Decentralized Authorization and Data Security in Web Content Delivery *

Danfeng Yao
Computer Science Department
Brown University
Providence, RI 02912
dyao@cs.brown.edu

Yunhua Koglin
Computer Science Department
Purdue University
West Lafayette, IN 47906
luy@purdue.edu

Elisa Bertino
Computer Science Department
Purdue University
West Lafayette, IN 47906
bertino@cerias.purdue.edu

Roberto Tamassia
Computer Science Department
Brown University
Providence, RI 02912
rt@cs.brown.edu

ABSTRACT

The fast development of web services, or more broadly, service-oriented architectures (SOAs), has prompted more organizations to move contents and applications out to the Web. Softwares on the web allow one to enjoy a variety of services, for example translating texts into other languages and converting a document from one format to another. In this paper, we address the problem of maintaining data integrity and confidentiality in web content delivery when dynamic content modifications are needed. We propose a flexible and scalable model for secure content delivery based on the use of roles and role certificates to manage web intermediaries. The proxies coordinate themselves in order to process and deliver contents, and the integrity of the delivered content is enforced using a decentralized strategy. To achieve this, we utilize a distributed role lookup table and a role-number based routing mechanism. We give an efficient secure protocol, *iDeliver*, for content processing and delivery, and also describe a method for securely updating role lookup tables. Our solution also applies to the security problem in web-based workflows, for example maintaining the data integrity in automated trading, contract authorization, and supply chain management in large organizations.

Categories and Subject Descriptors

D.4.6 [**Operating System**]: Security and Protection—*Access Controls*; K.6.5 [**Management of Computing and Information Systems**]: Security and Protection—*Authentication*

General Terms

Security

Keywords

Data Integrity, Authorization, Content Delivery Networks, Web

*This work was supported in part by National Science Foundation Grants CCF–0311510, IIS–0324846, and 0430274, and by the sponsors of CERIAS.

SAC'07 March 11-15, 2007, Seoul, Korea

1. INTRODUCTION

Content delivery networks (CDNs) [1, 4, 7, 8, 16, 17, 22] represent an important infrastructure for web-services and E-applications in general, such as distance-learning and virtual museums, requiring the delivery of tailored contents to very large user communities. CDNs not only support efficient publication, storage and retrieval of static files but also make dynamic web content and real-time streaming broadcasts easily accessible by devices with various capacities and configurations. CDNs support fast and adaptive information delivery, through the use of servers performing different types of modifications, such as aggregation and transcoding operations [4, 8, 16] on the contents, thus greatly improving end-user experiences. For example, web contents such as images and videos owned by yahoo.com can be cached by proxies; the user's browser can fetch and aggregate cached contents from proxies without having to request them from yahoo.com.

Transformations by proxies yield a web data delivery process that is more efficient than alternative approaches. In particular, one alternative approach is to let the publisher transform the contents into the corresponding format upon each request. Another approach is that the server transforms the contents into different formats *a priori*, and when a request arrives, the server retrieves the corresponding content and sends it out. However, neither approach is satisfactory. While dynamic web-content generation may work

for HTML pages, it has several drawbacks for large media data: (1) low efficiency, as the client needs to wait while the content is transformed and transmitted; (2) the server is the bottleneck, as many requests are issued — to address this problem one needs to use very high capacity computing infrastructures which cannot always be afforded by companies in today's fragmented economy; (3) the server may repeat the same transformation for many times. Concerning the aforementioned second approach, even though the efficiency is not a problem, the publisher would need huge storage capacity. For contents of size n, it is possible that the server would need exponential space to store it. Figure 1 shows an example of transformations performed on a piece of original data in order to accommodate different configurations and preferences of requesters. This approach is not suitable when contents frequently change.

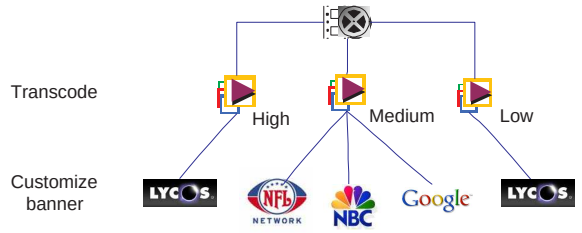

Figure 1: An example of transformations performed on a piece of original data. The first operation changes the resolution of the data to three different levels: high, medium, low. The second operation adds customized banners.

The rapid growth of business transactions conducted on the Internet has drawn much attention to the problem of data security in CDNs, in particular, the integrity of contents. Solutions have been proposed for data authentication in online publisher-requester model [13]. However, the integrity problem in CDN is more challenging, because requested contents usually need to be processed and modified by proxies who may not be known or trusted by users.

We study the problem of maintaining content *integrity* and *confidentiality* in CDNs in which contents may be modified by several proxies. The application scenario we refer to includes a publisher (e.g. yahoo.com), a set of end users, and a set of proxies owned by the publisher or by a third party (e.g., akamai.com) trusted by the publisher. A content may be processed by one or more proxies before it is received by the user. For example, one proxy performs a transcoding operation [4, 8, 16] to reduce the resolution of the media contents; the next proxy performs data filtering and executes value-added services, such as watermarking [9]. It is worth mentioning that SSL protocol [23] cannot be directly applied to our server-proxy-client model.

Our proposed solution is based on an access control approach, in which proxies can perform transformations on contents based on their authorizations. Access control mechanisms have been extensively investigated in conventional server-client settings. However, there are two unique requirements for the data security problem of CDN: (1) scalability and (2) robustness.

A scalable solution should be able to manage large number of proxies and frequent reconfigurations of proxies in the network. In addition, it should not contain any performance bottleneck, and require minimal involvement of a centralized authority. If the requested contents needs multiple proxies to process, the proxies have to be properly coordinated. Because of the large number of proxies in the network, it is inefficient for the data publisher to keep the up-to-date global proxy information.

For the robustness requirement, a secure content delivery protocol should prepare for failed nodes. When a proxy is unavailable for data processing, the protocol should provide alternative routes. In the meantime, an end user should still be able to verify the content's integrity. We use role-based model combined with lookup tables to realize a secure, flexible, and robust content delivery network.

1.1 Contributions

We describe a service architecture for secure content delivery using roles to manage web proxies. The main features of our approach are that the proxies coordinate themselves to process and deliver contents, and the security of the delivered contents is enforced using a decentralized strategy. A role in our context represents a specific function, or set of functions, that can be performed by proxies. Authorizations are expressed in terms of roles, therefore reducing the number of permissions that need to be granted. Also, the use of roles improves reliability in that a proxy can be easily replaced by a proxy playing the same role. The requirement of distributed access control is addressed by the use of special control information enclosed with the delivered contents. Such information allows a client agent to verify that all of the modifications satisfy the policies of the content publisher.

We give a secure protocol, *iDeliver*, for maintaining data integrity and confidentiality in a semi-honest adversarial model. The protocol does not assume any pre-established trust relationship among proxies. Trust can be established only after the required credentials and authorizations are validated. Similarly, the trust between the user and a proxy is also *ad hoc*, contingent on the validity of authorizations and digital signatures. The end user is only required to trust the content publisher and the role certificate authority delegated by the publisher. The *iDeliver* protocol uses cryptographic primitives including hash functions, signature schemes, symmetric and asymmetric encryption schemes.

We utilize the *distributed role lookup table*, which is a table maintained by each proxy containing the role information about its neighboring proxies. The proxy uses the lookup table and the content-specific control information for processing or delivery, according to a distributed strategy without requiring the participation of the publisher. We give a routing mechanism based on *role numbers* for CDNs, which is similar to the suffix-based routing in Tapestry [26]. A role number is used to locate a proxy with the required role. In comparison to routing with specific proxy identities, routing using role numbers enhances the flexibility and robustness. The number of overlay hops (hops from one proxy to another proxy) for locating a proxy is $O(h)$, where h is the height of the tree containing the role hierarchy. The number of overlay hops for processing the requested content is $O(mh)$, where m is the number of roles required for the content.

Organization of the paper. The CDN model is in Section 2. Section 3 describes the role lookup table. The secure *iDeliver* protocol is given in Section 4. Related work and conclusions are given in Section 5 and 6, respectively.

2. OVERVIEW

In this section, we present an overview of operations and the adversarial model. Below, we define data confidentiality and integrity that are specific to our context.

Data confidentiality: The delivered contents cannot be viewed by unauthorized entities, including unauthorized proxies and other users besides the requester. Proxies authorized by the publisher to process the contents can view the data.

Data integrity: Delivered content which is modified by unauthorized entities should not be accepted. Modifications can only be valid when accompanied by proper authorization and certification.

A general solution for the direct verification of data authenticity as in third-party data publication [12, 21] is difficult to obtain in CDN when proxies need to modify (e.g. insert, delete, transform) the contents. This is because an authentication method using conventional signature schemes is inefficient. Solutions have been given for specific types of data and operations [15, 18]. The goal of the work presented in this paper is to devise a more general solution.

Our model includes a publisher, end users, a set of proxies with roles, and a role certificate authority. The roles represent the functionalities of a proxy, such as transcode role, virus scan role, etc. The role certificate authority may be the publisher or an entity trusted by the publisher. It has the following main components: role certificate generation, control information generation, distributed lookup table and its maintenance, content processing, role-number based routing, and verification. Next, we give definitions of *control information*, *role number*, *permission*. A schematic representation of our model is reported in Figure 2.

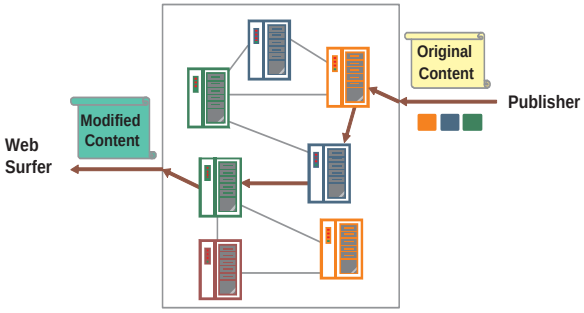

Figure 2: A schematic representation of CDN. The three squares under the "original content" represent the sequence of roles specified in the control information by the publisher. Thick lines represent the path of proxies used for delivering and processing the requested content. The thin lines represent the neighboring relations of proxies.

DEFINITION 1. *The control information of a requested content is generated by the publisher, and specifies the sequence of processing steps; each step represents a specific function to be performed on the content and the function is directly mapped to a role. In our model, such a sequence is specified according to regular expressions.*

Because of page limit, the syntax for the control information and regular expressions is not presented. An example of a regular expression is $(r_a, \{r_b, r_c\}, r_d)$, which means that the sequence of proxy roles can be either (r_a, r_b, r_d) or (r_a, r_c, r_d).

DEFINITION 2. *A role number is a unique base-b number assigned for routing purpose by the publisher and the role certificate authority.*

Role number is discussed in more details in the next section.

DEFINITION 3. *The permission associated with a role expresses the authorization for a proxy to perform a certain action on certain types of content.*

We define the operations in our model as follows.

1. **Role certificate generation**: The role certificate authority issues digital role certificates to qualified proxies.

2. **Control information generation**: The control information is generated and signed by the publisher with its private key.

3. **Distributed lookup table and its maintenance**: Each proxy maintains its distributed lookup table, which contains the mappings between roles and local proxies.

4. **Content processing**: The content is processed by a proxy. The transformed content and its integrity proof are passed on to the next required role member for processing or to the end user.

5. **Role-number based routing**: Given a role, the role number combined with the lookup table is used for locating a proxy server with that role.

6. **Content verification**: Once the requested content is delivered to the user, the user checks the integrity of the content by verifying the proof associated with it.

For our adversarial model, a polynomial-time bounded adversary is allowed to launch the following attacks.

- Attempt to add itself into a proxy's role lookup table through a forged or stolen certificate. This is equivalent to attempting to join the CDN as a valid proxy.
- Insert packets into the system (assuming that this attack does not flood the network).
- Intercept, replay, and modify data packets.
- Attempt to decrypt contents.

In model, a certified proxy is assumed to follow protocols, which implies that a proxy with a certain role assignment does not perform unauthorized operations. For almost all credential systems and distributed access control models (e.g. Delegation Certificates [2]), the above is the standard security assumption that is practical. An analogy in a physical access control scenario is that, for example, Alice with a college ID can get access to the library. We also assume that the role certificate authority is trusted and issues certificates according to publishers' data security policies.

3. ROLE NUMBERING SCHEME

In our model, the control information generated and signed by the publisher specifies the sequence of roles needed to process the content before it is delivered to the end user. Once a proxy finishes processing, it either passes the content to another proxy or to the end user. The latter happens when the proxy is the last one to process the content. In the former case, the proxy first consults the control information to determine the next required role, and then determines the location of a proxy having this role. We use a table to store the mappings between roles and neighboring proxies according to a decentralized strategy. The lookup table is similar to a routing table, and is used for our role-number based routing. Each node, including the publisher, maintains a role lookup table for storing information of its neighbor.

3.1 Hierarchical Role Numbers

In our model, each role in a CDN is assigned a role number that uniquely identifies the role. In hierarchical role-based access control, roles are organized into a hierarchy so that the role associated with a parent node has the permissions of all the child nodes. Our use of role hierarchies is to reduce the number of role lookup tables and improve the flexibility of the search process when having to locate a proxy with a required role.

A role hierarchy can be an arbitrary directed acyclic graph (DAG). In this paper, we consider a restricted type of role hierarchies, namely *trees*, where each node (except the root) has only one parent node. Figure 3 shows an example of a role hierarchy. Our tree labeling method is as follows. Let T be the tree containing the role hierarchy. Let b be the maximum degree of the nodes in T. Transform T into a balanced b-nary tree T' by adding null nodes that do not correspond to actual roles. Label the root with ϵ. For each node of tree T', label its child nodes from left to right with $0, 1, \ldots, b-1$. The role number of a node is the string that represents the path from the root to the node. Without loss of generality, we assume that the root is not associated with a role, so ϵ can be omitted from the role number. The length of a role number is bounded by the height h of the role hierarchy T. The role number of a non-null internal node of T' is the prefix of the role numbers of all its child nodes. For example, in Figure 3, a's number is the prefix of its children a_1 and a_2. Role numbers of null nodes in tree T' are for future roles, and correspond to empty entries in role table, as proxies with these role numbers do not exist yet.

With the support of the role hierarchy, it is convenient for the publisher to specify a group of roles using regular expressions in the control information. For example, $(1*, 22)$ means that the first role to process the content should be a role with prefix 1, the second role should be 22 (neither role 2 nor 221 could be used). Degree b can be chosen to be slightly larger than what is required for the current role hierarchy, to accommodate future expansions. Using our labeling method, deleting and replacing a node in T' do not change the numbering of other roles.

The role hierarchy does *not* introduce load-balance problem, where the concern is that a parent node might have more loads than its children. As a newly-joint (child) node propagates its information through the network, the parent and the child nodes have the same chance of being selected for a certain task. This will become clearer in Section 3.3.

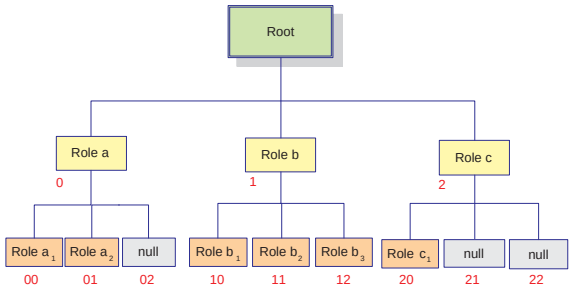

Figure 3: An example of the role numbering method. The balanced tree is derived from a role hierarchy by adding null nodes. Null nodes correspond to non-existent roles. The number below each node is the role number. Role a is the parent role of a_1, a_2, and a_3, and has the permissions associated with all its child roles. The same applies to roles b and c. The height and degree of the tree are both 3.

3.2 Components of a Role Lookup Table

In our model, each proxy maintains a role lookup table that corresponds to its role number. If a proxy is authorized to play multiple roles, for simplicity, there could be one lookup table for each role. Similarly to Tapestry [26], the table is organized into routing levels. Each routing level corresponds to the level of digit in role number. For each cell in the table, there is a *primary entry* and a *secondary entry* in the table for fault-tolerance purpose. Each entry points to the address and the public key of a neighboring proxy server with that role. Each proxy also maintains a *backpointer list* that points to the proxies that are stored in the table. We use backpointers for distributing update information of role tables. An example of a distributed role lookup table for role number 2312 in base-4 is shown in Figure 4.

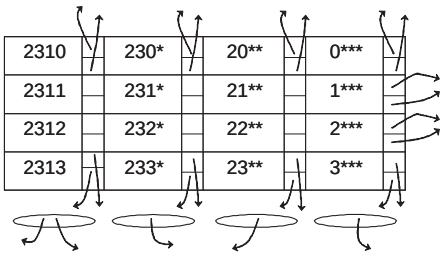

Figure 4: The lookup table for a node with the quadruple-based role number 2312. An arrow corresponding to each role number refers to the information (IP address and public key) of a proxy having that role. Not all arrows are shown here. Symbol * represents any number from 0 to 3. This means that the role number stored at 230* may be 2300, 2301, 2302, etc. Circles and arrows at the bottom represent backpointers.

The publisher also maintains a local role lookup table. As the publisher does not have a specific role, it may maintain a table associated with any existing role.

3.3 Table Setup

The lookup table of a new proxy is populated by its neigh-

bors when it joins the CDN. For example, a new proxy learns (some of) the existing proxies from the role authority. It multicasts a message indicating that it possess a role and requests role lookup table updating. Existing proxies with certified roles then send their own role lookup tables to the new proxy. It is important that the information in the table is authenticated before use.

Our model does not assume trust relationship between proxies, i.e. proxies do not have to trust each other. Thus, requiring neighboring proxies to sign the table information does not provide any security guarantee on the correctness of the table data. Therefore, for each entry in its new lookup table, the new proxy needs to contact the corresponding proxy for its role certificate. The certificate signature and the certificate holder's public key are verified. A similar verification process is required when an table entry is replaced or a new entry is inserted. The authentication prevents skewed table entries to be propagated that causes contents to be sent to wrong or malicious servers. These steps are standard operations when using certificates [14], and are not described.

To update a role table, neighboring proxies periodically send information to refresh each other's table. In order to prevent spoofing, the update information needs to be authenticated, which can be achieved using an existing lightweight authentication scheme (e.g., [19]).

In what follows, we assume that in the lookup tables of all certified proxies, empty entries only belong to non-existent role numbers. That is, for each valid role, the information of at least one proxy is stored. This assumption is reasonable because neighboring role tables are periodically refreshed and updated. The information of new proxies is propagated by ripple effects. Note that a proxy is *not* required to store the global knowledge of the network. Since there may exist several proxies that possess the same role, for any entry in a table, only two such proxies are kept. These two proxies are not fixed and may be changed by the proxy that maintains the table. This random choosing of proxies also keeps the system free of load balance problem. It is easy to conclude that the size of the lookup table is bounded by $O(bh)$, where b and h are the maximum degree and the height of the role hierarchy, respectively.

3.4 Role-Number Based Routing

We present a simple yet effective role-number based routing scheme for a proxy to locate another proxy with a required role. Proxies use local role lookup tables to incrementally route contents to the destination role number digit by digit (e.g., 0*** ⇒ 03** ⇒ 031* ⇒ 0312). (As any overlay routing mechanism, this relies on the underlying network routing infrastructure, such as routers and switches to physically transmit packets.) To route contents to any proxy with role number r, a proxy A with role table T executes the following steps.

- Proxy A consults its table T for the entry associated with role number r. It distinguishes two cases.
  - If T contains role number r, that is, r exists in T, then A retrieves the proxy B's information (IP address and public key) stored under r.
  - If T does not contain role number r, A chooses a role number r' in table T that has the maximum matching prefix with the role r. Let B be the proxy associated with role r'.
- The delivered content is encrypted (as will be described in 4) and sent to that proxy B.

Figure 4 illustrates the role-number based routing process. Suppose the table belongs to proxy A that has role number 2312 and the target role number is 2311. Because the table contains entry 2311, the primary proxy stored under that entry is chosen to receive the content. If the target role number is 0312 and the table does not contain that role number, proxy A chooses the proxy stored under 0*** because it best matches 0312. In this case, the number associated with 0*** corresponds to any role number that starts with 0, except 0312. The content is then sent to the chosen proxy, which tries to resolve role number 0312 using its own lookup table in the same way. For example, the path of role numbers that process the content could be 0112 ⇒ 0320 ⇒ 0311 ⇒ 0312. In general, the path may be 0*** ⇒ 03** ⇒ 031* ⇒ 0312, or shorter. Once all the roles required by the control information have processed the content, the content is sent to the user via regular routing mechanism.

The routing method is the same when role numbers encode the role hierarchy information. Suppose the target role number is still 0312. A possible path is 0112 ⇒ 0320 ⇒ 031. Role 031 is the parent role of 0312, and therefore can handle the processing task of role 0312. The proxies with roles 0112 and 0320 are not parents of role 0312, and thus do not have the authorizations of role number 0312.

Our approach is efficient as stated by the following theorem.

THEOREM 1. *Given a role number, it takes a proxy at most $O(h)$ overlay hops to locate another proxy with that role number, where h is the height of the tree representing the role hierarchy.*

Proof: In our CDN, at least one digit of the target role number is *corrected* at each routing level. Note that this is not affected by empty table entries that correspond to null nodes in the b-nary balanced tree T'. They are never used in our routing algorithm, since we assume that the rest of the lookup table entries contain non-empty proxy information. The length of a role number is $O(h)$ in a content delivery network with a role hierarchy with height h. Therefore, the bound holds. □

From Theorem 1, it is easy to draw the following conclusion for the overall number of overlay hops of the requested content.

COROLLARY 2. *The number of overlay hops for processing the content is bounded by $O(mh)$, where h is the height of the tree containing the role hierarchy in the content delivery network and m is the number of roles required to process the content.*

As every single lookup table of a node assumes that the preceding digits all match the current node's prefix, it only needs to keep a small constant size (b) entries at each route level, yielding a lookup table of size $O(bh)$.

$$\begin{aligned} \text{LookupTableSize} &= \text{entries per table} \times \text{size per entry} \\ &= O(bh) \end{aligned}$$

The above upper bounds hold independently of whether roles are organized according to a hierarchy or not. The average number of hops can be reduced when a role hierarchy

is used, because parent proxies serve as *shortcuts* for content delivery. Role-number based routing ensures that a qualified proxy can be found efficiently without requiring global information. Because role-lookup tables are up-to-date with respect to proxy availability, our routing scheme minimizes service disruption caused by unavailable nodes.

4. SECURE *iDeliver* PROTOCOL

In this section, we present our protocol, *iDeliver*, for maintaining data integrity and confidentiality in CDN. We assume that public and private key pairs of all the nodes in CDN are already set up.

Notations: We denote by $(r_1, \ldots, r_l)$ the numbers of roles that are required for processing content M; l is the total number of required roles. The public and private key of an entity E are denoted as Pub_E and $Priv_E$, respectively. We denote by H a cryptographic hash function. SIGN and VERIFY are the signing and verification algorithms of a secure public key signature scheme. For example, the RSA signature scheme can be used, and the public/private keys can be generated by the key-generation algorithm in RSA. The signature on message M by entity E is denoted as $Sig_E(M)$. A role certificate is denoted by *Cert*. We denote by E_K and D_K the encryption and decryption algorithms with secret key K of a symmetric encryption scheme, respectively. A public-key encryption scheme is used to transmit symmetric keys. We denote by $Encrypt(M, Pub_E)$ and $Decrypt(M, Priv_E)$ the encryption and decryption of message M with public key Pub_E and private key $Priv_E$, respectively. To simplify notations, $Encrypt(M, Pub_E)$ also refers to the ciphertext produced by the encryption algorithm.

4.1 Role Certificate Generation

At the setup, a set of roles and their role numbers for processing contents are specified by the publisher. The publisher delegates a role certificate authority to issue role certificates to proxy servers in the network. By doing this, the publisher trusts the judgement of the role certificate authority on the trustworthiness of proxies. Proxy servers may be commercial machines owned by third-party companies. Techniques in reputation management [11] can be used to determine the trustworthiness of proxies. How this is done is outside the scope of this paper and is not discussed. Decisions on role assignments to proxies are made by the role authority based on individual proxy's computing resource and capability.

A role certificate for a proxy Q having public key Pub_Q contains the following information: the public key Pub_Q, one or more role number assignments denoted by $(r_{Q_1}, \ldots, r_{Q_k})$, and the permissions associated with the roles, where k is the number of role assignments. The certificate *Cert* is signed with the private key $Priv_A$ of A, and is issued to the proxy.

4.2 Control Information Generation

We denote the publisher by Q_0. The details of the publisher's operations are as follows.

1. The publisher receives a request for a content from a user u with public key Pub_u. Q_0 computes the sequence of roles $(r_1, \ldots, r_l)$ that are required to process the content. The control information includes: role numbers $(r_1, \ldots, r_l)$, the public key Pub_A of A, the public key Pub_u of requester u, and a time-stamp. The publisher Q_0 signs the control information, which gives $Sig_{Q_0}(info)$, and signs M_0, which gives $Sig_{Q_0}(H(M_0))$.
2. The publisher uses its role lookup table to locate a proxy with role r_1, or chooses a proxy whose role number has the maximum matching prefix as r_1 in case that no proxy with r_1 can be found in the table. Denote the selected proxy by Q_1. The public key Pub_{Q_1} of Q_1 is also obtained from the table. The publisher chooses a symmetric key K_1 and encrypts M_0 with K_1; it also encrypts K_1 with the public key Pub_{Q_1} of proxy Q_1. These two encrypted messages, along with the signed control information, and $Sig_{Q_0}(H(M_0))$ are sent to Q_1.

4.3 Content Processing and Verification

A legitimate intermediate proxy needs to verify the integrity of the intermediate content received. Contents that fail the verification are rejected. This chained trust relationship reduces the trust dependency to adjacent proxies in the role sequence, and simplifies the verification process for the end user. Note that this does not reduce our security guarantees because of control information and our security model. Details of a proxy Q_i's operations are as follows.

- Q_i receives the encrypted content $E_{K_i}(M_{i-1})$, the encryption $Encrypt(K_i, Pub_{Q_i})$ of symmetric key K_i, and the following information: control information *info* from the publisher, the role certificate $Cert_{i-1}$ of Q_{i-1}, the signature $Sig_{Q_{i-1}}(H(M_{i-1}))$. Q_i first verifies the signature of the control information. Q_i then uses its private key $Priv_{Q_i}$ to obtain the symmetric key K_i. Q_i also generates a random symmetric key K_{i+1}. Then, Q_i distinguishes two cases.
  - **Case 1:** If Q_i has the role certificate for the i-th role r_i in the control information, then it proceeds to work on the content as follows. Uses key K_i to obtain the plaintext content M_{i-1}. Verifies the signature $Sig_{Q_{i-1}}(h_{i-1})$. From the control information, Q_i obtains the public key Pub_A of the role authority A, a time-stamp, and a sequence of role specifications $(r_1, \ldots, r_l)$, where l is the total number of roles required to process the content. Q_i examines certificate $Cert_{i-1}$ to verify that the previous proxy Q_{i-1} has proper authorizations as role r_{i-1}. Q_i then transforms content M_{i-1} as specified and produces M_i. Q_i signs M_i, which gives signature $Sig_{Q_i}(h_i)$. Q_i sends the following to the next entity Q_{i+1} (user or proxy): control information *info* and its signature $Sig_{Q_0}(info)$, $E_{K_{i+1}}(M_i)$ and $Encrypt(K_{i+1}, Pub_{Q_{i+1}})$, its role certificate $Cert_i$ for r_i, and signature $Sig_{Q_i}(H(M_i))$.
  - **Case 2:** If Q_i does not have the required role number r_i, then it locates a proper proxy Q'_{i+1} in its role table and forwards the content to Q'_{i+1} as above.

Requester u verifies the content integrity as follows:

1. u first verifies the signature of the control information. From the control information, u obtains the public

key Pub_A of the role authority A. u also obtains the sequence of role specifications $r_1, \ldots, r_l$, and a time-stamp.

2. u verifies that: (1) certificate $Cert_l$ is a valid role certificate issued by role authority A; (2) certificate $Cert_l$ states that proxy Q_l with public key Pub_{Q_l} has role number r_l; (3) the permissions stated in $Cert_l$ about role r_l are consistent with the content type. u verifies the signature of Q_l on M_l, and checks the time-stamp against the current-time. If the verifications succeed, the integrity of the content is correctly verified.

4.4 Security Analysis

Our *iDeliver* protocol does not assume any pre-established trust relationship among proxies. Trust is established after the required credentials and authorizations are validated. Similarly, the trust between the user and a proxy is also *ad hoc*, which is contingent on the validity of authorizations and digital signatures. The end user is only required to trust the publisher A and the role certificate authority delegated by A. In our protocol, if the content is accepted by a trustworthy proxy, then the previous modifications are authorized. A chained trust relationship is built on the proxies that process the content. The chained trust can be thought of as a *state diagram of trust*: if one is in a trusted state then the previous states are trusted.

THEOREM 3. *The* iDeliver *protocol ensures data integrity and confidentiality.*

The proof of Theorem 3 is to reduce an attack to our *iDeliver* protocol to an attack on the existential unforgeability of a secure signature scheme or an attack on a semantic-secure encryption scheme. Due to space limit, we omit the proof of Theorem 3.

An approach to relax the security assumption. One possible approach to relax the security assumption about certified proxies is to bring redundancies into the content processing that provide security guarantees for the data. For example, it is conceivable that a few number of *corrupted* yet certified proxies can be allowed to exist, if each content is required to be processed through two different routes. The end user only accepts the content if both of the received contents are the same and are successfully verified. With reasonable assumptions about the topology of the content delivery networks, the probability that contents delivered via two different routes are both tampered with and yet produce the same exact results can be small. Both theoretical and experimental studies need to be carried out on this topic as future work.

4.5 Efficiency Analysis

We analyze in Table 1 the complexities of cryptographic operations performed by the publisher, role authority, proxy, and user. Because of our security protocol where intermediate proxies are required to verify previous modifications, the user only needs to check the last modification, and hence performs a constant number of operations.

Operations	*Hash*	*Enc/Dec*	*Sign/Verify*
Role authority	O(N)	O(N)	O(N)
*Publisher**	O(m)	O(1)	O(1)
*Proxy**	O(1)	O(1)	O(1)
*User**	O(1)	O(1)	O(1)

Table 1: Complexities of cryptographic operations. N is the total number of proxies. m is the number of roles required for the content. * This refers to the operations for one request.

5. RELATED WORK

Intermediary Services. Several research efforts on content services using intermediary proxies focus on caching provided by proxies [1, 7, 17, 22]. Besides caching and delivering contents, there are other relevant new requirements for content services by intermediaries [4]. Some content services have been identified that include, but are not limited to: content transcoding [4, 8, 16], in which data is transformed from one format into another, data filtering and value-added services, such as watermarking [9].

Though much research on intermediary content services has been carried out [4, 8, 16], the problem of data security in this context has not been much investigated. Part of the reason is because it is difficult to enforce security when intermediaries are allowed to modify the data. Chi and Wu [10] proposed a Data Integrity Service Model (DISM). In this model, the integrity of intermediaries is enforced by using meta-data expressing modification policies by content owners. However, in DISM every entity in the system can access the delivered content.

One approach for secure content delivery is based on developing encryption methods that enable an entity to securely adapt or transcode the resulting protected stream without requiring decryption [15, 18]. Recently, Merkle hash tree is used to verify the integrity of transcoded MPEG-4 media streams [18], and also to address data integrity in media streaming when there are multiple sources [15]. Our model differs from those approaches because we address the security problem of CDN with a role-based access control approach, which allows us to build a general and flexible secure framework.

Distributed Hash Tables. Recent development in *Distributed Hash tables* (DHTs) [24, 26] provide location and routing infrastructures in which the routing only uses point-to-point links and does not require centralized resources. One main difference between our role lookup table and DHT in overlay networks is that in our model, different nodes may share the same role number, whereas in DHT each node has unique identifier – the probability of two nodes hashing to the same identifier is negligible. Our lookup table and number based routing mechanism can easily be modified based on other DHTs such as Chord and Pastry.

Access Control. The abstraction of roles has been previously used in the web and distributed environments [3, 6, 20, 25] for the authorization and authentication of web users. Recently, Bonatti and Samarati [6] proposed a framework for regulating service access and release of private information in web-services. They designed a policy language for the information disclosure in distributed environments such as WWW. The RBAC model in our paper is deployed for a use that is different from the conventional RBAC use in that: (1) the protected data is mobile on the web from node to node; (2) the roles are assigned to proxies; and (3) proxies and end users perform verifications. There are certain

similarities between our *iDeliver* protocol and the protocol for cooperative updates of XML documents [5]. The main difference is that their protocol requires each entity to maintain the global knowledge of the system, whereas this is not required in our protocol.

6. CONCLUSIONS

We have proposed a role-based authorization model for maintaining data security in content delivery networks. Our approach enhances the flexibility and scalability of data processing, as proxies coordinate among themselves to process contents. New concepts and techniques introduced include distributed role lookup table and role-number based routing mechanism. Our secure protocol *iDeliver* uses cryptographic primitives such as signature schemes and hash functions, and is efficient to deploy.

7. REFERENCES

[1] C. Aggarwal, J. L. Wolf, and P. Yu. Caching on the world wide web. *IEEE Transactions on Knowledge and Data Engineering*, 11(1):94–107, January 1999.

[2] T. Aura. Distributed access-rights management with delegation certificates. In *Secure Internet Programming – Security Issues for Distributed and Mobile Objects*, volume 1603 of *LNCS*, pages 211–235. Springer, 1999.

[3] J. Barkley, A. Cincotta, D. Ferraiolo, S. Gavrila, and D. Kuhn. Role based access control for the world wide web. In *20th National Computer Security Conference*, 1997.

[4] G. Berhe, L. Brunie, and J. M. Pierson. Modeling service-based multimedia content adaptation in pervasive computing. In *Proceedings of the First Conference on Computing Frontiers*, April 2004.

[5] E. Bertino, E. Ferrari, and G. Mella. An approach to cooperative updates of XML documents in distributed systems. *Journal of Computer Security*, 13(2):191–242, 2005.

[6] P. A. Bonatti and P. Samarati. A uniform framework for regulating service access and information release on the web. *Journal of Computer Security*, 10(3):241–272, 2002.

[7] L. Breslau, P. Cao, L. Fan, G. Phillips, , and S. Shenker. Web caching and Zipf-like distributions: evidence and implications. In *INFOCOM* '99, March.

[8] V. Cardellini, P. S. Yu, and Y. W. Huang. Collaborative proxy system for distributed web content transcoding. In *Proceedings of 9th ACM Int'l Conf. on Information and Knowledge Management*, November 2000.

[9] C. H. Chi, Y. Lin, J. Deng, X. Li, and T. Chua. Automatic proxy-based watermarking for WWW. *Computer communication*, 24(2):144–154, February 2001.

[10] C. H. Chi and Y. 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.

[11] P.-A. Chirita, W. Nejdl, M. T. Schlosser, and O. Scurtu. Personalized reputation management in P2P networks. In *ISWC Workshop on Trust, Security, and Reputation on the Semantic Web*, 2004.

[12] P. Devanbu, M. Gertz, C. Martel, and S. G. Stubblebine. Authentic third-party data publication. In *DBSec*, pages 101–112, 2000.

[13] P. T. Devanbu, M. Gertz, and A. Kwong. Flexible authentication of XML documents. *Journal of Computer Security*, 12(6):841–864, 2004.

[14] A. Fiat and A. Shamir. How to prove yourself: practical solutions to identification and signature problems. In *Proceedings of Advances in Cryptology – Crypto'86*, pages 186–194, 1986.

[15] A. Habib, D. Xu, M. Atallah, B. Bhargava, and J. Chuang. A tree-based forward digest protocol to verify data integrity in distributed media streaming. *IEEE Transactions on Knowledge and Data Engineering*, 17(7):1010 – 1014, July 2005.

[16] J.-L. Huang, M.-S. Chen, and H.-P. Hung. A QoS-aware transcoding proxy using on-demand data broadcasting. In *Proceedings of IEEE International Conference on Computer Communications (INFOCOM '04)*, March 2004.

[17] B. Li, M. J. Golin, G. F. Italiano, X. Deng, and K. Sohraby. On the optimal placement of web proxies in the Internet. In *Proceedings of IEEE International Conference on Computer Communications (INFOCOM '99)*, March 1999.

[18] T. Li, Y. Wu, D. Ma, H. Zhu, and R. 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.

[19] A. Lysyanskaya, R. Tamassia, and N. Triandopoulos. Multicast authentication in fully adversarial networks. In *Proceedings of IEEE Symposium on Security and Privacy (SSP 2004)*, pages 241–255, 2004.

[20] J. S. Park, R. Sandhu, and G.-J. Ahn. Role-based access control on the web. In *ACM Transactions on Information and Systems Security*, volume 4(1), 2001.

[21] D. J. Polivy and R. Tamassia. Authenticating distributed data using web services and XML signatures. In *Proceedings of the 2002 ACM workshop on XML security*, pages 80 – 89, 2002.

[22] S. Sivasubramanian, M. Szymaniak, G. Pierre, and M. V. Steen. Replication for web hosting systems. *ACM Computing Surveys*, 36(3):291–334, September 2004.

[23] SSL specification. Available at: `http://wp.netscape.com/eng/security/SSL_2.html`.

[24] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: a scalable peer-to-peer lookup service for internet applications. In *SIGCOMM'01*, August 2001.

[25] D. Yao, K. B. Frikken, M. J. Atallah, and R. Tamassia. Point-based trust: Define how much privacy is worth. In *Proceedings of the Eighth International Conference on Information and Communications Security (ICICS '06)*, December 2006.

[26] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. Kubiatowicz. Tapestry: A resilient global-scale overlay for service deployment. *IEEE Journal on Selected Areas in Communications*, 22(1):41–53, January 2004.