A System for Digital Rights Management Using Key Predistribution

Mahalingam Ramkumar
Department of Computer Science and Engineering
Mississippi State University
Nasir Memon
Department of Computer and Information Science
Polytechnic University

***Abstract*— We propose a system for digital rights management (DRM) which facilitates large scale deployments of heterogeneous devices, manufactured by different vendors, to interact and authenticate with each other securely in order to ensure fairness of transactions. The proposed system allows for both pluggable (and transferable) security modules for end-user authentication and built-in (non-transferable) security modules for mutual authentication of compliant devices (manufactured by different vendors). For the underlying security mechanism we propose the use of a recently proposed random key predistribution scheme, HARPS (HAshed Random Preloaded Subsets).**

I. Introduction

Systems for digital rights management (DRM) strive to achieve fairness of transactions between suppliers (content creators) and consumers (end-users) of digital content. Typically the exchange consists of monetary promises by the consumer in return for "rights" to consume the content. Rights to consume the content, in turn, may be subject to many restrictions. For instance, an end-user may purchase rights to one-time viewing of a movie. In another instance, the end user may purchase rights for unrestricted access to the content for a duration of one month. The job of the DRM is to ensure that the end user does not violate his/her rights. The system should also provide for flexibility in allowing different models for the rights of the end user, with respect to consumption of the content.

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 components - the content creator, the content, and 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 components of the DRM come to light. For instance the "content" initially perhaps resides in a camera which is then transferred to other devices where it undergoes further processing. Processed content may be copied into several storage media for distribution or may be loaded to a device (like 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 device 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.

Practical implementation of a system for DRM demands a mechanism for establishing "trust" between various components of the system. In other words, all the components of the system should in some way *guarantee* "compliance" to pre-established rules. The trust mechanism, would then be a way of *ensuring* that a device is indeed "compliant." There are two fundamental approaches to establishing "trust relationships." A first approach involves a *centralized* "client-server" model. In such an approach, each component involved in the DRM first establishes a trust relationship between itself and the trusted server. This trust can then be leveraged to develop trust between various components of the DRM by *involving the trusted server in the mediation process* between any two components. 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. A second distributed approach is that of a mechanism to establish trust between components in an ad-hoc manner. Such an architecture would permit establishment of trust between *any two* components without an *active involvement* of a trusted authority (TA).

In this paper, we propose a system for DRM based on the second approach. In the next section we discuss alternatives for enabling trust relationships between devices. Key predistribution schemes, and in particular HARPS, is identified as a suitable enabler. Section III provides an overview of the proposed system. A key element of the proposed system is its ability to permit different vendors manufacture "compliant devices" without compromising the overall security of the system. Conclusions are offered in Section IV.

II. Mechanism for Trust

From a cryptographic perspective, if two nodes (we shall interchangeably use the terms node, device or component to mean the same thing) share a secret that is not privy to any *other* node, they can then leverage this secret to establish a secure and trusted communication channel. The ability to establish such shared secrets is precisely what a key distribution scheme (KDS) does. In the absence of a persistent third party TA, there are two ways of establishing

trust between nodes - the use of public key cryptography in conjunction with an offline certifying authority (CA), or key predistribution schemes (KPDS) with an offline TA.

For the former scenario, each node would be loaded with the public key of the CA, the node's private key, and a certificate (signed by the CA) containing public key and ID of the node.

For the latter, some secrets may be preloaded in each node by a TA prior to deployment. For the most basic form of KPDS, a node may be preloaded with a secret it shares with *every other node*. However, this would imply an unreasonably large storage requirement in each node, especially if the network size (or the total number of nodes in the system) N is large. Fortunately, it is possible for KPDSs to perform *trade-offs between complexity and security*.

For instance, with some guarantee of tamper resistance, it may be possible to ensure that an attacker is not able expose keys in more than say n devices. There exists mechanisms, then, to build KPDSs which are n-secure (the KPDSs are secure as long as the number of compromised nodes is not greater than n, but no guarantee is provided if the number of compromised nodes is greater than n). Some KPDSs like the Blom's KPDS [3], needs only $k = O(n)$ keys to be preloaded in each node in order to ensure that the KPDS is n-secure. However such schemes have the problem of catastrophic failure. As long as the number of compromised nodes is less than n the system is completely secure. The moment the number goes above n the system is completely compromised. In practice it is definitely desirable to use KPDSs which do not such a catastrophic onset of failure.

Many KPDSs [6], [7] have been subsequently proposed, which eliminate the problem of catastrophic failure, for such schemes, the number of keys needed for each node turn out to be dependent on N (the network size or the total number of nodes). Random KPDSs [8], [9], [1] which have received considerable attention in literature recently, which achieve this requirement without introducing the dependency of k on N. For example, the scheme due to Leighton and Micali (LM), achieves $k = O(n^2)$. RPS (random preloaded subsets) [9] and HARPS do even better, by achieving $k = O(n)$.

In practice, to restrict the size of attacker coalitions to n, apart from a mechanism for tamper-proofing, we also need a mechanism for *periodic renewal* of keys. However with millions of components in the system, it may be impractical to perform renewal *synchronously*. In order to avoid "update floods", it is necessary to be able to deploy the KPDS in a hierarchical fashion. HARPS facilitates such a hierarchical deployment. While this alone would alleviate the problem to a large extent, it is also necessary to enable two nodes, one of which may have updated secrets, and the other which has not had the opportunity to perform updates yet, to continue to communicate securely. Once again, HARPS provides a highly flexible mechanism for updates which makes this feasible.

A. HARPS

HARPS [1] is a simple random KPDS, the performance of which is dictated by 3 parameters - P the size of the key-pool, k the size of the "key-ring" in each node, and L the maximum "hash depth." A HARPS system consists of a TA who chooses P secrets. Each node is given a unique ID. The ID of each node determines the subset of k keys that the node receives. Each key in the key-ring is further hashed a variable number of times (uniformly distributed between 1 and L, the exact values of which are also determined by the ID through a public one-way function). Two nodes just need to exchange their IDs to determine the secrets they share.

From their IDs, they can arrive at a unique secret (with a very high probability) by determining the keys they share, and the corresponding hash depths of the shared keys. The node which has a lower hash depth for a particular shared key needs to hash forward (that specific key) to reach the same hash depth as the other node. Once both nodes, in this fashion, have reached a common hash depth for all shared keys, they can hash the shared keys (now with same hash depths) together to calculate the shared secret independently. Under proper selection of parameters of the system (P, k, L) we can ensure that the probability of compromise of shared secrets by coalitions of n nodes (this includes the probability that two nodes *cannot* arrive at a shared secret) is kept to vanishingly small levels. That the nodes can arrive at shared secrets which are tied to their ID, serves also as a mechanism for mutual authentication of the nodes.

As an example, a HARPS systems designed to ensure that an attacker who has managed to sniff *all* keys buried in 20 nodes can only "eavesdrop" on any arbitrary communication with a probability less than 10^{-20}, would need $k = 1610$ keys (with $P = 19390$ and $L = 64$). For larger n k increases linearly.

In addition to being the most efficient of the three random KPDSs, HARPS also offers an excellent feature of a tree-hierarchy. In such a hierarchy, the node at the top-most level (say level H) has P keys. Each node at level $H-1$ has k keys. Nodes at level $H-1$ are the child nodes of the node at level H. Each node in level H may serve as the parent node for many nodes in level $H-2$. Each child node gets a subset of keys from the parent nodes. The subsets at each level are hashed a variable number of times. For example, nodes in level $H-1$ have hash depths between 1 and L. Nodes in level $H-2$ will have hash depths between $L+1$ and $2L$. Thus even if nodes in a lower level are compromised, they reveal nothing about the secrets at a higher level (if the hash function used is pre-image resistant).

III. System for Digital Rights Management

In this section we describe the proposed system for DRM. For the sake of brevity, we focus simply on the part of the DRM system that enables two nodes to establish a trusted relationship. We understand that a full blown DRM system would consist of protocols and modules for tracking, billing etc. and our system can be incorporated into any such architecture proposed in the literature [2], [4]. The security component we propose consists of low complexity security modules employing the HARPS key pre-distribution scheme. Some security modules may be pluggable devices, which can perhaps be plugged into a mobile phone or PDA or desktop /

laptop computers, for communicating over open channels like the Internet with external devices. Some security modules may be built into compliant devices.

Any compliant device has some device specific parts, a communication module and a built-in security module, in a tamper-proof casing. The security module serves as a "proof of compliance" by enabling mutual authentication of such devices. Figure 1(b) is a block diagram of a compliant device.

Each security module is a HARPS enabled node. Each node has a unique hierarchical ID. For a 3 level deployment, there is a single "root" node at level 3. Nodes or security modules at level 2 (say with IDs α_i are distributed to various vendors, who manufacture pluggable security modules or compliant devices with in-built security modules. A vendor with level 2 module α_1 for instance, manufactures level 3 devices with IDs $\alpha_1\beta_j$. Such devices would have a subset of keys belonging to the module α_1 which are hashed further.

The protection of higher levels from lower levels ensures that even though the vendor may have access to keys buried in the devices $\alpha_1\beta_j$ that he manufactures, he cannot obtain the secrets loaded in his own security module α_1 from the lower level secrets. Thus HARPS permits different vendors to manufacture devices. Two arbitrary devices, say $\alpha_1\beta_2$ and $\alpha_2\beta_3$ will be able to arrive a shared secret based on their ID's and therefore be able to authenticate each other.

Such a 3-layer hierarchical deployment is shown in Figure 1 (a). The boxes in level 1 represent compliant devices, and the portable communication equipment actually represent pluggable security modules plugged into such devices.

Each security module exposes a simple interface to encrypt / decrypt data that is sent to / from the module. For encryption, the input to the modules is a data packet and the ID of the destination node (security module). The output is the encrypted data packet (using the shared key of the module performing the encryption and the destination module) which only the destination module can decrypt. For decryption, the input is the encrypted packet with the ID of the sender (or sender's security module ID). Thus the security modules needs to provide only two APIs to the outside world. Internally, it needs secure storage of keys and a processor for performing *symmetric cryptography*. A block diagram of such a security module is shown in Figure 1 (c).

We shall now see an example of how such a deployment can be used for enforcing rights for digital video distribution. The system we consider for illustrative purposes, consists of the following components:

1) Digital content C_O.
2) An encryption device E.
3) A device D employed by the content distributor
4) A pluggable end user module P plugged perhaps in a hand held device of the end user.
5) A device S, a set-top box (STB), with the end user.
6) The renderer M (say a monitor or a TV).

All devices mentioned above are "compliant" devices, probably manufactured by different vendors. We shall assume that S for instance, represents the (hierarchical) ID of the device S. In the discussion below we use the notation $\mathcal{E}_{K_{AB}}()$ to represent encryption using key K_{AB}, where K_{AB} is the shared secret

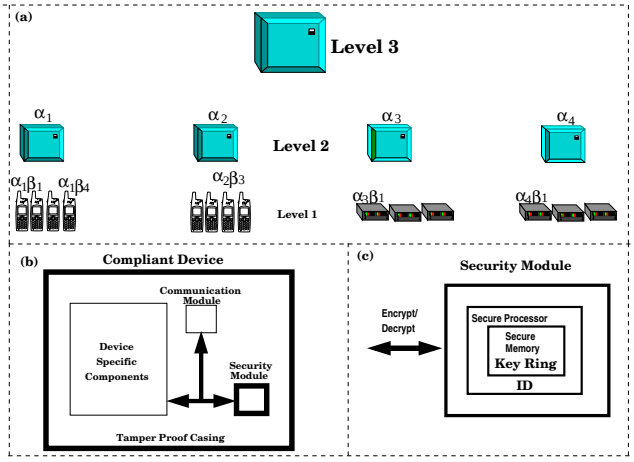

Fig. 1. (a) Hierarchical deployment of security modules. (b) Block diagram of a compliant device. (c) Block diagram of a security module.

between nodes A and B. $\mathcal{D}_{K_{AB}}()$ represents decryption. The actual (symmetric) cryptographic algorithm employed is of no concern to us.

The content creator employs device E to encrypt the content, with content key K_C. We shall represent the encrypted content as $C = \mathcal{E}_{K_C}(C_0)$. The encrypted content through some channel, is received by the end user (may even be distributed in DVDs for example).

A guarantee that both devices P and D for instance, can arrive at a secret K_{PD} that no other device (even other devices manufactured by the same vendor) can, serves as a mechanism of mutual authentication of the handheld device P of the end user, and the distributor's module D. Having confirmed the authenticity of each other, the device D and P negotiate the terms and conditions for access to the content C. The end user, also submits to D, the ID S of the STB that the end user would use to unscramble the content. After D and P agree on the terms and conditions R, the device D submits the authorization R to the content creators device E. Among other things, set of rights R will also contain the ID of the user P and the ID of the decrypting device S.

Now E encrypts the content key K_C first with K_{ES} (the shared key between E and the STB S) to get $K_{C_1} = \mathcal{E}_{K_{ES}}(K_C)$, followed by an encryption with key K_{EP} (shared key between E and P) to get $K_{C_2} = \mathcal{E}_{K_{EP}}(K_{C_1})$.

Thereafter, E sends K_{C_2} to the device D (all exchanges between E and D are further encrypted using key K_{ED}). Device D appends to the rights R, the encrypted content key K_{C_2}. All exchanges between E and D may occur over the Internet.

R (now with K_{C_2}) is passed to device P by D. All exchanges between D and P are encrypted with K_{PD} to ensure that only P can decrypt messages sent by D. P decrypts R, and also decrypts the attached K_{C_2} to get $K_{C_1} = \mathcal{D}_{K_{PE}}(K_{C_2})$. Exchanges between D and P may occur over the Internet if the device P is plugged into is say a PDA / laptop / desktop. It may even occur over phone lines if the communication device is a mobile phone.

K_{C_1} and the rights R are transmitted to the device S

(once again, all exchanges between P and S are encrypted with K_{PS}). The device S decrypts K_{C_1} with K_{SE} to obtain the content key K_C. The device S proceeds to decrypt the content using K_C, decompresses it to get raw video C_{OO}. Communication between P and S may be utilize IR or blue tooth channels that both devices (the device with P and the compliant STB S) possess.

S encrypts the video C_{OO} using K_{MS} before sending it to the renderer M. The renderer M decrypts the encrypted video and possibly inserts invisible and indelible fingerprints into the video before it is sent to the screen. Exchanges between S and M could occur through physical cables.

The STB S which is also supplied with the rights R ensures compliance to the rights. The rights may include, for instance, the number of times the content may be viewed, duration of validity of the "license" R etc. Note that the content is not restricted to be on a particular form of media device. The content could be distributed in a DVD or could be streamed over the Internet.

In the scenario above, every device is "trusted" by every other device as the security modules enables verification of authenticity, by means of shared secrets (tied to the IDs). The device S for instance, is trusted not to reveal the key K_C. Similarly S trusts M to insert a fingerprint before each frame is painted on the screen. If M were not compliant, it would obviously not be able to decode the encrypted raw-video it receives from S.

If the device P is portable, an end user who has purchased rights to view the content for a period of a week, may use perhaps another STB S_1 to watch the content - the user just has to specify the device he/she wishes to employ (for example the user may wish to watch the movie when he/she is travelling).

While it is possible for the device S to establish a connection (eg. over the Internet) directly with the device E, most home users, sensitive of privacy issues, would be averse to the idea of employing devices at home, whose usage can perhaps be monitored by external agents. However, in the proposed scenario the device S only communicates with the end user's hand-held device P (perhaps using blue-tooth or IR channels).

It is also possible for device S itself to add a fingerprint and thereby eliminate the need for another compliant device M. But adding additional features may increase the cost of devices. For example not all content creators may demand that the content be fingerprinted before consumption at the end-user's site. Some end users may employ an alternate device M_1 which perhaps does not add fingerprints, but probably ensures compliance to some form of analog copy protection scheme. The rights R would dictate the class of monitors the device S may send its output to, for that *particular* content. The "class" of course, is easily identifiable, perhaps by a device's "level 2 part" of the hierarchical ID. The system also allows for vendors to obtain multiple level 2 devices. Yet another possibility is for vendors to introduce additional levels of hierarchy below each level 2 device they possess.

IV. Conclusions

Digital rights management systems depend on a method for establishing trust relationships between different components involved in executing the transactions between creators and consumers of content. Public key cryptography based techniques, are perhaps one of the more common approaches to realize the requirement for trust relationships. In this paper we have proposed a system that employs key predistribution to establish a trusted relationship. A recent flurry of research, principally aimed at securing ad hoc resource constrained nodes forming MANETS (Mobile Ad hoc NETworks) and sensor networks [1], [9], has resulted in significant advances in key predistribution schemes. An obvious advantage of KPDSs, is the drastic reduction in resource requirements (in terms of processing power and bandwidth) when compared to nodes relying on public key cryptographic techniques to establish trust relationships.

The main disadvantage of KPDSs is that they depend on some form of assurance on tamper-resistance. However, systems employing the other alternative of public key cryptography, will also need to protect nodes from exposure of their *private* keys. The principal difference then is in the *extent* of tamper resistance needed. For public key cryptography, compromise of private keys of one node does not affect exchanges not involving the compromised node. However, preliminary results from our ongoing work, indicate that under certain realistic assumptions on a model for tamper-resistance, practical deployments of HARPS could reasonably resist tampering even with tens of thousands of nodes (all of which has to be carried out within a single update cycle).

Another supposed disadvantage of KPDSs is the necessity for nodes to perform periodic key updates - for which they need to communicate with the TA (or parent node). However, systems employing a public key based infrastructure also need to contact the CA periodically for the purpose of revocation of nodes. Additionally, HARPS offers an easier mechanism for revocation of nodes. Revocation of nodes is automatically achieved with updates, by not allowing revoked devices to participate in updates. All that is necessary is for the TA (the parent security module) to maintain a list of nodes to be revoked - the nodes do not maintain a revocation list.

References

[1] M.Ramkumar, N. Memon, "An Effi cient Key Predistribution Scheme for MANET Security," submitted to the IEEE Journal on Selected Areas of Communication.

[2] ContentGuard. Rights management from Xerox. Available at www.contentguard.com, 2000.

[3] 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.

[4] InterTrust Technologies Corp. Digital rights management. Available at www.intertrust.com/de/index.html, 2000.

[5] L. Gong, D.J. Wheeler, "A Matrix Key Distribution Scheme," *Journal of Cryptology*, **2**(2), pp 51-59, 1990.

[6] C.J. Mitchell, F.C. Piper, "Key Storage in Secure Networks," *Discrete Applied Mathematics*, **21** pp 215–228, 1995.

[7] M. Dyer, T. Fenner, A. Frieze and A. Thomason, "On Key Storage in Secure Networks," *Journal of Cryptology*, **8**, 189–200, 1995.

[8] T. Leighton, S. Micali, "Secret-key Agreement without Public-Key Cryptography,"*Advances in Cryptology* CRYPTO 1993, 456-479, 1994.

[9] M. Ramkumar, N. Memon, R. Simha, "Pre-Loaded Key Based Multicast and Broadcast Authentication in Mobile Ad-Hoc Networks," Globecom-2003.