Part I

Security in Distributed Computing

1

Security for Content Distribution Networks — Concepts, Systems and Research Issues

Elisa Bertino and Yunhua Koglin

CONTENTS

1.1 Introduction 4
1.2 Security Concepts 5
1.3 Access Control Models 6
1.4 Systems 7
1.4.1 Secure Distributed File Systems 7
1.4.2 Publish/Subscribe Systems 11
1.4.3 Content-Aware Intermediary Transforming Systems 14
1.4.4 Peer-to-Peer Content Distribution Systems 15
1.4.5 Collaborative Data Access and Updates Systems 15
1.5 Other Research Issues 18
Bibliography 19

Abstract Previous research on content distribution networks (CDNs) mainly focuses on improving system performance by deploying replication such that latency for data access could be reduced and bandwidth could be saved, especially when dealing with large amounts of data. Centrally-managed, trusted replicas are important characters in these traditional CDNs.

However, there is not enough attention given to the security of data in CDNs, even though data security is a crucial need for most Internet-based applications. Moreover, with the emergence of various network appliances and heterogeneous client environments, intermediaries are used for dynamic content delivery. Enforcing data security in such environments is more challenging than the traditional CDNs (client-server communication). Besides, new systems (such as publish/subscribe systems, peer-to-peer content distribution systems) are developed to meet different requirements of content distribution. Different mechanisms should be used in different systems to ensure content security.

In this chapter, we first review the security concepts related to CDNs and then present several systems, focusing on how they enforce content security. Finally, we discuss the other challenges in CDNs.

1.1 Introduction

Content distribution networks (CDNs) are all those applications that support data dissemination, searching, and retrieval. With the widespread use of Internet, CDNs have been studied extensively [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21]. Most previous research focuses on enhancing performance of CDNs by replication. Different mechanisms (such as [22, 23, 24, 25, 26]) are used to deploy content replication on *trusted* cache proxies scattered around the Internet. When receiving a client request, instead of asking a content server for the requested contents, a proxy first checks if these contents are locally cached. Only when the requested contents are not cached or out of date are the contents transferred from the content server to the clients. If there is a cache hit, network bandwidth consumption can be reduced. A cache hit also reduces access latency for clients. System performance therefore improves, especially when large amounts of data are involved. Besides these improvements, caching makes the system robust by letting caching proxies provide content distribution services when the server is down or the network is congested.

Secure content distribution has received more attention from both academia and industry than before, due to the increasing emphasis on security in many applications. Ensuring content security in distributed environments is challenging. For example, content may be easily modified or accessed when it is transmitted across the Internet; a compromised replica may violate access control of content or damage integrity by maliciously modifying the content.

Different kinds of systems have been developed recently in order to meet the new requirements of content distribution. For example, with the emergence of various network appliances and heterogeneous client environments, content-aware systems are developed that involve intermediaries to transform content; publish/subscribe systems are developed to distribute content where publishers do not need to know the addresses of subscribers. These systems are different from the traditional client-server communication. They have different service requirements and different security challenges.

In this chapter, we first introduce the concepts of security related to CDNs, then present several systems, with focus on their security mechanisms. For each kind of these systems, we present its current research. Finally, we discuss some other research issues in CDNs.

P1: Naresh Chandra
January 30, 2007 11:4 AU7921 AU7921˙C001

1.2 Security Concepts

In this section, we briefly review some security concepts that are related to CDNs.

For any systems designed with security among its goals, a detailed security policy should exist.

Definition 1.1
A security policy is a statement of what is, and what is not, allowed ([27]).

Definition 1.2
An access control policy states the privileges of principals or users over content and services under certain conditions.

Security policies could be represented in high-level languages in which policy constraints are expressed abstractly, or low-level languages in which policy constraints are expressed in terms of program options, input, or specific characteristics of entities on system ([27]). Policies should be expressed precisely and unambiguously.

After specifying the security policies, a mechanism is chosen to enforce these policies.

Definition 1.3
A security mechanism is a method, tool, or procedure for enforcing a security policy ([27]).

In general, the security of content distribution systems is measured by how it supports data confidentiality, data integrity, and system availability.

Definition 1.4
Confidentiality is the assurance that content is shared only among authorized subjects.

Definition 1.5
Integrity is the assurance that the information is authentic and complete.

Definition 1.6
Availability is the assurance that the system which is responsible for dissemination, storing, and processing information is accessible when needed by those who need these services.

Both confidentiality and integrity are defined by *access control policies*.

In the next section, we will review some access control models that describe how the access policies for content are generated.

P1: Naresh Chandra
January 30, 2007 11:4 AU7921 AU7921˙C001

1.3 Access Control Models

In CDNs, an access control model specifies who is allowed to perform what kinds of operations on content under certain conditions. The following types of access control model are commonly used:

- *Discretionary access control (DAC)*: Access policy is completely determined by the owner of the content. The owner decides who is allowed to access the data and with what privileges (such as *read, write*, etc.).

  This type of access control has been widely used, even beyond CDNs. For example, Alice creates a file called temp.c. She can specify which subjects may access it and with what type of access (such as read or write). An access control list is normally used to make access decisions. Users usually present credentials (such as login and password) for authentication.
- *Mandatory access control (MAC)*: Access policy is determined by the system, not the owner of the content. In such a system, subjects receive a clearance label and objects (data) receive a classification label, also referred to as security level. A subject cannot read anything up, which means that a subject cannot read any objects that have labels higher than the subject's clearance. Moreover, a subject cannot write anything down, which means that a subject cannot write to objects or create new objects with lower security labels than the subject's clearance. This prevents subjects from sharing secrets with subjects with a lower security label, keeping information confidential.

  Note in MAC, only administrators can change the security labels of data. Data owners cannot make such a change.

  MAC is often used in systems that process highly sensitive data with confidentiality as the highest priority, such as classified government and military information. The original MAC model [28] (also called Bell-LaPadula model) was later expanded to Multi-Level Security (MLS), which handles multiple classification levels (i.e., "top secret," "secret," "confidential," and "unclassified") between subjects and objects.
- *Role-Based Access Control (RBAC)*: Access is dependent on functionality, not identity. In RBAC models ([29, 30, 31, 32, 33, 34, 35]), an administrator defines a series of roles that are created for various job functions. The permissions to perform certain operations are assigned to specific roles. An administrator assigns members of staff (or users) some roles, and through those roles members (or users) acquire the permissions to perform particular functions.

RBAC can save an administrator from the tedious job of defining permissions per user within an organization.

When defining an RBAC model, it normally includes the following relations:

– $UA \subseteq U \times R$ User-role assignment (a many-to-many mapping)

– $PA \subseteq P \times R$ Permission-role assignment (a many-to-many mapping)

– $RH \subseteq R \times R$ Partially ordered role hierarchy

where U = User, R = Role, P = Permissions

Moreover, a RBAC model normally includes a set of sessions (SESSIONS) where each session is a mapping between a user and an activated subset of roles that are assigned to the user. Such a model may also include function *session_roles* that returns the roles activated by the session and the function *user_sessions* that returns the set of sessions that are associated with a user.

A RBAC model may also have other features such as: 1) roles are granted permissions based on the principle of least privilege; 2) roles are determined with a separation of duties; 3) roles are activated statically or dynamically.

Some other access control models include:

- *Originator Controlled Access Control (ORCON)*: The originator (subjects or organizations who create data) controls data access. Note that the originator may not be the data owner. ORCON is a combination of MAC and DAC ([27]).
- *Rule-Based Access Control model*: This is sometimes referred to as Rule-Based Role-Based Access Control (RB-RBAC). It includes mechanisms that dynamically assign roles to subjects based on their attributes and a set of rules defined by a security policy ([36]).

1.4 Systems

In this section, we present several types of systems in CDNs, focusing on the current research in these systems.

1.4.1 Secure Distributed File Systems

One important application in CDNs is file distribution. Instead of storing files on the machines owned by the data owners, some owners put their data in a data server, which is responsible for distributing the data according to the access control policies related to the data. This approach not only removes the space requirement for the data owners, but also makes the data distribution scalable.

Most previous file distribution approaches are based on the assumptions that the data servers are trusted: They keep the confidentiality and integrity of the data, and they enforce the access control policies related to the data. However, these assumptions are hard to prove true. In the following text, we present some current research on distributed file systems that removes these assumptions.

Current Research

Current research on distributed file systems with untrusted data servers includes the following aspects:

- **Cryptographic access control**. Harrington and Jensen propose a cryptographic access control mechanism in [37]. Files are encrypted and stored on an untrusted server. Access control is enforced by distributing symmetric keys that are used for encrypt/decrypting files. Integrity of the files can be verified by the server with signature verification, even though the server may not access the file content. The files are maintained with modifications recorded in a log.

  The above approach provides a nice solution that gets rid of a centralized reference monitor, such that the server does not need to maintain an access control list for the file and enforces this access control policy. Users can read the log that is signed by the data owner with timestamps or version numbers.
- **Supporting operations on encrypted data:** Moving the computation to the data server that stores only encrypted data seems very difficult; the data server should perform the computation without decrypting the data. Song and others [38] propose a practical technique for searching on encrypted data. Their solution supports the following:
  - *Provable Secrecy:* The untrusted server cannot learn anything about the plaintext given only the ciphertext.
  - *Controlled Searching:* The untrusted server cannot search for a word without the user's authorization.
  - *Hidden Queries:* The user may ask the untrusted server to search for a secret word without revealing the word to the server.
  - *Query Isolation:* The untrusted server learns nothing more than the search result about the plaintext.

  Before presenting the protocol, we first introduce the notations it uses. If $f : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ represents a pseudorandom function or permutation, then $f_k(x)$ is the result of applying f to input x with key $k \in \mathcal{K}$. $\langle x, y \rangle$ means concatenation of x and y.

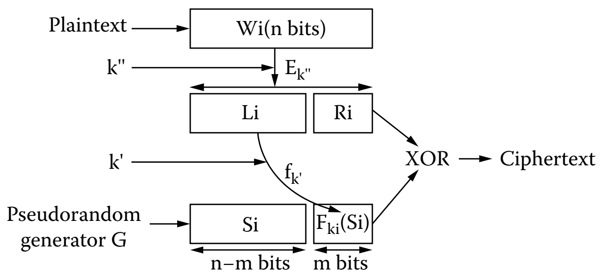

FIGURE 1.1
Encryption scheme (from [38]).

The protocol [38] has the following components:

1. *Storing data on the untrusted servers:* For each block W_i which has the fixed length of n, Alice gets the pseudorandom value S_i ($n - m$ bits long) from the pseudorandom generation G. Alice computes the ciphertext to be stored for W_i as $C_i = E_{k''}(W_i) \oplus \langle S_i, F_{k_i}(S_i)\rangle$ where $k_i = f_{k'}(L_i)$ and $E_{k''}(W_i) = \langle L_i, R_i\rangle$. L_i (respectively, R_i) denotes the first $n-m$ bits (respectively, the last m bits) of $E_{k''}(W_i)$. At the end, Alice keeps k', k'', and S_i, and sends the ciphertext to Bob (untrusted server) who stores the ciphertext. Figure 1.1 shows the encryption steps.
2. *Search operations:* To search the positions for word W_j, Alice sends Bob $X_j = E_{k''}(W_j) = \langle L_j, R_j\rangle$ and $k_j = f_{k'}(L_j)$. Bob performs a sequential scan on the encrypted data and returns $\langle p, C_p\rangle$ if $C_p \oplus X_j = \langle S_p, S'_p\rangle$ and $S'_p = F_{k_j}(S_p)$. In the returns, p denotes the position of the word. Note that there is small chance that some answers returned by Bob are garbage. This is due to the encryption collision.
3. *Retrieval operations:* To retrieve the data stored at position p, Alice sends Bob p. After Bob returns the ciphertext C_p at position p, Alice recalculates W_p by $C_p = \langle C_{p,l}, C_{p,r}\rangle$ where $C_{p,l}$ (respectively, $C_{p,r}$) denotes the first $n - m$ bits (respectively, the last m bits) of C_p, $X_{p,l} = C_{p,l} \oplus S_p$, $k_p = f_{k'}(X_{p,l})$, $T_p = \langle S_p, F_{k_p}(S_p)\rangle$, and finally, $W_p = D_{k''}(C_p \oplus T_p)$.

From the above description, we can see that each query takes one round of interaction and Bob performs one sequential scan on the ciphertext per query.

- *Proxy Re-encryption*
In 1998, Blaze, Bleumer, and Strauss (BBS) [39] proposed an application called atomic proxy re-encryption, in which a semitrusted proxy converts a ciphertext for Alice into a ciphertext for Bob without seeing the underlying plaintext. This strategy is useful when Alice would like temporally to let Bob check the messages that are

addressed to her, without revealing to Bob her secret keys that are needed to decrypt these messages.

Ateniese et al. ([40]) present an application for proxy cryptography in securing distributed file systems. A centralized access control server is used to manage access to encrypted files stored on distributed, untrusted replicas. A proxy re-encryption scheme is proposed such that the access control server could re-encrypt the appropriate decryption key to clients without learning the key in the process. Thus, there is no need to grant full decryption rights to the access control server.

- *Byzantine fault tolerance*
Besides using replication to increase content availability, other research focuses on byzantine fault tolerance. There are two types of system failure: fail-stop, which means data servers simply do not reply to clients' requests, and malicious failure, which means the data servers may behave arbitrarily; that is, they may reply with the wrong information to clients' requests.

Castrol and Liskov ([41]) propose an approach that tolerates byzantine fault in *asynchronous* systems like the Internet. Their solution ensures that the system that includes a set of replicas performing deterministic services could survive byzantine faults. Moreover, their solution guarantees safety and liveness. In the system, a client sends the request for an operation to the primary of the replicas. The primary then multicasts the request to the other replicas, which then execute the request and send a reply to the client. After the client receives $f + 1$ replies from different replicas with the same conclusion, this is the result of the operation. The algorithm performed by replicas only requires five rounds of messages.

The protocol in [41] has the following steps[1]:

1. **Request:** Client c sends a request message $m = \langle REQUEST, o, t, c\rangle_{\sigma c}$ to the primary p, where *o=operation, t=monotonic timestamp*.
2. **Preprepare:** Primary p assigns sequence number n to m and sends a message $\langle PRE\text{-}PREPARE, v, n, m\rangle_{\sigma p}$ to other replicas where *v=current view*.
3. **Prepare:** If replica i accepts the message from p, it sends $\langle PREPARE,v,n,d,i\rangle_{\sigma i}$ to all other replicas, where d is the hash of the request m from client c. This indicates that i agrees to assign n to m in v.
4. **Commit:** When replica i has a *PREPREPARE* and $2f + 1$ matching *PREPARE* messages, it sends $\langle COMMIT, v, n, d, i\rangle_{\sigma i}$ to all other replicas. At this point, correct replicas agree on an order of requests within a view.

[1] Message m signed by node i is denoted as $\langle m\rangle_{\sigma i}$.

5. **Reply:** Once replica i has $2f + 1$ matching $PREPARE$ and $COMMIT$ messages, it executes m, and sends to client c a message $\langle REPLY, v, t, c, i, r\rangle_{\sigma i}$ where r is the result of the operation.

The above approach requires at least $3f + 1$ replicas, where f is the max number of faulty replicas. It can tolerate malicious clients. Number of optimizations are described in [41] in order to have the proposed approach perform well in real systems.

1.4.2 Publish/Subscribe Systems

Publish/Subscribe (pub/sub) systems provide a new distributed paradigm for content dissemination. In such systems, a publisher publishes an event (or message) through a broker (also referred to as an event dispatcher). Subscribers specify their interests by registering with a broker. Brokers form a network in which they forward events to each other and, when needed, deliver events to subscribers that have registered with them. One major advantage of these systems is scalability: A publisher does not need to maintain subscription information, which may be changed dynamically, and a subscriber does not need to know which publishers may publish events of interest. Since there are no explicit destination addresses associated with an event, brokers are responsible for delivering each event to subscribers whose subscriptions are satisfied by the event, which is called event *matching*.

Figure 1.2 presents a general structure of pub/sub systems. Decoupling publishers from subscribers makes pub/sub systems scalable and powerful.

Basically, there are two types of pub/sub systems. The first, referred to as *subject-based* or *type-based* pub/sub, is a system in which events are labeled with predefined subjects (or types) to which subscribers may subscribe.

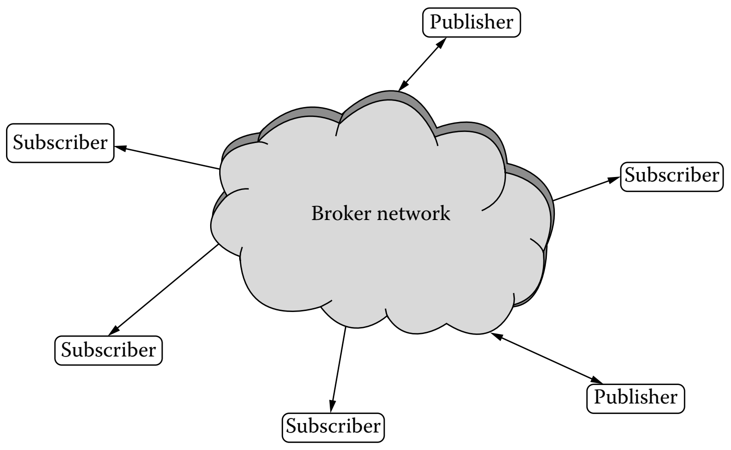

FIGURE 1.2
A general pub/sub system structure.

Since subscribers interested in a particular subject (or type) may be managed as a group, multicasting is an efficient method for event delivery in this kind of pub/sub system. The second one, referred to as *content-based pub/sub* system, is more flexible and powerful than the previous one. In this kind of system, both subscriptions and content are specified with respect to a set of *attributes*. Each attribute is an ordered pair of *name* and *type*. An attribute value is the type of the attribute. A subscriber subscribes to events by specifying *predicates* against attributes. For example, a classic schema used for a stock trade pub/sub system is (*company: string, price: integer, shares: integer*), a subscription could be: *(price < 20) AND company = "IBM."*

Current Research

Two important security issues for content-based pub/sub systems are *availability* and *confidentiality*. Availability in such a system means that after a subscriber registers with a broker by specifying subscription predicates, if the subscription request is accepted, the broker is responsible for delivering any event that satisfies such predicates to the subscriber in a timely manner. Furthermore, any broker failure along the event delivery path should not prevent the subscriber from receiving this event. This requirement means that if there is a broker failure, either the event delivery route should be reconfigured dynamically or multiple event delivery routes should be established at the beginning in such systems. For a large pub/sub system, it takes time to propagate subscription information into the whole network, therefore it may take a while for a subscriber to receive events. As a result, the subscriber may miss some matching events, even though these events are published after the registration. It is thus important that the *registration information propagating time (RIP time)* be minimized.

Confidentiality in pub/sub systems means that events should be available only to authorized subscribers. Malicious users must be prevented from reading events for which they do not have the proper authorization. Furthermore, even subscribers whose predicates do not match an event must not access the event. Therefore, key management and efficient encryption/decryption schemes play an important role in enforcing event confidentiality.

Next, we describe the current research on these two issues in the context of content-based pub/sub systems, since solutions in these systems could be easily applied to the subject-based pub/sub systems and the reverse is not true.

- **Availability** In most approaches, such as [42, 43], an event is distributed along a spanning tree, in which the root is the broker from which an event is published. Leaf nodes and some inner nodes are brokers that have subscribers requesting such events. If a broker at the root of a tree fails, either the events are lost or a reconfiguration (such as [44]) must be performed to rebuild the tree. Such a reconfiguration can be very expensive when pub/sub systems are large and a number of brokers are involved. Maintaining a tree structure for event forwarding also requires each broker replicate the whole network's subscription information. Therefore, the RIP time delay could be large and could further increase if brokers perform computations in order to minimize routing table information.

A simple approach to increase availability is to let each broker broadcast each event it receives. However, such an approach has the major disadvantage of resulting in system floods. Carzaniga et al. ([43]) propose an approach that broadcasts events only along the spanning tree, therefore, some unnecessary event broadcast could be avoided and event availability could be improved.

Srivatsa and Liu ([45]) propose a resilient network, which, instead of providing only a single path from each publisher to its subscribers, which is inherited from the spanning tree structure, several independent paths from a publisher to each of its subscribers are provided. These paths are built in a deterministic way. In their approach, building several independent paths from a publisher to every subscriber involves complex topology computations. In dynamic environments where subscriptions or unsubscriptions occur quite frequently, such computation is expensive.

Other approaches to improve availability such as multicasting an event by the broker that publishes the event also requires replicating subscription information at each broker. Besides the long time of RIP delay, broker space requirement is another challenge for the multicast approach in large scale pub/sub systems. This approach also causes the load unbalance, as some brokers where events are published frequently are overloaded, while other brokers that do not have an event published are idle.

- **Confidentiality** An event should be encrypted when it is delivered to subscribers, so that only authorized subscribers are able to decrypt it. Usually, a group key shared by the group members and the brokers is used to encrypt the event. However, since there could be many attributes and therefore a large number of complex predicates, for n subscribers, there are possibly 2^n subscription groups that may be interested in an event. Encrypting the event with a group key therefore could result in significant performance costs. Moreover, different events may be of interest to different sets of groups. In large-scale content-based pub/sub systems where the volumes of published events are huge, inefficiency may undermine availability.

  Opyrchal and Prakash [46] discuss how a broker can encrypt an event and deliver it to a possibly very large number of groups. As each group has a secret key shared by members and brokers, encrypting the event using a group key may involve performing many encryption operations, and there may be several groups to which this event should be delivered. Caching and clustering are therefore used to make fewer encryptions for an event.

Security issues in content-based pub/sub systems have not been so widely investigated. More detailed discussion on these issues can be found in [47].

1.4.3 Content-Aware Intermediary Transforming Systems

With the emergence of various network appliances and heterogeneous client environments, besides caching, there are other new requirements for content services by intermediaries [6, 7]. For example, content from the server needs to be transformed in order to adapt to the requirements of a client's security policy, device capabilities, preferences, and so forth. Several *content services* have been identified that include, but are not limited to, content transcoding [6, 7, 8, 13], in which data is transformed from one format into another, data filtering and value-added services, such as watermarking [10].

Intermediaries providing content services can be placed at the clients' end, at the servers' end or between them [12, 26]. Placing intermediaries at the client's end may not always be possible because of resource limitations. Because of these limitations, it is not possible to execute certain computation intensive transcoding functions at the clients' end. Placing intermediaries at the servers' end may result in reduced sharing. It is difficult to have one version of some content that satisfies diverse requests from clients. Placing intermediaries between clients and servers provides a better solution for content services.

Current Research

Though a lot of research on intermediary content service has been carried out [6, 7, 8, 13], there is not enough research on data security in this context. The approaches provided for securely transferring data from server to clients are not suitable when data are to be transformed by intermediaries. When a proxy mediates data transmission, if the data are completely enciphered during transmission, security is ensured; however, it is impossible for intermediaries to modify the data. It is difficult to enforce security when intermediaries are allowed to modify the data. Next, we list several research topics in this area:

- *SSL Splitting:* SSL splitting ([48]) is a technique that supports data integrity from untrusted caches. Upon receiving a request from clients, a proxy gets the data from caches and the Message Authentication Code (MAC) from the data server. Then, the proxy re-encrypts the merged data with the key shared by the server and the client and sends the encrypted data to the client. SSL splitting does not support data confidentiality, as the proxy has to access the key. The primary advantage of SSL splitting is that it reduces the bandwidth load of the data server.
- *Data Integrity Service Model:* Chi and Wu propose a Data Integrity Service Model (DISM) in [9]. In this model, integrity of intermediaries is enforced by using metadata expressing modification policies of content owners. However, in DISM everyone can access the data.

Thus confidentiality is violated. Another problem with DISM is the lack of efficiency. In several applications, such as multimedia content adaptation [6], efficiency is a vital factor.

- *JPSEC:* Wee and Apostolopoulos [19] present encryption methods and signaling syntax for JPEG-2000 images that allow an intermediary to transcode a JPEG-2000 codestream (JPSEC) without decryption. After unlocking the transcoded JPSEC, the transcoded JPEG2000 can be decoded to get the transcoded image. Moreover, an end user can verify that the transcoding operation was performed in a valid and permissible way.

1.4.4 Peer-to-Peer Content Distribution Systems

Peer-to-peer systems are characterized by the direct sharing of computer resources (such as content, storage, or CPU), rather than requiring the intermediation or support of a centralized authority.

Current Research

Many distributed file systems have been developed in peer-to-peer networks ([49, 50, 51, 52, 53, 54, 55]). These systems (such as Napster [49], Gnutella [50], and Freenet [51]) demonstrate a lot of benefit for content distribution. These benefits include node self-organization, load balance, fault tolerance, and scalability. Due to the lack of centralized administration and management, it is hard to ensure security in such environments. Androutsellis and Spinellis [56] have an extensive survey on the peer-to-peer content distribution technologies. Therefore, we will omit the discussion on the security issues in these systems. Interested readers are encouraged to read [56].

1.4.5 Collaborative Data Access and Updates Systems

The widespread use of the Internet for exchanging and managing data has pushed the need for techniques and mechanisms that secure information when it flows across the net. When several parties collaboratively perform certain transactions, each party needs to retrieve content and then perform certain authorized operations on it. Integrity and confidentiality have to be ensured for the data that flow among these parties.

Current Research

Several issues need to be addressed to support decentralized and cooperative document updates over the Web. A first requirement, which was investigated in [57], is the development of a high-level language for the specification of *flow policies*, that is, policies regulating the set of subjects that must receive a document during the update process, and *access control policies*. Starting from these policies, a server can determine the path that the document must follow and the privileges of each receiver. The second requirement is the development of an infrastructure and related algorithms to enforce confidentiality and integrity during the process of distributed and collaborative document updates.

- *Author-χ System:* This Java-based system ([58, 59, 60]) supports selective, secure, and distributed dissemination of XML documents. Specifically, Author-χ supports
  - the specification of security policies at varying granularity levels
  - the specification of user credentials
  - content-based access control
  - controlled release of XML documents according to the push-and-pull dissemination modes
  - document updates

  The system includes three Java server components: 1) X-Admin, 2) X-Access, and 3) X-Update.

  X-Admin component provides functions for administrative operations. Through this component, security administrators manage security policies, XML documents, subjects and credentials.

  X-Access component consists of two subcomponents: X-Push and X-Pull. X-Push supports document broadcast to clients at the server site. X-Pull supports the selective documents distribution upon clients' requests. All these kinds of distribution follow the policies stored in Policy Base ($\mathcal{PB}$).

  X-Update component manages the collaborative and distributed document update that we will describe later.

  Author-χ also includes X-bases repositories that consist of the following:
  - Policy Base ($\mathcal{P}B$) that contains the security policies for the XML documents and DTDs
  - Credential Base ($\mathcal{C}B$) that contains the user credentials and credential types
  - Encrypted Document Base ($\mathcal{E}DB$) that contains encrypted copy of portions of the documents in XML source
  - Authoring Certificate Base ($\mathcal{A}CB$) that contains generated certificates
  - Management Information Base ($\mathcal{M}IB$) that contains information required for updating process
  - XML Source
  - Push

  Next, we describe some protocols that are used to implement the X-Update component.

- *Self-Certifying Document Updates*: One important feature of these protocols that are used to implement the X-Update component is that the document integrity can be verified by each receiver. Before presenting these implementations, we first introduce the following notations for XML ([61, 62]) documents. These notations are used to enforce access control.

Definition 1.7
In an XML document, an atomic element (AE) is either an attribute or an element that includes the starting and ending tags of the element ([17]).

Definition 1.8
In an XML document, an atomic region (AR) includes a set of atomic elements. It is the smallest data portion to which the same access control policies apply ([17]).

Each atomic region is identified by an identifier. Therefore, an XML document could be divided into a set of atomic regions such that atomic elements of the same region are distinct and there is no atomic element that belongs to two different regions.

A region can be either *modifiable* or *nonmodifiable*. A region is nonmodifiable by a subject if this subject can only read it. A region is modifiable by a subject if this subject possesses the authorization to modify it, according to the access control policies.

Definition 1.9
In an XML document, a region object O is an instance of the information in a region R. A region object is associated with the region identifier R, the subject who authors it, and the time when the subject authors it.

Bertino et al. ([58, 63] propose a self-certifying document updating protocol in distributed systems. In their approach, the document is encrypted by the document server with the minimum number of keys such that different keys are used for encrypting different portions (a set of ARs) of the same document. Each participating subject receives only these keys for the portions that it is authorized to access from the document server. The encrypted document then circulates in sequence among the participating subjects.

When a subject receives the document, it could verify the correctness of the operations performed so far on the document, based on the control information the subject received from the document server. If there is no error, the subject can exercise its privileges on the document, sign these updates with its signature, then encrypt the portions it accessed, and send the encrypted document to the next subject. Only when the document fails the integrity check, a subject contacts the document server for document recovery.

A major limitation of this approach is that it does not exploit the possible parallelism that is inherent in data relationships and in the access control policies. Koglin et al. ([17]) propose an approach based on the use of a security region-object parallel flow (S-RPF) graph protocol. S-RPF graph protocol allows different users to simultaneously update different regions of the same document, according to the specified access control policies.

In an S-RPF graph, each node represents a subject in the flow path. An edge with label L from node i to node j denotes that there are region objects L sending from i to j. The S-RPF graph that the document server generated has the following properties:

- If no participating subject has access privilege to a region with the identifier of R, then no region object O associated with R will appear in the S-RPF graph.
- If a region object is modified by a subject *subj*, then this region object will not flow out from *subj* and a new region object with the same region identifier will start at *subj*.
- The same region object may be accessed by several subjects at the same time.
- The flow of each region object among the subjects is acyclic. This means that no region object flows back to the subject who authored it.
- If no subject has update rights on a region R, but there is at least one subject that has access privilege to this region, then a region object O associated with R starts its flow among the subjects from the document server and its author is the document server.

The S-RPF protocol is secure with respect to confidentiality and integrity. The proofs can be found in [17].

In all these mentioned approaches, the data server is not the bottleneck during the updating process. However, these approaches are not scalable. The data server has to perform some initial computation before the updating process starts. Furthermore, each participating subject is predefined. They cannot be changed once the updating starts. Also, these participants need to receive some control information from the data server in order to perform integrity checking of the document.

Further research in this area includes using roles to make the solution scalable. Moreover, the document server has too much control on the updated document, mechanisms should be proposed to enforce the principles such as separation of duty and least privilege.

1.5 Other Research Issues

Privacy preserving in content distribution networks is one important research area for study ([64]). Most research (such as [51]) in this area is on the techniques for supporting anonymity such that users could anonymously publish

P1: Naresh Chandra
January 30, 2007 11:4 AU7921 AU7921˙C001

or retrieve various kinds of information; furthermore, the transaction between data servers and clients should be unlinkable.

Research on censorship-resistant document publishing (e.g., [51, 65, 66]) also demands further study. In these systems, the content stored on and distributed by servers should be free of censoring. Peer-to-peer systems are one promising area for such study, since they do not have a centralized administration.

Other research issues in CDNs include location-based access control ([67]). Different mechanisms are needed to ensure that content could be accessed only within certain locations. Therefore, precise location verification techniques are important to enforce this kind of access control.

Bibliography

1. M. Srivatsa and L. Liu. Securing publish-subscribe overlay services with eventguard. In *Proceedings of the 12th ACM Conference on Computer and Communications Security (CCS '05)*, 2005.
2. Yanlei Diao, Shariq Rizvi, and Michael J. Franklin. Towards an internet-scale XML dissemination service. In *VLDB Conference*, August 2004.
3. Fengyun Cao and Jaswinder Pal Singh. Efficient event routing in content-based publish-subscribe service networks. In *Proceedings of IEEE INFOCOM '04*, 2004.
4. Michael J. Freedman, Eric Freudenthal, and David Mazires. Democratizing content publication with coral. In *Proceedings of the USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI '04)*, March 2004.
5. Tieyan Li, Yongdong Wu, Di Ma, Huafei Zhu, and Robert H. Deng. Flexible verification of mpeg-4 stream in peer-to-peer cdn. In *Proceedings of the 6th International Conference on Information and Communications Security (ICICS)*, pages 79–91, 2004.
6. Girma Berhe, Lionel Brunie, and Jean-Marc Pierson. Modeling service-based multimedia content adaptation in pervasive computing. In *Conf. Computing Frontiers*, pages 60–69, 2004.
7. Armando Fox, Steven D. Gribble, Yatin Chawathe, and Eric A. Brewer. Adapting to network and client variation using active proxies: lessons and perspectives. *IEEE Personal Communications*, August 1998.
8. V. Cardellini, P. S. Yu, and Y. W. Huang. Collaborative proxy system for distributed web content transcoding. In *Proceedings of 9th ACM Intl Conf. on Information and Knowledge Management*, November 2000.
9. Chi-Hung Chi and Yin Wu. An XML-based data integrity service model for web intermediaries. In *Proceedings of the 7th International Workshop on Web Content Caching and Distribution*, August 2003.
10. Chi-Hung Chi, Yan-Hong Lin, Jing Deng, X. Li, and T.-S. Chua. Automatic proxy-based watermarking for www. *Computer Communications*, 24(2):144–154, 2001.
11. P. Thuraisingham, A. Gupta, E. Bertino, and E. Ferrari. Collaborative commerce and knowledge management. *Knowledge and Process Management*, 9(1):43–53, August 2002.

12. P. Maglio and R. Barrett. Intermediaries personalize information streams. *Communications of the ACM*, 43(8):99–101, August 2000.
13. J.-L. Huang, M.-S. Chen, and H.-P. Hung. A qos-aware transcoding proxy using on-demand data broadcasting. In *Proceedings of the IEEE INFOCOM Conference*, March 2004.
14. Yunhua Koglin and Elisa Bertino. Secure content services from cooperative internet intermediaries. Technical report, Purdue University, 2005.
15. S. Chandra and C. S. Ellis. Jpeg compression metric as a quality aware image transcoding. In *Proceedings of USENIX 2nd Symp. on Internet Technology and Systems*, October 1999.
16. R. Han, P. Bhagwat, R. LaMaire, T. Mummert, V. Perret, and J. Rubas. Dynamic adaptation in an image transcoding proxy for mobile web browsing. *IEEE Personal Communications*, 5(6):8–17, December 1998.
17. Yunhua Koglin, Giovanni Mella, Elisa Bertino, and Elena Ferrari. An update protocol for XML documents in distributed and cooperative systems. In *Proceedings of International Conference on Distributed Computing Systems*, June 2005.
18. John Apostolopoulos. *Secure media streaming and secure adaptation for non-scalable video*. Technical Report HPL-2004-186, Hewlett-Packard Laboratories, October 2004.
19. Susie Wee and John Apostolopoulos. *Secure transcoding with jpsec confidentiality and authentication*. Technical report HPL-2004-185, Hewlett-Packard Laboratories, October 2004.
20. Susie Wee and John Apostolopoulos. Secure scalable streaming enabling transcoding without decryption. In *IEEE International Conference on Image Processing*, 2001. Available as Hewlett-Packard Laboratories Technical Report HPL-2001-320.
21. Susie Wee and John Apostolopoulos. Secure scalable video streaming for wireless networks. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, May 2001.
22. Charu C. Aggarwal, Joel L. Wolf, and Philip S. Yu. Caching on the world wide web. *Knowledge and Data Engineering*, 11(1):95–107, 1999.
23. Bo Li, Xin Deng, Mordecai J. Golin, and Kazem Sohraby. On the optimal placement of web proxies in the internet. In *Proceedings of Infocom Conference*, March 1999.
24. S. Sivasubramanian, M. Szymaniak, G. Pierre, and M. V. Steen. Replication for web hosting systems. *ACM Computing Surveys*, 36(3):291–334, September 2004.
25. Lee Breslau, Pei Cao, Li Fan, Graham Phillips, and Scott Shenker. Web caching and zipf-like distributions: Evidence and implications. In *Proceedings of IEEE INFOCOM Conference*, March 1999.
26. S. Buchholz and A. Schill. Adaptation-aware web caching: caching in the future pervasive web. In *13th GI/ITG Conference Kommunikation in Verteilten Systemen (KiVS)*, 2003.
27. Matt Bishop. *Computer Security: Art and Science*. Addison Wesley Professional, 2002.
28. D. Elliott Bell and Leonard J. LaPadula. *Secure computer systems: unified exposition and multics interpretation*. Technical Report MTR-2997, MITRE Corporation, March 1976.
29. R. S. Sandhu. Lattice-based access control models. *IEEE Computer*, 26(11):9–19, November 1993.

30. J. Barkley, A.V. Cincotta, D.F. Ferraiolo, S. Gavrila, and D.R. Kuhn. Role-based access control for the world wide web. In *20th National Computer Security Conference*, 1997.
31. James Joshi, Elisa Bertino, Usman Latif, and Arif Ghafoor. A generalized temporal role-based access control model. *IEEE Trans. Knowl. Data Eng.*, 17(1):4–23, 2005.
32. Roberto Tamassia, Danfeng Yao, and W. H. Winsborough. Role-based cascaded delegation. In *Proceedings of the ACM Symposium on Access Control Models and Technologies (SACMAT '04)*, pages 146–155. ACM Press, June 2004.
33. E. Barka and R. Sandhu. Framework for role-based delegation models. In *Proceedings of the 16th Annual Computer Security Applications Conference (ACSAC'00)*, December 2000.
34. E. Freudenthal, T. Pesin, L. Port, E. Keenan, and V. Karamcheti. dRBAC: Distributed role-based access control for dynamic coalition environments. In *ICDCS 2002*, pages 411–420, 2002.
35. J. S. Park, R. Sandhu, and G.-J. Ahn. RBAC on the web. In *ACM Transactions on Information and Systems Security*, volume 4(1), 2001.
36. Mohammad A. Al-Kahtani and Ravi S. Sandhu. A model for attribute-based user-role assignment. In *ACSAC*, 353–364, 2002.
37. Anthony Harrington and Christian D. Jensen. *Cryptographic access control in a distributed file system*, 2003.
38. Dawn Xiaodong Song, David Wagner, and Adrian Perrig. Practical techniques for searches on encrypted data. In *IEEE Symposium on Security and Privacy*, pages 44–55, 2000.
39. Matt Blaze, Gerrit Bleumer, and Martin Strauss. Divertible protocols and atomic proxy cryptography. *Proceedings of Eurocrypt*, 1403:127–144, 1998.
40. Giuseppe Ateniese, Kevin Fu, Matthew Green, and Susan Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. In *Proceedings of the 12th Annual Network and Distributed System Security Symposium (NDSS)*, February 2005.
41. Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance. In *OSDI: Symposium on Operating Systems Design and Implementation*. USENIX Association, Co-sponsored by IEEE TCOS and ACM SIGOPS, 1999.
42. Rongmei Zhang and Y. Charlie Hu. Hyper: A hybrid approach to efficient content-based publish/subscribe. In *Proceedings of International Conference on Distributed Computing Systems*, 2005.
43. A. Carzaniga, M. J. Rutherford, and A. L. Wolf. A routing scheme for content-based networking. In *IEEE INFOCOM*, 2004.
44. G. Cugola, D. Frey, A. L. Murphy, and G. P. Picco. Minimizing the reconfiguration overhead in content-based publish-subscribe. In *Proceedings of the 19th ACM Symposium on Applied Computing (SAC04)*, 2004.
45. M. Srivatsa and L. Liu. Securing publish-subscribe overlay services with eventguard. In *Proceedings of the 12th ACM Conference on Computer and Communication Security*, 2005.
46. Lukasz Opyrchal and Atul Prakash. Secure distribution of events in content-based publish-subscribe systems. In *Proc. of the 10th USENIX Security Symposium*, 281–295, 2001.
47. Chenxi Wang, Antonio Carzaniga, David Evans, and Alexander L. Wolf. Security issues and requirements for internet-scale publish-subscribe systems. In *Hawaii International Conference on System Sciences*, 2002.

48. Chris Lesniewski-Laas and M. Frans Kaashoek. Ssl splitting: Securely serving data from untrusted caches. In *Proceedings of 12th USENIX Security Symposium*, August 2003.
49. Napster. Available at: http://www.napster.com.
50. Gnutella. Available at: http://gnutella.wego.com.
51. Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W. Hong. Freenet: A distributed anonymous information storage and retrieval system. *Lecture Notes in Computer Science*, 2009:46–52, 2001.
52. Ion Stoica, Robert Morris, David Karger, Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the 2001 ACM SIGCOMM Conference*, pages 149–160, 2001.
53. Ben Y. Zhao, Ling Huang, Jeremy Stribling, Sean C. Rhea, Anthony D. Joseph, and John Kubiatowicz. Tapestry: A resilient global-scale overlay for service deployment. *IEEE Journal on Selected Areas in Communications*, 22(1):41–53, January 2004.
54. John Kubiatowicz, David Bindel, Yan Chen, Patrick Eaton, Dennis Geels, Ramakrishna Gummadi, Sean Rhea, Hakim Weatherspoon, Westly Weimer, Christopher Wells, and Ben Zhao. Oceanstore: An architecture for global-scale persistent storage. In *Proceedings of ACM ASPLOS*. ACM, November 2000.
55. Frank Dabek, M. Frans Kaashoek, David Karger, Robert Morris, and Ion Stoica. Wide-area cooperative storage with cfs. In *Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP)'01*, October 2001.
56. Stephanos Androutsellis-Theotokis and Diomidis Spinellis. A survey of peer-to-peer content distribution technologies. *ACM Comput. Surv.*, 36(4): 335–371, 2004.
57. Elisa Bertino, Silvana Castano, Elena Ferrari, and Marco Mesiti. Specifying and enforcing access control policies for xml document sources. *World Wide Web*, 3(3):139–151, 2000.
58. Elisa Bertino, Barbara Carminati, Elena Ferrari, and Giovanni Mella. Author-chi — A system for secure dissemination and update of xml documents. In *DNIS*, pages 66–85, 2003.
59. Elisa Bertino, Silvana Castano, and Elena Ferrari. Securing xml documents with author-x. *IEEE Internet Computing*, 5(3):21–26, 2001.
60. Elisa Bertino, Silvana Castano, and Elena Ferrari. Securing xml documents: The author-x project demonstration. In *SIGMOD Conference*, page 605, 2001.
61. Extensible markup language (XML). Available at: http://www.w3.org/XML/.
62. W3C XML schema. Available at: http://www.w3.org/XML/Schema.
63. Elisa Bertino, Elena Ferrari, and Giovanni Mella. An approach to cooperative updates of xml documents in distributed systems. *Journal of Computer Security*, 13(2):191–242, 2005.
64. Fatih Emekçi, Divyakant Agrawal, and Amr El Abbadi. Abacus: A distributed middleware for privacy preserving data sharing across private data warehouses. In *Middleware*, pages 21–41, 2005.
65. Marc Waldman. Censorship-resistant publishing systems-survey and thesis proposal.
66. Aviel D. Rubin Marc Waldman and Lorrie Faith Cranor. Publius: A robust, tamper-evident, censorship-resistant, web publishing system. In *Proc. 9th USENIX Security Symposium*, pages 59–72, August 2000.
67. Elisa Bertino, Barbara Catania, Maria Luisa Damiani, and Paolo Perlasca. Geo-rbac: A spatially aware rbac. In *SACMAT*, pages 29–37, 2005.