Parameterizing Access Control for Heterogeneous Peer-to-Peer Applications

Ashish Gehani
SRI

Surendar Chandra
University of Notre Dame

***Abstract*—Peer-to-peer overlays are being used for domain name resolution, massive multiplayer games, cooperative spam filtering, content sales and distribution, digital libraries, and data storage. As a result, applications often have conflicting access control needs. For example, an interactive game that needs fast response times for permission requests may prefer a capability-based access control subsystem (since the capabilities could be replicated). On the other hand, a digital library would choose an access control list approach (since it needs the ability to revoke permissions efficiently). Overlay designers are forced to either make an *a priori* choice for all applications, or to provide no access control functionality.**

We introduce DAAL (Decentralized Authentication and Authorization Layer) to allow application designers and users to select differing access control characteristics for each object. This allows a developer to use capability-like characteristics for objects whose access requests must complete quickly, while employing access control list-like functionality for other objects whose access needs to be efficiently revocable. Further, users can trade the efficiency of permission request and revoke operations for each object by adjusting its access control parameters. We empirically identify a simple criterion for parameter selection that guarantees good performance in the face of any predefined fraction of malicious peers in the overlay.

I. Introduction

Peer-to-peer protocols have been part of the Internet's fabric for several decades. For example, NNTP (Network News Transfer Protocol), ICP (Internet Cache Protocol), BGP (Border Gateway Protocol), OSPF (Open Shortest Path First), and DNS (Domain Name Service) zone transfers are completely decentralized. File sharing programs, such as Napster and Gnutella, pioneered the paradigm for end user applications almost a decade ago. In the intervening years, structured peer-to-peer overlay networks have become widely deployed as a substrate for a broad range of applications. To wit, OpenDHT [16] provides public access to a 200-node overlay distributed across the world using Planetlab [14]. Recently, a broad range of applications has been developed using the peer-to-peer framework. OverCite [28] provides citations of scientific publications. CoDNS [12] facilitates DNS lookups. Chord has been used to locate objects in a multi-player game [6]. Peer-to-peer sales of digital content have been explored [24]. SpamWatch [27] performs cooperative email filtering. As the applications grow in variety, their needs begin to diverge. In particular, their data access control requirements may differ significantly.

DAAL parameterizes access control operations by splitting each permission into β fragments, of which α are required for access to an object. As long as $(\beta - \alpha + 1)$ can be removed, revocation succeeds. Reducing α or increasing β improves the rate of finding sufficient fragments for an access request to complete. Decreasing $(\beta - \alpha)$ increases the probability of successfully revoking a right. By tuning α and β, the efficiency of granting, revoking, and requesting a right can be traded. Selecting α and β so that $\frac{\alpha}{\beta}$ exceeds μ, the fraction of subverted nodes in the overlay, ensures that access control operations are effected with high reliability.

II. Motivation

Current peer-to-peer authorization mechanisms fall broadly into two classes. Either they adapt access control lists to operate in wide area networks, like CRISIS [2], or they provide traditional capability semantics in a distributed environment, like Sirius [8]. The two classes have different benefits and limitations. For example, capabilities can be easily replicated to increase their availability, thereby decreasing the time it takes for a permission request to complete. However, once they have been granted, they cannot be revoked. This is not a problem when content is sold since rights are never revoked, but would be an issue if an object was on loan from a digital library. Access control lists enable revocation, but effect this by requiring all requests to go to a reference monitor. In a peer-to-peer environment, a centralized service can become a single point of failure, a performance bottleneck, or halt operations when network partitions occur. Such performance limitations would be a significant problem for an application that did DNS lookups. Thus, current security subsystems force the overlay designer to make *a priori* choices about which operations to optimize. Since the underlying overlay is shared, what is needed is an access control mechanism that is flexible enough to address conflicting application needs.

Table I illustrates the variation in the performance requirements for three access control operations used by five peer-to-peer applications. The bounds listed are weak. In practice, an application may need significantly better performance. CoDNS was created to speed up name resolution operations. Therefore, when a DNS query occurs, requests for permission to access data must complete rapidly. Similarly, when a program needs to determine if a piece of email should be marked as spam, or objects need to be located during a multiplayer game, permission requests must complete within seconds since a user may be interacting with the programs in real time. On the other hand, when digital content is borrowed online, the permission request occurs just once before the user proceeds to use the object for a long duration. In such a setting, users can tolerate

TABLE I
OPERATION TIMEFRAME

	Grant	Revoke	Request
Name resolution	Minute	Minute	Second
Spam filtering	Minute	Hour	Second
Digital library	Minute	Day	Minute
Content sales	Second	Never	Second
Multiplayer game	Second	Minute	Second

the operation taking a minute to complete. In the case of some applications, a new permission is granted occasionally, such as when a user is given the right to resolve names in a new domain, starts using information from another user for spam filtering, or borrows digital content from a library. Since the grant operation is infrequent, a delay of a minute from the time it is granted to when a user can utilize it would be accepted. When users are interacting in an online game, or they have just paid for content, they expect to get rapid access to their data. In such cases, the grant operations would need to complete in seconds. Finally, the speed with which a revocation operation must occur varies. Since games are interactive and name resolution is used in security protocols, revocation for these applications must complete within minutes. If a user's access to spam metadata or library content exceeds the target period, the consequences are limited. Hence it would tolerable if the revocation takes an hour or a day, respectively. Access to content that has been purchased never needs to be revoked.

III. DESIGN

A. Assumptions

Every peer has equivalent functionality. Subjects can store and retrieve objects using the overlay's insertion and access mechanism. To effect access control, our system must allow a user to specify which subjects should have read or write access to each object that they insert into the overlay. The confidentiality of the data must be retained regardless of where the data and any associated metadata are stored in the overlay. Only users with read permission must be able to see it. When a valid user effects a write operation, it can be verified as legitimate by other users. However, since the storage system is oblivious to the access control mechanism, users who do not have write permission for an object may still be able to delete, append, or otherwise alter it. These changes must be detectable as fraudulent. For example, if an object is stored at a malicious peer, the node can modify it. An authorized user who retrieves the object and tries to decrypt it will detect the unauthorized changes. Versioned storage can be used to allow users to retrieve previous commits from alternate nodes.

Peer-to-peer systems are designed to be scalable. So must their security mechanisms. In general, peers are assumed to operate in good faith. However, as more nodes turn malicious, the system's security guarantees should degrade gracefully. In particular, no single node or small clique should be able to subvert the entire system's assurances, including the access control protection mechanisms. A robust model of trust for peer-to-peer systems assumes that each node operates correctly at least some known fraction of the time or that at any given time, a known fraction of the nodes is operating correctly.

Identity administration is effected offline, exempting it from consideration as a target of attack. However, authentication and authorization occur as part of the online operation of the overlay. They can be implemented in one of two ways. The first method implicitly performs them as part of the data access protocol itself, typically through the use of cryptographic primitives. The second approach is the use of explicit communication with authentication or authorization services. Such an approach requires distributed and decentralized protocols.

In principle, even the operation for creating new subjects can be altered so each subject can create itself without any central intervention, as with PGP [15]. However, in this case each subject must create an identity for every other subject that it wishes to grant access rights to. Though this imposes an administrative burden on users, in practice there exist scenarios where it is worthwhile. In particular, it is useful when there does not exist any common root of trust, such as a shared bank, employer, university, government, or other institution. In such cases, prior to granting access to an object, the grantor and grantee need to establish a trust relationship. This can then be leveraged so they can simultaneously create identities for each other.

B. Goals

When designing the system, our aims include explicit tradeoffs that need to be made because of the constraints of peer-to-peer environments.

a) Parameterizing the protocol: Bandwidth, storage, and computational resources of the nodes in a peer-to-peer network vary widely. When one resource is scarce, it may be possible to continue operating correctly by using more of another resource. However, detecting where the optimal tradeoff will lie depends on global knowledge of the overlay. Without this information, a choice can be made by the application that is using the access control system. For example, using more network bandwidth and overlay storage when granting a permission could increase the likelihood of a subsequent access request completing within a given interval of time. Either the user or the application could set configuration parameters to yield the desired performance profile.

b) Establishing trust: There are three primary methodologies for establishing trust. The first is the use of a reputation-based system. Each subject in the system monitors the behavior of others with which it interacts. Such a scheme does not require any *a priori* mutual introduction between users by a third party. However, it is particularly susceptible to "Sybil attacks" [5]. Also, if the identifiers are not permanently bound to subjects, a malicious user's negative record is lost when it leaves the system and subsequently returns after a suitable timeout. If the profile persists in the overlay, a malicious user can repeatedly reconnect till they become attached to a profile that has a positive record, and can then exploit the imbued trust.

A variant of the above methodology is the web of trust, which is a formal representation of a multirooted graph of trust relationships. Each user acts as its own certification authority, issuing credentials to others that it believes are trustworthy. In addition, a user having an established trust relationship with another user can accept that user's attestations regarding the trustworthiness of previously untrusted users. Such transitive relations allow proofs of trustworthiness to be constructed in the absence of a central certification authority. PGP [15] popularized this approach and it is the basis of SDSI's linked local namespaces [1]. Since identities can be clearly defined, "Sybil attacks" are difficult to mount. The absence of a central authority makes the scheme useful for peer-to-peer environments. However, it has weaknesses as well. If a trusted peer is subverted, its attestations can no longer be relied upon until that peer is able to reestablish its veracity and recertify its previous statements. This can disrupt the trust relationship between any pair of users that used the subverted one in their certification chain. In the interim, while the subverted user's revocation propagates through the system, the trust can also be abused. Further, the overhead of bootstrapping trust in this model is significantly higher than when a single root certification authority exists. Nevertheless, it provides a compromise between the convenience of the first model and the security of the next model. This makes it useful for certain applications and DAAL aims to support it.

The third alternative is to use a single logical root of trust. The certification is recursive, forming a hierarchical certification tree. Verifying the trustworthiness of a node is accomplished by checking the chain of certificates from the node to the root of the tree. Such a system is unsuitable for peer-to-peer environments where all nodes have equivalent functionality. DAAL provides the described underlying trust relationship without requiring a hierarchy of network servers.

c) Authorizing users, not nodes: In a peer-to-peer network, the same access control subject may appear to connect to the rest of the overlay from a number of points. This may be due to the way the overlay manages nodes joining and leaving the network. When the same host computer reconnects and reintroduces itself into the overlay, it may receive a new node identifier each time. Alternatively, a user may be mobile and reconnect the host at a different point in the underlying network with the result that the overlay node identifier may change. Finally, the user may wish to connect to the overlay from a different host but still have access to remote data which that user is authorized to use. In such a scenario, the subject's connections will be coming from a different overlay node. As a result, it is important to ensure that an object's owner grants access rights to a subject, not a specific other node in the overlay. These rights should be usable by the subject independently from the overlay node from which access requests are made.

d) Allowing unordered access control operations: Traditional access control implementations require an administrator to create a subject's identity in the system before any permissions can be delegated to the subject. Similarly, an object must be created before its access rights can be defined. Further, a subject typically retrieves an object prior to making a request for rights to access it. However, this ordering is not strictly necessary. In principle, it is possible to treat the operations as commutative. The flexibility afforded by doing this is useful in peer-to-peer environments. For example, consider the problem of a subject wishing to grant a permission to another user. If the recipient was required to first have a certified identity, then it would be necessary for the object's owner to ensure that this property held by either contacting a directory service or requesting a public key certificate from the recipient. This lookup can be avoided by allowing the permission to be constructed and distributed with a cryptographic constraint imposed on it. The check would allow the permission to be used only by a subject that has subsequently been certified as the legitimate recipient. Making identity and rights creation commutative thus reduces the subset of the network that needs to be reachable for these operations to complete.

IV. Architecture

A user can opt to control access to an object by using DAAL. This involves two steps. In the first, the object is transformed into a protected format of the form in Figure 1. A signed hash of the object is computed. The signing key will be used as a capability for granting write permission. The object is then encrypted with a symmetric cipher. The encryption key used will serve as a capability to limit read access to the object. A signed version of the hash is prepended to the object so its integrity can be verified after decryption. A certified signature verification key is prepended. Next, two positive integer parameters α and β are selected and prepended before the hash. (α must be strictly less than β.) $\frac{\alpha}{\beta}$ characterizes the extent to which DAAL can continue to service requests in the face of subverted overlay nodes. Finally, the object's name and owner are prepended. The name is used as part of the protocol for requesting the access rights of the object. The owner is used in the process of verifying the object's integrity.

The second step is invoked when the object's owner wishes to grant access to another user. Access control metadata is created as shown in Figure 2. This is done once for each user that the owner wishes to delegate read or write permissions to. It must be repeated for each object for which permissions are to be granted. The read and write capabilities constructed

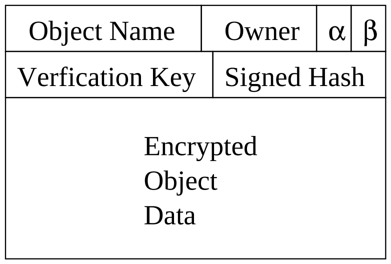

Fig. 1. Sealing an object encrypts it and prepends the object's name and owner, the share parameters α and β, a certified verification key, and a hash of the plaintext.

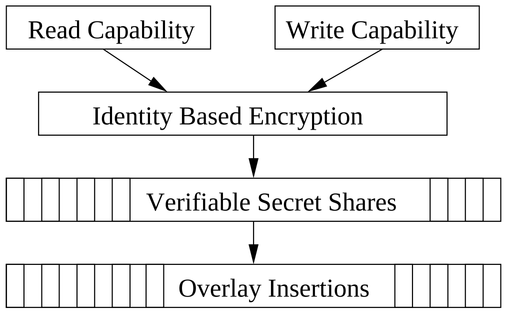

Fig. 2. Read and write capabilities are encrypted with the receiver's identity, and then a hash of them is signed using the sender's identity. The result is split using a secret sharing scheme. Each piece is inserted at a different location in the overlay.

during the object's encryption are encrypted with an *identity-based cipher* [3]. This allows the recipient's public key to be derived without a public key infrastructure. The key used is selected so only the intended recipient of the access rights can decrypt the capabilities. A data structure with space for both the sealed read and write capabilities is constructed. If read permission is to be granted, the appropriate capability is filled in. Otherwise, the field is left empty. Similarly, only if write permission is to be granted is the relevant capability inserted. The data structure is now transformed into β pieces using *verifiable secret sharing* [13]. These are dispersed over the overlay as illustrated in Figure 3. Finally, the owner makes a note of the delegation. This is needed for revocation.

A user who gains access to the object can reverse the above process to obtain permissions. The object's header, shown in Figure 1, contains enough information for the user to be able to locate all β pieces because each fragment's name in the overlay is the catenation of the delegatee's identity, the object's name, and the fragment's index number (that determines which of the β pieces it contains). Only α of the β pieces are required to reconstruct the capabilities. The user retrieves these from the overlay as illustrated in Figure 3. The pieces are processed with the inverse of the algorithm used to split the capabilities.

Pedersen's verifiable secret sharing scheme includes a witness with each share. If a set of peers provides fraudulent shares, then when the capabilities are being reconstituted, the witness information suffices to determine which peers were responsible for supplying incorrect shares. Trust management schemes can utilize the outcome of these checks to score peers. Since DAAL checks each share, it does not need to establish trust profiles of peers. If at least α peers provided legitimate shares, the interpolation step of Pedersen's scheme will produce a correct capability set as its output.

The output is then decrypted using a key that only the intended recipient knows. If the read capability field is not null, it can be used to decrypt the object. The signed hash in the object's header is verified to ensure that it was not modified without authorization. If the check fails, an earlier version of the object is retrieved. A user who wishes to modify the object can do so if the write capability field is filled. The

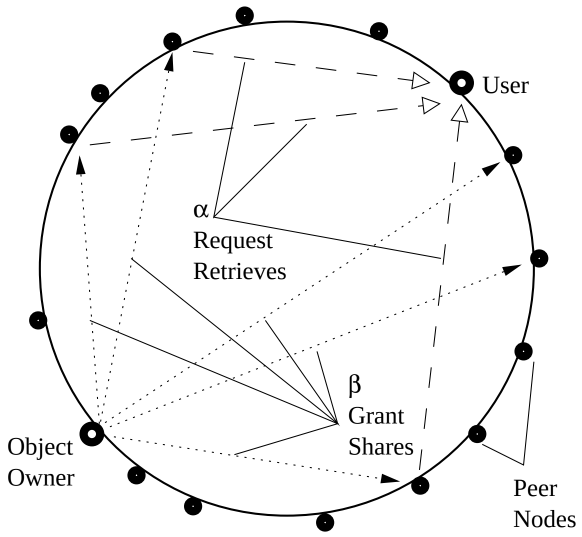

Fig. 3. Granting a permission results in β shares being inserted in the structured peer-to-peer overlay. Requesting a permission requires at least α shares to be retrieved.

user hashes the new version of the object, signs the hash with the write capability, and replaces the hash in the encrypted object's header. The user then encrypts the new version of the object and replaces the encrypted object data with this.

Revoking a permission stops a user from being able to gain access to an object. This is done by the removal of the relevant shares from the overlay. As long as at least $(\beta - \alpha)$ nodes comply with the deletion request, it will no longer be possible for the revoked user to utilize the remaining pieces to reconstruct the capability. This differentiates DAAL from previous distributed capability systems that either do not provide revocation or require every node to run a trusted reference monitor, which is clearly untenable in a peer-to-peer environment.

V. Implementation

We have prototyped DAAL with a combination of the Java Cryptography Architecture, an implementation of Boneh-Franklin identity-based encryption using the modified Tate pairing [7], Pedersen's verifiable secret sharing [13], and the Bamboo [16] distributed hash table running on Planetlab. The underlying platform of each DAAL peer was Mac OS 10.4.3 on a 1.2 GHz PowerPC. The remaining peers are varied but conform to the Planetlab node specification.

Overlay operations are effected by contacting random known Bamboo nodes (to prevent a single gateway from becoming a central point of failure). Objects are inserted into the overlay with a unique key that is needed for their removal. This prevents unauthorized users from revoking permissions.

Java applications use DAAL through the programming interface defined in the **daal.Access** class and shown in Figure 4. The **Globals** class contains a number of global constant values for the identity-based encryption and verifiable secret sharing cryptographic computations. An instance of the class must be created when the overlay initializes. Thereafter that

```
static void grant(Globals globals,
   Delegation delegation, User owner,
   String user, String filename,
   boolean grantRead, boolean grantWrite,
   int alpha, int beta);

static void revoke(Globals globals,
   Delegation delegation, User owner,
   String user, String filename,
   int alpha, int beta);

static Capabilities request(Globals globals,
     User user, String filename,
     int alpha, int beta);
```

Fig. 4. DAAL's Java API

instance is passed to all DAAL methods. It can be stored in any public location since none of the values are secret. The **Delegation** class acts as a key store. Thus, each user has an instance of it. When a user grants permission for another user to access an object, DAAL adds the new keys to the **delegation** instance passed to the *grant()* operation. The constructors for **Globals** and **Delegation** handle all initial housekeeping, such as creating the necessary large primes and hash tables.

The **User** class encapsulates the private credentials of a user. These are transparently created by the class's constructor. Since the user may have multiple roles, each with its own credentials, the appropriate instance of the **User** class must be explicitly provided. Each permission being granted, revoked, or requested is associated with a specific data object. The object's **filename** must be passed to the *grant(), revoke(),* or *request()* method. The object's α and β parameters must also be provided. These can be retrieved from the metadata of an object. The *grant()* operation takes two Boolean flags indicating whether to delegate read, write, or both permissions to an explicitly named **user**. *revoke()* and *request()* methods do not require any other parameters. The *request()* operation returns an instance of the **Capabilities** class that contains a key to decrypt the object if read permission has been granted and a signing key if writes are permitted.

VI. Properties

Grant

We now consider the effect of varying the parameters α and β on the access control operations' properties. When a new permission is being granted to a user, a larger value of β results in more fragments. Since they are created using cryptographic secret sharing, their construction cannot be parallelized. Therefore, the time to effect a grant operation increases as β becomes larger. Each fragment's size is on the order of a kilobyte. If they are being inserted over a limited bandwidth connection, such as a dialup line, the grant operation takes even longer for larger values of β. Over broadband lines, latency dominates the insertion time, allowing the fragments to be inserted in parallel. Increasing α makes the computation of each fragment take longer but has no effect on the insertion time since all β fragments must be put into the overlay. These factors dictate that both α and β should be minimized to speed up grant operations.

Revoke

Revoking a permission requires that at least $(\beta - \alpha + 1)$ fragments must be deleted from the overlay (since this ensures that α fragments can no longer be found and reassembled). This requires fragment deletion requests to be sent to the appropriate peers. No cryptographic computation is necessary. In principle, only $(\beta - \alpha + 1)$ network connections need to be made. Therefore, minimizing β and maximizing α will speed up the rate of revocations.

As β increases, so does the number of network connections needed for revocation. Since the remove request takes only a few bytes, the *revoke()* operation is not bandwidth limited. Instead, the constraint is the latency of the slowest $(\alpha + 1)$ peers if the message is sent to all β peers holding fragments.

In practice, the latency for deleting an object from a remote node depends on several factors. Peers may be unreachable because of a partition in the underlying network. Remote hosts may simply be powered off or disconnected from the network. Finally, we must account for the fact that some fraction of the overlay may consist of adversarial nodes, intent on disrupting the protocol by refusing to cooperate. We model the collected behavior by assuming that a peer performs correctly with probability $(1 - \mu)$. We can then derive $\rho_{revoke}(\alpha, \beta)$, the reliability of a revocation operation, as a function of α and β:

$$\rho_{revoke}(\alpha, \beta) \quad = \quad \sum_{i=0}^{\alpha-1} \binom{\beta}{i} \; (1-\mu)^{\beta-i} \;\; \mu^i$$

Request

When an application requests permission to read or write an object, the appropriate capabilities must be constructed by retrieving α fragments from the overlay. The cryptographic reassembly of the permission is effected by interpolating the pieces of data recovered from the fragments. If each piece came from a cooperating peer, the computation time is proportional to α. In this scenario, the value of β is immaterial since DAAL has sufficient data to complete the request after α network connections. Therefore, minimizing α will decrease the completion time for a permission request.

Peers in the overlay may malfunction for a variety of reasons. When this happens, some of the α fragments may be corrupted or irretrievable. As before, we model this by assuming that a peer will be unreachable, malfunctioning or malicious with probability μ. We refer to a node in this state as *compromised*. When a permission's fragments are distributed, they are routed to a random set of nodes. If more than $(\beta - \alpha)$ shares are stored at nodes that are compromised when a request is made, then it will fail. We can compute $\rho_{request}(\alpha, \beta)$, the reliability with which authorized permission requests will succeed, as a function of α and β:

$$\rho_{request}(\alpha, \beta) \quad = \quad \sum_{i=0}^{\beta-\alpha} \binom{\beta}{i} \; (1-\mu)^{\beta-i} \;\; \mu^i$$

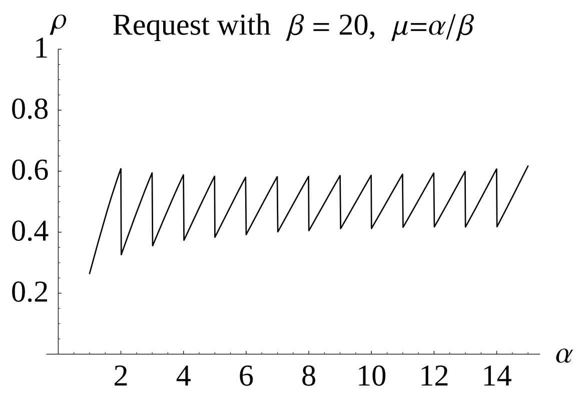

Fig. 5. μ is $\frac{\alpha}{\beta}$. α is varied while β is 20. The reliability of requests ρ oscillates around $\frac{1}{2}$.

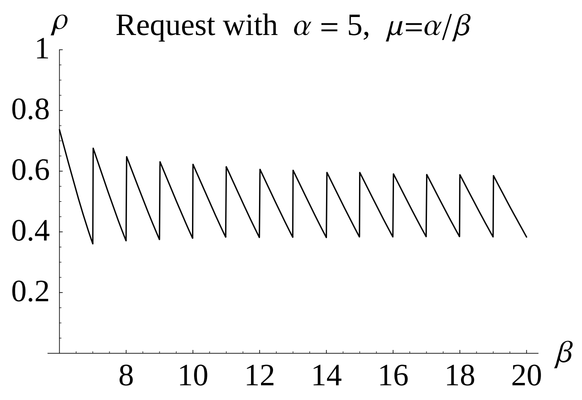

Fig. 6. μ is $\frac{\alpha}{\beta}$. α is held constant at 5. β is varied. Again, the reliability ρ oscillates around $\frac{1}{2}$.

Intuitively, finding and retrieving α fragments of a permission becomes progressively easier as the total number of pieces distributed in the overlay increases. Similarly, if fewer fragments are required to reconstruct a permission, the likelihood of finding enough of them increases. Thus, in addition to minimizing α, we can increase β to speed up permission requests.

VII. Parameter Selection

A. Performance

The criteria that yield the fastest access requests conflict with those that optimize permission revocation. Keeping either α or β constant and varying the other one forces a tradeoff between the rates at which access and revocation requests complete. However, this does not constrain application developers from optimizing the performance of both operations simultaneously. This is analytically possible since they have the freedom to adjust two variables (the parameters α and β) given two constraints (minimizing the time for requests and revocations). Intuitively, the value of α can be increased while it yields fast enough request completion times. Then the value of β can be adjusted so that revocations are fast enough. Note that increasing β may reduce request times but will never make them worse.

B. Reliability

In practice, parameter selection is complicated by the fact that it affects the reliability of request and revoke operations. Therefore, α and β cannot be selected to optimize performance alone. Using the equations in Section VI, we can infer that altering α and β has a complex effect on the reliability of the access control operations. The reliability characteristics are dependent on a third variable, μ, over which the application developer and individual user have little control. However, the value of μ (which represents the fraction of overlay nodes that are unreachable, malfunctioning or malicious) can be estimated empirically.

1) Request: We first examined the reliability of access requests when α and β were selected so that they mirrored the fraction of compromised nodes in the overlay. We found that it did not matter what specific values of α and β we chose. In Figure 5, the reliability of requests is plotted with β fixed at 20. Regardless of the value chosen for α, which we varied from 2 to 15, if the fraction of compromised nodes in the overlay was the same as the fraction $\frac{\alpha}{\beta}$, then the reliability oscillated around $\frac{1}{2}$. Similarly, Figure 6 plots the effect of fixing α at 5 and using different values for β, ranging from 5 to 20. The reliability is very similar to that observed when α was varied with a fixed β. Further examination shows that when $\frac{\alpha}{\beta} = \mu$, there is a balance between the cooperating and compromised forces in the overlay. Larger values of α mean that there are more cooperative nodes from which legitimate shares can be retrieved. However, to complete an operation, more shares are also then required. Similarly, if β is larger, the likelihood of a particular share being corrupt or unreachable is lower but there are simultaneously more shares that can potentially be used.

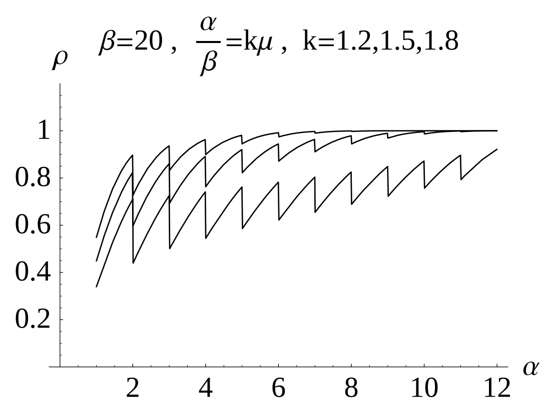

Fig. 7. $\frac{\alpha}{\beta}$ is chosen to be 1.2, 1.5 and 1.8 times μ, the fraction of compromised nodes. As the multiple increases, the reliability of requests ρ rapidly improves. In each plot, β is held constant at 20 while α is varied. When μ has a higher value, there are more legitimate shares accessible. As a result, the reliability is higher. Thus, the $k = 1.2$ is the lowest plot, while $k = 1.8$ is the top line.

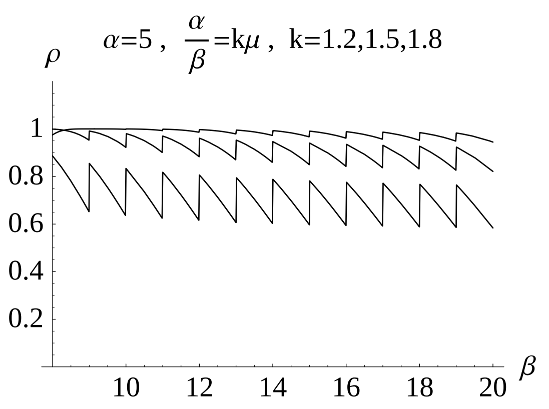

Fig. 8. Again, the reliability ρ grows as $\frac{\alpha}{\beta}$ increases over μ, giving DAAL an advantage over the compromised nodes. Here α is held constant at 5 while β is varied. As before, increased values of k yield higher reliability. ($k = 1.2$ plot is the lowest, while the $k = 1.8$ plot is the top line.)

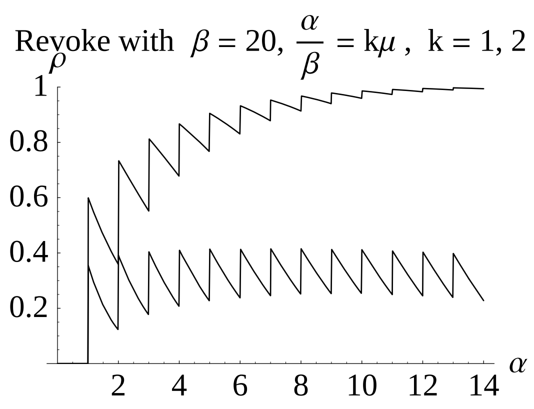

Fig. 9. Revocation reliability as α is varied with $\beta = 20$. $\frac{\alpha}{\beta} = 2\mu$ is significantly higher than $\frac{\alpha}{\beta} = \mu$.

Thus it is important to select α and β so that the cooperating peers have an advantage over the compromised ones.

With an understanding of the relationship of α and β to μ, we can now test our hypothesis that we need to select α to be large enough to overcome the effect of the compromised nodes. We can do this by selecting α and β so that the ratio is larger than μ, the fraction of compromised nodes in the overlay, by a factor k. In Figure 7, the reliability of access request operations is plotted as a function of α, which is varied from 1 to 12. In all cases, each permission is fragmented among 20 nodes, that is $\beta = 20$. When k is 1.2, the reliability rises gradually as α grows. Since the fraction of compromised nodes tracks $\frac{\alpha}{\beta}$, a higher value of α (with a fixed β) means there are more compromised nodes. Despite this, the reliability increases with α. As k increases, so does the reliability. This can be seen from the plots for $k = 1.5$ and $k = 1.8$ in Figure 7, which have successively higher values. Thus, if we wish to boost the reliability of request operations, for a given fraction of compromised nodes and a fixed value of β, we should use a higher value of α. Figure 8 shows the result of varying the total number of fragments that a permission is split into, that is β, from 10 to 20, while requiring a fixed number of fragments (i.e., $\alpha = 5$) to reconstruct a permission. Regardless of β's value, a higher value of k ensures higher request reliability. Thus, α needs to be increased to the point where $\frac{\alpha}{\beta}$ is sufficiently larger than the empirical estimate of μ.

2) Revoke: We now consider the effect of α and β on the reliability of revocation. As before, we first examine the behavior when the parameters are selected to mirror the fraction of compromised nodes, that is $\frac{\alpha}{\beta} = \mu$. Figure 9 shows the reliability when the number of permission shares, β, is fixed at 20 while the threshold of recovery, α, is varied from 1 to 14. The lower plot in that figure corresponds to $k = 1$, where $\frac{\alpha}{\beta} = \mu$. We see that the revocation fails with significant probability. Similar results are found when we fix α at 5 and vary β from 7 to 20, as shown in the lower plot of Figure 10 (where $k = 1$).

When the parameters were varied, they had inverse effects on the performance of request and revoke operations. Therefore, since boosting α improved the reliability of requests, we may expect the need to reduce it to improve the reliability of revocation. However, this is not the case since revocation also relies on having sufficient nodes that cooperate. Revocation requires $(\beta - \alpha + 1)$ nodes holding a share to cooperate. Therefore, as α's value draws closer to β, the number of cooperating nodes needed drops quickly and revocation reliability grows commensurately. Thus, a higher value of k improves revocation reliability. We selected parameters so that $\frac{\alpha}{\beta}$ is twice the fraction of compromised nodes, μ. The upper plot in Figure 9 shows the reliability of revocation when α is varied from 1 to 14 while β is fixed at 20. The same behavior is observed when β is varied from 8 to 20 while α is held constant at 5, as can be seen in the upper plot in Figure 10. Compared to the case where $k = 1$, the reliability drastically improves when α and β are selected so that $k = 2$.

3) Optimization: The criteria determined in Sections VII-B1 and VII-B2 can be collected into a greedy deterministic algorithm that yields values for α and β that satisfy a user's performance criteria. It is shown in Algorithm VII.1,

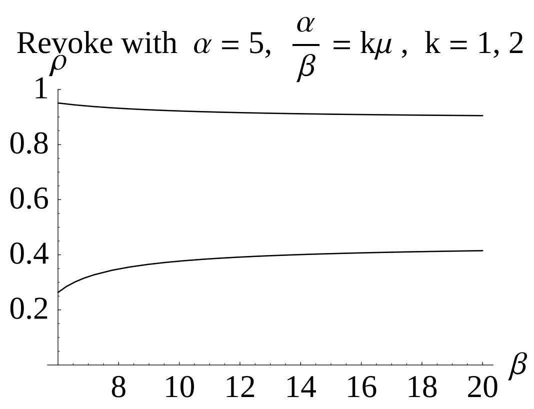

Fig. 10. Revocation reliability as β is varied with $\alpha = 20$. Again, $\frac{\alpha}{\beta} = 2\mu$ is higher than $\frac{\alpha}{\beta} = \mu$.

where the maximum latency tolerable for permission requests and revocation operations is denoted by $request_{threshold}$ and $revocation_{threshold}$, respectively. The $Time()$ function measures how long it takes to execute the operation passed as a parameter. We assume that an estimate of μ is available and that k has been selected to provide a sufficient margin of reliability for the application.

```
Algorithm VII.1: SELECT($\alpha, \beta$)

$\alpha \leftarrow 1, \beta \leftarrow 2$

while $Time(request(\alpha, \beta)) < request_{threshold}$
  do $\begin{cases} \alpha\text{++} \\ \beta\text{++} \end{cases}$

while $Time(revoke(\alpha, \beta)) < revoke_{threshold}$
  do $\beta$++

$\beta_0 \leftarrow \beta$

while $\frac{\alpha}{\beta} <= k.\mu$
  do $\beta$--

if $\beta < \beta_0$
  then Optimization Failed
  else $Output(\alpha, \beta)$
```

The algorithm starts with the smallest possible values for α and β, where a permission is split into two shares, either of which suffices for gaining access. The required and total number of shares is increased in lockstep as long as access requests do not start taking too much time. When this point is reached, the threshold for reconstructing a permission is held constant and the total number of shares is increased until the time for revocation grows too long. At this point the ratio $\frac{\alpha}{\beta}$ is continuously increased by decreasing β until it is larger than the fraction of compromised nodes, μ, by the required margin of reliability, k. If β is reduced to the point where the performance constraints are violated, then the algorithm is deemed to have failed at finding parameters that satisfy the user's criteria. Otherwise, the values of α and β are output. An application developer needs to run this only once per class of objects that needs particular access control characteristics. The values are added to the object metadata when it is created. Thereafter these values are transparently utilized.

VIII. Related Work

Peer-to-peer Security

Systems providing remote access to data using centralized services have existed for several decades. For example, Xerox PARC's Interim File Server was operational in 1975 [23]. Unstructured peer-to-peer overlay networks do not depend on such online centralized services after the completion of the initial bootstrapping procedure. The latter consists of downloading source or binary code to implement the protocol and a list of seed nodes with which to initialize routing. This facilitated a rapid growth in the number of deployed nodes in systems like Gnutella [17]. However, it also had unwanted side effects such as queries that frequently failed or resulted in long response times and excessive network traffic from message flooding. Structured peer-to-peer overlays, typically using a distributed hash table, like Chord [22] are able to ameliorate these problems. They implicitly require all participating nodes to follow a uniform, predefined protocol for operations such as choosing which neighbor to route each message to. Since these overlays operate in the absence of any prior trust relationships between nodes, the system becomes vulnerable to attacks by malicious nodes that are willing to violate the central specification.

Sit and Morris examined the underlying assumptions of distributed hash table-based structured peer-to-peer overlay systems [20]. They found that it was possible to disrupt the correct operation of the overlays in a variety of ways. Potentially, the network could be partitioned, messages could be misdirected during routing, routing tables could be corrupted, availability guarantees through replication could be violated, malicious nodes could not be identified reliably, garbage in messages could result in computation power denials of service, rapid node churn could create network bandwidth denials of service, and responses could be forged. They suggested alternate designs to address the issues. Douceur argued that verifiable node identification is necessary to avoid "Sybil" attacks [5], those that can be launched by a single node masquerading as many. Further, if the maximum degree of a node in the overlay graph is bounded, then it is possible to prevent "eclipse" attacks [19] where most of a victim's neighbors act maliciously to control the traffic flow between it and the rest of the overlay. Using appropriate choices [20], secure routing between nodes can be effected [4]. In particular, nodes need to obtain identity certificates from a trusted authority, updates to routing table entries must satisfy protocol-specific constraints, and routing success is verified by checking that the mean distance between destination replicas does not cross a predefined threshold. The latter test evaluates whether the destination set is likely colluding, in which case redundant routing is used.

Distributed Authorization

Traditional distributed storage systems [25], [9] had a different trust model from current peer-to-peer networks. While the users were not trusted, the client hosts from which they connected were assumed to be running system software that was controlled by administrators. As the size of the deployments increased, ensuring the integrity of every machine's software became more difficult. Athena and Andrew [18] addressed this by redefining the trusted computing base to exclude all software running on workstations. Kerberos's [21] authentication and authorization services ran on a few machines where users could not log in. CRISIS [2] was designed to provide the same services across the wide area where network connectivity is unreliable. However, their dependence on specialized security nodes limits their utility in peer-to-peer environments for the reasons described in Section III.

A number of projects [10], [26] adapted the idea of allowing clients to store data on untrusted remote servers. Access control is provided using cryptographic primitives. These systems require specialized group servers that cannot be run on an untrusted node. Also, each server manages its own authentication. SFS [11] was extended with a decentralized authentication server, which allows cryptographic credentials to be transparently exchanged between SFS file servers in different protection domains. Its reliance on these specialized servers rules out its use in the peer-to-peer context. SiRiUS [8] was designed for use over a range of storage technologies, including peer-to-peer overlays. However, it does not address the problem of continued operation in the face of malicious nodes. In contrast, DAAL's use of verifiable secret sharing for the authorization metadata allows it to tolerate a fraction of the overlay operating maliciously.

IX. Conclusion

A traditional reference monitor becomes a central point of failure if the overlay is exposed to the Internet. Capability-based systems scale well in distributed environments but do not provide efficient revocation mechanisms. DAAL provides a hybrid system that allows the performance and reliability of grant, revoke, and request operations to be traded. This allows application developers and users to utilize a single underlying data authorization framework while varying its characteristics at the granularity of individual objects.

DAAL uses two parameters for each object, dictating the extent to which permission fragments are replicated and the threshold of cooperation necessary before the permission can be reconstructed. We first characterized the effect of these parameters on performance. Next, their relationship to the reliability of operations was examined. Finally, we established a criterion for parameter selection that relates them to the fraction of nodes in the overlay that are likely to be compromised. This suffices for users to determine how to select parameters for their particular application.

References

[1] Martin Abadi, On SDSI's linked local name spaces, Journal of Computer Security, 1998.

[2] E. Belani, A. Vahdat, T. Anderson, and M. Dahlin, CRISIS wide area security architecture, 7th USENIX Security Symposium, 1998.

[3] Dan Boneh and Matt Franklin, Identity-based encryption from the Weil Pairing, SIAM Journal of Computing, 32(3), 2003.

[4] Miguel Castro, Peter Druschel, Ayalvadi Ganesh, Antony Rowstron, and Dan S. Wallach, Security for structured peer-to-peer overlay networks, Proceedings of the 5th Symposium on Operating Systems Design and Implementation, 2002.

[5] John R. Douceur, The Sybil attack, Proceedings of the 1st International Workshop Peer-to-Peer Systems, 2002.

[6] Scott Douglas, Egemen Tanin, Aaron Harwood, and Shanika Karunasekera, Enabling massively multi-player online gaming applications on a P2P architecture, Proceedings of the IEEE International Conference on Information and Automation, 2005.

[7] A. Duffy and T. Dowling, An object oriented approach to an identity based encryption cryptosystem, 8th IASTED International Conference on Software, 2004.

[8] Eu-Jin Goh, Hovav Shacham, Nagendra Modadugu, and Dan Boneh, SiRiUS: Securing Remote Untrusted Storage, Network and Distributed Systems Security Symposium, 2003.

[9] A. Hisgen, A. Birrell, T. Mann, M. Schroeder, and G. Swart, Availability and consistency tradeoffs in the Echo distributed file system, 2nd IEEE Workshop on Workstation Operating Systems, 1989.

[10] Mahesh Kallahalla, Erik Riedel, Ram Swaminathan, Qian Wang, and Kevin Fu, Plutus: Scalable secure file sharing on untrusted storage, 2nd Conference on File and Storage Technologies, 2003.

[11] David Mazieres, Michael Kaminsky, M. Frans Kaashoek, and Emmett Witchel, Separating key management from file system security, 17th ACM Symposium on Operating Systems Principles, 1999.

[12] KyoungSoo Park, Vivek S. Pai, Larry Peterson, and Zhe Wang. CoDNS: Improving DNS performance and reliability via cooperative lookups, Proceedings of the 6th Usenix Symposium on Operating Systems Design and Implemetation, 2004.

[13] T. P. Pedersen, Non-interactive and information-theoretic secure verifiable secret sharing, Advances in Cryptology, Lecture Notes in Computer Science 576, 1991.

[14] Larry Peterson, Andy Bavier, Marc E. Fiuczynski, and Steve Muir, Experiences building PlanetLab, Proceedings of the 7th USENIX Symposium on Operating System Design and Implementation, 2006.

[15] http://www.pgp.com/

[16] Sean Rhea, Brighten Godfrey, Brad Karp, John Kubiatowicz, Sylvia Ratnasamy, Scott Shenker, Ion Stoica, and Harlan Yu, OpenDHT: A public DHT service and its uses, ACM SIGCOMM, 2005.

[17] M. Ripeanu, I. Foster, and A. Iamnitchi, Mapping the Gnutella network: Properties of large-scale peer-to-peer systems and implications for system design, IEEE Internet Computing Journal, 2002.

[18] Mahadev Satyanarayanan, Integrating security in a large distributed system, ACM Transactions on Computer Systems, 7(3), 1989.

[19] Atul Singh, Tsuen-Wan Ngan, Peter Druschel, and Dan S. Wallach, Eclipse attacks on overlay networks: Threats and defenses, Proceedings of IEEE INFOCOM, 2006.

[20] Emil Sit and Robert T. Morris, Security considerations for peer-to-peer distributed hash tables, Proceedings of the 1st International Workshop on Peer-to-Peer Systems, 2002.

[21] J. G. Steiner, B. C. Neuman, and J. I. Schiller, Kerberos: An authentication service for open network systems, Winter Usenix Conference,1988.

[22] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan, Chord: A scalable peer-to-peer lookup service for Internet applications, ACM SIGCOMM 2001.

[23] C. P. Thacker, E. M. McCreight, B. W. Lampson, R. F. Sproull, and D. R. Boggs, Alto: A personal computer, Computer Structures: Principles and Examples, 1981.

[24] A. Wierzbicki and K. Goworek, Peer-to-peer direct sales, Fifth IEEE International Conference on Peer-to-Peer Computing, 2005.

[25] Edward Wobber, Martin Abadi, Michael Burrows, and Butler Lampson, Authentication in the Taos operating system, 14th ACM Symposium on Operating System Principles, 1994.

[26] Fareed Zaffar, Gershon Kedem and Ashish Gehani, Paranoid: A global secure file access control system, 21st Annual Computer Security Applications Conference, 2005.

[27] Feng Zhou, Li Zhuang, Ben Y. Zhao, Ling Huang, Anthony D. Joseph, and John Kubiatowicz, Approximate object location and spam filtering on peer-to-peer systems, Proceedings of the 4th ACM/IFIP/Usenix Middleware Conference, 2003.

[28] Feng Zhou, Li Zhuang, Ben Y. Zhao, Ling Huang, Anthony D. Joseph, John Kubiatowicz, Jeremy Stribling, Jinyang Li, Isaac G. Councill, M. Frans Kaashoek, and Robert Morris, OverCite: A distributed, cooperative CiteSeer, Proceedings of the 3rd USENIX Symposium on Networked Systems Design and Implementation, 2006.

X. Acknowledgment

This work was supported by National Science Foundation Grants IIS-0515674 and CNS-0447671.

This work was produced in part with support from the Institute for Information Infrastructure Protection (I3P) research program. The I3P is managed by Dartmouth College, and supported under Award number 2003-TK-TX-0003 from the U.S. Department of Homeland Security, Science and Technology Directorate. Points of view in this document are those of the author(s) and do not necessarily represent the official position of the U.S. Department of Homeland Security, the Science and Technology Directorate, the I3P, or Dartmouth College.