

Security Issues in Inter-Domain Data Management

B. Hoon Kang
Division of Computer Science
UC Berkeley
Berkeley, CA 94720
hoon@cs.berkeley.edu

Jaein Jeong
Division of Computer Science
UC Berkeley
Berkeley, CA 94720
jaein@cs.berkeley.edu

ABSTRACT

We discuss the security design issues in providing secure updates to the write-shared object among users across different administering domains. In general, it is difficult to assume a dedicated central server for serializing updates and authenticating collaborators in a write-write sharing across administering domains. Hence, we have proposed a decentralized inter-domain data management method, which has been exemplified in Self-administering Data Handler Model [1]. This paper discusses security issues of the SDH: the authentication (identity), the authorization (access control), the data integrity, content confidentiality, and the traffic confidentiality. We also discuss that the SDH model is not more vulnerable to the computer virus attack than other inter-domain applications.

1. Introduction

Traditionally, Network File System provides a model of transparent file access/management semantic that allows the user to access/manage remote files in the same way as local files are accessed. In most real implementation such as Sun's NFS, Sprite File System, and Andrew File System, read/write caching is used to provide such transparency to the user by amortizing the network delay over multiple subsequent reads or writes. For example, in a typical write-write sharing of remote file between more than two users, each user's write is immediately applied to the write cache on their local machine and waits 2-30sec (Sun's NFS) or 30sec (Sprite FS) to absorb any subsequent writes before the written-cache is sent to overwrite the original remote file.

The model of transparent file management has limitations. First, although each user keeps their most-recent update in their local cache, they cannot undo or redo their own update. If they maintain explicit copies and its versions instead of the most-recent cache, the redo/undo/versioning can be done locally without involving remote repository. Second, in this model, one of the collaborators has to dedicate a shared resource that serves as a file repository to provide a single coherent view among collaborators, which is not feasible all the time especially across different administrative domains. Each domain (realm) likes to

maintain their own explicit copies rather than implicit caches that depend on a centralized shared repository.

We have proposed an inter-domain data management protocol as exemplified in Self-administering Data Handler Model [1]. In this paper, we address the security issues of the SDH: the authentication (identity), the authorization (access control), the data integrity, content confidentiality, and the traffic confidentiality. We present the data-domain concept and compare it with the concept of realm in Kerberos [2] [3]. In traffic analysis section, we proposed a radical approach using a "rendezvous server" to deter the traffic analysis based on network packet header. We also investigated whether the SDH model is more vulnerable to the computer virus attack than other collaborative inter-domain applications such as email and web browser.

2. Self-administering Data Handler Model

In this section, we briefly introduce the SDHandler (SDH), as an inter-data domain data management mechanism that we proposed in [1]. In this model, the SDD(Self-administering Data Description), a declarative description of how a data object should behave, is attached to the data object. The description specifies how and to whom the data should be transferred, how it should be incorporated when it is received, and the kind of relation that should exist between distributed copies of the data object.

2.1 Network of SDHandlers

As shown in [1], we envision a network of SDHandlers, each "close" to a user or device that it serves. To a first approximation, there would be one SDHandler per networked device, perhaps more. Some would be associated primarily with users, some with data collection in devices, others with services, such as digital object repositories, each supporting basic SDHandler functionality, but perhaps implementing services associated with the particular characteristics of its application. Such a network is illustrated in Figure 1. SDHandlers form a network, within which data is moved in accordance with the SDHandler's discipline. In addition, each SDHandler may provide an interface to a local collection or stream of data. The data may be a user's file system, web space, database, or other collection, administered by some mechanism other than the SDHandler. While these may be administered by a wide variety of mechanisms, the data looks the same once it

is with the SDHandler network. We refer to each diverse collection of data as a data *domain*. In effect, the SDHandler bridges a data-domain into the SDHandler infrastructure.

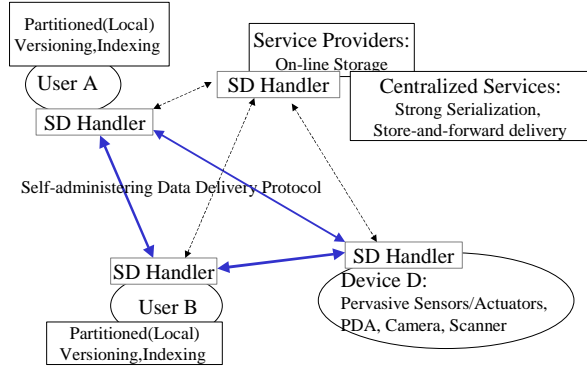


Figure 1: Network of SDHandlers

2.2 Scalable Update Propagation Model

We have developed a scalable update propagation model based on cliques. We define a clique as a strongly connected group of users who share the same SDD (Self-administering Data Description). SDD is meta-data describing how the data should behave, to whom the data needs to be send and how the data need to be incorporated when it is received. As shown in Figure 2, the update for the data, A:foo, is reported back to the central coordinating user/server, A in the diagram, and then propagated back to the rest of the participants in the central model. In a peer to peer model, as shown in Bayou's anti-entropy algorithm [6], the updates are reconciled among peers in an arbitrary order whenever they are connected to each other; then the update is propagated to other members. In our model, changes are immediately propagated within cliques; secondary propagation to other cliques is through the junction point, as shown in B, C in the "cliques" diagram. We can imagine that B and C can filter and aggregate the changes made by members in the clique, so that the tiny changes made within B's clique and C's clique are not visible to A. We believe this approach will fit naturally to group interaction.

Both central and the peer-to-peer propagation models use the same name for all the propagation. However, in our model, the naming carries more clique interaction information. For example, given C:A:foo, one can deduce that this object originated from data A:foo and that user C created a new clique by creating a new SDD associated with C:A:foo. The SDD for A:foo would have a different

member's policy and serialization preference. For the data, A:foo, the clique composed of A,B,C would specify strong serialization and back-up support, while, the clique composed of B,D,E,F might not need to use such strong serialization and back-up support among them.

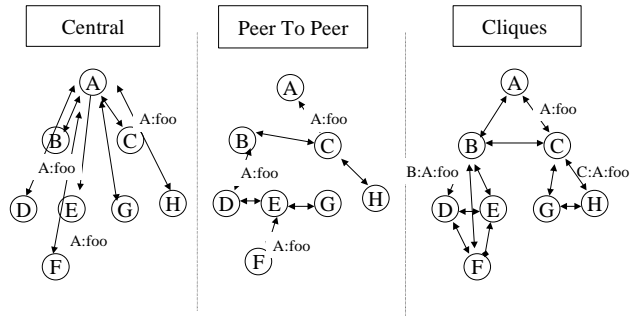


Figure 2: Update Propagation Models

2.3 Self-administering Data Delivery Protocol

The Self-administering Data Delivery Protocol is illustrated in Figure 3.

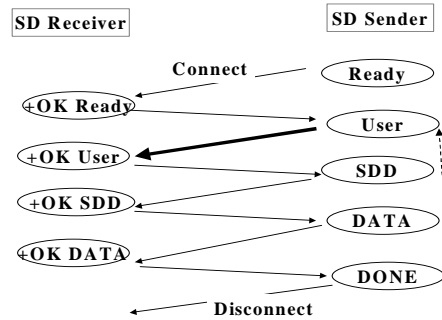


Figure 3: Self-administering Data Delivery Protocol

The SD Sender parses the SDD file written in SDDL (Self-administering Data Description Language) and makes a connection to each Sharer's SD Receiver. When it receives the "+OK Ready" message from the SD Receiver, it sends the SD Sender's user id, (generally an email address). The SD Receiver then checks the membership of this user's id in its SD Receiver's trustee list. If it is not on the list, the SD Receiver refuses the connection and the SD Sender sends an email notification to the user of the SD Receiver. If the user id is found on the trustee list then the SD Receiver tries to verify whether the user of the SD Sender is who she claims to be. For user authentication, we use a public-private key encryption system, where the user of the SD

Sender's identity are verified using a digital signature. To avoid the man-in-the-middle-attack, all the links between the SD Sender and the SD Receiver can be encrypted.

After the authentication is finished, the SD Sender sends the SDD to the SD Receiver, followed by the DATA command. The SD Receiver then parses the received SDD and follows its description, which may involve synchronizing the shared document with latest copy given by the DATA command.

If the SD Sender has more files to be sent, it repeats its protocol from the authentication point onwards, until all files have been received; then the connection is closed. The SDDP thus provides a selective delivery acceptance mechanism using a trustee list. Only the owner of the trustee list can add a new user id into its own trustee list.

3. Authentication and Authorization

In general, authentication means to verify the claimed identity of the principal and the authorization means the process of looking up the access rights of the principal and allowing an appropriate access to the requested service.

3.1 Definition of Data-Domain

We define a data-domain as a trusted boundary that contains three components: authenticated (certified) users (including owners), data objects, and the access control list (ACL) to specify the user's access right to the objects (i.e. authorization). We define the owner of the data-domain as a principal who has administrative access to the data-domain. The owner can set up the policy and specify the access rights of authenticated users to the objects in a given data-domain.

3.2 Data-Domain vs. Realm

In Kerberos, the *realm* is defined as subset of the users/servers/services registered with a particular authentication server[2]. The concept of user's data-domain may look similar to the concept of *realm* in Kerberos system; however it is quite different in that the realms are divided by the authentication boundary while the data-domains are divided by the authorization boundary.

Basically, realm maintains the mapping between the user identity and its credential such as hashed password or shared secret key; however data-domain maintains a mapping (ACL) between the data objects and the access rights of authenticated users. For example, a file repository server/service in realm can be a data-domain and the data-domain verifies the authenticated user's identity by trusting the authentication server of the realm and authorizes the user's requests based on the ACL that it maintains.

3.3 Authentication

In SDH, we use a CA (Certificate Authority in PKI) as an authentication server and verify the certificate based on the trust-relationship with the CA. Once we have trusted CA's public key in SDH, the verification of certificate can be done offline. We could also use the web-of-trust model, as in PGP; however, since it requires a management of key-ring, we decided to use CA's certificate as an authentication method so that SDH can focus on the management of authorization (ACL) and the update incorporation of data-domain. In essence, the data-domain takes the certified statement about the user's identity either from trusted CA or the consensus of trusted friends.

3.4 Authorization (ACL per Data-domain)

The data-domain maintains the ACL specifying each collaborating users' access rights for each data objects within the data-domain. In Unix File System, the ACL is maintained per data objects for three principal: owner, group, and public. It requires human administrator to add/delete the member to the group principal and hence rarely used for specifying fine-grained access controls in collaboration. However, the Microsoft NT file system allows the data owner to specify the access rights per each collaborating users in the sharing group. In SDH, the owner of the domain can set up the rights of the each principal per given data objects.

3.5 Replay Attack

An active attacker, named Malory, can intercept the encrypted and authenticated packet and can replay it to bypass the authentication step in SDH. In general, the replay attack can be dealt with by using timestamp or nonce. Timestamp method requires clock synchronization and its clock skew can open a window of vulnerability, and the nonce method requires extra round-trip and computation cycle.

3.6 Data Integrity

Malory can tamper the data bits in transit between SDHandlers. SDH uses Digital Signature Algorithm (DSA + SHA1) to provide the integrity of the data and verify the authenticity of the data based on sender's identity.

3.6.1 Forward/Backward Integrity

Assume one of the collaborators, named Malory, has to leave the group, then how we can prevent him from hampering the copies that I had before in my domain and the copies that I will have in the future. By removing the Malory's write/execution access rights from the ACL of the collaborated data object, we can have the forward integrity. The backward integrity can also be provided since the SDH will not allow the Malory to write any previous versions in the data-domain. If there is new member who recently acquired the write access cannot make changes to the

previous versions either since the update will create a new version not overwriting any previous versions.

3.6.2 Undo/Redo with Versioning

Since the SDH provides versioning of the data objects, we can always undo from honest mistake or malicious collaborator's legitimate update. The undo function is basically performed by creating a new version with the content of the previous version. Hence there is no undo functionality but we can have undo effect. This feature has desirable security feature that we don't need to worry about losing a data by allowing an update from a malicious users.

4. Confidentiality (Privacy)

Suppose Alice is an investor and Bob is the CEO of a hypothetical company 'CalBest', and they communicate with each other using the SDH. Alice and Bob want to hide what they are doing from any eavesdropper. An adversary can take important information not only by seeing the content of their conversation (snooping), but also simply inspecting the network traffic such as packet header, time, and its burstiness (traffic analysis) and thus can find: (i) the existence of the inter-collaborative relationship between Alice and Bob, (ii) the fact that the traffic between Alice and Bob is unusually high or low for the moment. In SDH, we assume that the Alice and Bob are in good faith and they don't need to hide their identity from each other but they want to protect properties (i) and (ii).

4.1 Content Confidentiality (Snooping)

We discuss how we can provide the secrecy of the content of the data in SDH.

4.1.1 Hybrid Method

We can prevent an eavesdropper from snooping the message text (content confidentiality) by combining the secret key cryptography and the public key cryptography. After encrypting the text with a randomly generated session key, Alice encrypts the session key with Bob's public key. Then, she prefixes encrypted session key to the encrypted text and sends it. Bob reads the plain text in a symmetric way. After getting the session key by using his private key, he decrypts the cipher text with the session key.

This is the typical way of enforcing confidentiality and it is also used in SDH.

4.1.2 Forward/backward Secrecy

Assume one of the collaborators, named Malory, has to leave the group, then how we can prevent him from snooping the copies that I will have in the future. By removing the Malory's read/write/execution access rights from the ACL of the collaborated data object, we can have the forward secrecy.

The backward secrecy is much difficult in general. If the snooper has recorded all of the previous encrypted data and were able to crack one of the collaborator's private keys then the snooper can be able to read all the previous secrets. By revoking the compromised public key, we can have forward secrecy. Since we assume the size of clique in SDH is small, we don't need to worry about the scalable group key revocation. We simply unicast the revocation message to all the participants and the CA.

Since we use the session key to encrypt the data, the newly joined member cannot be able to look it up even if he/she has collected all the previous encrypted traffic. Hence, we have backward secrecy.

4.2 Traffic Confidentiality (Traffic Analysis)

Usually, traffic confidentiality means disguising who is participating (anonymity of sender or receiver) and who sends to whom (the unlinkability of sender and receiver). And these can be achieved by simply introducing an anonymizing proxy (agent) server. However the agent server has to be trusted not to reveal the client's identity. We show some ways to evade attacks of traffic analysis including "rendezvous server" which is a radical solution used in SDH.

4.2.1 Periodic Connection Avoiding Traffic Analysis: Mixes

We can think of a way to avoid traffic analysis by pretending to send messages periodically and this meets property (ii) but not (i) which are mentioned above.

At each time a proxy called a mix makes a message either by taking a message from a sender, Alice, or by generating a bogus message. After encrypting the message using public key cryptography, the mix sends it to the recipient, Bob. There may be more than one mixes in the path from Alice to Bob. In this way, we can achieve sender and receiver unlinkability against a possible eavesdropper. Mixes have been used in many types of communication such as e-mail, ISDN, and general synchronous communication.

However, this strategy is subject to some caveats. First, its dependence on computationally expensive public key cryptography makes it not fast. Next, a message gets inefficiently large to cover any big data in a chunk.

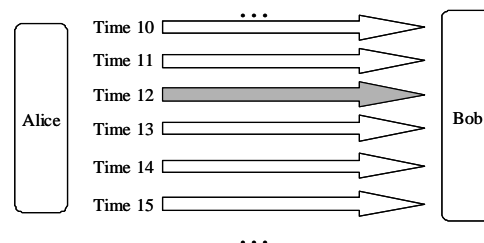


Figure 4: Using Mixes

4.2.2 Avoiding Traffic Analysis: Anonymous Remailer

This protect both property 1 and 2 by introducing a trusted agent.

We can disguise the identity of participants by using agents we trust, called an anonymous remailer or anonymizer.

In this scheme, Alice sends a mail for Bob to a remailer rather than to Bob directly. Then, the remailer changes the sender from Alice to the remailer and forwards it to Bob. An eavesdropper may find that Alice sends something to a remailer and Bob receives something from the remailer, but cannot easily find that Alice sends something to Bob as long as there are many others users in the remailer.

The idea of hiding identity through a remailer has already been used in different kinds of applications such as e-mail, World-Wide-Web, and so on. The Anonymizer (<http://www.anonymizer.com/>) and the Lucent Personalized Web Assistant are among the real examples.

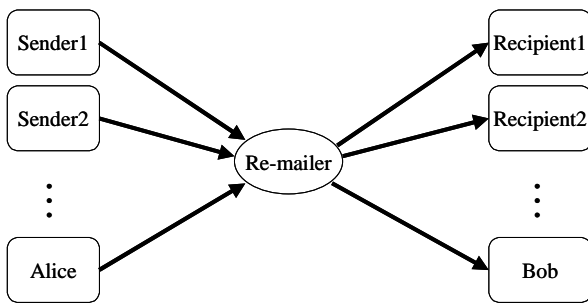


Figure 5: Using anonymous remailer

4.2.3 Avoiding Traffic Analysis: Rendezvous Server

We can have anonymity and unlinkability by using the anonymizing agent. But the agent has to be trusted and it has to keep the mapping between the sender and its next destination in the database. In addition, using an anonymizing agent may not work well when the network is cut off or the recipient is out of reach when the remailer tries to send data.

Let's assume that many people share a rendezvous server to which each user makes a connection periodically. During this scheduled connection time, the user uploads data (either real or bogus) and downloads all the data that has been uploaded on the server presumably by other users.

Since all the members make connections at the same rate and downloads all the messages, an eavesdropper has difficulty in detecting whether a couple of members communicate with each other or a member sends or

receives unusually many messages. Thus it meets property (i) and (ii).

How can a member, Bob, pick up the data sent from another member Alice to him while avoiding attacks by traffic analysis? Alice and Bob can share an identifier only between them by using public key cryptography. Before uploading data, Alice encrypts the identifier and prefixes it to the data. When Bob receives his data, he downloads all the data in the file pool at first, and chooses only the data whose header matches the predetermined identifier to confuse an eavesdropper.

By scheduling an expiration period and a user's connection time, we can prevent the recipient from spending unexpectedly too much time getting his data; all the uploaded data is deleted after a expiration time and senders and recipients do their jobs only once in a period.

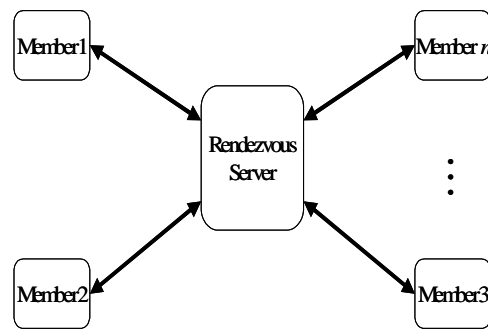


Figure 6: Using shared file pool

5. Vulnerabilities to computer viruses

One may ask whether the SDHandler is safe from the viral attacks. Before we talk about this, we need to remind ourselves of computer viruses and their characteristics.

5.1 Computer Viruses and their characteristics

A computer virus is a program that spreads itself by infecting executable files and making copies of itself. The effect of a virus depends on how it was programmed, but its effect can be categorized to some groups: damaging files, disclosing secret information to others or reformatting disks. And a virus can exist in different forms: An executable program in a file system, a bootstrap code, or an attached executable file in e-mail [10],[11],[13].

In whatever form it exists, a virus is activated only when it is executed. We can prevent the attacks of viruses by not executing programs whose sanity is dubious. An anti-virus program helps us to determine whether an executable program is safe. It scans a file and compares its segments of code with the code of known viruses. If we update the anti-

virus program frequently enough, we can detect most viruses.

5.2 Policy against Computer Viruses in SDH

Suppose that a virus try to spread itself through the SDHandler. Since the SDHandler updates data as described in the Self-administering Data Description (SDD), it can transfer a virus if the data is infected. But, it doesn't run malicious executable programs automatically like some e-mail clients. If we let anti-virus programs cooperate with the SDHandler, we can prevent the situation that the infected data is executed and spread.

However, viruses are made nearly everyday and get more complicated. So even the most frequently updated anti-virus programs cannot detect all the viruses. We need to rely on reasonable policy as well as good anti-virus programs. In most cases, the shared files are pure data like documents, pictures and video clips. And they can be shared without special concerns. We can accomplish safety by enforcing that all the participants check the sanity of the executables and make announcement before sending executables.

This strategy works well even in the situations as follows [12]: In a Windows host, a script file 'picture1.jpg.vbs' can be seen as a picture file 'picture1.jpg', since Windows operating system hides file extensions by default. This is why Anna Kournikova virus was able to attack a large number of hosts. Many were lured to click the attachment thinking of it as a picture of a beautiful tennis star, Anna Kournikova. If we enforce sanity rule for executable files, we can prevent the spread of those kinds of viruses.

Remember that the golden rule is "Wait until it is safe."

6. Discussion

We found the inter-data-domain data management has a few distinctive requirements. First, the protocol has to be designed with failure case in mind without assuming the complete trust relationship between data-domains. If there is tight trust across data-domain, it is not inter-data-domain anymore it becomes a single virtual data-domain. Second, it is hard to keep a state outside data-domain. Each data-domain has to maintain their own state without depending on other data-domain's state.

Another lesson we learned is that the design of secure application requires balancing the performance and the security requirements of the application.

7. References

- [1] B.Hoon Kang and Robert Wilensky, Toward A Model of Self-administering Data, ACM/IEEE Joint Conference on Digital Libraries, JCDL 2001
- [2] B. Clifford Neuman and Theodore Ts'o. Kerberos: An Authentication Service for Computer Networks, *IEEE Communications*, 32(9):33-38. September 1994, <http://www.isi.edu/gost/publications/kerberos-neuman-tso.html>
- [3] J. Kohl and C. Neuman, The Kerberos Network Authentication Service RFC1510, 1993 <http://www.faqs.org/rfcs/rfc1510.html>
- [4] Peter J. Keleher, Decentralized Replicated-Object Protocols, In The 18th Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing, April 1999.
- [5] Jim Whitehead, Collaborative Authoring on the Web: Introducing WebDAV, Bulletin of the American Society for Information Science, Vol. 25, No.1,1998, <http://www.webdav.org/papers/>
- [6] K. Petersen, M. J. Spreitzer, D. B. Terry, M. M. Theimer, and A. J. Demers. Proceedings of the 16th ACM Symposium on Operating Systems Principles (SOSP-16), Saint Malo, France, 1997, <http://www.parc.xerox.com/csl/projects/bayou/>.
- [7] CVS (Concurrent Versions System), <http://www.cvshome.org/>
- [8] PGP (Pretty Good Privacy), <http://www.pgpi.org/>
- [9] Michael K. Reiter, Aviel D. Rubin, Crowds: Anonymity for Web Transactions.
- [10] Nick FitzGerald, Computer Virus FAQ for New Users, Jun. 1999, <http://www.faqs.org/faqs/computer-virus/new-users/>
- [11] Nick FitzGerald, Computer Virus Frequently Asked Questions 2.0, Oct. 1995, <http://www.faqs.org/faqs/computer-virus/faq/>
- [12] MMCC, Inc. E-mail Virus FAQ, <http://www.mmcctech.com/faq-0126.htm>
- [13] Computer Associates Virus Information Center, <http://www.cai.com/virusinfo/faq.htm>