SECURED CONTENT DISTRIBUTION NETWORK USING OPTIMIZED RSA

Liji Merlin Kurian[1] and Saiju Punnoose[2]

[1]Department of Computer Science & Engineering, EPCET, Bangalore
lijisaiju@gmail.com
[2]Dell Services, Bangalore
saiju_punnoose@dell.com

ABSTRACT

Securing data is becoming a crucial need for most internet-based applications. Whereas the problem of data confidentiality has been widely investigated, the problem of how to ensure that data, when moving among different parties, are modified only according to the stated policies has been so far not deeply investigated. At present data security is an area where corporates spend huge amount to ensure that their data and their client's data are not tampered with. This paper proposes an approach that addresses data integrity and confidentiality in content adaptation and caching by intermediaries. This approach permits multiple intermediaries to simultaneously perform content services on different portions of the data. To enforce authenticity and integrity, standard cryptographic primitives such as public keys generated using optimized RSA are used for digitally signing the data. The optimized RSA approach reduces the processing time, computational overheads and key storage requirements which are the major drawbacks of RSA. The protocol defined for this project supports decentralized proxy, key management and flexible delegation of services. The experimental results show that this approach is efficient and minimizes the amount of data transmitted across the network. Ultimately, this provides a trusted way of transporting data without causing a performance hit.

KEYWORDS

Data Sharing, Distributed Systems, Integrity, Security.

1. INTRODUCTION

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. *Confidentiality* and *integrity* are two main security properties that must be ensured to data or information in all those distributed cooperative applications, such as collaborative e-commerce [7], distance learning, telemedicine and e-government. With the emergence of various network appliances and heterogeneous client environments, there are other relevant new requirements for content services by intermediaries [2].Therefore, several content services have been identified that include but are not limited to content transcoding [2], [5], in which data is transformed from one format into another, data filtering, and value-added services such as watermarking [7]. Many studies have been carried out on intermediary content services however; the problem of data security in these settings has not caught much attention. Confidentiality means that data can only be accessed by subjects who are authorized by the stated access control policies. Integrity means that data can only be modified by authorized subjects. The problem of integrity has not been much investigated, even though it is a common requirement in many application environments that not all parties be authorized to modify any data that is exchanged. In order to enhance the performance of content distribution networks (CDNs), several approaches have been developed based on the use of content management services provided by intermediary proxies. In most of these approaches, content caching is the main service provided by proxies [1], [3]. That is, instead of asking a content server for contents upon each client request, 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, the network bandwidth consumption can be reduced. System performance thus improves, especially when a large amount of data is involved. Besides these improvements, caching makes the system robust by letting caching proxies provide content distribution services when the server is not available.

When a proxy mediates data transmission, if the data is enciphered during transmission, security is ensured; however, it is impossible for intermediaries to modify the data. On the other hand, when intermediaries are allowed to modify the data, it is difficult to enforce security. The proposed architecture uses the efficient optimized RSA algorithm for digital signature. This new algorithm is an adaptation of the RSA algorithm that overcomes the shortcomings of the RSA system (processing time & computational overheads). This paper presents a general and improved protocol to meet the high availability requirement for large-scale network services with an efficient digital signing algorithm.

2. RELATED WORKS

Among the existing protocols concerning Web services, Security Socket Layer (SSL) is the only one that provides protection to data integrity. SSL relies on the concept of a secured channel. This channel guarantees confidentiality in that all messages that pass over it are encrypted. The SSL technology provides two functions: encrypting the information flow between client and server and forming the basis for mutual client/server authentication. SSL secures the data stream from the TCP/IP layer, eliminating the interference of any proxy server. Therefore, proxy caching for SSL is unavailable. Other value-added service like webpage customisation, virus scans etc. are also impossible. Another drawback of SSL is that it does not support pre-processing. Data are encrypted when the request arrives and a new encryption session must be created for each incoming request. This seriously limits the server response speed especially when the number of requests is vast.

3. SYSTEM ARCHITECTURE

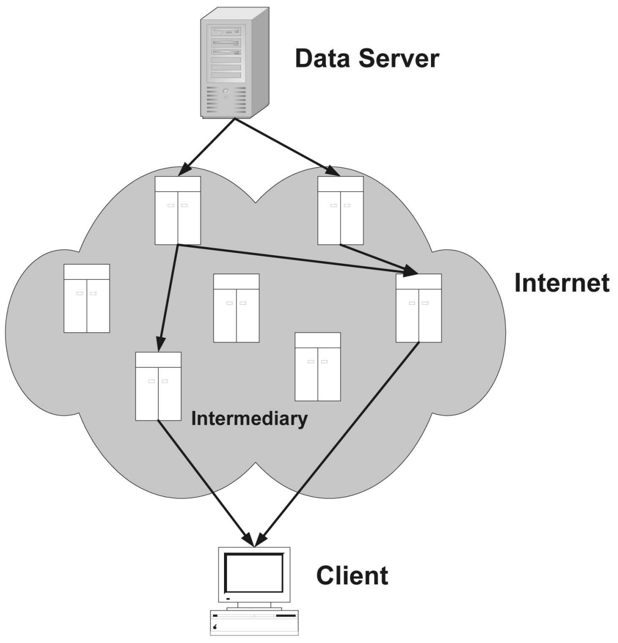

Figure 1. System architecture

In the model (Figure 1), there are three types of entities:

3.1. Data Server

This is an entity that originally stores the data requested by a client.

3.2. Client

This is any entity that requests data from a data server. When a client submits a request, besides the data it requests, it may also include some content service requirements, arising from device limitations and data format limitations [4].

3.3. Intermediary

This is any entity that is allowed by a data server to provide content services in response to requests by clients. Intermediaries include caching proxies and transforming proxies.

4. PRELIMINARIES

This section introduces the notions and terminology used in the paper.

4.1. Content Service Functions

Each content service belongs to a service function. The mapping from a content service to a service function is a many-to-one mapping. For example, a content service may compress images with less precision in order to reduce their size, or a content service may perform media conversion such as from text to audio or a format change such as from PDF to HTML. All these services belong to a transcoding function that changes the data from one format into another. We summarize the basic content service functions that intermediaries can perform in Table 1. We include some important classes of functions that are related to security services, such as the function of virus scanning.

To ensure data security, an intermediary must have certain privileges in order to access the data. Based on a client request, the data server decides the privileges for each participating proxy. For example, if a proxy needs to transcode the data from text to audio, then it needs to have certain privileges from the data server that authorizes this proxy to perform this transcoding function. Based on whether a service function needs to modify the requested data or not, there are two types of privileges that allow intermediaries to perform content service functions: read and update. The read privilege allows a proxy to read and store the data. The update privilege allows a proxy to read and modify the data. For each content service function, the corresponding privilege types are listed in Table 1.

Table 1. Functions and corresponding privileges

Function	**Purpose**	**Privilege**
Customize	Tailor for particular users	Update
Filter	Remove information	Update
Annotate	Add information	Update
Transcode	Change to different format	Update
Cache	Store for later use	Read
Aggregate	Combine from multiple sources	Read
Virus scan	Scan virus	Read, Update
Watermark	Add watermarking	Update

4.2. Data Representation

We cast our approach in the framework of XML [9], [8] because of its widespread use in Web services. XML can be used to manage data, documents, graphics, and even multimedia data. Also, XML organizes data according to hierarchical nested structures, thus facilitating the parallelization. It organizes data into tagged elements. We define an atomic element (AE) as either an attribute or an element including its starting and ending tags. A data segment is a set of elements to which the same access control policy applies. That is, if a proxy has a read (or write) privilege over a segment, the proxy has a read (or write) privilege over all the elements in the segment. We enforce confidentiality by allowing a proxy to access only the segments that are permitted by access control policies. We assume that each segment is uniquely identified.

4.3. Data Provider (DP) and P-Proxy

A DP is any entity that can provide the data requested by a client. Thus, a DP may be either a data server or a cache proxy caching the data requested by clients. In order to provide content services to clients, a DP has a group of cooperative intermediaries that can perform different content services.

A P-proxy is a list (size$\geq$1) of cooperative proxies that perform certain content services on the data on behalf of the DP. Each DP maintains the information about the services provided by each cooperative proxy in an intermediary profile table. The intermediary profile table stores the public keys and the authorizations of proxies. Table 2 shows an example of such a table. Because a proxy may provide several content services, it may appear in several different P-proxies maintained by a DP. Even though a P-proxy may group several proxies, only one proxy in such group performs the content service associated with the P-proxy on each requested data. The proxy that is initially assigned to execute the operation on the data is referred to as the primary proxy (prim) of this P-proxy for the requested data. The purpose of P-proxies is therefore to enhance both the availability and the efficiency of the system.

Table 2. Intermediary profile table

P-proxy name	Content service	Proxy/public key
P-proxy1	Virus scan	Proxy1/pubk1, Proxy2/pubk2
P-proxy2	Logo-adding	Proxy3/pubk3
P-proxy3	Audio to text conversion	Proxy4/pubk4, Proxy5/pubk5
P-proxy4	Content filter	Proxy1/pubk1

4.4. Access Control System

Each DP has its own security policy related to its data. The access control system of each DP enforces which proxies and clients can access which data. The access control system can return three possible access decisions.

4.4.1. Deny

This indicates that the DP does not have the data requested by the client, the client is not allowed to access the data according to the DP's policy, or no intermediaries in the DP's intermediary profile table exist or are allowed to transform the data into the version requested by the client.

4.4.2. Empty path

This indicates that the client's request can be satisfied without any intermediary's involvement.

4.4.3. Path with ACIS

This indicates that the client's request can be satisfied with the involvement of the P-proxies listed in the returned path. ACIS denotes access control information structure, which specifies the privileges over the data for each Pproxy in the path. We now provide details concerning paths and ACIS.

4.5. Control Information

The control information specifies which segments the primary proxy of a P-proxy in a path will receive and how the primary proxy can verify the integrity of each received segment. It also includes information on how to securely communicate with the P-proxy's successors (*succ*) and predecessor (*pred*).

4.6. Optimized RSA

In this algorithm the message to be signed is input to a one-way hash algorithm producing an unrecoverable digest of the message. The digest is encrypted with receiver's public key then sender's private key to produce signature, appended to the original message and transmitted. The proposed algorithm is shown in Figure 2. It is clear that the transmitted message is not encrypted because the signature function is sufficient for securing the transmission process. At the receiver, the message and signed digest (the signature) are separated. The original message is passed through the same hash function used by the originator and the signature is decrypted using sender's public key then receiver's private key to produce another copy of the original digest. The two digests are presented to a comparator. If they are equal, the message is accepted as genuine. If they do not match, the message is rejected, as shown in Figure 2.

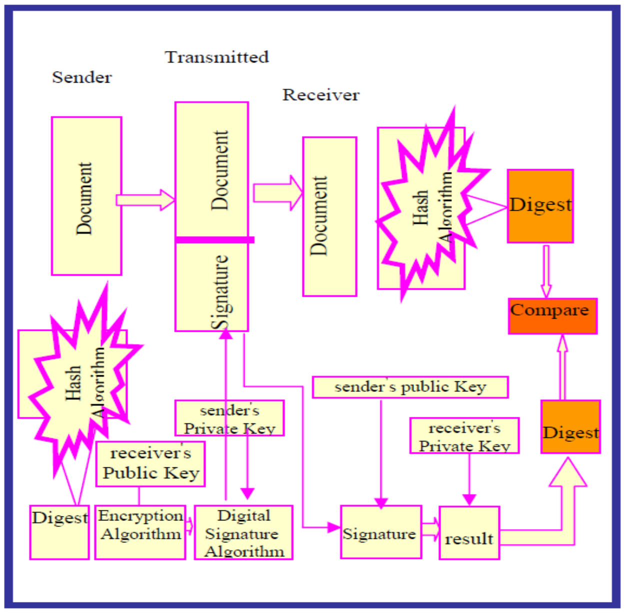

Figure 2. Optimized RSA Algorithm

5. PARALLEL SECURE CONTENT SERVICE PROTOCOL

Before presenting the PSCS protocol, we give the security assumptions in the model:

1. Each DP has a group of P-proxies that are cooperative with the DP in order to provide content services to clients.

2. All proxies in a P-proxy are equal. That is, there is no proxy considered more trustworthy than another by the DP.

3. All P-proxies that can perform the content services requested by a client are operative.

4. They will execute their permitted update privileges correctly and will not collude with each other.

5. If a P-proxy in a path cannot correctly perform the required content service, which involves updates, the request from the client cannot be satisfied.

6. A P-proxy may attempt to access data without permission, thus violating data confidentiality, or it may attempt to modify data without permission, thus violating data integrity.

5.1. Parallel Secure Content Service Data Server Protocol

5.1.1. Data Server Protocol

The data server sends the client's request to its access control. If the access decision on the request is deny, the data server notifies the client of the result. If the output is an empty path, the data server computes and signs the hash value of the data. Then, the data is transferred to the client using SSL/TLS. Both integrity and confidentiality are trivially satisfied in this case. We focus on a more complex case where the outputs from the access control system are a path and an ACIS.

5.1.1.1. Control Information Generation

The purpose of the control information is to help the P-proxies and the client to verify the integrity of the data and to securely communicate with each other.

5.1.1.2. Initialization

First, for each participating P-proxy, the data server randomly orders the proxies in each P-proxy. The first one in the list is the primary proxy for this P-proxy. This random ordering avoids making certain proxies overloaded, especially when a proxy is being used in many P-proxies in a data server or a proxy appears in many DPs' intermediary profile table. This step also initializes each Pproxy's predecessors (*pred*), successors (*succ*) and segments (*seg*) that this P-proxy is authorized to access.

5.1.1.3. Segment Labeling

For each P-proxy in Path, this step labels each segment that this P-proxy is authorized to access with a list of public keys of the P-proxy that can modify this segment. The list of keys starts with the primary proxy of the P-proxy, followed by other proxies' public keys in the Pproxy. We use array *Seg[i]* to store the public keys of the P-proxy that most recently modified segment i and a structure *i* that contains the set of accessible read and update segments of the P-proxyi in Path.

5.1.1.4. Generating the Data Server's CI

The data server needs to send segments to corresponding P-proxies or to the client. Each P-proxy must receive the segments that are allowed to access and send some or all of these segments to subsequent P-proxies or the client. For each segment, the data server scans the P-proxies according to the content service path; if a P-proxy needs to read this segment, then the data server adds this P-proxy to its succ, and this P-proxy will receive this segment from the data server. The data server repeats this activity until the first P-proxy that needs to update this segment. After the P-proxy updates the segment, it sends out the segment to the rest of the P-proxies or the client if they need to access it.

5.2. Key Generation

The sender and the receiver uses the following algorithm to generate a public key and a private key [10].
INPUT : Bit length of modulus, k.
OUTPUT: Public key (E; N), and private key (D; N).
1) Generate prime numbers (P_a)and (P_b)of bit length [k/2]
2) Generate prime numbers (Q_a) and (Q_b)of bit length k – [k/2]
3) Compute N $\leftarrow$ P. Q for both a & b
4) Select E to be an integer, where GCD (e; (P - 1).(Q - 1)) = 1
5) Compute D such that E . D ≡ 1 (mod (P - 1) . (Q - 1))
6) Return ((E_a; N_a); (D_a ;N_a) for A And ((E_b; N_b); (D_b ;N_b)) for B.

5.3. Signature Generation

INPUT : Private Key (Da ; Na) for the sender, Public key (Eb ; Nb) for the receiver , and message to be signed, M.
OUTPUT: s, signature of M
1) A $\leftarrow$ $h(M)^{E_b}$ (mod N_b)
2) S $\leftarrow$ A^{D_a} mod Na.
3) Return (s)
A common hash algorithm used is SHA-1.

5.4. Signature verification

INPUT : Private Key (Db; N_b) for the receiver, Public key (E_a ;N_a)for the sender , message(M) and signature (S).
OUTPUT: VALID or INVALID.
1) B $\leftarrow$ S^{Ea} mod Na
2) Q $\leftarrow$ B^{Db} mod N_b
3) If Q = h(M), Return VALID
Else, Return INVALID

5.5. Intermediary Proxy and Client Protocol

During content service processing, the primary proxy in each P-proxy can either delegate the content services to another proxy in the same P-proxy or execute the function by itself. If no delegation needs to be performed, the primary proxy executes a protocol that is similar to the client protocol. Such protocol has the following steps: check the integrity according to incoming package templates received from the data server, execute operations according to privileges, form package(s) according to outgoing package templates received from the data server, and send out packages. The details of these steps are given as follows:

Step 1. Integrity check

Upon receiving a package P, the receiver with control information CIi verifies whether there has been any transmission error; if there is any such error, it asks the sender to send the package again. It verifies that the package is from one of its predecessors.

Step 2. Executing functions on the data

After correctly receiving a package from each predecessor, the receiver executes its functions on the data. If it has update privileges on some segments, it will update the segments, calculate the hash value for each segment it updated, and cipher this value with its private key for future authorization checking.

Step 3. Forming new package(s)

For each su_y *E CIi.succ*, the receiver forms an outgoing package U such that U:*pid* = *i*. For each r *E* su_y.*seg*, the receiver fills r in U. After this, the subject encrypts U with $su_{y.}sk_{iy}$ and sends it to the primary proxy of P-proxy. The receiver should also keep a copy for later recovery. If a primary proxy decides to delegate its function to another proxy q in the P-proxy, the primary proxy first performs the integrity checking as in step 1. If there is no error, the primary proxy sends all the received packages to q, which completes the second step. q sends back only the updated segments with the delegateHash attributes signed by q. The primary proxy calculates the hash value for each segment that is updated by q, makes sure that these values are equal to those signed by q, and then signs these hash values. At last, the primary proxy executes the last step of forming and sending out the outgoing packages.

6. Complexity And Security Analysis

We analyze the security for the case that a client's request is handled by a data server. Due to space limitations, the analysis about the security of the PSCScp Protocol is not provided, as it is similar to the analysis of the PSCSds Protocol. Our protocol assumes a digital signature scheme that is secure against existential forgery attacks. Intuitively, this means that a polynomial-time adversary cannot successfully forge the signature of a message that a signer has not signed. We also assume a symmetric key encryption scheme with one-way security, which intuitively means that a polynomial-time adversary cannot successfully guess the plaintext after seeing the cipher text.

Theorem 1. The PSCSds Protocol is secure with respect to integrity.

We need to prove that a proxy cannot update a segment over which it does not have update privileges. If a primary proxy delegates another proxy q in the Pproxy for the execution of content services, according to the protocol executed by intermediary proxies, proxy q cannot modify a segment that it is not allowed to update and send the segment out. Thus, we only need to consider the primary proxies. There are two cases:

Case 1: A primary proxy modifies a segment that is not authored by itself, and the proxy does not have the authorization to update the segment.

In this case, integrity is enforced by digital signatures. When a proxy i updates a segment seg, it signs the hash value that it generates from seg with its private key. If a primary proxy j has read authorization on seg, j receives control information from the data server, which contains an incoming template. The incoming template includes the public key of i for deciphering the hash. j will calculate the hash of the segment and check the signature. Suppose an adversarial proxy can modify segment seg before it reaches j and the signature on the modification is accepted by j. This means that the adversarial proxy can successfully forge i's signature and thus break the digital signature scheme. This contradicts with the assumption on the security of the signature scheme. If a primary proxy m receives two segments authored by the same proxy i, it cannot switch the information in these two segments, as a segment represented in XML has the segment identifier in its tag and this identifier is included when the hash value is generated. Although a primary proxy may not indicate that it delegates another proxy for the update of a segment, this does not violate the integrity, because the primary proxy also has the privilege to update the segment. It can write the segment with the result of the delegated proxy. Thus, no proxy can modify a segment that has not been authored by it and for which it does not have an update privilege.

Case 2: A primary proxy modifies a segment authored by itself, even though it does not have update authorization over it now.

First, we discuss the case in which no delegation happens. Suppose a segment seg is updated by proxy A, then read by proxy B, and, after that, sent by B to A for reading (A cannot update seg this time) and then by B to C for reading. In this case, A cannot send C a segment that is different from the one it sends to B. As in the control information generation, B will send a copy to C instead of A. Thus, C receives the segment that A authored at the beginning. In the case of delegation, suppose a segment seg is updated by A, then read by B, and then distributed by B to C for reading. Suppose A delegates to B, updating seg; after B finishes updating, A also needs to sign the hash value of the segment. B cannot change seg when accessing it the second time, because C will also verify the segment with A's signature as A is the primary proxy. Thus, data integrity is enforced.

Theorem 2. The PSCSds Protocol is secure with respect to confidentiality.

We need to prove that if a proxy is not authorized to access a segment, then it cannot access it. During the communication, the confidentiality is enforced by the symmetric key that enciphers/deciphers a package so that only the designated receiver can access it. When a proxy receives a package, it uses the control information received from the data server to decipher the package. Suppose an adversarial proxy has successfully deciphered the package and broken the confidentiality. This means that the adversary has broken the symmetric key encryption scheme, which contradicts with the assumption of a secure symmetric key encryption scheme. Besides, the control information generation ensures that a proxy only receives the segments that it is authorized to access. Thus, the protocol is secure with respect to confidentiality. It is important to notice that under our approach a receiver could learn the access privileges of its predecessor(s) or successor(s). The disclosure of such access privileges, however, does not violate confidentiality and integrity, because these requirements only concern the data contents.

7. EXPERIMENTS AND RESULTS

In this section, we report some results for the experiments we have carried out to evaluate the performance of our approach. The primary goals of our experiments are to understand the characteristics of the PSCS Protocol with respect to three properties.

1. *Size of transmitted data*. How much bandwidth our approach saves?
2. *Integrity checking time.* How much does this security requirement cost?
3. *Content servicing time.* How much could the PSCS Protocol save in the overall servicing time and how does the recovery affect the overall servicing time?

We compare our approach with a centralized approach. In the centralized approach, the data server sequentially contacts each P-proxy to complete the requested content services. Once an intermediary has completed the operation, it sends back only the data that it has updated to the data server, and the data server then prepares the data for the next P-proxy. Both the server and intermediaries use message digests and digital signatures to ensure data integrity. Once the data server receives the data from the last intermediary, it finally sends the data to the client. The centralized model has a star-shaped communication pattern, where the data server is at the centre, and each proxy only communicates with the data server. In addition, the communication is always a round-trip traffic, that is, from the data server to a proxy and then back. The experiments have been carried out in a high-speed local area network with an average bandwidth of about 100 Mbps. The optimized RSA algorithm generates the signature at the receiver end and the same is verified at the receiver's end.

7.1. Data Size

Reducing the total size of the data transmitted in the system could improve system performance by reducing congestion and collision. In a low-bandwidth system, the time difference between transferring a large amount of data and a small amount of data is quite significant. First, we should notice that even if the content is divided into different segments such that the sum of all segment sizes is equal to the size of the whole content, for a symmetric encryption algorithm, the size of the ciphertext is the same as the cleartext. Thus, the ciphertexts under these two approaches have the same size. Second, in XML encryption, the cleartext uses UTF-8 encoding (8 bits per character), whereas the ciphertext is base64 (6 bits per character) text. Thus, the ciphertext is about 1.33 times of the size of the cleartext. Enforcing the confidentiality of XML data during communications requires transmitting more data. Figure 3 reports the total size of the data transmitted by all entities in the system under the two approaches when no recovery is executed. In the experiment, we considered the case of 100-Kbyte data equally divided into 25 segments. All segments are accessed by four P-proxies, and RSA signatures of size 128 bytes are used. From the graph, we can see that our approach requires transferring less data than the centralized approach when the update/access ratio is higher than 4 percent. Because most content service functions require an update privilege that has an update/access ratio of 100 percent, our approach is much more efficient than the centralized approach with respect to the amount of data to be transmitted on the network.

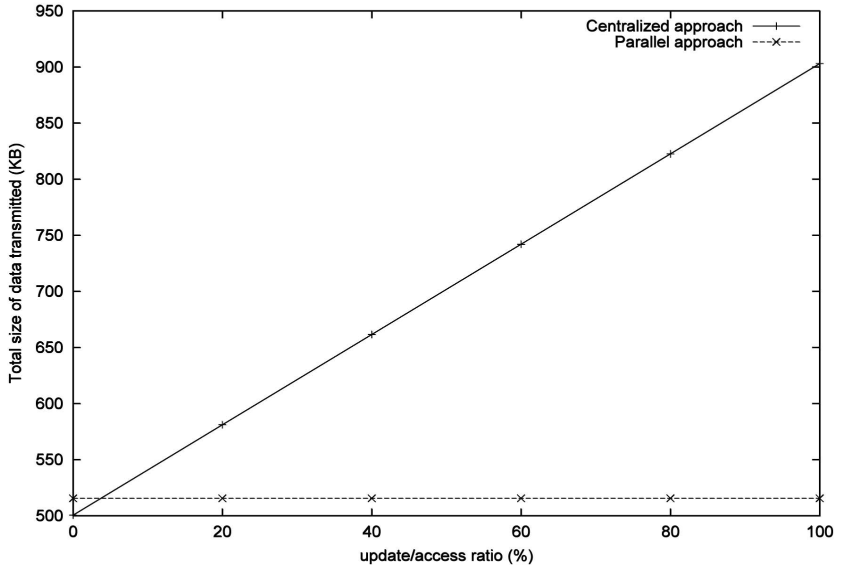

Figure 3. Comparison of the total size of data transmitted

7.2. Integrity Check Time

Since our approach requires each P-proxy to perform an integrity check, we have evaluated the time of this check by dividing a whole document equally into five segments. For the parallel approach, the time of the check is given by the sum of the times required for computing the hash values of the five segments and for the optimized RSA signature verification. The signature verification is much faster as in Table 4 compared with RSA as the proposed system uses two keys for the verification.

Table 4. RSA Vs Optimized RSA

	Key generation (sec)	**Signature generation (sec)**	**Signature verification (sec)**
Experiment 1			
RSA	4.485000	0.016000	13.594000
Optimized RSA	10.719000	0.015000	0.015000
Experiment 2			
RSA	7.735000	0.016000	13.281000
Optimized RSA	10.750000	0.031000	0.047000
Experiment 3			
RSA	12.324000	0.016000	19.625000
Optimized RSA	11.45600	0.040000	0.0500000

7.3. Overall Content Servicing Time

First, we investigate how much our approach could save in the overall content servicing time against the centralized approach. The experiments have been run with four proxies (as data normally require about four different content services), and all segments have the same size (size ¼ N=M, where N is the size of a document, and M is the number of segments). Each proxy transcodes the segments it receives from the data server, and then, these results are sent to the client. We use a ratio to represent the data size distribution.

8. CONCLUSIONS

In this paper, we have presented a solution for secure content services characterized by scalable and robust network architecture using XML. This paper can definitely benefit medium and large scale industries for customised services. Our protocol allows a client to verify that the received data is authentic and transformations on the data are properly authorized thus we present a general and improved protocol to meet the high availability requirement for large-scale network services. Our approach also assures data confidentiality during transmission. It highlights load distribution through P-proxies to improve system performance and supports parallel content services. Because no modification is required to current content distribution systems in order to adopt our approach, our work is easy to deploy for many applications. In addition, our approach is extensible; if a new type of content service is required, our architecture can be easily adapted to the new requirements.

REFERENCES

[1] C. Aggarwal, J.L. Wolf and P.S. Yu, (2000) "Caching on the World Wide Web", *IEEE Trans. Knowledge and Data Eng.,* Vol. 11, No. 1, pp94- 107.

[2] G. Berhe, L. Brunie, and J.M. Pierson, (2004) "Modeling Service-Based Multimedia Content Adaptation in Pervasive Computing", *Proc. First Conf. Computing Frontiers*.

[3] L. Breslau, P. Cao, L. Fan, G. Phillips and S. Shenker, (1999) "Web Caching and Zipf-Like Distributions: Evidence and Implications", *Proc. IEEE INFOCOM '99*.

[4] S. Buchholz and A. Schill, (2003), "Adaptation-Aware Web Caching: Caching in the Future Pervasive Web", *Proc. 13th GI/ITG Conf. Kommunikation in Verteilten Systemen (KiVS).*

[5] V. Cardellini, P.S. Yu and Y.W. Huang, (2000) "Collaborative Proxy System for Distributed Web Content Transcoding", *Proc. Ninth ACM Int'l Conf. Information and Knowledge Management (CIKM '00).*

[6] S. Chandra and C.S. Ellis, (1999) "JPEG Compression Metric as a Quality- Aware Image Transcoding", *Proc. Second Usenix Symp. Internet Technology and Systems (USITS '99).*

[7] C.H. Chi and Y. Wu, (2005) "An XML-Based Data Integrity Service Model for Web Intermediaries", *Proc. Seventh Int'l Workshop Web Content Caching and Distribution (WCW'05).*

[8] Chen M.S., Huang J.L. and Hung H.P., (2006) "A QoS-Aware Transcoding Proxy Using On-Demand Data Broadcasting", *Proc. IEEE INFOCOM '06,* pp315-317.

[9] Freedman M.J., Freudenthal E. and Mazie`res D., (2007) "Democratizing Content Publication with Coral", *Proc. Usenix/ACM Symp. Networked Systems Design and Implementation (NSDI '07),* pp201-212.

[10] Lum W.Y. and Lau F.C.M., (2009) "On Balancing between Transcoding Overhead and Spatial Consumption in Content Adaptation", *Proc. ACM MobiCom '09,* pp239-250.

[11] Markatos E., (2000) "Main Memory Caching of Web Documents", *Computer Networks and ISDN Systems*, Vol. 28, pp893-905.

[12] Pierre G., Sivasubramanian S., Szymaniak M. and Steen M.V., (2008) "Replication for Web Hosting Systems", *ACM Computing Surveys,* Vol. 36, No. 3, pp291-334.

[13] Wang J., (2005) "A survey of web caching schemes for the Internet", *ACM Computer Communication Review 29(5),* pp10-21.

[14] *Content Distribution Networks*, http://en.wikipedia.org/wiki/Content_Delivery_Network, 2010.

[15] *Content Distribution Networks*, http://encyclopedia2.thefreedictionary.com/CDN, 2010.

[16] *Extensible Markup Language (XML),* http://www.w3.org/XML/, 2010.

[17] Andrew S. Tanenbaum, (2009) *Computer Networks,* Pearson Education.

[18] Larry L. Peterson & Bruce S. Davie, (2009) *Computer Networks-A Systems Approach,* Morgan Kaufmann Publishers.

[19] Scot Hull, (2008) *Content Delivery Networks,* Tata McGraw-Hill Edition.

[20] William Stallings, (2009) *Cryptography & Network Security,* Pearson Education Asia.

Authors

Liji Merlin Kurian received Master of engineering degree in Computer Science from Anna UniversityCoimbatore, india in 2010. She is now an assistant professor in East Point College of Engineering, Bangalore. She has presented many papers in National and International conferences on network security and cryptography. Her research interests are network security and cryptography

Saiju Punnoose received Masters degree in computer science from Bharathiar University Coimbatore, India in 2002. He is with Dell services. His research interests are secured web applications and security.