

# A Hybrid Enforcement Model for Group-Centric Secure Information Sharing

Ram Krishnan  
George Mason University  
Email: rkrishna@gmu.edu

Ravi Sandhu  
University of Texas at San Antonio  
Email: ravi.sandhu@utsa.edu

**Abstract**—Group-Centric Secure Information Sharing (g-SIS) is motivated by the need to dynamically share information amongst a set of authorized users for a specific purpose. Authorized group users may read and contribute new objects to the group. An important usability objective in g-SIS is to allow users to access group objects offline without having to contact a server every time an access is requested. Thus a fundamental requirement for g-SIS is that protection needs to extend to clients. Henceforth we assume that a Trusted Reference Monitor (TRM) is present on the client platforms that can enforce the group policies in a trustworthy manner. In this paper, we discuss three different approaches for realizing a scalable and high-assurance g-SIS. In a Micro-Distribution (MD) architecture, objects are individually encrypted for each group user. Thus the server shares a unique key with each user. In a Super-Distribution (SD) architecture, a single key is shared amongst all group users and thus group objects are uniformly encrypted. SD promotes “protect once, access when authorized”. We discuss the pros and cons of both MD and SD architecture and propose a novel split-key RSA based hybrid architecture. As we will see, this architecture incorporates the high-assurance aspect from MD and the usability and scalability advantages from SD approach respectively.

## I. INTRODUCTION

Secure Information Sharing (SIS) or sharing information *while* protecting it is one of the earliest problems to be recognized in computer security, and yet remains a challenging problem to solve. The central problem is that, in a distributed system, copies of digital information are easily made and controls on the original typically do not carry over to the copies. One main roadblock was the lack of hardware-rooted trust on client platforms. The advent of Trusted Computing Technology [2] and the feasibility to verify the trustworthiness of software running on trusted hardware (e.g., [19]) has made it possible to address SIS. Here, a Trusted Platform Module (TPM), a security chip that can store keys and provide trusted capabilities, is integrated in the system in an inseparable manner. With this hardware-root of trust, access control can now be trustworthily enforced on client platforms where the content is decrypted and displayed, so as to ensure that only authorized users get to see the content and that they are unable to make plaintext unprotected copies. There has been considerable interest in this approach, initially driven by the forces of digital rights management for entertainment content seeking to protect revenue but more generally seeking to protect content for its sensitivity. An overview of SIS motivation and solution approaches was presented in [11].

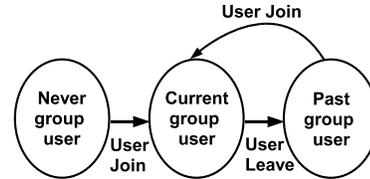


Fig. 1. User Membership States.

The g-SIS problem, introduced in [9], [10], is motivated by the need to share sensitive information amongst a group of authorized users. For simplicity, we only consider read operations and addition of new objects to the group in this paper. In g-SIS, users may join, leave and re-join the group (figure 1). Similarly, objects may be added, removed and re-added to the group. Access decision is made based on the relative membership state of user and object. For example, a user joining a group may only access objects that are added after join time. In another scenario, the user may also be allowed to access objects added prior to join time. The metaphor is that of a secure meeting room (although virtual) where members’ access depend on their participation period. For example, in a program committee meeting, when discussing Alice’s paper, she may be required to step out the room and thus cannot access discussions during her absence. In another scenario, the discussions during Alice’s absence may be recorded on a whiteboard and she may access them on her return. Our architecture can support a wide variety of such policies.

In [8], the authors discuss two distribution approaches, namely Super-Distribution (SD) and Micro-Distribution (MD), for g-SIS. In the context of g-SIS, Super-Distribution (SD) refers to widespread distribution of protected group objects. That is, a group object is encrypted once using a group key and released into the infospace through any available means such as Email, P2P, WWW, USB drives, etc. Although any user may obtain a copy of the encrypted object, only authorized group users may read the object using access machines running trustworthy and verifiable software. This can be briefly summarized as “protect once and access where authorized”. This is significantly different from the Micro-Distribution approach for sharing. In MD, objects are typically custom encrypted individually for each authorized group user, thereby incurring scalability and performance costs. In this

paper, we discuss the pros and cons of each approach and propose a new hybrid approach. As we will see later, both SD and MD suffers from many practical limitations and our hybrid approach combines the advantages of SD and MD approach and minimizes their disadvantages.

The remainder of this paper is organized as follows. In section II, we review the g-SIS architecture and discuss the SD and MD approach. In section III, we present the hybrid approach and compare the three approaches. In section IV, we discuss related work and conclude.

## II. G-SIS ARCHITECTURE REVIEW

In this section, we review the g-SIS architecture that was proposed in [8] and discuss the SD and MD approach for object access in g-SIS. In the next section, we discuss our hybrid approach that addresses the limitations of SD and MD.

An important g-SIS objective that we are interested in is to enable offline access. We strongly believe that some degree of offline access is highly desirable in SIS where disconnected access attempts are likely. Offline periods of access can be restricted by various measures such as time, usage, etc. Thus, we expect that sometimes access decisions will be made locally without contacting a central authority. This requires that authorization information such as user attributes be stored locally and used in a trustworthy manner. Nevertheless, authorization information needs to be periodically refreshed with the central authority. Motivations for offline access in the context of SD in P2P networks in mobile devices can be found in [14].

We follow the well-known architecture-mechanism separation principle as illustrated in [20]. In this section, we assume that the users interact with other entities using access machines running a Trusted Reference Monitor (TRM) whose software configuration can be verified. Authorization information such as the group key(s) will only be available to the TRM in a trustworthy state. Thus the TRM can faithfully enforce group policies locally. It is beyond our current scope to discuss implementation model to realize the TRM. A number of work on building trustworthy and verifiable systems for enforcing mandatory policies using Trusted Computing Technology can be found in the literature (see [19], [18], [12] for example).

### A. System Characterization

The g-SIS system consists of users and objects, trusted access machines (using which users access group objects), a Group Administrator (GA) who is responsible for updating user and object attributes and a Control Center (CC) that maintains the attributes. An access control policy is specified using user and object attributes. A user's access machine has a Trusted Reference Monitor (TRM) that maintains a local copy of user attributes which are refreshed periodically with the CC so as to reflect changes, if any, in their values. There are many approaches to trigger a refresh of user attributes. For example, a refresh could be triggered based on time-out or a count on the number of times the user attributes (e.g. group key) may be used by the TRM to decrypt group objects. In this paper, we take the usage count based approach for refresh

(see for example [18], [12]). Object attributes are embedded in the object itself. An object removed from the group is listed in the Object Revocation List (ORL) which is provided to the TRM as part of refresh. A g-SIS system can be characterized as follows:

User attributes	{id, Join_TS, Leave_TS, ORL, gKey, N}
Object attributes	{id, Add_TS}
Access Policy	Authz( $u, o, read$ ) $\rightarrow o \notin \text{ORL}(u)$ $\wedge \text{Leave\_TS}(u) = \text{NULL} \wedge$ $\text{Join\_TS}(u) \leq \text{Add\_TS}(o)$

User attribute Join\_TS is join time of a user, Leave\_TS is the leave time of user (if the user has left the group, NULL otherwise), ORL is the Object Revocation List that identifies the list of objects that have been removed from the group along with their history of add and remove time-stamps, gKey represents the group keys (symmetric or asymmetric key pair) using which objects can be encrypted and decrypted and N is the usage count that limits usage of gKey before a refresh of user attributes is required with CC. Object attribute Add\_TS is the time at which an object was added to the group. Attribute id represents a unique identity for each user and object. Any access policy (based on figure 1) can be specified using these sets of attributes. Authz here specifies that a user is allowed to read an object as long as both the user and object are current members of the group and the object was added after the time at which the user joined the group. This policy is used as an example for our discussion below but in general the CC may instruct the TRM to enforce any such policy.

### B. System Architecture

Figure 2 shows the g-SIS architecture and illustrates the interaction between various components. The GA controls group membership and policies. The CC is responsible for maintaining authoritative attributes of group users and objects on behalf of the GA.

- *User Join (steps 1.1-1.4)*: Joining a group involves obtaining authorization from the GA followed by obtaining group credentials from the CC. In step 1.1, the TRM on user's access machine contacts the GA and requests authorization to join a group. The GA authorizes the user join in step 1.2 (by setting AUTH to TRUE). The user furnishes the authorization to join the group and the evidence that the access machine is in a good software state to the CC in step 1.3. The integrity evidence is a signed hash that proves that the chain of software loaded during the boot-up process including the system steady-state is trustworthy. The CC remotely verifies GA's authorization, that the user's access machine is trustworthy (using the integrity evidence) and has a known TRM that is responsible for enforcing g-SIS policies. In step 1.4, the CC provisions the attributes after setting them with appropriate values. We refer to these attributes as group credentials or simply credentials. Note that we assume

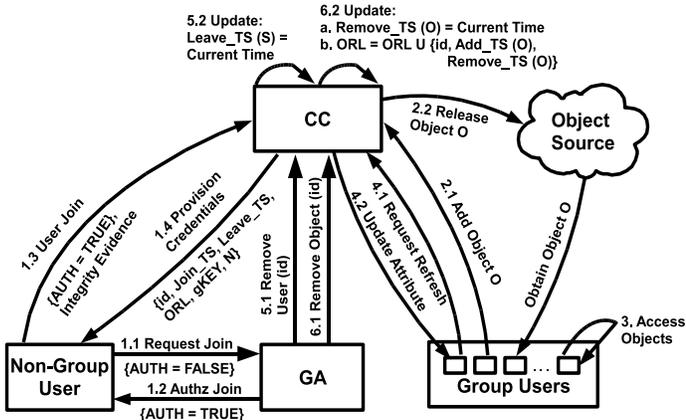


Fig. 2. g-SIS Architecture.

that these credentials are provisioned in a manner such that only the TRM in the user's machine can access them.

- *Add Object (steps 2.1-2.2)*: From here on, the user is considered a group member. Objects may be added to the group by users after the CC sets the Add\_TS (step 2.1). The CC verifies the object<sup>1</sup> and sets the Add\_TS attribute. We assume object attributes are embedded in the object itself. The CC then releases the protected object to the Object Source (step 2.2). The Object Source acts as the object repository from which users may obtain objects. It is possible for the CC itself to act as the Object Source, but in practice, this may be an independent entity.
- *Access Objects (step 3)*: Users may access group objects as per the group policy using the credentials obtained from the CC. This is locally mediated and enforced by the TRM. Note that the objects may be obtained from Object Source and stored locally. Because of the presence of a TRM on user's access machines, these objects may be accessed offline conforming to the policy. Recall that various access policies are possible as discussed in section II-A.
- *Attribute Refresh (steps 4.1-4.2)*: The TRM refreshes user attributes with the CC periodically. More generally, g-SIS access policy, like the one discussed in section II-A, may be updated or replaced in these steps. A usage count limits the number of times the credentials (e.g., gKey) may be used to access group objects (like consumable rights). Thus objects may be accessed until the usage count is exhausted and the TRM will be required to refresh user attributes in steps 4.1 and 4.2 before any further access can be granted.
- *Administrative Actions (steps 5.1-5.2 and 6.1-6.2)*: The GA may have to remove a user or object from the group. In step 5.1, the GA instructs the CC to remove a user. The CC in turn marks the user for removal by locally setting

<sup>1</sup>We treat this as an abstract step because verification of object depends on the specific application. In one case, this could simply be proof-reading while in another it could be a verification that the content is appropriate.

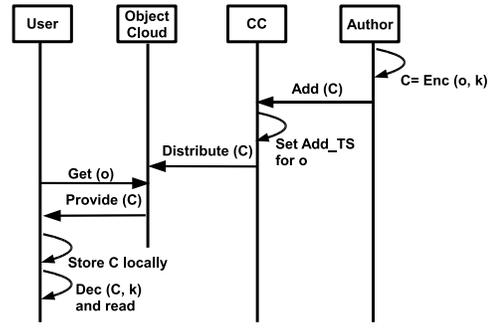


Fig. 3. Super-distribution in g-SIS.

the user's Leave\_TS attribute in step 5.2. This attribute update is communicated to the user's TRM during the refresh steps 4.1 and 4.2. In the case of object remove, an Object Revocation List or ORL is provisioned by the CC on the user's access machine. Thus for object removal (steps 6.1-6.2), the object's id, Add\_TS and Remove\_TS are added to the ORL.

### C. Super Vs Micro-Distribution based g-SIS Architecture

We now discuss SD and MD approaches for object distribution and access in the g-SIS architecture. In figure 2, steps 2.1 and 2.2 are object distribution steps and step 3 is the offline object access step. We quickly compare and point out the pros and cons of each approach and present a superior hybrid approach in the following section. In this paper, *Super-Distribution* (SD) refers to widespread distribution of protected group objects while only authorized users may read them. Here all group users share a single group key using which objects are encrypted and decrypted. *Micro-Distribution* (MD) refers to custom encryption of objects for each authorized user. Figures 3 and 4 illustrate the difference between SD and MD in g-SIS. For simplicity, we assume that a symmetric key is used for encrypting and decrypting objects in both SD and MD. Our discussions below apply equally well to SD and MD approaches using asymmetric keys.

In SD based approach all group users share a single group key for encrypting and decrypting group objects. Thus in SD (figure 3), an Author (a group user) creates an object, encrypts the object using the group key (mediated by TRM) and sends it to the CC for approval and distribution. The CC verifies the object, time-stamps object add and releases this protected object into the infospace (referred to as Object Cloud) through conventional networks such as WWW, Email, etc. or sneakernet such as USB flash drives. These are steps 2.1 and 2.2 in figure 2 for SD approach. Other group users can obtain such encrypted objects and store them locally in his/her access machine for later offline access. Note that group objects need not be obtained from CC or a specific Object Source in SD. Since the group key is shared with all group users, the TRM can decrypt the object and display to the user offline without involving the CC every time.

In MD (figure 4), the CC shares a unique key with each group user. In contrast, in SD the same key is shared with all

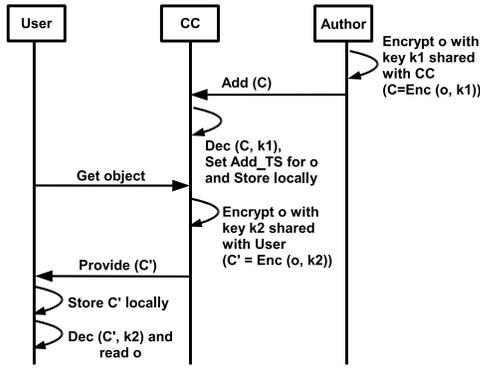


Fig. 4. Micro-distribution in g-SIS.

users in the group. The Author (a group user) creates an object, encrypts the object using the author’s key shared with the CC (mediated by TRM) and sends it to the CC for approval and distribution. The CC approves the object, time-stamps object add and saves it locally. These are steps 2.1 and 2.2 in figure 2 for the MD based approach. When a User requests access to the object, the CC specifically encrypts the object with the key shared with that User. This is a critical difference from SD. Since objects need to be individually encrypted/prepared for each group user, the scalability of the system is gravely affected. Here scalability refers to performance in the context of number of objects shared amongst group users. Since a large number of objects may be shared within the group, the requirement that the CC decrypt and then custom encrypt each object for each group user affects the scalability of MD based enforcement model. In SD, the object is encrypted once and all authorized group users are able to access them without the intervention of the CC. We now discuss how MD differs from that of SD for g-SIS.

- User Join: In SD, the CC provisions the group key for each joining user. In MD, the CC needs to create and share a new key with each joining user. This requires maintenance of a large number of keys.
- Object Add: In MD, the TRM would send the object to the CC. The CC needs to encrypt this object with every other user’s key so that it is accessible to them. This suffers from scalability and performance issues.
- Object Access: In MD, the first time a user needs to access an object, it needs to be obtained from the CC where it is custom encrypted for that user. It can thereafter be accessed offline whenever authorized. Note that every group object needs to be initially obtained from CC in MD. In contrast, the objects could be obtained through any means in SD since all objects are encrypted with the group key.
- Attribute Refresh, User and Object Remove: There is no difference between SD and MD approach for g-SIS.
- Assurance and Recourse: In SD, if any one of the group user’s access machine is compromised, the group key can be exposed and all group objects can be read in plaintext. A new group key can be provisioned after

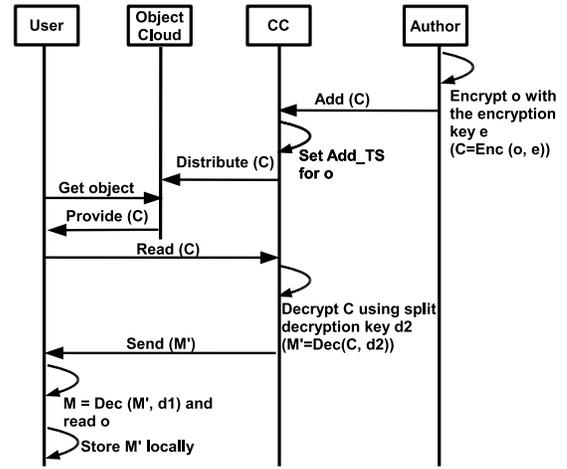


Fig. 5. Hybrid approach in g-SIS.

recovering from the compromise—although this can only guarantee secrecy of new objects that will be created. Due to this single point of failure problem, the sensitivity of information that be can be distributed using SD model may be limited. In MD, if any one access machine is compromised, only objects that were encrypted for the specific user using that access machine will be compromised. Other group objects remain safe because they were encrypted with different keys. In summary, in SD a compromise can result in large-scale access violation while in MD the violation is limited. This is the trade-off between scalability/usability and assurance of each model.

Clearly, the SD based architecture has important scalability and performance benefits of simplified key management (single key per group) and the usability and convenience of offline access and “encrypt once and access where authorized” where group users can quickly share objects with other users. The MD based architecture on the other hand has important failsafe benefits such as limited damage in case of access machine compromise. Note that MD suffers from key management issues regardless of whether symmetric or asymmetric key cryptography is used. For the later, a public-private key pair needs to be shared by the CC with each group user. A more useful and practical architecture should combine the benefits of SD and MD based architecture (thereby minimizing the disadvantages of respective approaches). The hybrid architecture that we discuss in the following section attempts to achieve this by using split-key RSA.

### III. HYBRID APPROACH USING SPLIT-KEY RSA

In this section, we present a hybrid approach that addresses the drawbacks of SD and MD based architecture thereby achieving the benefits of each approach.

#### A. Split-Key RSA

We provide a brief overview of split-key RSA (see [4], [5], [6]) in this section. A proof that the security of split-key RSA

TABLE I  
COMPARISON OF SD, MD AND HYBRID APPROACH IN G-SIS ARCHITECTURE.

Aspect	SD	MD	Hybrid
Key type	Symmetric/Asymmetric	Symmetric/Asymmetric	Asymmetric
Number of encryption keys per group	One	One per group user shared with CC	One
Number of decryption keys per group	One	One per group user shared with CC	One split variant for the same RSA exponent per group user
Add object	Encrypt with group key	Encrypt with key shared with CC	Encrypt with private key
Read object	Decrypt with group key and read (encrypt once and access where authorized). Offline access enabled throughout.	First time, CC custom encrypts the requested object for the user (encrypt differently for each user). Subsequent reads can be carried out offline.	First time, CC decrypts the object with its split decryption key. Subsequent reads can be carried out offline (encrypt once and access where authorized).
Usability (with respect to users)	Very high (offline access, no CC participation).	Medium (To add object, need to encrypt with the key shared with the CC. The CC in turn decrypts and custom encrypts for each user.).	High (Encryption is performed with a uniform encryption key).
Performance (with respect to CC)	Very high (CC never participates in encryption/decryption).	Medium (CC participates in decrypting and custom encrypting each object for each group user).	High (CC performs a one time split key decryption operation per document).
Assurance	Low (compromising one user's access machine exposes group key thereby potentially exposing all group objects).	High (Only objects in the compromised access machine are exposed)	High (Only objects in the compromised access machine exposed).

is equivalent to that of the classical RSA is given in [21]. In split-key RSA, the decryption key is comprised of multiple parts each held by various parties (or users) involved in the decryption process. Thus if  $e$  and  $d$  denote encryption (public) and decryption (private) key respectively,  $d$  can be split into  $n$  parts— $d_1, d_2, \dots, d_n$  and shared with possibly  $n$  different parties. Thus a message encrypted with  $e$  can be decrypted only if all  $n$  parties participate. For simplicity, let us consider only two splits (and thus two parties) for our discussion below.

$$e * d = 1 \text{ mod } \varphi(n) \quad (1)$$

$$d_1 * d_2 = d \text{ mod } \varphi(n) \quad (2)$$

$$C = M^e \text{ mod } n \quad (3)$$

$$(M)^{d_1 d_2} \text{ mod } n = \quad (4)$$

$$(M)^{d_2 d_1} \text{ mod } n =$$

$$(M)^{d_1 * d_2} \text{ mod } n =$$

$$M^d \text{ mod } n$$

In the classical RSA [17], the encryption ( $e$ ) and decryption ( $d$ ) keys for a given  $n$  are related by equation (1). In split-key RSA with two splits,  $d$  can be split into two portions as guided by equation (2). A message  $M$  is encrypted using a single operation as shown in equation (3). Finally, the fundamental operation of exponentiation in RSA is given by equation (4). Thus decrypting a message  $M$  using the split keys  $d_1$  and  $d_2$  in any order is equivalent to decrypting the message using  $d$ . Also, note that  $d$  can be split into two parts in any number of ways, there by yielding pairs of different splits such as  $d_1$  and  $d_2, d_3$  and  $d_4$ , etc. Thus a message encrypted with  $e$  can be decrypted using  $d_1$  and  $d_2$  or  $d_3$  and  $d_4$ , etc.

### B. Hybrid Approach in g-SIS

Figure 5 illustrates the hybrid approach. Split-RSA keys are created for each group. When a user joins a group, the CC creates a unique split decryption key pair ( $d_1$  and  $d_2$  for this user), keeps one split ( $d_2$ ) and shares the other decryption split key ( $d_1$ ) with the joining user. The CC also shares the same encryption key ( $e$ ) with every joining user. Thus, in figure 5, the Author (a group user) adds an object by encrypting it with  $e$  and sending it to the CC. The CC approves the object, sets the add time-stamp and releases it into the object cloud similar to SD. The first time other group users need to access the object, they send a request to the CC. The CC performs the decryption on the object using its split decryption key  $d_2$  and sends it back to the user. The user then decrypts this blob using his/her split-key  $d_1$  (mediated by TRM) to get the final plain-text object. The blob from CC can be stored locally and future read accesses can be performed completely offline. Note that, in practice, a symmetric object key would be used and the split-key decryption operation will be carried out on the object key instead of the entire object to minimize performance penalty of asymmetric key operation. Table I summarizes and compares g-SIS architectures based on SD, MD and hybrid approaches. Clearly, the hybrid approach for g-SIS architecture combines the advantages of both SD and MD approach and minimizes their disadvantages.

## IV. RELATED WORK AND CONCLUSION

We presented a flexible g-SIS architecture that supports SD, MD and a hybrid approach for object distribution and offline access. The g-SIS problem was introduced in [11], [9], [10] and a discussion on SD and MD and TPM based protocols

for the SD approach were presented in [8]. However, the SD and MD approach suffers from various practical issues. In this paper, we addressed the limitations of SD and MD by using a novel hybrid approach using split-key RSA.

Solution approaches for SD using specialized hardware or purely software-based architectures for Digital Rights Management of entertainment content are plentiful in literature ([1], [16], [22]). Mori et al [13] was the first to propose the concept of SD—although it was applied to widespread distribution of software, the usage of which needs to be authorized and monitored using specialized hardware. In [15], the authors classify security architectures for information dissemination based on three major factors: *virtual machine* that employs control functions on objects to be accessed, *control set* that specify the access rights and usage rules and *distribution style* which specifies whether the objects are pushed to users or the users obtain them from an external repository. An SD infrastructure for secure storage, retrieval and use of copyrighted data in mobile devices can be found in [7]. The solution is based on secure multimedia card extensions for mobile phones. A classification and analysis of current content distribution networks and technologies (called Darknet) can be found in [3]. Our work is different in that it focuses on a generic g-SIS enforcement model for enabling SD, MD and a hybrid approach that maximizes the benefits of SD and MD.

#### V. ACKNOWLEDGMENTS

Ravi Sandhu is partially supported by NSF grant IIS-0814027, AFOSR MURI award FA9550-08-1-0265 and grants from the State of Texas Emerging Technology Fund and Intel Corporation. Ram Krishnan is partially supported by a grant from Intel Corporation.

#### REFERENCES

- [1] Architecture of windows media rights management. <http://www.microsoft.com/windows/windowsmedia/drm/default.aspx>.
- [2] TCG Specification Architecture Overview. <http://www.trustedcomputinggroup.org>.
- [3] P. Biddle, P. England, M. Peinado, and B. Willman. The Darknet and the future of content distribution. *Proc. of ACM Work. on DRM*, 2002.
- [4] C. Boyd. Digital multisignatures. In *Cryptography and Coding*, pages 241–246, Oxford University Press, 1989.
- [5] R. Ganesan. Yaksha: Augmenting Kerberos with public key cryptography. In *Proc. of the Symp. on Network and Dist. Syst. Security*, 1995.
- [6] R. Ganesan. *Yaksha: Towards reusable security infrastructures*. PhD Thesis, Johns Hopkins University, 1996.
- [7] T. Hatakeyama, H. Maruyama, and T. Chiba. Web computing. Superdistribution and the security of music content. *Fujitsu*, 2001.
- [8] R. Krishnan and R. Sandhu. Enforcement Architecture and Implementation Model for Group-Centric Information Sharing. *Proc. of 1st International Workshop on Security and Comm. Networks*, 2009.
- [9] R. Krishnan, R. Sandhu, J. Niu, and W. Winsborough. A conceptual framework for group-centric secure information sharing. *Proc. of 4th ACM Symposium on Information, Computer and Comm. Security*, 2009.
- [10] R. Krishnan, R. Sandhu, J. Niu, and W. Winsborough. Foundations for group-centric secure information sharing models. *Proc. of 14th ACM Symposium on Access Control Models and Technologies*, 2009.
- [11] R. Krishnan, R. Sandhu, and K. Ranganathan. PEI models towards scalable, usable and high-assurance information sharing. *Proc. of 12th ACM Symposium on Access Control Models and Technologies*, 2007.
- [12] U. Kühn, M. Selhorst, and C. Stübke. Realizing property-based attestation and sealing with commonly available hard- and software. In *Proc. of ACM workshop on Scalable trusted computing*, 2007.
- [13] R. Mori and M. Kawahara. Superdistribution: The concept and the architecture. *The Transactions of the IEICE*, 73(7):1133–1146, 1990.
- [14] A. Osterhues, A. R. Sadeghi, M. Wolf, C. Stübke, and N. Asokan. Securing Peer-to-peer Distributions for Mobile Devices. In *4th Information Security Practice and Experience Conference*, 2008.
- [15] J. Park, R. Sandhu, and J. Schifalacqua. Security architectures for controlled digital information dissemination. *ACSAC*, 2000.
- [16] B. Popescu, B. Crispo, A. Tanenbaum, and F. Kamperman. A DRM security architecture for home networks. *Proceedings of the 4th ACM workshop on Digital rights management*, pages 1–10, 2004.
- [17] R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21:120–126, 1978.
- [18] A. Sadeghi, M. Scheibel, C. Stübke, and M. Wolf. Play it once again, Sam—Enforcing stateful licenses on open platforms. In *2nd Workshop on Advances in Trusted Computing*, 2006.
- [19] R. Sailer, X. Zhang, T. Jaeger, and L. van Doorn. Design and implementation of a TCG-based integrity measurement architecture. *Proc. of the 13th USENIX Security Symposium*, 13:16–16, 2004.
- [20] R. Sandhu. Engineering authority and trust in cyberspace: the OM-AM and RBAC way. In *Proc. of 5th ACM workshop on RBAC*, 2000.
- [21] R. Sandhu, M. Bellare, and R. Ganesan. Password-enabled PKI: Virtual smartcards versus virtual soft tokens. In *Proceedings of the 1st Annual PKI Research Workshop*, 2002.
- [22] S. Sovio, N. Asokan, and K. Nyberg. Defining authorization domains using virtual devices. *Symposium on Applications and the Internet Workshops*, pages 331–336, 2003.