Secure Group Key Management for Storage Area Networks

Yongdae Kim, University of Minnesota — Twin Cities; Fabio Maino, Andiamo Systems; Maithili Narasimha, University of California Irvine; Kyunghyun Rhee, Pukyung University; Gene Tsudik, University of California Irvine

An earlier version of this article was presented at [1]

This research was supported in part by DISC (DTC Intelligent Storage Consortium), at University of Minnesota — Twin Cities

ABSTRACT

Storage area networks offer high availability, reliability, and scalability, and are a promising solution for large-scale storage needs of many enterprises. As with any distributed storage system, a major design challenge for SANs is to provide secure storage, which implies data integrity and data confidentiality. In this article we propose a solution that addresses these core security requirements. In particular, we focus on mechanisms that enable efficient key management for SAN entities and allow scalable data sharing. We use strong cryptographic techniques to achieve data security and integrity. Further, we delegate the bulk of the cryptographic processing to the SAN entities, thereby removing bottlenecks at disks and causing minimal inconvenience to hosts. By recognizing the peer nature of the group of SAN entities, we propose a novel security architecture for SAN that uses a secure group communication protocol to provide efficient group keying without involving any centralized servers. This fosters both scalability and fault tolerance.

INTRODUCTION

Continued growth and popularity of the Internet fuels increased reliance on e-business, which often involves data-intensive applications. The amount of information that needs to be stored and managed can become quite intimidating. Traditional centralized servers with SCSI interfaces to peripheral storage devices, which have been the workhorses of the industry, are often unable to meet the storage needs of large organizations. To this end, they are being replaced by network attached disks and, more recently, by *storage area networks* (SANs). SANs provide efficient any-to-any connectivity between hosts and storage devices, and represent a major step in the evolution of network storage.

A critical requirement in any distributed (e.g., storage) system is the security of stored data, which, for the most part, implies data integrity and data privacy. Although, as discussed below, this has been studied intensively in the past, certain unique features of the SAN setting result in some new security challenges.

PRIOR WORK

Security of storage systems has been an area of active research for well over a decade now. Several systems have been proposed and analyzed (e.g., CFS [2], Cepheus [3], and NASD [4]). Prior results vary widely with respect to trust assumptions and security primitives/services offered. For example, one of the earliest systems, Cryptographic File System (CFS) [2], is tailored toward single-user workstations and relies on user-supplied passwords for data encryption. In contrast, Network-Attached Secure Disks (NASD) [4] proposes a distributed system comprising intelligent disks and uses user supplied keys as proofs of authorization.

We partition previously proposed secure storage systems into:

- Those focusing on protecting data in transit
- Those attempting to safeguard data while it is stored on disk
- Those providing end-to-end protection (on both wire and disk)

Approaches where the underlying storage server is trusted (e.g., NASD and SFS [5]) focus mainly on securing network traffic and preventing *outsider* attacks. Other approaches like Cepheus and SNAD [6] do not trust the storage servers and therefore propose security measures to protect data in transit as well as on disk. We follow the latter approach and aim to provide both on-wire and on-disk data protection.

Many of these storage systems provide mechanisms for efficient group sharing of data. In other words, identical data access permissions are given to groups of users, and any user who can prove group membership is authorized to access data based on the group permissions. Group sharing reduces the total number of keys to be stored and distributed in the system. These group keys are typically used to secure the symmetric keys used for data encryption (e.g., the notion of a *group lockbox* in Cepheus). Systems such as SNAD and NASD rely on centralized

0163-6804/03/$17.00 © 2003 IEEE

group servers to distribute these group keys. Although a centralized server simplifies key distribution, it is a single point of failure and represents an enticing target for attacks. (In contrast, our approach does not require any centralized entities while providing efficient group key management.)

Focus

In this work we concentrate on safeguarding data (stored on a SAN) from various threats and attacks with a further emphasis on efficient key management. Specifically, we propose a security architecture for preserving privacy and integrity of SAN data. We use on-disk as well as on-wire encryption to protect data from unauthorized insiders and malicious outsiders. Only authorized SAN entities and hosts (through these authorized SAN entities) can gain access to the unencrypted disk contents. We employ cryptographically secure hashing and digital signatures to provide data integrity. Our approach, in addition to providing strong security, offers good systemwide performance since the bulk of the cryptographic operations are relegated to SAN entities, which are typically equipped with powerful processors and/or hardware acceleration to support wire-speed encryption.

Organization — The rest of this article is organized as follows. We describe the details of our system model and trust assumptions, and specify our design goals in the following section. We then introduce the cryptographic primitives used in the article. The actual architecture is described next. The security and performance of the proposed system are compared with those of a popular variant, and we conclude in the final section.

System Model

A *storage area network* can be viewed as a collection of SAN entities (e.g., switches, routers, and other network elements) connecting hosts to remote disks. As seen from the perspective of a host or a disk, a SAN is a network infrastructure that forwards, in an efficient and reliable way, both data blocks and commands required to retrieve and store these data blocks on disks (Fig. 1).

In most SAN frameworks the actual data owner can control, fully or in part, the SAN administration. This is the case in a typical enterprise scenario as well as in the storage service provider (SSP) model (where SSP companies sell storage as a service to their customers).

In fact, in an enterprise scenario, user data is protected according to the security policy established by the enterprise and enforced within the enterprise SAN. As an example, data availability is ensured by applying a backup policy that is enforced in the SAN itself, without user intervention (e.g., disks where user data are stored are mirrored, replicated, or backed up under the control of the SAN administrators, not of the users). In the same way, data integrity and privacy should also be ensured and enforced by the SAN administrators, in accordance with the security policy of the enterprise.

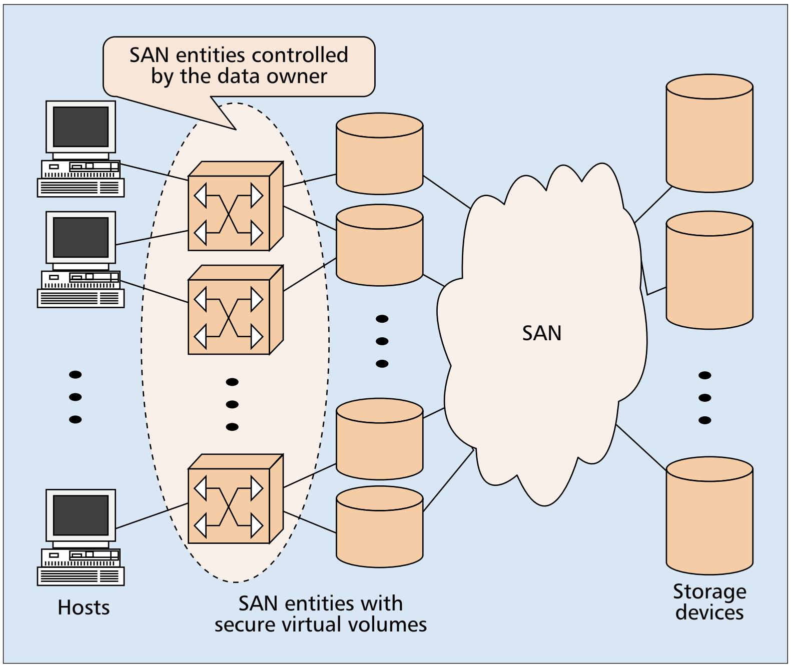
SAN entities controlled by the data owner
SAN
Hosts
SAN entities with secure virtual volumes
Storage devices

Figure 1. *System architecture.*

Even in the SSP model, the SAN is at least partially under control of the data owner. The SAN entities located on customer premises (that provide connectivity with the provider's part of the SAN) are, in fact, managed and controlled by the customer's administrators.

The fact that the data owner controls the SAN enables the powerful concept of virtualization for data security: the SAN entities can actively enforce data security policies by encrypting and decrypting on-the-fly blocks of data that are written to, or read from, the storage subsystem. In practice, the host sees the remote disk as a secure virtual volume with security attributes transparently provided by the SAN. The mapping between the secure virtual volume, the physical disk(s) where data is actually stored, and the security parameters/transforms applied to the data is performed by the SAN entities controlled by the data owner.

Since the SAN entities are responsible for active enforcement of data security, there is a need to effectively protect them from unauthorized access on their management interface. This is a well-known problem addressed by management architectures and beyond scope of this article. We assume that management access to the SAN entities is governed by a strong access control mechanism, ensuring that only authorized storage administrators can modify the configuration of parameters for a secure virtual volume.

System Events

Events that take place in our model are summarized below.

Initialization: A storage administrator, through a management action in one of the SAN entities, creates a new secure virtual volume mapped over one or more physical disks. The initial encryption keys for the volume are chosen.

Disk access: A disk read or disk write event is triggered when a host accesses a secure virtual disk through a SAN entity.

Key update: This occurs when the encryption key must be changed.[1] This can be prompted by:
- Compromised key(s)
- A compromised SAN entity
- Periodic key refresh

We use the term *compromised SAN entity* to describe an entity that has been removed from the SAN for one of the following reasons: changes in network topology, reallocation of SAN resources for performance optimization or administrative reasons, or because the SAN entity was subverted. We assume that it is possible to detect subverted SAN entities (e.g., by using an intrusion detection system) and propagate this information to other SAN elements.

SAN entity join: This is triggered when a secure virtual volume is instantiated for the first time by a new SAN entity (e.g., because a host connected to that SAN entity attempts to access that volume).

We assume that the storage subsystem (essentially, the set of disk arrays) is not intrinsically trusted by the data owner, whereas, at least some of the SAN entities are. In other words, the data owner controls the hosts and part of the SAN but not necessarily the physical disk array.

Assumptions and Scope — We assume that the storage subsystem (essentially the set of disk arrays) is not intrinsically trusted by the data owner, while at least some of the SAN entities are. In other words, the data owner controls the hosts and part of the SAN, but not necessarily the physical disk array. There are many solutions for authentication and authorization between the host and the SAN (e.g., the iSCSI [7] architecture proposes a password-based approach to authentication); we do not address these issues here.

We provide security at the SCSI block level. File-level encryption and access control are not dealt with here. Furthermore, we note that the use of block-level encryption is independent of the physical organization of the storage subsystem. Thus, data redundancy and high availability can be provided by the usual approach followed on disk arrays (it may be structured with RAID organization that better addresses the requirement in terms of resiliency to catastrophic failures of the disk drives). For example, a secure virtual volume may be physically mapped over a disk array organized as a RAID 5 storage subsystem without compromising robustness. We note that even the backup strategy is completely unaffected by block-level encryption, since traditional backup strategies can be applied to the physical disks over which a secure virtual volume is mapped.

Other aspects beyond the scope of this article pertain to the actual mechanisms used to provide block-level integrity and encryption for a secure virtual volume. Many well-known block encryption and data integrity protection algorithms can be applied. Furthermore, any write operation must be authenticated (e.g., key information should not be modified by unauthorized entities). This article will not address the details of the authentication mechanisms between the SAN and the storage subsystem.

Cryptographic Building Blocks

In order to support secure sharing of data among a group of SAN entities without relying on any centralized authority, public key cryptography is a natural choice due to its simple key management. At the same time, the use of public key cryptography must be minimized because of its relatively high cost. Therefore, a two-tiered approach is often used: bulk data is encrypted using a fast symmetric cipher, and the symmetric encryption keys are themselves encrypted under the public keys of all authorized SAN entities. (We will compare this simple approach with our approach later). One very viable alternative is to encrypt symmetric (bulk data encryption) keys under a single group key known only to all authorized SAN entities. This can be achieved through the use of a secure group key agreement mechanism [8].

Group Key Agreement

Group key agreement [9] is a process whereby a shared secret key is jointly computed by a group of users.[2] Fundamental properties of group key agreement include:
- A group key is computed using key contributions randomly and uniformly chosen by each group member.
- No information about the group key can be obtained by observing the group key agreement protocol without knowledge of at least one of the key contributions.
- All key contributions are kept secret; if a member is honest, even if all other parties collude, they cannot extract any information about the secret contribution of the member from their combined view of the protocol.

Many group key agreement protocols support only secure group creation, that is, they enable a static group of users to compute a shared key. Recently, some of these protocols have been extended to handle group membership changes (GDH [8], STR [11] and TGDH [9]). We are particularly interested in the following features:
- **Condition 1**: Provide efficient mechanisms for member join and member evict events (in order to add a new member or expel a member).
- **Condition 2**: Do not require all current group members' contribution for join and evict operations (since all members may not be available/active at a given time).

We do not consider STR since it is quite inefficient in handling group eviction events, requiring, at worst, a number of exponentiations linear to the total number of members. GDH is equally expensive for both evict and join events. (Recent results on the performance of practical group key agreement protocols can be found in Amir *et al.* [12].) Therefore, we focus on the TGDH protocol, which requires, on average, only a logarithmic number of modular exponentiations to handle any group event.

TGDH Protocol

TGDH is a group key agreement technique combining Diffie-Hellman key exchange [13] with key trees. It implements fully distributed contributory group key agreement and handles key adjustments due to group membership changes and periodic rekeying needs. A group key is derived from the individual contributions of all

[1] *We stress that key update is an* atomic *operation. Consequently, any failure of this operation will cause the system to revert back to a previously known* stable *state.*

[2] *A group* key distribution *[10] mechanism also enables a group of users to share a secret key. However, it requires a centralized server, which is a potential single point of failure or corruption.*

group members using a binary key tree. TGDH assumes reliable communication among group members for protocol correctness and fault tolerance (but not for security). In the SAN setting, however, instead of counting on the presence of a reliable group communication system, we use shared storage to maintain the group key tree. Below, we describe the protocol in detail. (A detailed description of TGDH appears in Kim *et al.* [9].)

A group key tree is organized in the following manner: each node $\langle l, v\rangle$ is associated with a key $K(l, v)$ and a corresponding blinded key (bkey) $BK\langle l, v\rangle = g^{K}\langle l, v\rangle \bmod p$ where g is a generator of a subgroup of $\mathbb{Z}_p^*$ and p is a large prime. Node $\langle l, v\rangle$'s left and right children are denoted by $\langle l + 1, 2v\rangle$ and $\langle l + 1, 2v + l\rangle$, respectively. This shared key-tree includes **only blinded keys**. Note that it is computationally infeasible to derive a key from a blinded key. All keys, including the root key and the members' individual contributions, are private to each member.

Figure 2 shows a key tree example. The key at the root node is the secret group key shared by all members, and a key at the leaf node is the member's contribution. (Each leaf node is associated with a distinct member M_i). Every member knows its own contribution (i.e., key associated with its leaf node) as well as all bkeys on the key tree. Using this knowledge, a member can compute all other keys on the path from its leaf node to the root. Each key $K\langle l, v\rangle$ is computed recursively as follows:

$$
\begin{aligned}
&\left(BK_{\langle l+1,2v+1\rangle}\right)^{K_{\langle l+1,2v\rangle}} \bmod p \\
K_{\langle l,v\rangle} = &\left(BK_{\langle l+1,2v\rangle}\right)^{K_{\langle l+1,2v+1\rangle}} \bmod p \\
&g^{K_{\langle l+1,2v\rangle}\ K_{\langle l+1,2v+1\rangle}} \bmod p
\end{aligned}
$$

Clearly, computing a key at $\langle l, v\rangle$ requires knowledge of the key for one of the two children and the bkey of the other. For example, in Fig. 2, M_2 can compute $K_{\langle 2,0\rangle}$, $K_{\langle 1,0\rangle}$, and the group key $K_{\langle 0,0\rangle}$ (in that order) using its own contribution $K_{\langle 3,1\rangle}$ and the blinded keys $BK_{\langle 3,0\rangle}$, $BK_{\langle 2,1\rangle}$, and $BK_{\langle 1,1\rangle}$ as: $K_{\langle 2,0\rangle} = BK_{\langle 3,0\rangle}{}^{K\langle 3,1\rangle} \bmod p$, $K_{\langle 1,0\rangle} = BK_{\langle 2,1\rangle}{}^{K\langle 2,0\rangle} \bmod p$, and $K_{\langle 0,0\rangle} = BK_{\langle 1,1\rangle}{}^{K\langle 1,0\rangle} \bmod p$. Following each group membership change, a particular member (called a sponsor) recomputes all affected keys and bkeys and updates the shared key tree file.[3] Note that the role of a sponsor is unique to each membership event.

In general, the insertion point for a join event is the shallowest rightmost node, where the join would not increase the height of the key tree. Otherwise, if the key tree is fully balanced, the new member node is joined to the root. The sponsor is the rightmost leaf node in the subtree rooted at the insertion node. Figure 3 depicts the join protocol.

Figure 4 shows an example where M_4 joins a group and the sponsor (M_3) performs the following actions:

1 Renames node $\langle 1,1\rangle$ to $\langle 2,2\rangle$
2 Creates new intermediate node $\langle 1,1\rangle$ and new leaf node $\langle 2,3\rangle$
3 Promotes $\langle 1,1\rangle$ as parent node of $\langle 2,2\rangle$ and $\langle 2,3\rangle$

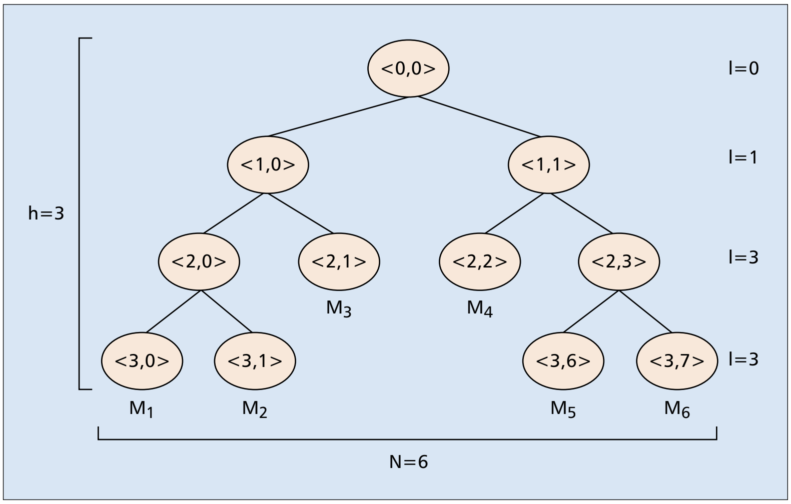

Figure 2. *TGDH key tree notation.*

Since $BK_{\langle 2,3\rangle}$ and $BK_{\langle 1,0\rangle}$ are publicly known, M_3 can compute the new group key $K_{\langle 0,0\rangle}$. Finally, M_3 saves the updated tree $\widehat{T}_s$, only with the bkeys, on the shared storage. (Note that since the updated tree is saved on the shared storage, other members can subsequently compute the new group key.)

Member eviction is similar to a join. The key tree is updated by deleting the leaf corresponding to the evicted member (M_d). The former sibling of M_d is promoted to replace M_d's parent node. The member performing the update (sponsor) computes all [*key*, *bkey*] pairs on the path up to the root, and reveals the updated key tree (containing a new set of bkeys) to the rest of the group. Thereafter, only current members can compute the new group key; equivalently, outsiders (including evicted former members) cannot compute subsequent group keys. The sponsor is always determined as the rightmost leaf of the subtree rooted at the evicted member's sibling.

TGDH protocol is proven secure under the well-known Decision Diffie-Hellman (DDH) assumption [14]. For more details of the proof and the actual protocol, please refer to Kim *et al.* [15].

TGDH Protocol for the SAN Environment

To the best of our knowledge, this is the first effort aimed at applying a group key agreement protocol in the SAN setting. In order to achieve this goal, however, we need to adapt the TGDH protocol for the SAN environment. In its original form, TGDH relies on the presence of a reliable group communication system to notify the group of all membership changes, and to provide reliable and sequenced protocol message delivery. However, in the present environment, two (related) issues arise:

- Reliable group communication requires constant online presence of all current members.
- Network partitions and congestion may cause membership changes; that is, group

[3] *We stress, once again, that the shared file contains only bkeys.*

Step 1: New member chooses a random contribution K_{n+1} and broadcasts request for join along with its bkey.

$$M_{n+1} \xrightarrow{BK_{(0,0)}=g^{K_{n+1}}} \{M_1,\ldots M_n\}$$

Step 2: Sponsor member

- updates key-tree by adding a new member node and a new intermideate node,
- computes all [*bkey*, *key*] pairs on the path from node associated with it to the root,
- saves the update tree $\hat{T}_s$ only with bkeys on the shared file.

Figure 3. *Join protocol.*

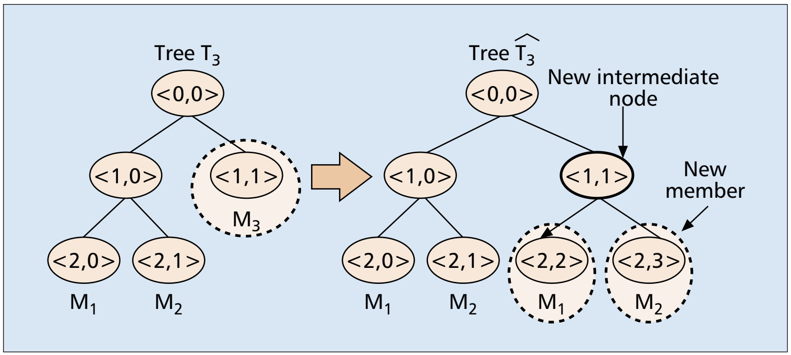

Figure 4. *Tree update: join.*

membership is dependent on the state of the network.

In a SAN, membership in a group of SAN entities is a long-term concept and should not be influenced by *short-term* network perturbations. Therefore, group membership changes should not be triggered by the instantaneous reachability or availability of members, but instead by explicit and infrequent events such as a new member being introduced into the group (join) or a current member being expelled (eviction) from the group.

In light of the above, some TGDH heuristics need to be amended. In particular, sponsor selection needs to change since constant online presence of sponsors (as defined above) cannot be assumed. Details of these changes can be found in [1].

Architecture

A trivial solution to safeguard data is to encrypt all data in a virtual disk with a single key and ensure that only authorized SAN entities know this key. Additionally, in order to maintain data integrity, digital signatures can be used. This solution is simple to implement, and the associated storage overhead is minimal. However, it is clearly impractical, since changing the key would require re-encryption of the entire virtual disk data, which can be very expensive.

One straightforward performance enhancement is to divide a virtual disk into multiple logical segments or *encrypted data units* (EDUs).[4] Data in each EDU can be encrypted with a separate key, and a key change operation would cause only the data in the relevant EDU(s) to be re-encrypted. The size of individual EDUs and therefore the total number of EDUs on a disk can be either fixed or variable. To simplify things, we assume that EDU size is a global parameter that is fixed and enforced by SAN administrators. Clearly, the choice of EDU size affects encryption granularity. In other words, the number of data encryption keys per virtual volume is determined by the number of EDUs. While we acknowledge that choosing the optimal EDU size is an important issue that affects the overall performance of the system, we do not address the details in this work.

In essence, this approach of encrypting data in each EDU with a separate key results in several data encryption keys for a given virtual disk. These keys need to be shared among the group of SAN entities authorized to virtualize that disk. Additionally, any authorized SAN entity should be allowed to unilaterally change an EDU-specific key. To avoid explicit communication of key updates, all EDU keys for a particular virtual disk should be stored on that disk. Of course, these keys are themselves encrypted to enable seamless retrieval by the authorized SAN entities.

Our approach is to use a single *master key* to encrypt individual EDU keys. All SAN entities authorized to virtualize a volume can be viewed as a peer group, and the master key used to encrypt individual EDU keys can be shared by the members of this peer group. The EDU keys are stored in a *key lockbox* (similar to the group lock box concept suggested in Cepheus [3]) secured by the master key, and the master key itself is securely shared by all group members. However, unlike Cepheus, our approach involves each authorized SAN entity taking part in the generation of the master key, without requiring any external key distribution server. (This is discussed in the next section.)

The key management problem is now essentially reduced to sharing the master key between the group members. Before going into the details of our proposal, we describe the fundamental components (building blocks) of a virtual volume taking into account the security-related information (EDU keys, lockboxes, and master keys) that will be stored along with encrypted data.

System Components

Figure 5 depicts a encrypted virtual volume and also shows the hierarchical key structure. A virtual disk, as the figure illustrates, contains three basic objects: *EDUs*, *key lockbox,* and *master key component*.

EDUs — An EDU is a container for encrypted data segment. A virtual disk includes one or more EDUs. The EDU id uniquely identifies an EDU on a virtual disk. Each EDU contains multiple 512-byte blocks of data encrypted under a symmetric encryption algorithm with a single key. Each EDU also stores a pointer to the location inside the key lockbox where its encryption

[4] *This will require storing additional meta-data corresponding to each EDU (a pointer holding the beginning address and a length field denoting the size of the EDU) on the disk.*

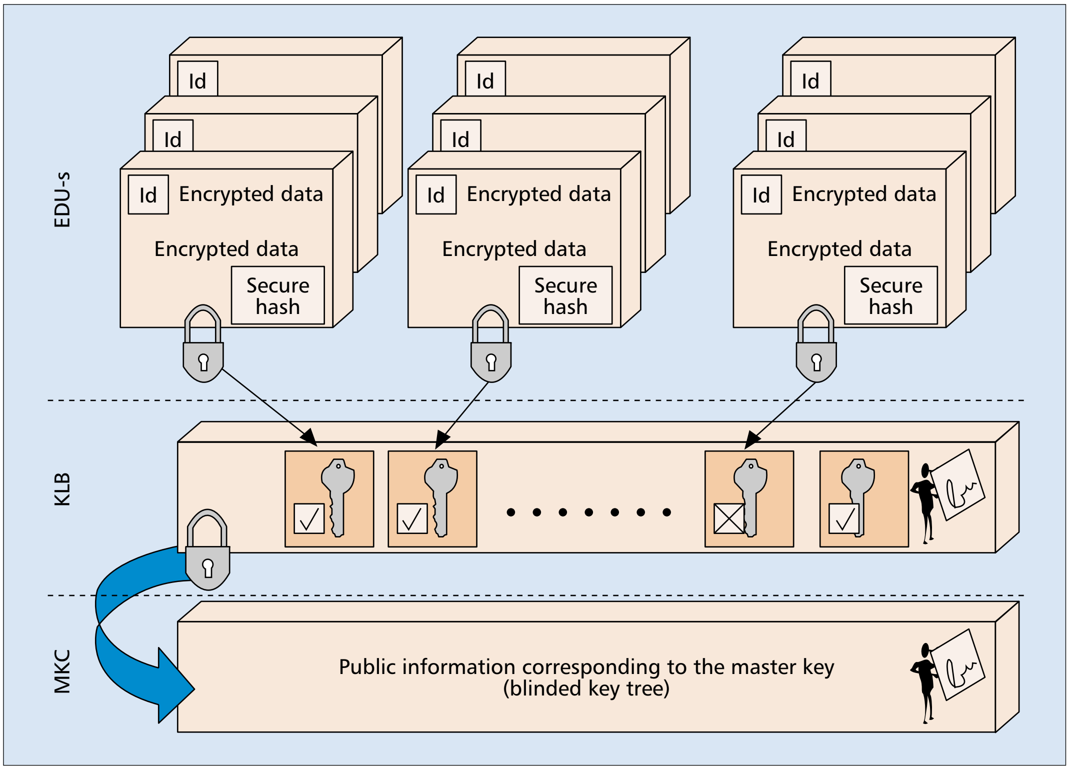

Figure 5. *Secure virtual volume.*

Our approach for sharing this Master key among all group members is based on group key agreement. Consequently, instead of relying on any one SAN entity to choose the Master key, the SAN entities jointly compute a Group Key that has contributions from all members.

key is stored. Finally, every EDU contains a secure checksum (i.e., a keyed hash of the cleartext data[5]).

Key Lockbox — Every virtual volume has a key lockbox (KLB) component that stores encrypted EDU keys. (As stated above, a master key unique to each virtual volume is used to encrypt individual EDU keys.) The KLB also stores a validity field for each EDU that denotes the *compromise* status of the data in the EDU. In other words, when the data has been read or written by a potentially compromised SAN entity, the value of this field can be set to indicate that the EDU needs rekeying. The validity field is useful when we employ so-called *lazy re-encryption* discussed below. In order to protect the integrity of its contents, the KLB stores the identity and digital signature of the SAN entity that last modified this object.

Master Key Component — The master key component (referred to as *MKC_GK*) contains the information necessary to retrieve the master key of a secure virtual volume. Specifically, this object stores public information (blinded key tree in TGDH) related to the group key derived from the contributions of all members. Any *current* member can compute the group key (master key) by combining this public information with its own secret contribution, as described earlier. Once again, the signature of the SAN entity that last modified this object is included.

Details

In this section we discuss the *secure* SAN architecture. More concretely, we discuss the details of the master key management scheme. As noted above, introducing a higher-level master key helps key management. Our approach for sharing this master key among all group members is based on group key agreement. Consequently, instead of relying on any one SAN entity to choose the master key, the SAN entities jointly compute a *group key* that has contributions from all members. All *blinded* (public) keys are stored in the MKC GK component of that disk (Fig. 5). Any authorized member can compute the group key by using the blinded keys on the key tree and its own secret contribution. The scheme is briefly explained below.

Virtual Disk Initialization Event — In this event, a new virtual disk is created by the storage administrator with the help of a SAN entity. This SAN entity randomly chooses its private key contribution that is also the initial *group key* (since the group currently has only one member). The SAN entity computes the corresponding public key (*blinded value*) and stores this public information in MKC GK along with its signature.

SAN Entity Join Event — When a new SAN entity attempts to instantiate an existing secure virtual volume, this event is triggered. The new SAN entity initiates the process by sending out a `join_request` to existing group members. One of the members (*sponsor*) updates the group key tree to include the blinded contribution of the new member. This results in a new group key for the virtual volume. The sponsor also updates the KLB by encrypting all EDU keys with the new group key. Finally, the sponsor recomputes the signatures on KLB and MKC GK.

[5] *Note that this key is different from an encryption key. In general, we can derive two keys from the same secret by applying distinct one-way functions to the secret.*

Our approach for sharing this Master key among all group members is based on group key agreement. Consequently, instead of relying on any one SAN entity to choose the Master key, the SAN entities jointly compute a Group Key that has contributions from all members.

Key Update Event — As mentioned earlier, a key update event may be triggered because of a compromised key, a compromised SAN entity or periodically when the key needs to be refreshed. These situations are dealt with differently as explained below.

EDU key refresh event: To refresh an EDU key, the SAN entity handling the event chooses a new data encryption key; encrypts the EDU data with the new key, and updates the KLB by encrypting the EDU key with the group key. Additionally, it sets the *validity* field corresponding to that EDU in KLB to indicate that the EDU key is valid. The SAN entity also generates a new signature on KLB.

Group key refresh event: A SAN entity refreshes the group key[6] by changing its secret contribution and updating the blinded key tree to reflect its new contribution, thereby changing the group key (see above). It also re-encrypts all the EDU keys in KLB with the new group key, and generates new signatures for KLB and MKC GK.

Key update due to key compromise: In the event of a EDU key compromise, the key needs to be changed, and the affected data needs to be re-encrypted by the SAN entity handling that event. This event handling is similar to a EDU key refresh event as described above.

Key update due to SAN entity compromise: When a group member is evicted, effectively the group key is compromised. This implies that the data on the entire virtual disk is compromised. Handling this event requires the group key and all individual EDU keys to be changed and the entire disk data to be re-encrypted. Since this can be a very expensive operation, we employ a *lazy re-encryption* mechanism: the SAN entity handling the event changes its private contribution; deletes the leaf node corresponding to the evicted member from the group tree and recomputes the new group key (see above for the evict protocol). Additionally, this SAN entity updates the KLB by encrypting all EDU keys with the new group key and sets the validity field value for every EDU in KLB as compromised. The EDU key is changed, and the data is re-encrypted subsequently during a disk access or a key refresh event on that EDU.

Disk Access Event — When an authorized SAN entity (group member) wants to access an EDU of a secure virtual volume: it first obtains the blinded key tree information from MKC GK; using this information and its own private contribution computes the group key; using the group key, unlocks the KLB to obtain the encryption key for the EDU. The SAN entity should change the encryption key during a disk access event if the validity field corresponding to that EDU is set as compromised. (Encryption key can be changed by triggering an EDU key refresh event).

Discussion

In this section we compare the security and efficiency of two key management approaches: ours and another approach that achieves roughly the same goals.[7] In the latter, the master key for a virtual disk is simply chosen by one of the SAN entities authorized to virtualize that volume. All EDU keys are then encrypted with the master key and stored in a KLB. The master key, in turn, is encrypted individually for all other authorized SAN entities using standard public key encryption. This is referred to as the *encrypted master key* approach, while our proposal (as described earlier) is referred to as the *shared group key* approach.

Security

One of the major differences in the two approaches concerns the master key. In the encrypted master key approach, the master key is chosen by a single SAN entity, whereas in the shared group key approach, the group key is determined by contributions from all SAN entities that have currently instantiated that virtual volume. Hence, in the shared group key approach, the randomness of the master key does not depend on the ability of a "single" SAN entity to choose cryptographically strong random keys. Additionally, in the encrypted master key approach, the master key is encrypted using each SAN entity's public key, whereas in the Shared group key approach, no such long term keys are used (since the group key and the individual contributions are periodically refreshed).

Another important distinction between the two approaches, from a security perspective, pertains to the shared group key approach's ability to provide *forward and backward secrecy* (and therefore *key independence*) (For details see [9]). In order to get the same level of security in the encrypted master key approach, an explicit master key update event needs to be triggered every time the group membership changes.

Efficiency

Table 1 compares the two approaches discussed in the previous section with respect to the number of modular exponentiations required to handle the basic system events. The cost of each modular exponentiation differs depending on the public key encryption algorithm. Therefore, in case of the encrypted master key approach (called PK in the table), we counted the number of public key operations (encryption or decryption). The cost of the shared group key approach (called GKA) is more straightforward: only pure modular exponentiations are considered. These operations are performed by the SAN entity handling an event. The count does not include signature generation (since same number of digital signatures are used in both the approaches). In Table 1, k denotes the total number of SAN entities that have instantiated a given virtual volume. We assume that the average height of the TGDH key tree is $\log k$. Please refer to [1] for a detailed description of the comparative cost analysis.

Conclusions

In this article we propose a security architecture for preserving privacy and integrity of SAN data with an additional emphasis on secure and efficient key management. In our approach, SAN entities are responsible for active enforcement of

[6] *The group key, like any other key, is subject to aging and prudent security practices require that it should be periodically refreshed.*

[7] *Refer to [1] for further details of this approach.*

			PK	GKA
Modular exponentiations	Initialization event		1	1
	Key update event	Periodic refresh	k	2 log k
		EDU key compromise	0	0
		SAN entity compromise	$k - 1$	2 log k
	Disk access event	Data read (normal)	1	log k
		Data read (cached key)	0	0
		Data write (normal)	1	log k
		Data write (cached key)	0	0
	SAN Entity join Event		$1 + k$	2 log k
	Memory requirement		k * (*keysize*)	2k – 1) * (*keysize*)

Table 1. *Cost comparisons.*

security. Our scheme utilizes the nascent computing power of the SAN entities to carry out computationally intensive cryptographic functions. We specifically address the key management problem. Exploiting the peer group nature of the SAN entities virtualizing a secure disk, we present a mechanism to enable key sharing that does not require any centralized servers.

In our approach, SAN entities are responsible for active enforcement of security. Our scheme utilizes the nascent computing power of the SAN entities to carry out computationally intensive cryptographic functions.

References

[1] Y. Kim *et al.*, "Secure Group Services for Storage Area Networks," *1st Int'l. IEEE Sec. in Storage Wksp.*, Dec. 2002, pp. 80–93.

[2] M. Blaze, "A Cryptographic File System for Unix," *1st ACM Conf. Comp. and Commun. Sec.*, Nov. 1993, pp. 9–15.

[3] K. Fu, "Group Sharing and Random Access in CryptoGraphic Storage File Systems," Master's thesis, MIT, 1999.

[4] H. Gobioff, "Security for a High Performance Commodity Storage Subsystem," Ph.D. thesis, Carnegie Mellon Univ., 1999.

[5] D. Mazieres *et al.*, "Separating Key Management from File System Security," *Proc. 17th ACM Symp. Op. Sys. Principles*, Dec. 1999, pp. 124–39.

[6] E. Miller *et al.*, "Strong Security for Network-Attached Storage," *Conf. File and Storage Tech.*, Jan. 2002, pp. 1–13.

[7] M. Krueger *et al.*, "Small Computer Systems Interface protocol over the Internet(iSCSI) Requirements and Design Considerations," IETF RFC 3347, July 2002.

[8] M. Steiner, G. Tsudik, and M. Waidner, "Key Agreement in Dynamic Peer Groups," *IEEE Trans. Parallel and Distrib. Sys.*, vol. 11, Aug. 2000, pp. 769–80.

[9] Y. Kim, A. Perrig, and G. Tsudik, "Simple and Fault-tolerant Key Agreement for Dynamic Collaborative Groups," *7th ACM Conf. Comp. and Commun. S.*, Athens, Greece, Nov. 2000, pp. 235–44.

[10] C. K. Wong, M. G. Gouda, and S. S. Lam, "Secure Group Communications using Key Graphs," *Proc. ACM SIGCOMM '98 Conf. Apps., Tech., Architectures, and Protocols for Comp. Commun.*, pp. 68–79, 1998; Appeared in *ACM SIGCOMM Comp. Commun. Rev.*, vol. 28, no. 4, Oct. 1998.

[11] Y. Kim, A. Perrig, and G. Tsudik, "Communication-efficient Group Key Agreement," Information Systems Security, *Proc. 17th Int'l. Info. Sec. Conf.*, 2001.

[12] Y. Amir *et al.*, "On the Performance of Group Key Agreement Protocols," *IEEE Int'l. Conf. Distrib. Comp. Sys.*, July 2002.

[13] W. Diffie and M. E. Hellman, "New Directions in Cryptography," *IEEE Trans. Info. Theory*, vol. IT-22, Nov. 1976, pp. 644–54.

[14] D. Boneh, "The Decision Diffie-Hellman Problem," *3rd Algorithmic Number Theory Symp.*, no. 1423, LNCS, Springer-Verlag, 1998, pp. 48–63.

[15] Y. Kim, A. Perrig, and G. Tsudik, "Tree-Based Group Key Agreement," http://eprint.iacr. org, rec. 2002/009, Cryptology ePrint Archive, Feb. 2002.

Biographies

YONGDAE KIM (kyd@cs.umn.edu) received his Ph.D. in computer science from the University of Southern California (USC) in 2002. Since August 2002 he has been an assistant professor in the Department of Computer Science at the University of Minnesota — Twin Cities. Between 2000 and 2002 he was a research associate at the University of California at Irvine (UC Irvine). From 1993 to 1998 he was a member of research staff at the Electronics and Telecommunication Research Institute in Korea. His research interests include network security and applied cryptography.

MAITHILI NARASIMHA (mnarasim@ics.uci.edu) is currently pursuing her Ph.D. in computer science at the School of Information and Computer Science, UC Irvine. She received her Master's degree in spring 2003 from UC Irvine. She is a member of the Secure Computing and Networking Center (SCONCE) at UC Irvine ; her research interests include storage and database security.

KYUNG-HYUNE RHEE (khrhee@pknu.ac.kr) has completed his M.Sc. and Ph.D. degrees in the Department of Applied Mathematics of the Korea Advanced Institute of Science and Technology (KAIST), Republic of Korea in 1985 and 1992, respectively. He served as a visiting professor at SCONCE, Department of Information and Computer Science, UC Irvine, 2001–2002. He is currently an associate professor at the Division of Electronic, Computer and Telecommunication Engineering, PuKyong National University, Busan, Republic of Korea. His main interests are mobile and ad hoc security, Internet application to security issues, and statistical analysis of cryptographic algorithms.

GENE TSUDIK (gts@ics.uci.edu) is an associate professor in the School of Information and Computer Science at UC Irvine. He has been active in the areas of internetworking, network security, and applied cryptography since 1987. He obtained a Ph.D. in computer science from USC in 1991 for his research on access control in internetworks. Before coming to UC Irvine in 2000, he worked at IBM Research (1991–1996) and USC Information Science Institute (1996–2000). Over the years, his research interests have included internetwork routing, firewalls, authentication, mobile network security, secure e-commerce, anonymity, secure group communication, digital signatures, key management, ad hoc network routing, and, more recently, database privacy and secure storage.

FABIO MAINO (fmaino@andiamo.com)