Efficient Proxy-Based Internet Media Distribution Control and Privacy Protection Infrastructure

Songqing Chen[1], Shiping Chen[2], Huiping Guo[3], Bo Shen[4], and Sushil Jajodia[2]

[1]Dept. of Computer Science
George Mason University
Fairfax, VA 22030
sqchen@cs.gmu.edu

[2] Center for Secure Information Systems
George Mason University
Fairfax, VA 22030
{schen3, jajodia}@gmu.edu

[3] Dept. of Computer Science
California State University
Los Angeles, CA 90032
hpguo@calstatela.edu

[4] Mobile & Media Systems Lab
Hewlett-Packard Laboratories
Palo Alto, CA 94304
boshen@hpl.hp.com

***Abstract*— Massive Internet media distribution demands prolonged continuous consumption of networking and disk bandwidths in large capacity. Many proxy-based Internet media distribution algorithms and systems have been proposed, implemented, and evaluated to address the scalability issue. However, few of them have been used in practice, since two important issues are not satisfactorily addressed. First, existing proxy-based media distribution architectures lack an efficient media distribution control mechanism. Without protection on the Internet, content providers are hesitant to use existing fast distribution techniques. Second, little has been done to protect client privacy during client accesses. Straightforward solutions to address these two issues independently lead to conflicts. For example, to enforce distribution control, only legitimate users should be granted access rights. However, this normally discloses more information (such as which object the client is accessing) other than the client identity, which conflicts with the client's desire for privacy protection. In this paper, we propose a unified proxy-based media distribution protocol to effectively address these two problems simultaneously. We further design a set of new algorithms for cooperative proxies where our proposed scheme works practically. Simulation results show that our proposed strategy is efficient.**

I. Introduction

Recent years have witnessed the dramatic and continuous increase of Internet media applications. Different from the Web object delivery, the delivery of media objects demands prolonged continuous use of networking, disk, and other resources in large capacity. To efficiently deliver Internet media, a lot of research has been conducted to extend proxy caching for media content and many proxy-based media distribution systems have been proposed to improve the scalability and efficiency for Internet media delivery (e.g. [1], [2], [3], [4], [5], [6], [7], [8], [9]).

Although these strategies and systems can improve caching performance and client playback quality, such as reducing the startup latency and minimizing playback jitter under various conditions, none of these systems has been practically deployed because the following two problems have not been well addressed.

The first problem is media distribution control. The proxy-based media delivery system addresses the scalability issue by caching media objects in a proxy closer to clients. However, this approach makes it difficult for distribution control. Once a media object is cached in the proxy, any client can access it without notifying the server. The server thus loses the ultimate control of who is allowed to access the object. In other words, the interest of the content provider is not protected, which prevents content providers from making their digital goods available on-line. Existing practice often uses pay-per-view, membership, or DRM schemes [10] without considering effective utilization of proxy caching. Some recently developed strategies tried to leverage from proxy caching to deal with the problem with the aid of encryption mechanisms [11], [12], [13]. But they either demand special hardware support or cannot work if some clients collude.

The second problem is that existing strategies have given little consideration to client privacy protection during content accesses. Through on-line transactions, E-Commerce makes it easier to get and compile a customer's personal information. Various kinds of activities could be conducted based on the collected information, such as advertisements based on client preferences extracted from client purchases. This issue raises serious concerns from not only individuals, but also law enforcement [14]. For example, when a client accesses a media server for a movie, it is difficult to prevent the server from knowing which movie is being accessed. But people sometimes do not want the server to know which object they are accessing. If this problem is not well addressed, clients may be reluctant to use Internet media distribution services. On the contrary, if a media service provider can protect client privacy, it can potentially attract more clients. Although there are some initial efforts considering privacy protection in the E-commerce context (e.g. [15]), the efficiency is poor and little work has been conducted to protect client privacy in the proxy-based media distribution architecture.

Efforts have been made to address these two problems

1-4244-0476-2/06/$20.00 ©2006 IEEE.

separately. However, a desirable proxy based media delivery system should be capable of simultaneously protecting the interests of both the content provider and the clients, i.e., to provide distribution control for the content provider and privacy protection for clients. But addressing these two problems at the same time may lead to conflicts: for distribution control, only legitimate users should be granted access rights. However, this normally requires revealing client identity, thus leading to an invasion of client privacy since the server may know which object the client is accessing. To the best of our knowledge, there is no efficient and practical proxy-based solution for simultaneous media distribution control and privacy protection.

In this paper, we propose a unified strategy that can provide distribution control and privacy protection at the same time in proxy-based Internet media delivery systems. This unified strategy consists of a key division based distribution control scheme and a commutative cipher based privacy protection scheme. Based on the unified strategy, we further design a set of algorithms in the cooperative proxy environment where our proposed strategy works practically. Extensive simulation-based experiments have been conducted to evaluate the proposed system. The preliminary results show that our proposed system is very effective. Note that our proposed strategy works when the proxy and the content provider do not collude. If they collude or the proxy belongs to the content provider, the scheme cannot be applied.

The reminder of this paper is organized as follows. Some related work is discussed in section II. We dissect the two components of our unified strategy and present distribution control and privacy protection in section III and section IV, respectively. We describe the unified strategy in section V, its analysis in section VI, and design the algorithm for cooperative proxies in section VII. The evaluation results based on simulations are presented in section VIII. We make concluding remarks in section IX.

II. Related Work

Internet content distribution control has been investigated for some time. Without considering proxy caching, existing schemes generally employ *pay-per-view*, *membership*, or DRM schemes for distribution control [16]. When a proxy is present, the major approach for distribution control relies on encryption and decryption and focuses on improving their efficiencies. For example, early work [17], [18], [19] considers how to speed up the decryption and encryption process. The scheme proposed in [11] relies on costly hardware support to deal with member collusion. A recent work [13] proposes a multi-key video proxy infrastructure without such a requirement. However, the scheme is also vulnerable to member collusion, as pointed out in [12].

On the other hand, privacy protection has been studied in many other areas (e.g. [20], [21] in databases) and solutions are mainly based on encryption or anonymizing. For example, work [15] provides privacy protection through encryption. However, this work relies on an independent third party and introduces lots of traffic overhead. Many anonymizing based approaches (e.g. [22], [23], [24], [25]) can protect client identity through mixing, but few of them work in the context of on-line media distribution systems.

Other than in proxy-based environments, key division strategies (e.g. [26]) and commutative ciphers (e.g. [27]) have been studied extensively for various purposes. Compared to existing studies, our work targets proxy-based media delivery applications and we propose a framework that achieves distribution control and privacy protection at the same time.

III. A Proxy-based Distribution Control Scheme

The essential goal of distribution control in the context of proxy caching is to enforce access control while leveraging the proxy caching capability. The naive approach is that the server encrypts objects and puts encrypted objects on a proxy. All clients can get any encrypted object from the proxy freely, but to access the plaintext of an object, they must contact the server to get the decryption key. The major limitation of this approach is that it is vulnerable to client colluding attack: once a client gets the key and shares with others, the server loses its control on the object distribution. Periodically changing the object encryption keys can partially mitigate the threat, but the overhead caused by frequent rekeying operations is overkill.

In this section, we propose a novel scheme via which the server can take advantage of proxy caching without losing control over object distribution. The scheme is based on key division cipher.

A. Key Division Cipher System

Definition 1 (Key Division Cipher System): For object M, there is a key pair (K_e, K_d) in a crypto system such that $M = D(E(M, K_e), K_d)$, where E( ) is the encryption function and D( ) is the decryption function. If K_d can be broken into K_{d_1} and K_{d_2} ($K_d = K_{d_1} \bigotimes K_{d_2}$, $\bigotimes$ is an operator) such that $M = D(D(E(M, K_e), K_{d_1}), K_{d_2})$, we call the crypto system a key division cipher system on operator $\bigotimes$.

El Gamal [28] is a reliable public cipher system based on discrete logarithms. We first briefly introduce El Gamal, and then prove it is also a key division cipher system.

Suppose client A wants to send a message to client B by the El Gamal cipher system, the protocol works as follows.

1) A and B select two common parameters, a prime number q and a primitive root of q, α, where $\alpha < q$.
2) Client B generates a key pair (X_B, Y_B), where $X_B < q$ and $Y_B = \alpha^{X_B} \ mod \ q$. Then B keeps X_B secret and sends the public key Y_B to user A.
3) To send a message M to B, A chooses a random integer k such that $1 \leq k \leq q-1$ and computes $K = (Y_B)^k \ (mod \ q)$. Then A encrypts M as the pair of (C_1, C_2), where $C_1 = \alpha^k \ (mod \ q), C_2 = KM \ (mod \ q)$, and sends them to B.
4) Client B decrypts the received data as follows: B computes $K = (C_1)^{X_B} \ (mod \ q)$ first, then decrypts the message as $M = (\frac{C_2}{K}) \ (mod \ q)$.

Theorem 1: El Gamal is a key division cipher system on operator "+".

Proof: Suppose we break the secret key X_B in the above description into two keys X_{B_1} and X_{B_2} ($X_B = X_{B_1} + X_{B_2}$). Client A does the encryption in the same way and sends the encryption result (C_1, C_2) to B. With the two keys X_{B_1} and X_{B_2}, B can get M via the following two steps: 1) computes $K_1 = (C_1)^{X_{B_1}}(mod\ q)$ and $M_1 = \frac{C_2}{K_1}(mod\ q)$; and 2) computes $K_2 = (C_1)^{X_{B_2}}(mod\ q)$ and $M_2 = \frac{M_1}{K_2}(mod\ q)$. It's obvious that $M_2 = M$ because 1) $K = (Y_B)^k(mod\ q) = (C_1)^{X_B}(mod\ q) = (C_1)^{X_{B_1}+X_{B_2}}(mod\ q) = K_1K_2$; and 2) $M = \frac{C_2}{K}(mod\ q) = \frac{C_2}{K_1}/K_2(mod\ q) = \frac{M_1}{K_2}(mod\ q) = M_2$. ■

B. Our Proposed Scheme

Now we present our scheme by extending El Gamal. The basic idea is as follows. To distribute an object M, the sever generates a key pair, using El Gamal cipher. The server encrypts M using the public key and stores the encrypted object on the proxy. To access the object, a client contacts the server for the decryption key after it gets the object. Upon a client request, the server breaks the corresponding secret key into two keys and sends them to the proxy and the client, respectively. Then the proxy decrypts the object using the key it receives following the El Gamal protocol and sends the result to the client. After receiving the partially decrypted object, the client finishes the decryption using the key received from the server. In detail, each step works as follows.

1) For object M_i, the server selects a key pair (X_i, Y_i), where $Y_i = (\alpha)^{X_i}\ (mod\ q)$;
2) The server then selects a random number k and computes C_1, where $C_1 = \alpha^k\ (mod\ q)$. Further, the server computes K_i where $K_i = (Y_i)^k\ (mod\ q)$ and encrypts the object with K_i to produce the encrypted object $C_2 = K_iM_i$. The encrypted object C_2 gets cached in the proxy.
3) The server breaks the secret key X_i into two parts, X_{i1} and X_{i2}, where $X_i = X_{i1} + X_{i2}$, and sends to the proxy and client pairs of (C_1, X_{i1}) and (C_1, X_{i2}), respectively.
4) The proxy partially decrypts the object with X_{i1} as follows: it computes $K_{i1} = C_1^{X_{i1}}\ (mod\ q)$, and decrypts through $M_{i2} = \frac{C_2}{K_{i1}}$ and sends to the client.
5) The client then decrypts the "partially" decrypted object M_{i2} with X_{i2}: it computes $K_{i2} = C_1^{X_{i2}}\ (mod\ q)$, and decrypts through $M_i = \frac{M_{i2}}{K_{i2}}$ to get the original object.

IV. Privacy Protection in Proxy-based Distribution System

As aforementioned, today E-Commerce makes it easier for the service provider to get and compile customer personal information so that various activities could be conducted based on the collected information. An increasing number of people are concerned about the misuse of their private information and desire privacy protection in on-line transactions.

Accessing an object on the server through a proxy naturally protects the client from identity exposure to the server. However, in this case the server may lose the access control on its objects, since clients can directly get the objects from the proxy without letting the server know (if the proxy is not associated with the server). On the other hand, in this case, the system cannot protect client privacy since the proxy knows which object the client has accessed. The challenge of privacy protection problem in the context of proxy caching is how to achieve the client privacy protection while not endangering the server's distribution control on the content.

In this section, we propose a new scheme which can protect the client's privacy in the proxy-based distribution system. Our scheme is based on a commutative cipher.

A. Commutative Cipher System

Definition 2 (Commutative Cipher): Given a crypto system, for an object M, for any two keys, (K_{e1} and K_{e2}), if the sequence of these two keys to encrypt the object does not matter, that is, $E(E(M, K_{e1}), K_{e2}) = E(E(M, K_{e2}), K_{e1})$, the system is commutative.

Two popular commutative cipher systems are Pohlig-hellman based on Elliptic Curve Cryptography (ECC) and Shamir-Omura based on RSA ([29], [30]). Our scheme is based on the Shamir-Omura algorithm. It is used to exchange symmetric keys between communication parties A and B. It works as follows.

1) A selects a key M it wants to use for communication with B. A encrypts M with its key K_{EA} (only known to A) and sends $E(M, K_{EA})$ to B.
2) B receives this and further encrypts with his key K_{EB} (only known to B) and sends $E(E(M, K_{EA}), K_{EB})$ to A.
3) A then decrypts the received message, and sends back to B. Since K_{EA} and K_{EB} are commutative, this enables B to get the communication key M successfully with another decryption.

B. Our Proposed Scheme

Supported by the commutative cipher, we thus design a scheme to protect the client privacy in a proxy-based media distribution system. The system should have the following features:

- The server generates a key pair (K_E, K_D) based on Shamir-Omura and advertises/lists the IDs (such as names) of objects in plaintext so clients can freely browse them;
- For each encrypted object, the server also encrypts the object ID with key K_E before the encrypted object gets cached in the proxy as $ID^S = E(ID, K_E)$. The corresponding decryption key is K_D.
- The objects cached in the proxy are only identifiable through their corresponding encrypted IDs (ID^S).

With the above features, our proposed design works as follows.

1) When a client wants to access an object, the client generates a key pair (K_e, K_d) based on Shamir-Omura and encrypts the object ID with K_e to get $ID^C = E(ID, K_e)$ and sends to the server.

TABLE I

NOTATIONS USED IN THE PROPOSED STRATEGY

(K_e, K_d)	client key pair for communicating with servers
(K_D, K_E)	server key pair for communicating with clients and the proxy
ID^S	server encrypted object ID
M	the client requested object
M^E	server encrypted object
$S_c + S_i$	object encryption keys
S_{c1}, S_{c2}	keys to proxy and client for decryption

2) Upon the arrival of the client message, the server further encrypts the ID^C with its encryption key K_E and sends back, the client can get $(ID^C)^S = E(E(ID, K_e), K_E)$.
3) Since K_e and K_E are commutative, the client can decrypt the $(ID^C)^S$ with K_d, and get $ID^S = D(E(E(ID, K_e), K_E), K_d) = E(ID, K_E)$.
4) Since the object is identifiable in the proxy via ID^S, the client can thus request the object from the proxy without letting the server or the proxy know which object it is accessing.

V. UNIFIED DISTRIBUTION CONTROL AND PRIVACY PROTECTION PROTOCOL

In the previous sections, we proposed schemes to address distribution control and privacy protection separately. As aforementioned, achieving both objectives simultaneously is difficult. In this section, we integrate the two proposed schemes to get a unified one which can achieve two goals seamlessly.

In this new scheme, we assume that a media provider always categorizes available media objects (such as movies) into different types and charges accesses to objects in the same category with the same price. The distribution control also needs to enforce different charges on different types of objects. Our proxy-based protocol is capable of comprehensive distribution control and client privacy protection simultaneously with such practical conditions.

A. A Unified Strategy

To present our unified strategy, we have the following notation.

- k anonymity: k is the privacy protection indicator, which means that the server only knows that the client accesses one of k objects, but could not know which one exactly. In this study, we assume if k anonymity is achieved, client privacy is well protected;
- On a media server, objects (videos) are classified into n classes according to their prices (we use price as the criteria as an example here). For each price class, there exist multiple ($\geq k$) objects;
- For simplification, each object is identified by an ID on the server, which could be the name of the object, or the URL of the object; each object is uniquely identifiable by its encrypted ID on the proxy;
- In an object class, a same key pair is used to encrypt/decrypt all the object IDs, while the objects themselves are encrypted with different keys through El Gamal.

We assume that encrypted objects have been cached in the proxy. Since the content server classifies objects into different classes, we describe the strategy for clients accessing objects in one class.

As aforementioned, different objects are encrypted with different El Gamal keys. In the unified strategy, each key consists of two parts: a shared part S_c (which is the same for all objects in an object class) and an individual part S_i (which is different for different objects in an object class). The notations are summarized in Table I.

To prepare for client accesses, the server needs to complete three tasks.

1) The server performs encryption as follows:
$$M^E = E(M, S_c + S_i), \quad (1)$$
and M^E is cached in the proxy, where $E()$ is El Gamal based encryption algorithm.
2) The server also generates a key pair (K_E, K_D) based on Shamir-Omura and encrypts the object ID with K_E. The object is only identifiable with its encrypted ID, ID^S, where $ID^S = E(ID, K_E)$, in the proxy.
3) The server also encrypts the S_i of each object with key K_E (the server can use a different key), and makes available an on-line tuple of object ID and encrypted S_i —- $(ID, E(S_i, K_E))$.

A client access follows the protocol as depicted in Figure 1. Details of each step are described as follows.

- ①: When a client wants to access an object, it gets the object ID and $E(S_i, K_E)$. There are many different ways to get these without letting the server know (e.g., simply copying them down). Then the client generates a key pair (K_e, K_d) based on Shamir-Omura and encrypts the tuple with K_e,
$$E(ID, K_e)||E(E(S_i, K_E), K_e), \quad (2)$$
and sends the encrypted message to the proxy.
- ②: The proxy forwards the request to the server. After the access is authorized (e.g., the server gets payment from the client), the server performs encryption on encrypted ID and decryption on $E(E(S_i, K_E), K_e)$ from the client. Then the server sends the result back to the client,
$$E(E(ID, K_e), K_E)||D(E(E(S_i, K_E), K_e), K_D). \quad (3)$$

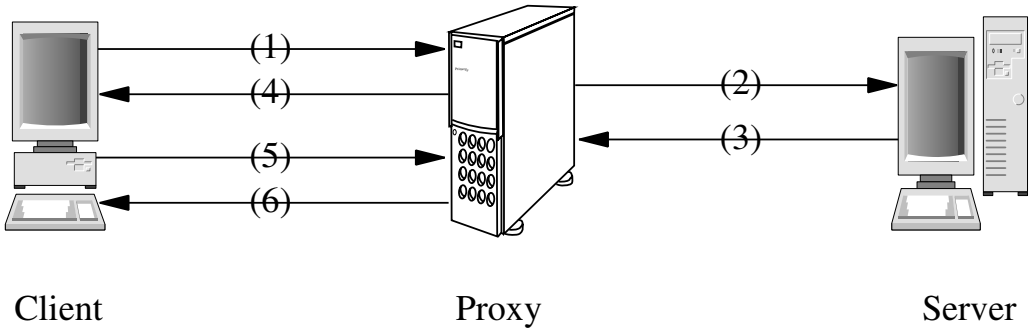

Fig. 1. A unified distribution control and privacy protection strategy

Upon this client request, the server also randomly breaks the key S_c for this object into two parts, S_{c1} and S_{c2}. S_{c2} is piggybacked to the client and the server sends S_{c1} to the proxy, where $S_c = S_{c1} + S_{c2}$.

- ③: Since (K_E, K_D) and (K_e, K_d) are based on Shamir-Omura and are commutative, the client received message $E(E(ID, K_e), K_E)||D(E(E(S_i, K_E), K_e), K_D)$ is the same as $E(E(ID, K_E), K_e)||E(S_i, K_e)$. Thus, the client further decrypts the received message with K_d, and gets $E(ID, K_E)$ and S_i, where $E(ID, K_E)$ is ID^S. Once obtaining the encrypted object ID, the client sends a request with the encrypted ID, ID^S, in the request URL to the proxy to request this object.
- ④: The proxy can successfully locate the object in its cache, and decrypts the object identified by ID^S with partial key S_{c1} from the server before sending to the client.
- ⑤: The client receives the partially decrypted object from the proxy, decrypts it with key S_{c2} and S_i.

VI. Security Analysis

Our unified strategy is based on El Gamal for access control and Shamir-Omura for privacy protection, and shares their security characteristics. The security of El Gamal has been proved and discussed in [30], [31], and the commutative property of Shamir-Omura has been proved in [29], [30]. In the following, we thus briefly analyze how the proposed scheme achieves our design goals and its potential vulnerability.

A. Design Goal Revisit

Our scheme achieves access control from the following three aspects. First, for each object the server only needs to perform encryption once and the encrypted object can be stored on the proxy forever. For each access, the server only needs to transfer two keys to the proxy and the client. In such a situation the server utilizes proxy caching very well for content distribution.

Second, the server does not lose control on the content it is distributing. Although a client can get the encrypted object freely from the proxy, a client has to get the key from the server before it can access the plaintext of the object. In our scheme, the server divides the decryption key into two partial keys and sends them to the proxy and the client. As long as the two parties holding the keys do not collude with each other, which is reasonable for the proxy-based distribution system, the server will not lose control on the object.

Third, client colluding attacks do not threat our scheme, since each client gets a different partial key S_{c2} even when multiple clients are requesting the same object. Since S_c is different for different objects and is given by the server only after the client access is authorized, client-proxy collusion cannot get more information about other objects in this class. To support this, for different client accesses, key S_c can be freely divided into different pairs of partial keys S_{c1} and S_{c2} for the proxy and the client. Thus, if they collude, they can get nothing more than the decryption keys of this object. The entire system is broken only when the proxy colludes with all clients.

Now let us look at how the client privacy is protected in our proposed scheme. First, without the server's authorization as above, the client cannot get the right object from the proxy. Second, since the only information that the server gets from a client is an encrypted object ID (ID^S), and the encryption key used is generated by the client for one access, the server only knows that the client is accessing one object, but it does not know which one exactly. On the other hand, on the proxy, all objects and their IDs are encrypted. Although the proxy knows which object (identified by the encrypted ID) the client is accessing, it does not know what the object is about. Thus, access control and privacy protection are achieved simultaneously.

Note that access control in our scheme is different from traditional access control in the sense that if a client just blindly requests objects from the proxy, it can get all objects without notifying the server. But without decryption keys from the server, the client still cannot access the plaintext of any object.

B. Further Discussion

Two other issues are worth more discussion. The first is that, in practice, some padding scheme should be used to avoid the flaw of Shamir-Omura cryptography technique under *chosen-cipher attacks* ([11], [29], [32], [33]). For a RSA-based scheme, an effective padding needs a random number

generator and a cryptographic hash function H ([33]). To encrypt a message M, a random number r is generated, and the ciphertext of M is $(r^{KA_e}, H(r) \bigoplus M)$. Thus, upon receiving the encrypted message (M_1, M_2), the recipient computes $H(M_1^{KA_d}) \bigoplus M_2$, where $\bigoplus$ is the bit-wise XOR operation. This padding avoids information leakage during the exponent operation on message M.

We further extend this padding scheme to make our proposed strategy robust against such attacks. Having object ID_i and its corresponding random number $r_i^{KA_e}$ available to a client, the client can encrypt with its key KB_e and send it to the server. The server thus decrypts with KA_d and sends back to the client. The client then gets r_i and $H(r_i)$. After XORing with ID_i, the client can correctly request the object identified by $H(r_i) \bigoplus ID_i$. An optimized step is to make $H(r_i)^{KA_e}$ available to the client.

The second is that, in our scheme, the proxy needs to perform decryption for each client request. Since media objects, such as movies, are generally very large, decryption can be computing intensive and it may overload the proxy. To mitigate the proxy workload, we can further optimize the strategy with the help of selective encryption. Selective encryption is known to be able to protect coded content by encrypting only parts of the content [34]. For example, if we selectively encrypt certain syntax bits in a compressed video, the content is still well protected with much less computing overhead.

VII. Cooperative Proxy Based System Design

In the previous sections, we have presented a unified mechanism that can provide seamless distribution control and client privacy protection at the same time. The scheme works when all encrypted objects have been $push$ed to the proxies before client accesses, which is feasible for CDNs. However, in practice, the proxy caches the object via a $pull$ approach where an object gets cached only after it is requested. Thus, our proposed scheme cannot be directly used to protect client privacy in such a $pull$ mode. Otherwise, the server can easily correlate an object (that is requested by a proxy) and a client (who is requesting the access key) based on the two request times if the client requested object is not cached, i.e., a cache miss.

To accommodate the $pull$ approach, upon a cache miss, in addition to the being requested object, some $(k-1)$ other objects must also be fetched from the server simultaneously to protect client privacy. This increases traffic overhead. To reduce and potentially utilize such traffic, in this section, we propose a set of new object caching policies as a complementary mechanism to our proposed scheme in a cooperative proxy environment where the system consists of multiple proxies, running Inter-Cache Protocol (ICP) [35] for inter-proxy communication. In the following design, we omit the cryptography part and focus on the policy design.

We assume that each proxy gets a list of available objects (e.g. movies) on the media server. This can be done actively or passively (e.g., upon the first access to a media server, the object list is piggybacked to the proxy). Each proxy also maintains three data items: the global object access frequency f, the locally cached object list *local_obj_list*, and *global_obj_list*, which contains all the available objects in all proxies. We further assume that proxies can request as many objects from the server as they want. This assumption is reasonable since all objects are encrypted.

The system works as follows. Upon a client request, if the client request is a hit, the proxy decrypts the object with the key received from the server before sending to the client. The client further decrypts the received data with the keys received from the server. The proxy updates the corresponding object access frequency f. If the client request results in a miss at the local proxy, but the object is cached in a peer proxy in the cooperative system based on the *global_obj_list*, the object is transferred from that proxy and forwarded to the client after the object is partially decrypted. The object access frequency, f, is updated.

When the client request is a miss, the following cost-amortized admission policy is activated to deal with the request, assisted by the aggressive object selection and pro-active replacement.

A. Cost-amortized Request Admission

If the object is not cached in local proxy or any peer proxy, the proxy collects the information regarding the number of requests of distinct objects in the same object class to the same media server at that instance. With guaranteed k anonymity, if there are r $(r < k)$ requests demanding different objects through p different proxies, each of these p proxies requests an additional $\lceil \frac{k-r}{p} \rceil$ objects.

In this admission policy, upon a miss, some additional traffic is incurred. This traffic is amortized since the server only sees k objects are fetched at that time, while they may be fetched by different proxies. An additional proxy can be used so that all p proxy requests go through the additional proxy when it is necessary to hide the proxy identities from the server.

B. Aggressive Object Selection

The above admission policy determines the number of objects a proxy should fetch to protect client privacy while amortizing proxy traffic through cooperation. After determining the number of objects to be fetched, the proxy also needs to determine which objects to fetch. Aggressive object selection thus works in two phases:

- In the first phase, the objects are selected according to the object popularity, reflected by f. The proxy selects the most popular objects that are not cached in any proxy. Then it relies on the pro-active replacement policy (see section VII-C) to deal with these fetched objects.
- In the second phase, where the non-cached objects are not as popular as the cached ones, the proxy selects objects according to object size. The smallest objects are selected until the privacy level is achieved.

Note in this selection policy, two different types of criteria are applied. In the first phase, the policy is based on object

TABLE II
EVALUATED STRATEGIES

Strategy Name	Privacy	Pro-active Replacement	Amortizing
base	No	No	No
strategy1	Yes	No	No
strategy2	Yes	Yes	No
strategy3	Yes	Yes	Yes

popularity. In the second phase, the policy is based on object size.

C. Pro-active Replacement

A replacement policy is normally required since cache space is always limited. When there is insufficient cache space to store an object, the replacement policy is responsible for reclaiming space by evicting some currently cached object(s). Thus, this type of replacement is passive in general.

To protect client privacy, another type of replacement is proposed, pro-active replacement. To protect client privacy, when a miss occurs, the proxy needs to fetch k objects simultaneously. This enforces the proxy to fetch $k-1$ objects other than the requested one to protect client privacy. Previously, the system has determined which objects to fetch using cost amortized admission and aggressive object selection. The following pro-active replacement policy determines the action upon arrival of requests at the proxy. Corresponding to the two phases defined in object selection:

- In the first phase where the fetched objects are more popular than the cached ones, a popularity-based replacement policy is implemented to replace the least popular object in the cache until the newly fetched popular objects get cached.
- In the second phase where the smallest objects are fetched, the fetched objects are simply discarded, since it is not worth caching them by replacing more popular objects.

VIII. PERFORMANCE EVALUATION

To protect client privacy upon a miss, a proxy has to fetch more objects than needed. Although with the amortized admission policy, the proxy still pays additional overhead for privacy protection. On the other hand, since the system always chooses more popular objects to cache in the proxy, it is possible that the additional traffic produces some caching benefit: some later requests hit these objects in the proxy.

To evaluate the overhead in the proposed cooperative proxy based system, we implemented a trace driven simulator. We extracted a server log from an enterprise media server. The extracted workload lasts for a duration of 2 months and covers various client activities. Because this is a single server-based workload and we want to evaluate the system performance when multiple proxies are present, we randomly duplicate some requests before distributing them to different proxies. The distribution also follows a random pattern.

In this workload, there is a total of 934 distinct media objects, with the size ranging from 288 KB to 638 MB. The total unique object size is 67 GB and the total number of requests is 642770. The total requested traffic is 139.5 TB. The averaged requested traffic per request is about 222 MB. We classify objects into different numbers of object classes. In the following results, a default value of 5 is used.

Four different strategies are evaluated for comparison, as shown in Table II. Strategy *base* is the reference scheme in which no privacy protection is considered. Strategies 1 to 3 all provide privacy protection with different ways to handle caching of objects. *Strategy1* works as follows: when a request is received and turns out to be a miss, the proxy works cooperatively to see if any peer proxy caches the requested object. If no peer proxy caches it, this proxy is responsible for getting the requested object from the server. At the same time, additional objects are fetched from the server to protect client privacy. If there is spare cache space in the proxy, these additionally fetched objects are selected based on object popularity and are cached in this proxy. Otherwise, these additional objects are selected based on object size (smallest object first) and are simply discarded when arriving at the proxy. Different from *Strategy1*, *Strategy2* replaces cached objects if the additionally fetched objects are more popular than the cached ones using pro-active replacement policy. We further design *Strategy3* to consider sharing the cost of fetching additional objects among peer proxies using cost-amortized admission policy.

The system is evaluated under various conditions: when the available cache size changes, when the desired privacy protection level varies, when the number of participating proxies increases, and when the number of object class increases. For each set of experiments, several metrics are evaluated. *Additional Traffic* reports the additional traffic caused due to privacy protection. *Local Hit Ratio* represents the number of requests served from the local proxies over the total number of requests, while *Peer Hit Ratio* represents the number of requests served from peer proxies divided by the total number of requests. Requests that cannot be directly served by any proxy are counted into *Miss Ratio* after averaging on the total number of requests. Corresponding to these three number based metrics for cache performance, we also have their corresponding byte based values evaluated and reported, as *Local Byte Hit Ratio*, *Peer Byte Hit Ratio*, and *Miss Byte Ratio*.

A. Cache Size

First, the system is evaluated when the available cache size varies from 5% to 100% of the total unique object size, with 5% as the interval. The cache size is evenly divided among all participating proxies. In this set of experiments, the privacy

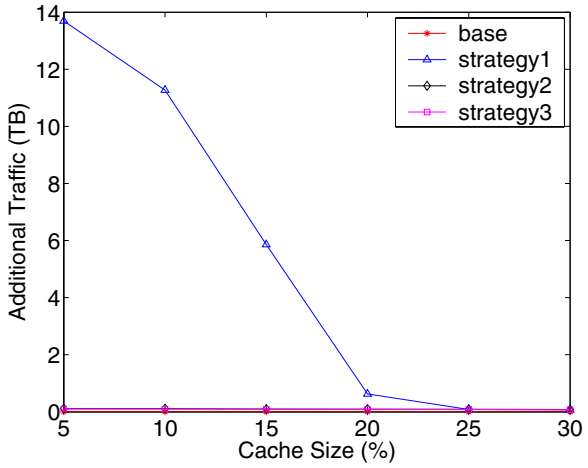

Fig. 2. Additional Traffic

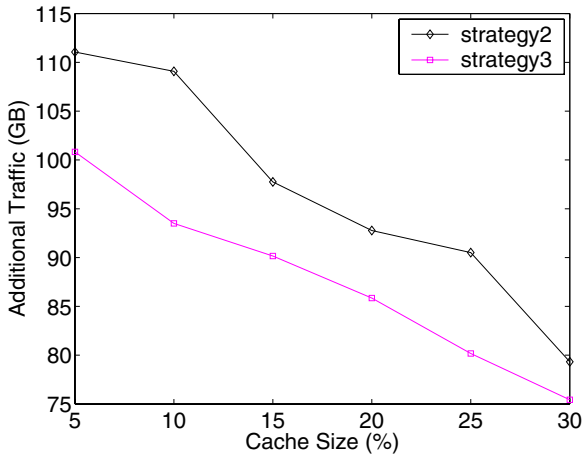

Fig. 3. Additional Traffic Enlargement

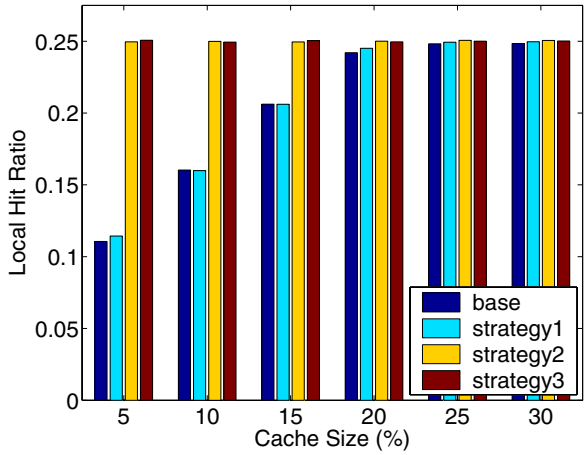

Fig. 4. Local Hit Ratio

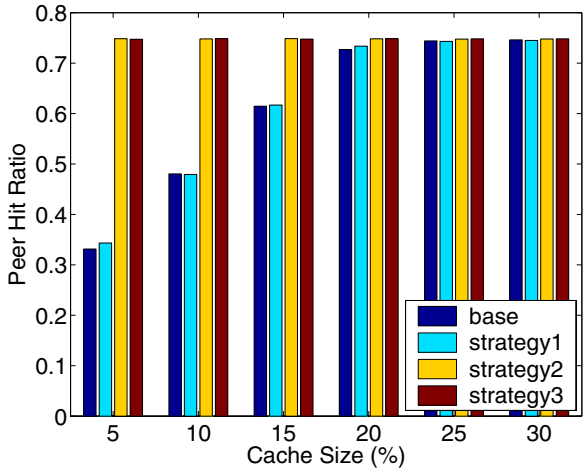

Fig. 5. Peer Hit Ratio

level is at a medium value of 4, and the number of proxies in the cooperative system is also 4. Since the performance difference among the four strategies becomes trivial after the cache size is beyond 30% of the total object size, we only report the performance under 30% in the following figures.

Figure 2 shows the *Additional Traffic* caused by the four strategies. *Strategy1* results in the largest additional traffic when the available cache size is very small, but the traffic quickly drops when the cache size increases to and beyond 20% of the total object size. This is largely due to the fact that when the cache size is very small (e.g. 5%), some requested large objects are frequently fetched, leaving little space to cache additional objects. Thus, more client requests cannot be served from the proxy and have to be fetched from the server. Note that although the absolute values of the additional traffic caused by *Strategy1* are large (about 14 TB) when the cache size is small (e.g., 5%), it is about 10% of the total amount of requested traffic.

Comparing the result of *Strategy3* with those of *Strategy1* and *base* shown in Figure 3 (the enlargement of Figure 2), we find that *Strategy3* achieves a much better result due to traffic amortizing. The additional traffic incurred in *Strategy3* is only about 1% of the total amount of requested traffic when the cache size is small (e.g., 5%). Compared with *Strategy2*, *Strategy3* achieves a better result but the gap is not very significant. We find this is because, in the workload, there is only a small percentage of requests arriving at the same time, resulting in only a few amortizing opportunities. In the future, we plan to cluster the requests arriving in a time window to re-evaluate these strategies.

Figure 4 and Figure 5 show the cache performance in terms of *Local Hit Ratio* and *Peer Hit Ratio*. It is interesting that when the cache size is small, such as 5% and 10%, *Strategy2* and *Strategy3* achieve a much higher hit ratio than *Strategy1* and *base*. When the cache size increases beyond 20%, *base* and *Strategy1* achieve similar performance to others in *Local Hit Ratio* and *Peer Hit Ratio*. Since *Strategy2* and *Strategy3* demand more operation overhead, a smart strategy selection policy should be considered according to the available cache size when designing a real system.

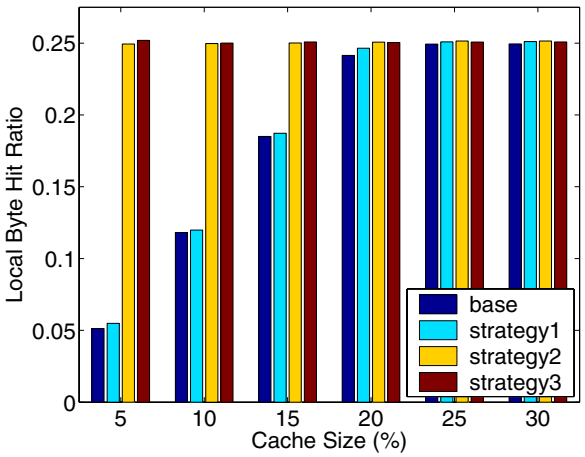

Fig. 6. Local Byte Hit Ratio

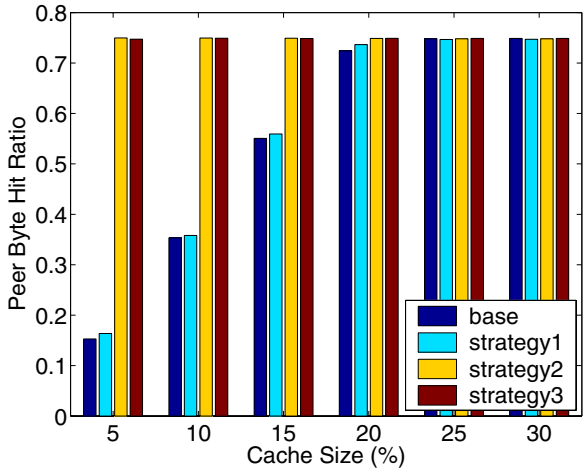

Fig. 7. Peer Byte Hit Ratio

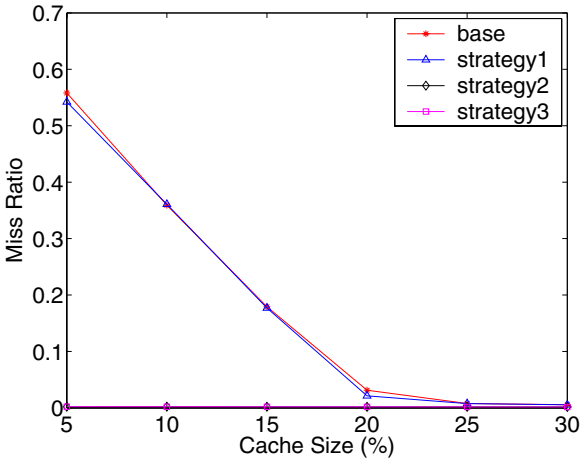

Fig. 8. Total Miss Ratio

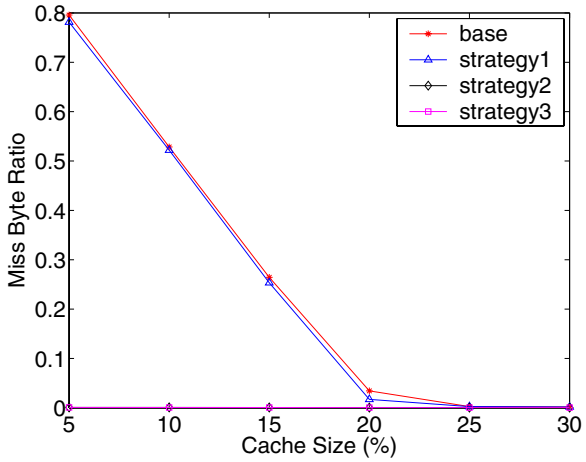

Fig. 9. Total Miss Byte Ratio

Figure 6 and Figure 7 show *Local Byte Hit Ratio* and *Peer Byte Hit Ratio* for these four strategies. Compared with their corresponding numbers based evaluation results in Figure 4 and Figure 5, the byte based performance trends are similar. However, it is interesting that they do not achieve the same percentage levels, particularly when the cache size is small, such as 5% and 10% of the total object size. For example, when the cache size is 5% of the total object size, *base* and *strategy1* achieve a *Local Hit Ratio* of about 11%, while their corresponding *Local Byte Hit Ratio* is about 5%. It is also worth noting that these evaluation results confirm that increasing the total available cache size does not always improve cache performance. When the cache size reaches beyond 20% of the total object size, the improvement slows down and eventually becomes trivial. Our previous evaluation on one proxy-based system reports similar results where a cache size of 8% to 16% can achieve near-best results for general workloads [36].

Figure 8 and Figure 9 show the *Miss Ratio* and *Miss Byte Ratio*. They reflect the traffic between the proxy and server directly caused by client requests. Both figures indicate that *Strategy2* and *Strategy3* achieve better performance results (between 0.002-0.008) due to the pro-active replacement policy, especially when the cache size is small. When the cache size is sufficiently large ($> 25\%$ of the total object size), the replacement policy can hardly bring any benefit, as implied by the comparisons with *Strategy1* and *base*.

With increased privacy level, the additional traffic caused by all four cooperative proxy based strategies is increasing. However, the overall cache performance is hardly impacted. When the number of participating proxies and the number of object classes increase, none of these performance metrics changes significantly. We omit these results for brevity.

In summary, the evaluation indicates that our proposed system is very efficient. When the privacy level is 4, the additional traffic overhead incurred in our system (*strategy3*) is less than 1% of the total amount of requested traffic when the cache size is relatively small (e.g., 5%). When the cache size is relatively large (about 20%), the additional traffic caused by our system is lower than 0.1% of the total amount

of requested traffic. In addition, our system can effectively utilize the additional traffic to improve the cache performance (compared with the *base* strategy) in terms of hit ratio and byte hit ratio.

IX. Conclusion

Plenty of research has been conducted to improve the performance of proxy-based content delivery systems for Internet media distribution. However, the adoption of such research is hindered by the difficulty in providing distribution control and privacy protection at the same time. In this work, through the analysis of distribution control and privacy protection requirements, we propose a unified strategy to address these two issues simultaneously by using a key division cipher and a commutative cipher. We further design a set of algorithms for cooperative proxies where our proposed protocol works for a pull approach. Simulation-based experiments have been extensively conducted and the results show that our proposed system is effective.

We are currently implementing our proposed scheme on a proxy based media distribution prototype. We plan to test the cache performance and the computational overhead for encryption and decryption used in our scheme in this real environment.

X. Acknowledgment

We would like to thank William L. Bynum, Xiaodong Zhang, and anonymous reviewers for their helpful comments on this paper. The work is supported by NSF grant CNS-0509061 and a grant from Hewlett-Packard Laboratories.

References

[1] Y. Chae, K. Guo, M. Buddhikot, S. Suri, and E. Zegura, "Silo, rainbow, and caching token: Schemes for scalable fault tolerant stream caching," in *IEEE Journal on Selected Areas in Communications*, September 2002.

[2] M. Y. Chiu and K. H. Yeung, "Partial video sequence caching scheme for vod systems with heteroeneous clients," in *Proceedings of the 13th International Conference on Data Engineering*, Birmingham, United Kingdom, April 1997.

[3] J. Kangasharju, F. Hartanto, M. Reisslein, and K. W. Ross, "Distributing layered encoded video through caches," in *Proceedings of IEEE Inforcom*, Anchorage, AK, April 2001.

[4] Z. Miao and A. Ortega, "Scalable proxy caching of video under storage constraints," in *IEEE Journal on Selected Areas in Communications*, September 2002.

[5] R. Rejaie, M. Handely H. Yu, and D. Estrin, "Multimedia proxy caching mechanism for quality adaptive streaming applications in the internet," in *Proceedings of IEEE INFOCOM*, Tel-Aviv, Israel, March 2000.

[6] P. Schojer, L. Boszormenyi, H. Hellwagner, B. Penz, and S. Podlipnig, "Architecture of a quality based intelligient proxy (qbix) for mpeg-4 videos," in *Proceedings of WWW*, Budapest, Hungary, May 2003.

[7] S. Sen, J. Rexford, and D. Towsley, "Proxy prefix caching for multimedia streams," in *Proceedings of IEEE INFOCOM*, New York City, NY, March 1999.

[8] K. Wu, P. S. Yu, and J. Wolf, "Segment-based proxy caching of multimedia streams," in *Proceedings of WWW*, Hongkong, China, May 2001.

[9] Z.L. Zhang, Y. Wang, D.H.C. Du, and D. Su, "Video staging: A proxy-server based approach to end-to-end video delivery over wide-area networks," in *IEEE Transactions on Networking*, Aug. 2000, vol. 8, pp. 429–442.

[10] "Windows media digital rights management (drm)," http://www.microsoft.com/windows/windowsmedia /drm/default.aspx.

[11] F. Bao, R. Deng, P. Feng, Y. Guo, and H. Wu, "Secure and private distribution of online video and several related cryptographic issues," in *Proceedings of the 6th Australia Conference on Information Security and Privacy*, Sedney, Australia, 2001.

[12] Y. Wu and F. Bao, "Collusion attack on a multi-key secure video proxy scheme," in *Proceedings of ACM Multimedia*, New York City, NY, October 2004.

[13] S. Yeung, J. Lui, and D. Yau, "A case for multi-key secure video proxy: Theory, design, and implementation," in *Proceedings of ACM Multimedia*, Juan-les-Pins, France, December 2002.

[14] "Federal trade commission - information privacy and security," http://www.ftc.gov/privacy/.

[15] F. Bao, R. Deng, and P. Feng, "An efficient and practical scheme for privacy protection in the e-commerce of digital goods," in *Proceedings of the 3rd International Conference on Information Security and Cryptology*, 2000.

[16] "Mediacom," http://www.mediacomcc.com/.

[17] C. Griwodz, O. Merkel, J. Dittmann, and R. Steinmetz, "Protecting vod the easier way," in *Proceedings of the 6th ACM Multimedia*, Bristol, England, September 1998.

[18] C. Shi and B. Bhargava, "A fast mpeg video encryption algorithm," in *Proceedings of the 6th ACM Multimedia*, Bristol, England, September 1998.

[19] A. S. Tosun and W. C. Feng, "Secure video transmission using proxies," Tech. Rep., Computer and Information Science, Ohio State University, 2002.

[20] B. Chor and N. Gilboa, "Computational private information retrieval," in *Proceedings of the 29th STOC*, 1997.

[21] Y. Gertner, Y. Ishai, E. Kushilevita, and T. Malkin, "Protecting data privacy in private information retrieval schemes," in *Proceedings of the 30th STOC*, 1998.

[22] "The freenet project," http://freenet.sourceforge.net/.

[23] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router," in *Proceedings of the 13th USENIX Security Symposium*, August 2004.

[24] M. J. Freedman and R. Morris, "Tarzan: A peer-to-peer anonymizing network layer," in *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS 2002)*, Washington, DC, November 2002.

[25] M. K. Reiter and A. D. Rubin, "Anonymous web transactions with crowds," in *Communications of the ACM*, 1999, vol. 42 (2), pp. 32–48.

[26] D. Boneh, X. Ding, G. Tsudik, and C.M. Wong, "A method for fast revocation of public key certificates and security capabilities," in *Proceedings of USENIX Security*, Washington, D.C., August 2001.

[27] H. Yang and S. Lu, "Commutative cipher based en-route filtering in wireless sensor networks," in *Proceedings of IEEE VTC Wireless Security Symposium*, Los Angeles, CA, September 2004.

[28] T.E. Gamal, "A public key cryptosystem and signature scheme based on the discrete logarithm," *IEEE Transactions of Information Theory*, , no. 4, 1985.

[29] W. Diffie and M. Hellman, "New directions in cryptography," in *IEEE Transactions on Information Theory*, November 1976, vol. 22 (6), pp. 644–654.

[30] W. Stallings, "Cryptography and network security: Principles and practice," in *Prentice Hall*, 1998.

[31] Y. Tsiounis and M. Yung, "On the security of elgamal based encryption," in *Proceedings of International Workshop on Practice and Theory in Public Key Cryptography*, Yokohama, Japan, February 1998.

[32] J. Coron, D. Naccache, and J. Stern, "On the security of rsa padding," in *Proceedings of Crypto*, Santa Barbara, August 1999.

[33] RSA Laboratories, "Pkcs 1 v2.1: Rsa cryptography standard," June 2002.

[34] J. Wen, M. Severa, W. Zeng, M. Luttrell, and W. Jin, "A format compliant configurable encryption framework for access control of video," in *IEEE Transactions on Circuits and Systems for Video Technology, Special Issue on Wireless Video*, June 2002, pp. 545–557.

[35] "Inter cache protocol," http://icp.ircache.net/.

[36] S. Roy, B. Shen, S. Chen, and X. Zhang, "An empirical study of a segment-based streaming proxy in an enterprise environment," in *Proceedings of the 9th International Workshop on Web Content Caching and Distribution*, Beijing, China, October 2004.