techniques[2] to probe the device to find out, with some degree of certainty, which of the legitimate DVD players were compromised to synthesize the illegitimate player. Those players could then be revoked - which would also simultaneously make it impossible for the illegitimate DVD players to play DVDs created in the future.

Another scenario where it would be possible to identify candidates for revocation is when the attacker uses a very different attack - which is to tap the analog video output. This is less attractive (though very simple) for the attacker because the attacker would need to redigitize and recompress the content for redistribution - resulting in some loss of content fidelity. However, if the DVD player has the capability to insert invisible and indelible "fingerprints" using steganographic techniques,[1] and if the repackaged content is discovered, the TA might be able to identify from the extracted fingerprint, the compliant DVD player that was used to decrypt the content (the fingerprint could contain information regarding the unique ID of the DVD player). Under this circumstance, the offending player would be revoked (and possibly some legal recourse sought against the owner of the offending DVD player).

The main problem with a DRM solely based on broadcast encryption is that there is usually no means of determining which devices have been tampered with and their secrets exposed (unless a pirate attempts to manufacture "pseudo compliant" devices or if some "fingerprint" has been detected in some illegally distributed content). Further, most broadcast encryption schemes in literature only cater for broadcasts by the TA. DRM systems of the future should ideally enable any

[‡]For instance, using the broadcast secret to encrypt the "title-key" which is used to encrypt the content

[§]They need the key too!

one to be able to publish and distribute content using the existing deployment of "compliant" devices - which means even peer devices should be able to broadcast secrets.

Both these shortcomings are effectively addressed by the proposed DRM scheme, employing HARPS. By forcing devices to renew their secrets periodically, and by ensuring that nodes that have been "tampered with" cannot participate in the renewal process, such devices get revoked even if tampering attempts are not *otherwise* detected. The second shortcoming is also eliminated as HARPS permits *broadcast encryption by peers*.

3. HARPS

HARPS is a simple random KPD where each node is preloaded with a hashed subset of keys belonging to its parent. The chosen keys are hashed a (random) variable number of times. The index of the chosen keys and their hash depths are determined by a public one-way function.

A hierarchical deployment of HARPS has a root node R with P secrets $\mathcal{R} = \{K_1 \cdots K_P\}$ (or $\mid \mathcal{R} \mid = P$), at the root of the tree (see Figure 1). The root node has many (say N_0) children, with IDs α_i, $1 \leq i \leq N_0$, at level 1. A node α_i has a set of k_1 secrets $\mathbb{A}_i$. The set of k_1 secrets $\mathbb{A}_i$ is a subset of the P secrets $\mathcal{R}$, which are further repeatedly hashed (using a cryptographic public hash function $h()$) a variable number of times. The choice of the subset of keys, and the number of times each chosen key is hashed, is determined by another public function $f_1()$, and the node ID. Or,

$$\{(I_1, d_1), (I_2, d_2), \ldots, (I_k, d_k)\} = f_1(\alpha_i).$$

The first coordinate $\{I_1, I_2, \ldots, I_k\}$ indicates the indexes of the preloaded keys (between 1 and P) in node α_i, and the second coordinates $\{d_1, d_2, \ldots, d_k\}$, their corresponding "hash-depths" - or the number of times each chosen key is hashed. As a concrete example, $I_1 = 23, d_1 = 31$ implies that the first preloaded key in node α_i is ${}^{31}K_{23}$ - or the key obtained by hashing the key indexed 23 (or K_{23} of the TA) repeatedly, 31 times. The hash depths of the keys in level 1 nodes are uniformly distributed between 1 and L_1.

The level 2 children of a node α_i are $\alpha_i\beta_j$, $1 \leq j \leq N_i$, and the level 3 children of the node $\alpha_i\beta_j$ are $\alpha_i\beta_j\gamma_l$, $1 \leq j \leq N_{i,j}$. A public function $f_2(\beta_j)$ determines the indexes and hash depths of the keys preloaded in $\alpha_i\beta_j$ (w.r.t the parent device α_i) and a similar function $f_3(\gamma_l)$ determines the indexes and hash depths of the keys preloaded in $\alpha_i\beta_j\gamma_l$ (w.r.t the parent device $\alpha_i\beta_j$). The indexes of the preloaded keys in level 2 range between 1 and k_1, and the hash depths between $L_1 + 1$ and L_2 ($L_2 > L_1$). Similarly the indexes of the preloaded keys in level 3 range between 1 and k_2 and the hash depths between $L_2 + 1$ and L_3 ($L_3 > L_2$). Note that as long as the hash function used is pre-image resistant, compromise of secrets in lower levels (say level 3) of the hierarchy does not affect the higher levels (levels 2, 1 and 0).

Shared Secret Discovery: Two nodes just need to exchange their IDs in order to determine their shared secret. From their IDs, the nodes can discover shared indexes and their corresponding hash depths in the nodes by application of the public function. If for instance, if two nodes A and B share m keys with indexes $i_1 \cdots i_m$, and the hash depth of the m keys in node A are $a_1 \cdots a_m$, and the corresponding hash depths in node B are $b_1 \cdots b_m$, the shared secret between the nodes A and B is obtained as follows by each node independently:

- For each of the m shared indexes, node A hashes jth key ($1 \leq j \leq m$) $b_j - a_j$ times if $b_j > a_j$.
- For each of the m shared indexes, node B hashes jth key ($1 \leq j \leq m$) $a_j - b_j$ times if $a_j > b_j$.
- The concatenation of the resulting m keys, $K_{i_1} \cdots K_{i_m}$ at hash depths $\max(a_j, b_j)$, are hashed together to obtain K_{AB}, the shared secret between A and B.

Broadcast Authentication: The source appends its messages with many key based message authentication codes (MAC) - one corresponding to each of the k keys it has in its key ring. The ability of the source to *choose* hash depths for each MAC key provides HARPS with a unique feature not possible in conventional broadcast authentication schemes - the ability to impose "preferred verifiers".[13] In other words, for scenarios where a message is of interest to only one or a select set of verifiers, the source could *choose* the hash depths such that it would be substantially more difficult for an attacker to forge authentication, with the intent of fooling the chosen set of verifiers, compared to the case where verifiers are not targeted.

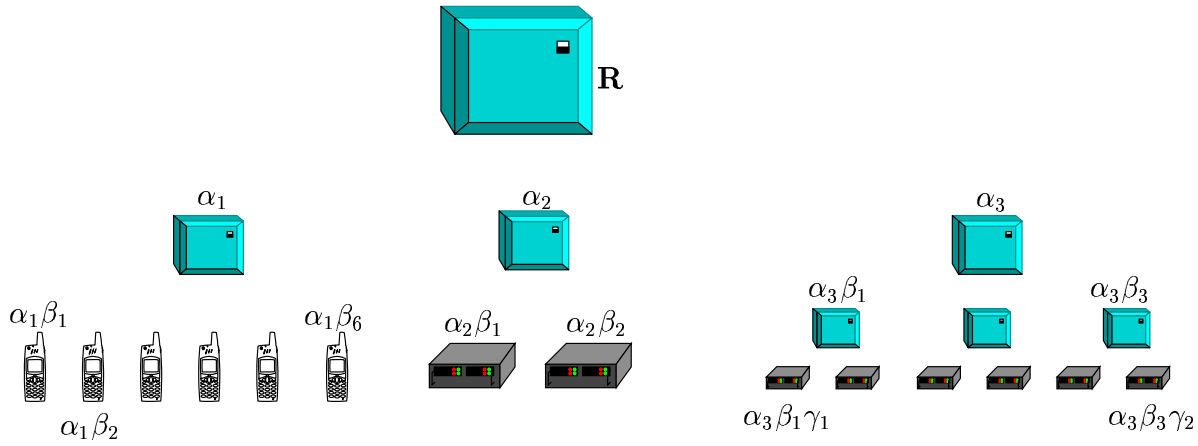

Figure 1. Hierarchical deployment of HARPS.

3.1. Broadcast Encryption

For broadcast encryption using HARPS, the sender employs a subset of all secrets not covered by the union of the r revoked nodes. If we represent by $\mathbb{R}$ the entire set of secrets and by $\mathbb{S}_r$ the secrets covered by the union of r nodes, each of the *independent* secrets in $\mathbb{R} \setminus \mathbb{S}_r$ can be used to encrypt the broadcast secret K_b.

Indexes	1	2	3	4	5	6	7	8
A	4	2	x	1	x	3	x	x
B	x	3	1	x	3	2	x	x
d_i	3	1	x	x	2	1	4	4
C	x	1	4	1	x	x	2	x
D	2	x	x	x	3	1	4	x

Consider the illustrative example above, where $P = 8$ (the total number of keys with the TA), and each node has 4 keys (or $k = 4$). We shall also assume that $L = 4$. In other words each node will have 4 keys, and the hash depth of a key in any node is uniformly distributed between 1 and 4. Node A in the example, has keys corresponding to indexes 1,2,4 and 6 - at hash depths 4,2,1 and 3 respectively. To transmit a secret to all other nodes (or to revoke nodes A and B), the source of the broadcast (let us assume that the source is the TA, who has access to all keys at hash depth 0), can encrypt the broadcast secret K_b with the secrets

1. ${}^3K_1$ (node A has ${}^4K_1$ but cannot get ${}^3K_1$,
2. ${}^1K_2$ (both nodes A and B have a key corresponding to the index 2, but the minimum hash depth is 2. So both A and B do not have access to ${}^1K_2$.
3. ${}^2K_5$, ${}^1K_6$, and
4. ${}^4K_7$, ${}^4K_7$.

Note that key indexes 3 and 4 cannot be used. While the TA can use ${}^0K_3$ and ${}^0K_4$, it does not serve any purpose - no node can decrypt the broadcast secret encrypted with those keys (as the hash depth of the keys in any node apart from the TA is at least 1). As neither A nor B have key indexes 7 and 8, ${}^4K_7$, and ${}^4K_8$ can be used to encrypt the broadcast secret.

To decrypt the broadcast secret, node C could use ${}^4K_7$, (by hashing its secret ${}^2K_7$ twice). Node D meanwhile could use ${}^3K_1$ or ${}^1K_6$ or ${}^4K_8$ to decrypt the broadcast secret.

For large r however, the cardinality of $\mathbb{R} \setminus \mathbb{S}_r$ may be small, an thus many of the $N - r$ nodes may not be able to decipher the broadcast secret too. This can be overcome by the following strategy. Divide the r nodes into a l smaller groups $\mathcal{N}_1 \cdots \mathcal{N}_l$ of size r/l. A secret $k_i, 1 \leq i \leq l$ is then broadcast to all nodes except the subset $\mathcal{N}_i$ (of r/l nodes). The l such keys that are transmitted could then be XOR-ed together to obtain the broadcast secret K_B - which none of the r revoked nodes can. Equivalently, this technique boils down to splitting one revocation broadcast into l revocation broadcasts.

Each key used by the source node to encrypt the secret, has some *probability* of being "useful" for any arbitrary node (useful only if the node has a key corresponding to the index, and if the hash depth of the key is less than or equal to the hash depth used by the source for encrypting the secret). With sufficiently large number of keys, the probability of "outage" or the probability that an intended node will miss the broadcast can be made arbitrarily small. Obviously for larger group sizes, the outage probability should be smaller. In other words, the number of encryptions that need to be used increases as the group size increases.

In general, values of $\frac{k}{P} \leq 1$ close to one is efficient for small r (number of nodes to be revoked) and small $\frac{k}{P} << 1$ is more efficient for larger r (number of encryptions needed *per* revoked node is lower).[8] As revocations can always be split into multiple revocations, at first sight it would appear that using $\frac{k}{P}$ close to one is a better option.

However, smaller values of $\frac{k}{P}$ are very useful for purposes of renewal (discussed in Section 5) and offer better security for pairwise secrets. Therefore in practice a combination of both would be a good trade-off. For instance each node may have 1000 keys. 500 of them may be assigned to each node by choosing 500 keys from say $P_1 = 550$ of the TA's (or parent node's) keys (however the hash depths of the keys in each node are different). The second set of 500 keys could be chosen from a set of $P_2 = 7500$ keys for each node (and $P = P_1 + P_2$).

With such an initial distribution of secrets, broadcast encryption using HARPS requires only about $0.75r, 1.8r, 3r$ encryptions respectively (on an average) for group sizes of a thousand, million and a billion respectively.[8]

3.1.1. Broadcast Encryption by Peers

For broadcast authentication by peers, the source node does not have access to all secrets $\mathbb{R}$, but only *it's* secrets $\mathbb{S} \subset \mathbb{R}$. The source node is restricted to using secrets $\mathbb{S} \setminus \mathbb{S}_r$ in this case (which would typically amount to employing smaller cluster sizes for revocation).

In the example above, if node C is the source of the broadcast, it could use ${}^4K_7$ and ${}^1K_2$ for encrypting the broadcast secret. Broadcast encryption by peers needs (on an average) about 10% more encryptions than broadcast encryption by the TA or parent node.

4. HARPS ENABLED DRM

Consider a deployment of heterogeneous trusted devices, which may include for instance,

1. Portable trust modules (like smart cards) assigned to every individual[¶].
2. Trust modules *built into* devices taking part in the DRM - like STBs - which render the devices "compliant."

A trust module is a secure processor and associated memory enclosed in a tamper-proof casing, and exposes only two interfaces to the outside world - encrypt() and decrypt(). Functionally, each module has an encryption / decryption block, a rules-engine and associated storage (for eg., access control list (ACL)), key ring (consisting of the preloaded KPD secrets), revocation / subscription secrets (explained shortly), and a unique ID. The purpose of the trust module is very simple - encrypt or decrypt packets of data it receives. The rules-engine inside the trust module determines if the trust module is *authorized* to decrypt a packet supplied to it - thereby converting a stateless trust module into to a stateful one.

A deployment of devices is illustrated in Figure 1. At the root of the deployment tree is a highly tamper resistant "root" module R, which spontaneously, the first time it is turned on, generates a set of P secrets randomly. No person or device would ever have access to the P secrets stored in the device R.

[¶]The trust module may have a ubiquitous interface like USB, and therefore could be plugged into any general purpose communication device - like laptops, desktop computers, PDAs, and mobile phones of the future.

Many trust modules $\alpha_1 \cdots \alpha_{N_1}$, similar to R are provided with secrets initially by physical contact with the module R. The devices are distributed to different vendors who manufacture trusted or compliant devices. In Figure 1, two such modules α_2 and α_3 are used by vendors who manufacture compliant devices. The module α_1 is provided to a manufacturer of portable trust modules for end-user authentication. The portable trust modules are represented by $\alpha_1\beta_1 \cdots \alpha_1\beta_6$ in the figure. Similarly $\alpha_2\beta_1$ and $\alpha_2\beta_2$ are STBs manufactured by a vendor who has access to α_2. The figure also shows a possible scenario where a vendor α_3 further distributes some devices $\alpha_3\beta_1 \cdots \alpha_3\beta_3$ to *other* vendors, who in turn manufacture compliant devices (device $\alpha_3\beta_1\gamma_1$ for instance, is manufactured by vendor $\alpha_3\beta_1$).

The portable trust modules are assigned to individuals after using some out-of-band mechanism to establish their identity (similar to methods used for signing public keys by a certificate authority in PKI). Each individual gets a "public key" which is just a unique ID of his / her trust module. For convenience, a descriptive string "Firstname, Lastname, Affiliation" of a person can be hashed using a public one-way cryptographic hash function to derive their unique ID (or public key). Depending on a users ID (or more specifically, the ID of their portable trust module), secrets are assigned to their trust modules. The portable trust modules can be plugged into any general purpose communication device to provide end-user authentication.

Every parent module is associated with (or chooses) a revocation secret. For example, node α_2 chooses a revocation secret (which changes from time to time) $R_{\alpha_2}(t)$. The node α_2 (whenever necessary) broadcasts its revocation secret to all its child nodes using broadcast encryption. So revoked child nodes (or STBs) will not have access to the *current* revocation secret. The root node R, similarly could revoke the vendor α_2 and all devices manufactured by the vendor. Similarly, α_1 could revoke end-users. Revoked nodes will loose their ability to take further part in the deployment (non-revoked nodes will also use the revocation secret in addition to the shared KPD secrets for mutual authentication). For purpose of revoking STBs, the actual "broadcasts" may be sent along with content(s) that are distributed. Broadcasts for revoking end-users trust modules pay be posted periodically on web-sites (which will be accessed by the trust module using any general purpose device it is plugged into).

All end-users (consumers of content) have a portable trust module. In addition they will also have a STB (like $\alpha_2\beta_1$ or $\alpha_3\beta_3\gamma_2$) to render their content. Any individual with trust module can also act as a *distributor* of content.

Consider a scenario where an individual with trust module $\alpha_1\beta_1$ wants to distribute content, playable using STBs manufactured by the vendor α_2. The content distributor chooses a content encryption key K_C (for some content C) and provides it to the vendor α_2. Communications between the distributor and α_2 are secured and authenticated by the unique pairwise KPD secret shared between α_2 and $\alpha_1\beta_1$ (or $K_{\alpha_2,\alpha_1\beta_1}$) The vendor α_2 provides the distributor $\alpha_1\beta_1$ with

$$K_{C_1} = E_{R_{\alpha_2}}(K_C) \tag{1}$$

The distributor at this point has many options for distributing his content C.

1. The distributor may decide that his content could be played in *any* legitimate player manufactured by vendor α_2.
2. The distributor may decide to distribute his content on a subscription basis.
3. The distributor wants to explicitly specify the STB that the end-user can use to render the content.

For the first approach, the distributor just needs to include the key K_{C_1} along (say in a header) with the encrypted content C (encrypted with key K_C). Any legitimate (non-revoked) player has access to R_{α_2} and therefore could obtain $K_C = D_{R_{\alpha_2}}(K_{C_1})$, and therefore decrypt the content.

For the second approach, the subscribers are provided with access to a "subscription" secret K_S. For instance a subscriber with portable trust module $\alpha_1\beta_3$ would authenticate herself to the distributor (with trust module $\alpha_1\beta_1$) using their shared KPD secret $K_{\alpha_1\beta_3,\alpha_1\beta_1}$ and receive the subscription secret K_S (say on some promise of payment). The key K_{C_1} is further encrypted with K_S to yield

$$K_{C_2} = E_{K_S}(K_{C_1}), \tag{2}$$

and the key K_{C_2} is now included with the encrypted content. Note that only users with legitimate (unrevoked) STBs *and* a valid subscription may now be able to decrypt the content. The trust module would first decrypt K_{C_2} to obtain K_{C_1} and

then encrypt K_{C_1} with the shared KPD secret between the trust module and the STB. For instance if the STB employed by the end user is $\alpha_2\beta_3$, the shared KPD secret between the the trust module $\alpha_1\beta_3$ and the STB is $K_{\alpha_1\beta_3,\alpha_2\beta_3}$. The trust module, employing a general purpose communication device that it is plugged into, could now transfer the encrypted secret $K_{C_3} = E_{K_{\alpha_1\beta_3,\alpha_2\beta_3}}(K_{C_1})$ to the STB, which can obtain K_{C_1}, and thereby K_C by decrypting K_{C_1} with the revocation secret R_{α_2} known to all non-revoked STBs manufactured by vendor α_2.

After the subscription period of a user expires (and if some users do not renew their subscription) then the distributor can revoke the users by broadcasting a new secret K'_S to all his continuing subscribers. The new subscription secret is now $K_{S_1} = K_S \oplus K'_S$ and will be used for distribution of content for all content distributed after this point in time. The broadcast for achieving this could be posted in the distributors website or even distributed along with the content. In the latter case, the STB would provide the end-user with the broadcast by the distributor and challenge the end-user to decrypt part of the content encryption secret (encrypted with the new subscription key) - which revoked users no longer have access to.

For the third approach, the encrypted content encryption key K_{C_1} would be encrypted further using the shared KPD key between the distributor $\alpha_1\beta_1$ and the STB $\alpha_2\beta_3$, (or $K_{\alpha_1\beta_1,\alpha_2\beta_3}$) before it is encrypted using the subscription secret K_S.

It is also possible for the different end-users to use devices manufactured by *different* vendors. For instance, the controller of node α_3 in the figure distributes devices $\alpha_3\beta_i, \forall i$ further to many manufacturers of compliant devices (say DVDs). In this case each manufacturer would choose their own revocation secret of the form $R_{\alpha_3\beta_i}$. Each device ($\alpha_3\beta_1\gamma_2$ for example) would have two revocation secrets - one issued by their parent and one issued by their "grand" parent α_3. It is possible in this case for α_3 to revoke a manufacturer (and therefore all devices manufactured by the manufacturer) with one broadcast!

In other words, the *vendors of devices control the devices* deployed (by periodically revoking devices that are suspected) employing broadcast encryption and the *distributor of content controls access to the content by end-users*, by employing peer-based broadcast encryption.

4.1. Authentication of Broadcasts

Broadcasts made for revoking nodes have to be authenticated in order to prevent a malicious user from broadcasting false revocation secrets. For instance, even an attacker who has managed to expose only a few KPD keys, may be able to broadcast some "new" revocation secret encrypted with the keys he / she has obtained. This would affect all nodes which are capable of decrypting the "new" secret.

Therefore the source of the broadcast should *authenticate* its broadcast. While HARPS caters for broadcast authentication, it is accompanied by a large increase in bandwidth needed - as k key based HMACs need to be appended to each broadcast (in addition to the encryptions necessary for revoking nodes). This may not be acceptable especially in cases where the number of revoked nodes are small (in which case the bandwidth necessary *per* revoked node would be substantially higher).

An efficient alternative is to use *self-authenticating* revocation secrets. For instance, the distributor could set-up a one way hash chain[14] and provide the last key in the chain as the first revocation secret (or the subscription secret provided to each subscriber). From this point on, for each revocation, the new revocation secret could be the pre-image of the old revocation secret (or one key earlier in the hash chain) - this is in essence, similar to the concept employed in the TESLA[15] broadcast authentication protocol.

5. RENEWAL OF THE KPD

A property unique to HARPS (unlike other KPD schemes) is its "high resistance to node synthesis." In other words if an attacker needs to tamper with and expose secrets from n_e devices $A_1 \cdots A_{n_e}$ in order to be able to determine shared secrets K_{BC} between two nodes B and C, or to accomplish *eavesdropping* attack, with some probability p, he may need to tamper with and expose secrets from a significantly larger number n_s of nodes $A_1 \cdots A_{n_s}$, in order to expose *all* secrets buried in B or C - or accomplish *synthesis* attack. Typically n_s may be two or three *orders of magnitude* higher than n_e for HARPS (for KPDs based on finite field arithmetic, for instance Blom's scheme,[5] $n_e = n_s$).

Note that the eavesdropping attack implies the ability to fool "siblings". The synthesis attack implies ability to fool the parent. For HARPS it is considerably more difficult for an attacker to fool the parent node than fooling siblings. Fortunately, for renewing secrets, an attacker has to fool the parent! This property of high resistance to node synthesis can be used to safely renew the KPD periodically, provided some assurance of tamper resistance / read-proofing is possible.

5.1. Some Possible Assurances

As an example, assume that it is possible to *unconditionally* protect the contents of a *single register* inside a CPU (inside a trust module). Some random value K_V is stored in the special register at all times. The k KPD keys, revocation keys and other subscription keys a node may acquire would be stored in non-volatile memory - however, always encrypted with the secret K_V. Whenever any tampering attempt is sensed, the key K_V is erased - which simultaneously renders the node unusable in future - as all the KPD keys (which are encrypted) become unusable.

For a KPD, at any point in time, only *one* key is needed. In other words only *one* key need to be stored in "less protected" RAM at any point in time for cryptographic calculations (arriving at shared secret or decrypting a broadcast secret). Therefore even if an attacker, is able to "freeze" the contents of the memory of a chip before the node has the ability to erase the contents of its RAM (however K_V is erased), only one key can be exposed by an attacker.

Another possible assurance (which would even make it unnecessary to unconditionally protect even the single secret K_V), is the use of physical un-clonable functions (PUF).[16] Gassend et. al[16] argue that it may be possible to build physically un-clonable one-way functions based on uncontrollable delays in the components of a fabricated chip. The manufacturer could collect a set of challenge-response pairs from the PUF in each device, which could later be used to identify the chip. However, such a PUF can also be used for encrypting the secrets! For example, the responses for various challenges can be used for encryption the secrets stored in the device. Every time a KPD secret is needed, the PUF could be challenged to provide the secret necessary (the response) to decrypt the KPD secret.

For a deployment of HARPS where say each node (say in the third level of hierarchy - which includes the the portable modules $\alpha_1\beta_j$, and STBs $\alpha_2\beta_i$) has 1000 keys (say 500 keys chosen from $P_1 = 550$ keys and 500 keys chosen from $P_2 = 7500$ keys, where $P = P_1 + P_2$) the attacker would have to tamper with (and expose one secret from)

1. about $n_e = 15,000$ nodes in order to be able to perform the eavesdropping attack - or impersonate nodes for purposes of interaction with other nodes, and
2. over $n_s = 5,000,000$ nodes - in order to be able to *synthesize* a node.

For renewal, each node authenticates itself to the parent node using *all* its old keys in order to receive a new set of keys, then accomplishing node synthesis is essential for an attacker to synthesize a non-compliant node that can take part in renewal (or "fool" the parent). If the attacker is able to perform eavesdropping (which by itself might be very expensive for an attacker to realize) but however is not able to synthesize nodes (because it is *prohibitively* expensive), all his hard work (compromising 15,000 nodes) is wasted after the next round of key updates - thus reducing the motivation of the attacker to even undertake eavesdropping attack. Note that nodes that have been tampered with automatically get "revoked" from the system as they cannot take part in the key renewal process.

6. CONCLUSIONS

We presented a security framework for a DRM based on a random key pre-distribution scheme, HARPS. HARPS has three major advantages over other broadcast encryption schemes based on tree based key distribution schemes. Firstly, it permits broadcast encryption by peers. Secondly, it is renewable. Thirdly, the KPD secrets also cater for broadcast authentication, and mutual authentication of devices, irrespective of the vendor manufacturing the device.

Renewability of the deployment secrets is very crucial for long-lived security of any deployment. This is perhaps one of the primary shortcomings of tree-based broadcast encryption schemes widely reported in literature today. By permitting a framework for many useful cryptographic primitives like pairwise authentication, broadcast authentication and broadcast encryption, and at the same time being very light on resource requirements, HARPS could be a practical solution for securing DRM schemes. Note that severely resource contrained devices (which may have lower storage for keys and may be less tamper resistant) could always be relegated to lower and lower levels of hierarchy without compromising the security of the deployment. In other words, HARPS offers scalable security, depending on the available resources in devices.

REFERENCES

1. H. T. Sencar, M. Ramkumar, A.N. Akansu, *Data Hiding Fundamentals and Applications: Content Security in Digital Multimedia*, Elsevier Academic Press, July 2004.
2. J. Lotspiech, S. Nusser, F. Pestonoi, "Anonymous Trust: Digital Rights Management using Broadcast Encryption," Proceedings of the IEEE, **92** (6), pp 898–909, 2004.
3. R. Gennaro, A. Lysyanskaya, T. Malkin, S. Micali, T. Rabin, "Tamper Proof Security: Theoretical Foundations for Security Against Hardware Tampering," Theory of Cryptography Conference, Cambridge, MA, February 2004.
4. R. Needham and M. Schroeder, "Using encryption for authentication in large networks of computers," Communications of the ACM, 21(12), December 1978.
5. R. Blom, "An Optimal Class of Symmetric Key Generation Systems," *Advances in Cryptology: Proc. of Eurocrypt 84*, Lecture Notes in Computer Science, **209**, Springer-Verlag, Berlin, pp. 335-338, 1984.
6. M. Ramkumar, N. Memon, "An Efficient Random Key Pre-distribution Scheme for MANET Security," to appear, IEEE Journal on Selected Areas of Communication, March 2005.
7. M. Ramkumar, N. Memon, "A System for Digital Rights Management Using Key Pre-distribution," Proceedings of ICME 2004, Taipei, Taiwan, June 2004.
8. M. Ramkumar, "Broadcast Encryption Using Random Key Pre-distribution Schemes," manuscript under preperation.
9. J. Garay, J. Staddon and A. Wool, "Long-Lived Broadcast Encryption," Proceedings of Advances in Cryptology - CRYPTO 2000, Mihir Bellare (Ed.), LNCS (1880), Springer-Verlag, pp. 333-352, August 2000.
10. A. Fiat, M. Noar, "Broadcast Encryption," Lecture Notes in Computer Science, Advances in Cryptology, Springer-Verlag, **773**, pp 480–491, 1994.
11. C.K. Wong, M. Gouda, S. Lam, "Secure Group Communications using Key Graphs," Proceedings of SIGCOMM 98, pp 68–79, 1998.
12. D. Noar, M. Noar, J. Lotspiech, "Revocation and Tracing Routines for Stateless Receivers," Lecture Notes in Computer Science, Advances in Cryptology, Springer-Verlag, **2139**, 2001.
13. M. Ramkumar, K.A. Sivakumar, "Broadcast Authentication with Preferred Verifiers," submitted to the 25th Annual International Cryptology Conference (Crypto 2005).
14. L. Lamport, "Password Authentication with Insecure Communication," Communications of the ACM, 24(11):770-772, November 1981.
15. A. Perrig, R. Canetti, D. Song, D. Tygar, "Efficient and Secure Source Authentication for Multicast," in Network and Distributed System Security Symposium, NDSS '01, Feb. 2001.
16. B. Gassend, D. Clarke, M. van Dijk, S. Devadas, "Delay-based Circuit Authentication and Applications," Proceedings of the 2003 ACM symposium on Applied computing, Melbourne, Florida, pp 294 – 301, 2003.