Achieving Simultaneous Distribution Control and Privacy Protection for Internet Media Delivery

SONGQING CHEN
George Mason University
SHIPING CHEN
Sybase Inc.
HUIPING GUO
California State University
BO SHEN
HP Laboratories
and
SUSHIL JAJODIA
George Mason University

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 and performance 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 copyright protection, content providers are hesitant to use proxy-based fast distribution techniques. Second, little has been done to protect client privacy during content accesses on the Internet. 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 article, we propose a unified proxy-based media distribution protocol to effectively address these two problems simultaneously. We further design a set of new algorithms in a cooperative proxy environment where our proposed scheme works efficiently and practically. Simulation-based experiments are conducted to extensively evaluate the proposed system. Preliminary results demonstrate the effectiveness of our proposed strategy.

Categories and Subject Descriptors: H.3.4 [**Information Storage and Retrieval**]: Systems and Software—*Distributed systems*; D.4.6 [**Oprating Systems**]: Security and Protection—*Access controls*

An earlier version of this article was published in Proceedings of the 14th IEEE Workshop on Quality of Service (IWQoS '06) [Chen et al. 2006].
This work is supported by NSF grants CNS-0509061 and CNS-0621631.
Authors' addresses: Songqing Chen, Department of Computer Science, George Mason University, Fairfax, VA 22030; email: sqchen@cs.gmu.edu; Shiping Chen, Department of ITSG/Security, Sybase Inc., Dublin, CA 94568; email: shiping.chen@sybase.com; H. Guo, Department of Computer Science, California State University, Los Angeles, CA 90032; email: hpguo@calstatela.edu; B. Shen, Media and Mobile Systems Lab, HP Laboratories, Palo Alto, CA 94304; email: bo@blueapple.mobi; S. Jajodia, Center for Security Info. System, George Mason University, Fairfax, VA 22030; email: jajodia@gmu.edu.

General Terms: Algorithms, Design, Security

Additional Key Words and Phrases: Distribution control, privacy, proxy caching, media delivery, cooperative proxy, distribution control

ACM Reference Format:

Chen, S., Chen, S., Guo, H., Shen, B., and Jajodia, S. 2008. Achieving simultaneous distribution control and privacy protection for internet media delivery. ACM Trans. Multimedia Comput. Commun. Appl. 4, 2, Article 9 (May 2008), 23 pages. DOI = 10.1145/1352012.1352013 http://doi.acm.org/ 10.1145/1352012.1352013

1. INTRODUCTION

Recent years have witnessed the dramatic and continuous increase of Internet media applications. Different from the Web object delivery, delivering Internet media objects demands prolonged and 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 [Chae et al. 2002; Chiu and Yeung 1997; Kangasharju et al. 2001; Miao and Ortega 2002; Rejaie et al. 2000; Schojer et al. 2003; Sen et al. 1999; Wu et al. 2001; Zhang et al. 2000].

However, 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 that originate from both the server and the client sides have not been well addressed.

The first problem is media distribution control concerned at the server side. 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. The reason is as follows. 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 online (http://www.microsoft.com/windows/windowsmedia/drm/default.aspx). Existing practice often uses pay-per-view, membership, or DRM schemes without considering effective utilization of proxy caching. Some recently developed strategies tried to leverage proxy caching to deal with the problem with the aid of encryption mechanisms [Bao et al. 2001; Wu and Bao 2004; Yeung et al. 2002, 2005]. But they either demand special hardware support or fail to work if some clients collude.

The second problem is that for clients, existing strategies have given little consideration for client privacy protection during content accesses over the Internet. Through online 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[1] [LAW]. 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 some initial efforts [Bao et al.

[1]Federal Trade Commission—Information Privacy and Security, http://www.ftc.gov/privacy/.

2000] considered privacy protection in the E-commerce context, the efficiency is poor and little work has been conducted to protect client privacy in a proxy-based media distribution architecture.

Although efforts as aforementioned have been made to address these two problems separately, a desirable proxy based media delivery system should be capable of simultaneously protecting the interests of both the content provider and the clients, that is, 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 revelation of client identity, thus leading to an invasion of client privacy since the server may know which object the client is accessing. Straightforward solutions that address these two problems independently conflict with each other. To the best of our knowledge, there is no efficient and practical proxy-based solution with simultaneous media distribution control and privacy protection. In addition, a practical strategy for a proxy-based media delivery system must also consider the performance cost when providing such functionalities, although it is a common belief that there is a trade-off between the performance and the security offered by a system.

In this paper, we propose a unified strategy that can provide distribution control and privacy protection simultaneously in proxy-based Internet media delivery systems while media content can still be delivered efficiently. 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 efficiently and practically. Extensive simulation-based experiments have been conducted to evaluate the proposed system. The preliminary results show that our proposed system is very effective.

In this study we assume that the proxy is independent from the content provider and the client, and it will not collude with either of them. Otherwise, the scheme cannot be applied. In our study, we also distinguish the client software from the clients themselves. We assume the client software (player) is trusted. It will implement the algorithms and scheme correctly, and will not disclose the keys it possesses in the process of decrypting objects. However, the clients (end users) are not trusted. They may collude with each other or distribute the content they receive via out-of-band channels. This is similar to a TV program, where a user can tape-record the program and give it to a friend who is not a subscriber. How to prevent out-of-band violations is out of scope for this paper. However, with the help of the trusted players, our scheme can effectively mitigate this kind of threats.

The remainder of this article is organized as follows. Some related work is discussed in Section 2. We dissect the two components of our unified strategy and present distribution control and privacy protection in Section 3 and Section 4, respectively. We describe the unified strategy in Section 5 and design the algorithm for cooperative proxies in Section 6. The evaluation results based on simulations are presented in Section 7. We make concluding remarks in Section 8.

2. RELATED WORK

The development of Internet techniques has provided an alternative vehicle for delivering all kinds of content, such as Internet media content. At the same time, it also brings new challenges. For Internet media content delivery, on the one hand, it requires effective distribution control schemes to protect the copyright of the delivered objects. On the other hand, for clients accessing the content on the Internet, the demand for client privacy protection is also increasing.

For the server or the content provider, the 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 (http://mediacomcc.com/). 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 [Griwodz et al. 1998; Shi and Bhargava 1998;

Tosun and Feng 2002] considers how to speed up the decryption and encryption process. The scheme proposed in Bao et al. [2001] relies on costly hardware support to deal with member collusion. Recent work [Yeung et al. 2002, 2005] proposes a multi-key video proxy infrastructure without such a requirement. However, the scheme is also vulnerable to member collusion, as pointed out in Wu and Bao [2004]. In addition, this RSA based multikey scheme requires producing a large number of asymmetric key pairs and the encryption load is simply too high to be realistic, even with the assistance of selective encryption. This scalability issue prevents it from dealing with a large number of client accesses simultaneously.

On the other hand, an increasing number of clients want privacy protection in their online transactions. Privacy protection has been studied in many other areas (e.g., Chor and Gilboa [1997] and Gertner et al. [1998] in databases) and strategies include selective attribute disclosure [Bonatti and Samarati 2002; Park and Sandhu 1999; Parsiano and Visconti 2000] (for example, no exact values are released in Camenisch and Herreweghen [2002] and Brands [2000]), using pseudonym credential systems [Camenisch and Herreweghen 2002; Brands 2000; Chaum 1985] to provide anonymity, or trust negotiation [Holt et al. 2003; Winsborough and Li 2002a, 2002b]. In general, these solutions are either based on encryption or anonymizing or both. For example, Bao et al. [2000] provide privacy protection through encryption. However, this work relies on an independent third party and introduces much more overhead traffic.

Anonymizing is another alternative to enable privacy protection. Anonymity can be provided based on either access control or mixing. The examples of the former case include FreeNet (http://freenet.sourceforge.net), Crowds [Reiter and Rubin 1999]. Onion routing [Dingledine et al. 2004] and Tarzan [Freedman and Morris 2002] fall into the latter category. Recently, parallel mixing [Golle and Juels 2004] allows servers to mix inputs in parallel to reduce the overall mixing time for moderate-to-large numbers of servers. With n inputs and M servers, the original sequential approach takes $M \times n$ time while parallel mixing takes at most $2n$. Fragile mixing [Reiter and Wang 2004] discourages an administrator from revealing some correspondence between its input and output messages. Through fragile mixing, revealing an input-message to output message correspondence discloses all such correspondences for all messages. However, to the best of our knowledge, none of these existing strategies work in the context of online media distribution systems, since the distribution control on the server will be difficult, if not possible at all.

Other than the proxy-based environments, key division strategies [Boneh et al. 2001] and commutative ciphers [Yang and Lu 2004] have also been studied extensively for various purposes. Compared to existing studies, our work focuses on the proxy-based media delivery architecture by proposing a framework that achieves distribution control and privacy protection at the same time.

3. 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 distributes to others, the server will lose its control on the object distribution. Periodically changing the object encryption keys can partially mitigate the threat, but the overhead caused by frequent rekey operations is an overkill.

In this section, we propose a novel scheme via which the server can benefit from proxy caching without losing control over object distribution. The scheme is based on a key division cipher. Before presenting the scheme we propose, we introduce the key division cipher first.

3.1 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 [Gamal 1985] is a reliable public cipher system based on discrete logarithms. We first briefly introduce El Gamal, and then show 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 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 by 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) $K = (Y_B)^k (mod\ q) = (C_1)^{X_B} (mod\ q) = (C_1)^{X_{B_1}+X_{B_2}} (mod\ q) = K_1 K_2$; and 2) $M = \frac{C_2}{K} (mod\ q) = \frac{C_2}{K_1}/K_2 (mod\ q) = \frac{M_1}{K_2} (mod\ q) = M_2$. □

3.2 Our Proposed Scheme

Now we present our scheme by extending El Gamal. The basic idea is as follows. To distribute an object M, the server generates a key pair, using the 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 $C_1 = \alpha^k \ (mod\ q)$. Furthermore, 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 $C_2 = K_i M_i$. 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.

3.3 Discussion

Our proposed scheme achieves the design goal 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 well utilizes the power of the proxy caching for content distribution.

Second, the server does not lose any 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 even when multiple clients are requesting the same object. In our scheme, the key X_i can be freely divided into infinite pairs of partial key X_{i1} and X_{i2}, which is scalable.

4. PRIVACY PROTECTION IN PROXY-BASED DISTRIBUTION SYSTEM

As mentioned, e-commerce today makes it easier for the service provider to get and compile customers' 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 thus desire privacy protection in online 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 access control on its objects, since clients can directly get objects from the proxy without letting the server know. 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 for privacy protection 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. Before presenting our proposed scheme, we introduce the commutative cipher first.

4.1 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 [Diffie and Hellman 1976; Stallings 1998]. 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.

4.2 Our Proposed Scheme

With a 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 (e.g., the hash result of the object names through SHA-1) 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 given 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.

(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.

4.3 Discussion

Now let us briefly look at how the client's privacy is protected in our proposed scheme. Since the only information that the server gets from a client is an encrypted object ID, and the encryption key 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.

At the same time, the server still keeps the object distribution under control. Without its authorization, the client cannot get the right object from the proxy. Notice that this access control is a little bit weak in the sense that if the client just blindly requests objects from the proxy, it can get all the objects without notifying the server. However, without decryption keys from the server, the client still cannot access the plaintext of any object. But if the objects are encrypted with the same key, a client who gets a key can disclose all objects. A comprehensive scheme which integrates both schemes we proposed so far will be introduced in Section 5 to address this.

Both Shamir-Omura (http://www.afn.org/~afn21533/keyexchg.) and [Bao et al. 2000] have discussed the Shamir-Omura based commutative cipher. We use one alike. Let $p = 2q + 1$, where both p and q are two prime numbers. The size of p should be large enough. Select a key $K \in Z_{2q}$ that is an odd number (not q), to encrypt a message M, $C = M^S mod\ p$.

Due to the well-known multiplicative property of RSA, in practice, some padding scheme should be used to avoid the flaw of Shamir-Omura cryptography technique under *chosen-cipher attacks* [Bao et al. 2001; Coron et al. 1999; Diffie and Hellman 1976; RSA Laboratories 2002]. For a RSA-based scheme, an effective padding needs a random number generator G and a cryptographic hash function H (e.g., SHAs or Whirlpool). RSA group proposed Optimal Asymmetric Encryption Padding (OAEP) [RSA

Laboratories 2002]. A padding function such as $P(m) = m \bigoplus G(R) \| RD \bigoplus H(m \bigoplus G(R))$ could be used. The padded message is called M. To encrypt 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 bitwise XOR operation. This padding avoids information leakage during the exponent operation on message M.

For our strategy, we simply extend the above strategy to be robust to such attacks. Having the object ID_i and its corresponding random number $r_i^{KA_e}$ available to the client, the client can encrypt with its key KB_e and send 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.

5. 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 different categories with different prices. The distribution control also needs to enforce different charges on different types of objects. In this section, we present our proxy-based protocol that is capable of comprehensive distribution control and client privacy protection simultaneously with such practical conditions.

5.1 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 do 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 criterion as an example here). For each price class, there exist multiple ($\geq k$) objects;

—For simplicity, each object is identified by an *ID* on the server, which could be the hash (e.g., through SHA-1) result of the name of the object, or the URL of the object. The ID of an object must be unique so that not only each object is uniquely identifiable by its encrypted *ID* on the proxy, but also they are not linkable with each other;

—In an object class, the same key pair is used to encrypt/decrypt all the object *ID*s, 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 a single 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 the following three tasks:

(1) The server performs encryption as follows:

$$M^E = E(M, S_c + S_i), \tag{1}$$

and M^E is cached in the proxy, where $E()$ is the El Gamal–based encryption algorithm.

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

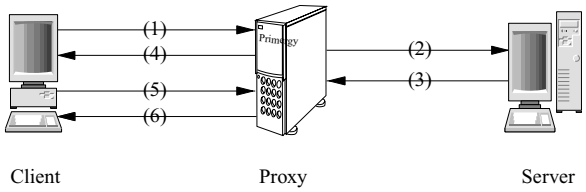

Fig. 1. A unified distribution control and privacy protection strategy.

(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 online tuple of object ID and encrypted S_i – $(ID, E(S_i, K_E))$.

A client access following the protocol is 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,

$$E(ID, K_e) \| E(E(S_i, K_E), K_e), \tag{2}$$

and sends the encrypted message to the proxy.

—②: The proxy forwards the request to the server. After the access is authorized, the server performs encryption on the encrypted ID and decryption on the $E(E(S_i, K_E), K_e)$ from the client. Then the server sends the results back to the client,

$$E(E(ID, K_e), K_E) \| D(E(E(S_i, K_E), K_e), K_D). \tag{3}$$

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 obtained 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.

5.2 Discussion

In our proposed strategy, since for the same object different clients get different S_{c1} for decryption, this effectively prevents client collusion that is easy if a naive approach is taken. Only when clients can drive the key held by the proxy from their partial keys, the security is broken. However, this is computationally impossible as our key division cipher is El Gamal.

In our scheme, the proxy needs to perform decryption for each client request. Since media objects, such as movies, are generally very large in size, encryption and decryption can be computing intensive. To improve its efficiency, we further optimize the strategy with the help of selective encryption. Selective encryption is able to protect coded content by only encrypting parts of the coded content [Wen et al. 2002]. For example, if we selectively encrypt certain syntax bits in compressed video, the content is still well protected with much less computing overhead.

To improve encryption and decryption efficiency, our selective encryption based approach works as follows. The server encryption key (containing S_c and S_i) now actually contains two sets. One set is used for El Gamal, and the other is for a symmetric crypto system, such as DES. For multimedia content, for example, a compressed video, we choose to encrypt its syntax bits using the El Gamal–based scheme. For the rest of the bits, the DES based scheme is used. Since the computing intensive El Gamal encryption is applied to smaller amounts of data and a much faster DES scheme is applied to the majority of the content bits, the encryption produces well protected content with less computing overhead. Note that this scheme requires that the content coding format is known to the server, the proxy, and the client, which is not a problem for standard based content such as MPEG video and audio.

6. COOPERATIVE PROXY BASED SYSTEM DESIGN

In the previous section, 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 proxies before client accesses, which is feasible for CDNs. However, in practice, a 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 time if the client requested object is not cached, that is, a cache miss.

To accommodate the *pull* approach, upon a cache miss, in addition to the requested object, some other objects ($k - 1$) 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: http: //icp.ircache.net/) 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 proactive replacement policies.

6.1 Cost-Amortized Request Admission

If the object is not cached in the local proxy or any peer proxy, the proxy collects the information regarding the number of requests for distinct objects in the same object class to the same media server at that instance. With guaranteed k anonymity, if there are r 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.

6.2 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 proactive replacement policy (see Section 6.3) 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 the object size. The smallest objects are selected until the privacy level is reached.

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

6.3 Proactive 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, proactive replacement, is proposed. To protect client privacy, when a miss occurs, the proxy needs to fetch k objects simultaneously. This forces 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 proactive 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.

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

7. 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 the fetched objects in the proxy.

To evaluate the overhead in the proposed cooperative proxy based system, we implemented a trace driven simulator, which runs on a machine with a 2.4 GHz CPU and 1 GB physical memory. 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 requests are also randomly directed through the available proxies.

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 average 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 of handling object caching. *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 the cached objects if the additionally fetched objects are more popular than the cached ones using proactive replacement policy. We further design *Strategy3* to consider sharing the cost of fetching additional objects among peer proxies using the 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 the 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*, respectively.

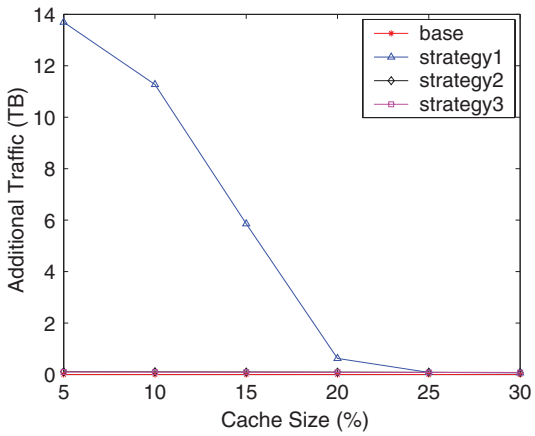

Fig. 2. Additional Traffic.

7.1 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 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 because 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 additional traffic amounts 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.

Compared the result of *Strategy3* with those of *Strategy1* and *base* shown in Figure 3 (the enlargement of Figure 2), *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. 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 the 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 available cache size when designing a real system.

Figure 6 and Figure 7 show *Local Byte Hit Ratio* and *Peer Byte Hit Ratio* for these four strategies. Compared with their corresponding results in Figure 4 and Figure 5, the performance trends of different

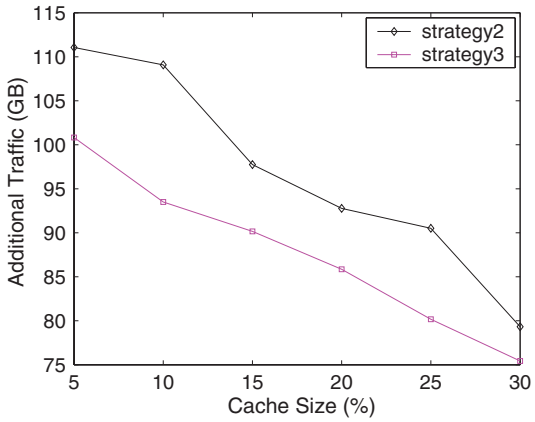

Fig. 3. Additional Traffic Enlargement.

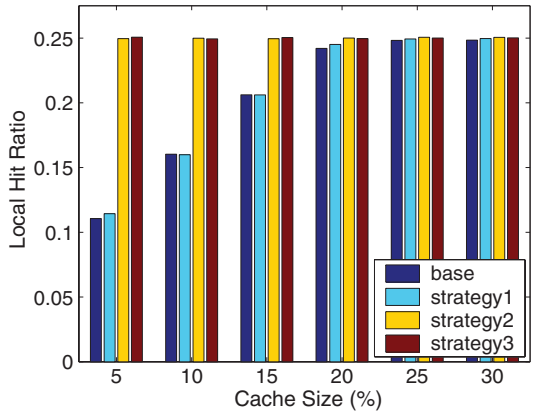

Fig. 4. Local Hit Ratio.

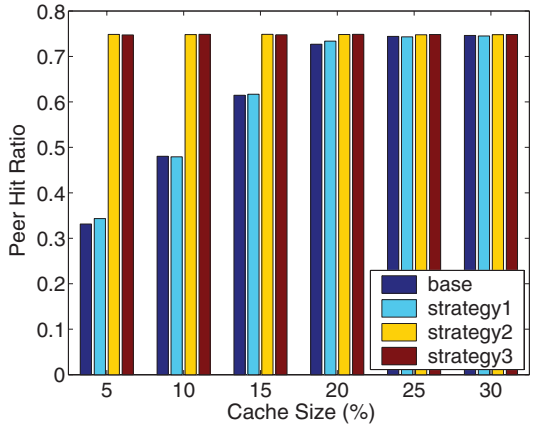

Fig. 5. Peer Hit Ratio.

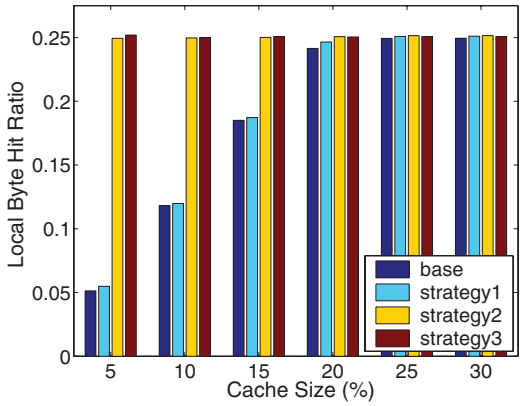

Fig. 6. Local Byte Hit Ratio.

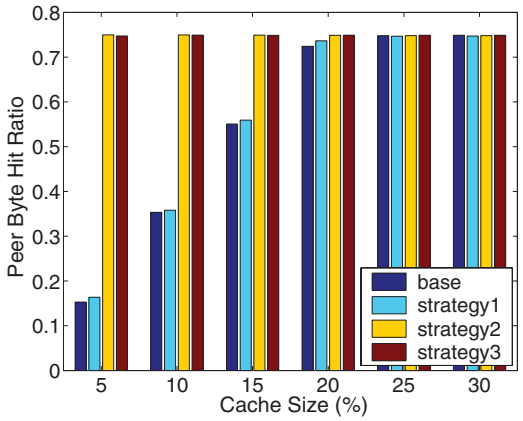

Fig. 7. Peer Byte Hit Ratio.

strategies 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*s are of 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 [Roy et al. 2004].

Figure 8 and Figure 9 show the *Miss Ratio* and *Miss Byte Ratio*. They reflect the traffic between the proxy and the 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*.

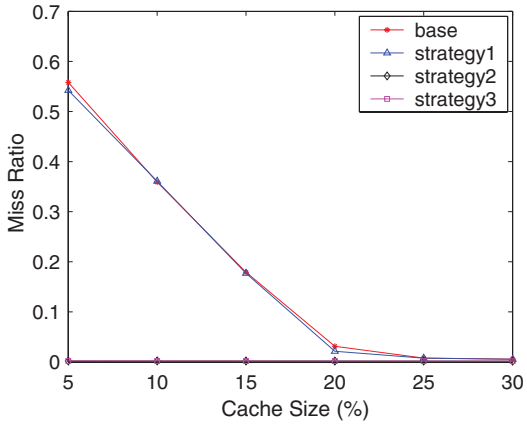

Fig. 8. Total Miss Ratio.

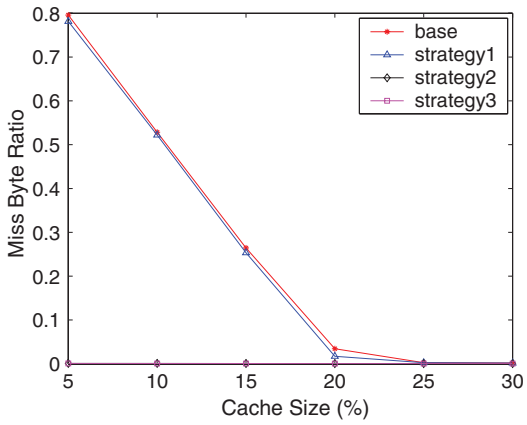

Fig. 9. Total Miss Byte Ratio.

7.2 Proxy Number

In this section, we evaluate the system performance when the number of participating proxies varies. In this set of experiments, the total cache size is 20% of the total object size, and the privacy level is 4. We increase the proxy number from 2 to 10. For a given participating proxy number, the cache size is equally divided among them. We have the results evaluated using the same set of metrics as before.

Figure 10 shows the *Additional Traffic* caused by the four strategies. Not surprisingly, *Strategy1* results in the largest amount of additional traffic in all cases, while *Base* does not have additional traffic. An interesting trend is that the additional traffic reaches a local lowest point when the number of proxies is 4 for all three strategies. The trend is very clear for *Strategy1*. For example, when the proxy number is 4, *Strategy1* causes only about 115 GB additional traffic, while the additional traffic amounts are about 140 GB and 207 GB when proxy numbers are 2 and 10, respectively. The reason is as follows. When the number of proxies is small, the privacy level dominates the results. When the number of proxies is large, the replacement on each proxy dominates the system performance. Thus, an appropriate selection of proxy number is important to the amount of additional traffic.

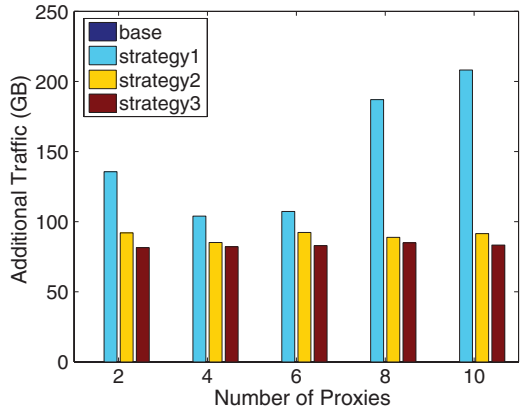

Fig. 10. Additional Traffic.

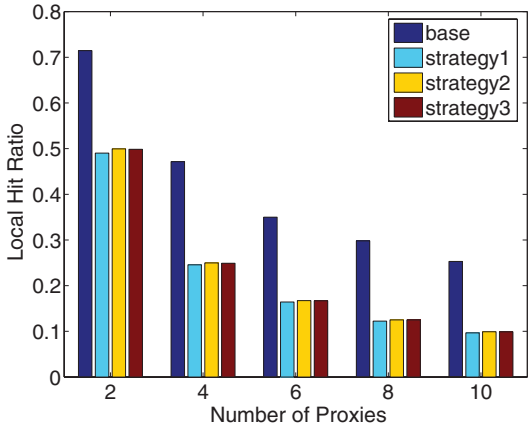

Fig. 11. Local Hit Ratio.

Compared with *Strategy1* and *Strategy2*, *Strategy3* achieves the best result due to traffic amortizing. When the number of proxies is 10, the additional traffic amount caused by *Strategy3* is smaller than half of that caused by *Strategy1*. In addition, Figure 10 shows that the additional traffic amount incurred in *Strategy3* is quite stable, around 80 GB, and does not vary much with the variation of the proxy numbers. This implies that, if *Strategy3* is adopted in a real system, the proxy number is not necessarily a high priority factor with respect to the traffic reduction.

Figure 11 and Figure 12 show the cache performance in terms of *Local Hit Ratio* and *Peer Hit Ratio*. These two figures demonstrate the following interesting facts. First, different from evaluation results on cache size, *Local Hit Ratio* and *Peer Hit Ratio* have opposite trends when the number of proxies increases. The *Local Hit Ratio* decreases with the increase of the number of proxies, while the *Peer Hit Ratio* increases with the increase of the number of proxies. This is expected since: with a given total cache size, the more participating proxies in the system, the less cache size a proxy has. Thus, the less chance a request can be served by a local proxy. On the other hand, the more proxies in the system, the less chance that the cache space on those proxies is occupied by large objects, thus more objects that are stored in the system, therefore the *Peer Hit Ratio* is improved.

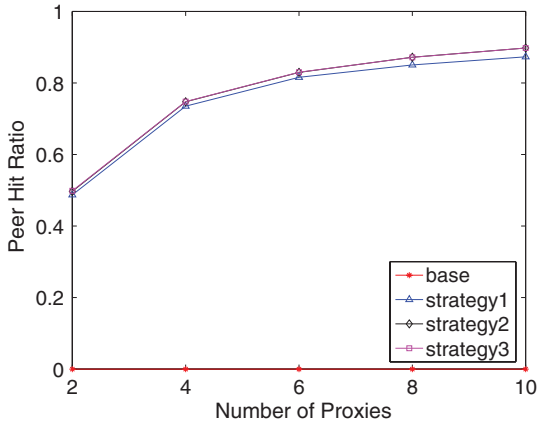

Fig. 12. Peer Hit Ratio.

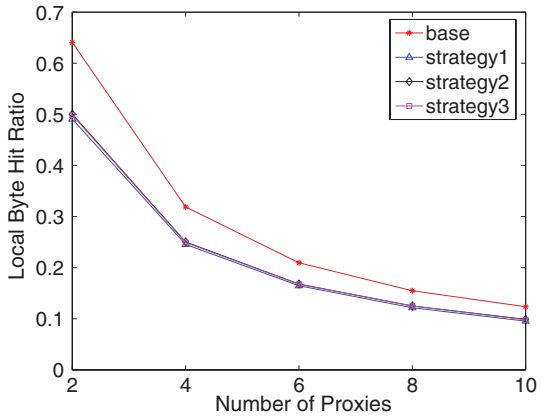

Fig. 13. Local Byte Hit Ratio.

Figure 11 and Figure 12 also indicate that the three strategies (except *Base*) achieve similar performance in the hit ratio. This is consistent with the experimental result on cache size in the previous subsection. When the total cache size is beyond 20% of the total object size, the performance differences between these three strategies are negligible.

Figures 13 and 14 show the corresponding *Local Byte Hit Ratio* and *Peer Byte Hit Ratio* for these four strategies. Compared with their corresponding number-based evaluation results shown in Figure 11 and 12, the performance trends on byte-based evaluation results are similar as before. They confirm that increasing the proxy number improves the *Peer Hit Ratio*, but decreases the *Local Hit Ratio*.

Figures 15 and 16 show the *Miss Ratio* and *Miss Byte Ratio*. They indicate the traffic amount between the proxy and the server directly caused by client requests. Similar to the evaluation results on cache size, *Strategy2* and *Strategy3* achieve better performance due to the proactive replacement policy, and changing the proxy number does not affect much on their *Miss Ratio*s and *Miss Byte Ratio*s. On the other hand, the increases of the *Total Miss Ratio* and the *Total Miss Byte Ratio* for *Base* are due to the decreasing number of objects that can be cached in a local proxy when the proxy number increases.

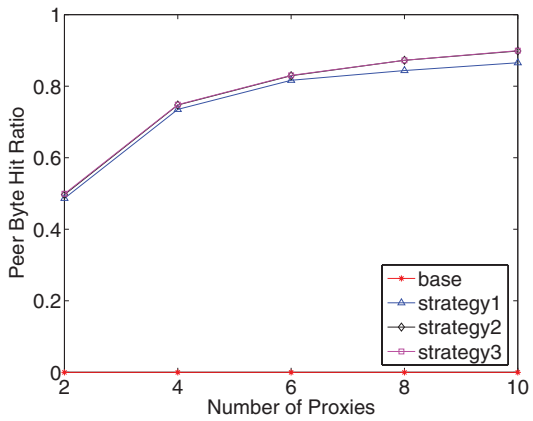

Fig. 14. Peer Byte Hit Ratio.

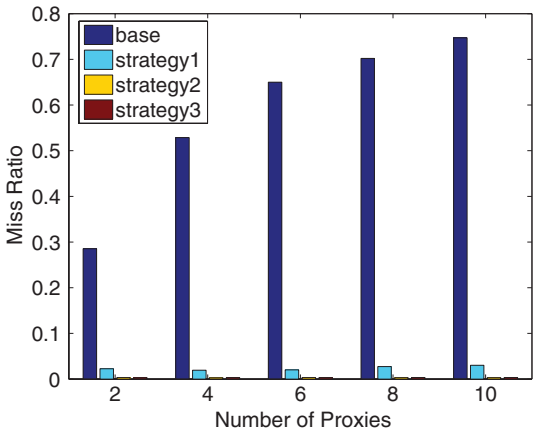

Fig. 15. Total Miss Ratio.

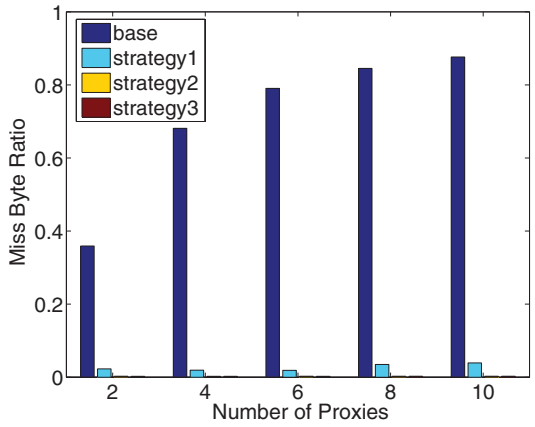

Fig. 16. Total Miss Byte Ratio.

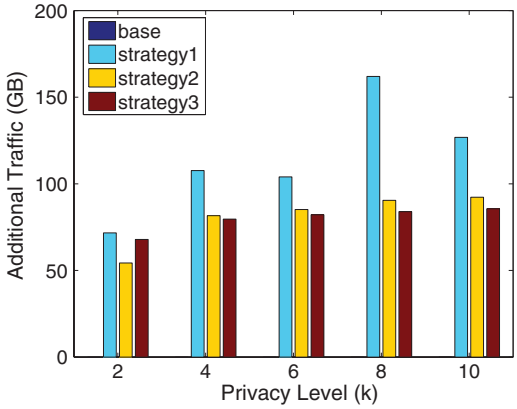

Fig. 17. Additional Traffic.

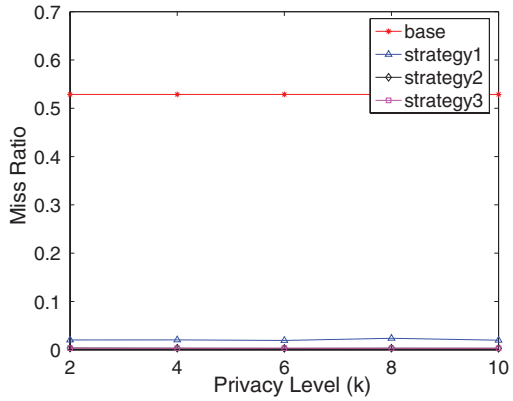

Fig. 18. Total Miss Ratio.

7.3 Privacy Level

Intuitively, the privacy level desired by clients is the most critical factor in terms of the system performance. The larger the privacy level, the larger the amount of additional traffic that could be caused to our proposed system. Without the cooperation among proxies, the increase of the additional traffic amount is linearly proportional to the increase of the privacy level, without considering the effect of temporal locality in client accesses.

To evaluate the effect of privacy level on the system performance, we conducted a similar set of experiments as before. In these experiments, we also used 20% of the total object size as the total available cache size, and set the number of participating proxies as 4. This total cache space is equally divided among the four participating proxies and we vary the client desired privacy level from 2 to 10.

As shown in Figure 17, increasing the privacy level does affect the additional traffic caused to the system. The larger the privacy level, the more the additional traffic amount the system brings. With effective replacement in *Strategy2* and traffic amortizing in *Strategy3*, they cause much less additional traffic than *Strategy1*. However, the increase of additional traffic amount is not linearly proportional to the increase of privacy level (especially for *Strategy2* and *Strategy3*). This is mainly due to a relatively flat *Total Miss Hit Ratio* increase as shown in Figure 18, and a high average request arrival rate

in our workload. Thus, part of the privacy requirement is naturally satisfied during request services without causing significant additional traffic. This indicates that if our proposed system has many users with enough cache space, the protection of users' privacy would cause much less traffic overhead than expected.

As shown in Figure 18, when the privacy level varies, the *Total Miss Ratio* has almost no change. This result is obvious since the *Miss Ratio* (and the *Hit Ratio*) are mainly affected by the total available cache size. For each privacy level, compared with *Strategy1*, *Strategy2*, and *Strategy3* result in smaller *Total Miss Ratio*. However, the absolute values of the differences are less than 2% for all three strategies. This is consistent with results shown in the previous subsections. The trend is very similar on the *Total Miss Byte Ratio*. As compared with the evaluation results in the previous experiments, similar performance trends are found, and we omitted them for brevity.

In summary, these evaluation results show 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 the hit ratio and the byte hit ratio.

8. 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 in practice 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 practically. Simulation-based experiments have been extensively conducted and the results show that our proposed system is effective.

We are extending the simultaneous conditions so that more requests can be clustered to further amortize the traffic. We also plan to collect real cooperative proxy workloads to verify our system and further explore usage control.

ACKNOWLEDGMENTS

We would like to thank William L. Bynum, Xiaodong Zhang, and anonymous reviewers for their helpful comments on this article.

REFERENCES

AN DI. VISCONTI, P. P. 2000. User privacy issues regarding certificates and the tls protocol. In *Proceedings of Conference on Computer and Communications Security*. Athens, Greece.

BAO, F., DENG, R., AND FENG, P. 2000. 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*.

BAO, F., DENG, R., FENG, P., GUO, Y., AND WU, H. 2001. 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.

BONATTI, P. A. AND SAMARATI, P. 2002. A uniform framework for regulating service access and information release on the web. In *J. Comput. Secur. 10*, 3.

BONEH, D., DING, X., TSUDIK, G., AND WONG, C. 2001. A method for fast revocation of public key certificates and security capabilities. In *Proceedings of USENIX Security*. Washington, D.C.

BRANDS, S. 2000. Rethinking public key infrastructures and digital certificates: Building in privacy. In *Rethinking Public Key Infrastructures and Digital Certificates*. MIT Press.

CAMENISCH, J. AND HERREWEGHEN, E. 2002. Design and implementation of the idemix anonymous credential system. In *Proceedings of the ACM Conference on Computer and Communications Security*. Washington, DC.

CHAE, Y., GUO, K., BUDDHIKOT, M., SURI, S., AND ZEGURA, E. 2002. Silo, rainbow, and caching token: Schemes for scalable fault tolerant stream caching. In *IEEE J. Select. Areas Comm*.

CHAUM, D. 1985. Security without identification: Transactions system to make big brother obsolete. In *Comm. ACM*. *24*, 2.

CHEN, S., CHEN, S., GUO, H., SHEN, B., AND JAJODIA, S. 2006. Efficient proxy-based internet media distribution control and privacy protection infrastructure. In *Proceedings of the 14th IEEE International Workshop on Quality of Service (IWQoS '06)*. New Haven, CT.

CHIU, M. Y. AND YEUNG, K. H. 1997. Partial video sequence caching scheme for vod systems with heteroeneous clients. In *Proceedings of the 13th International Conference on Data Engineering*. Birmingham, UK.

CHOR, B. AND GILBOA, N. 1997. Computational private information retrieval. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*.

CORON, J., NACCACHE, D., AND STERN, J. 1999. On the security of rsa padding. In *Proceedings of Crypto*. Santa Barbara, CA.

DIFFIE, W. AND HELLMAN, M. 1976. New directions in cryptography. In *IEEE Trans. Inform. Theory*. *22*, 6, 644–654.

DINGLEDINE, R., MATHEWSON, N., AND SYVERSON, P. 2004. Tor: The second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium*.

FREEDMAN, M. J. AND MORRIS, R. 2002. Tarzan: A peer-to-peer anonymizing network layer. In *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS'02)*. Washington, DC.

GAMAL, T. 1985. A public key cryptosystem and signature scheme based on the discrete logarithm. *IEEE Trans. Inform. Theory* 4.

GERTNER, Y., ISHAI, Y., KUSHILEVITA, E., AND MALKIN, T. 1998. Protecting data privacy in private information retrieval schemes. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*.

GOLLE, P. AND JUELS, A. 2004. Parallel mixing. In *Proceedings of the ACM Conference on Computer and Communications Security*. Washington, DC.

GRIWODZ, C., MERKEL, O., DITTMANN, J., AND STEINMETZ, R. 1998. Protecting vod the easier way. In *Proceedings of the 6th ACM Multimedia Conference*. Bristol, UK.

HOLT, J., BRADSHAW, R., SEAMONS, K., AND ORMAN, H. 2003. Hidden credentials. In *Proceedings of the ACM Workshop on Privacy in the Electronic Society*. Washington, DC.

KANGASHARJU, J., HARTANTO, F., REISSLEIN, M., AND ROSS, K. W. 2001. Distributing layered encoded video through caches. In *Proceedings of IEEE Inforcom*. Anchorage, AK.

MEDIACOM. http://www.mediacomcc.com/.

MIAO, Z. AND ORTEGA, A. 2002. Scalable proxy caching of video under storage constraints. In *IEEE J. Select. Areas Comm*.

PARK, J. AND SANDHU, R. 1999. Extending x.509 for secure attribute services on the web. In *Proceedings of the 22nd National Information Systems and Security Conference*. Crystal City, VA.

REITER, M. K. AND RUBIN, A. D. 1999. Anonymous web transactions with crowds. In *Comm. ACM*. *42*, 2, 32–48.

REITER, M. K. AND WANG, X. 2004. Fragile mixing. In *Proceedings of the ACM Conference on Computer and Communications Security*. Washington, DC.

REJAIE, R., H. YU, M. H., AND ESTRIN, D. 2000. Multimedia proxy caching mechanism for quality adaptive streaming applications in the internet. In *Proceedings of IEEE INFOCOM*. Tel-Aviv, Israel.

ROY, S., SHEN, B., CHEN, S., AND ZHANG, X. 2004. 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.

RSA LABORATORIES. 2002. PKCS 1 v2.1: RSA cryptography standard.

SCHOJER, P., BOSZORMENYI, L., HELLWAGNER, H., PENZ, B., AND PODLIPNIG, S. 2003. Architecture of a quality based intelligient proxy (qbix) for mpeg-4 videos. In *Proceedings of WWW*. Budapest, Hungary.

SEN, S., REXFORD, J., AND TOWSLEY, D. 1999. Proxy prefix caching for multimedia streams. In *Proceedings of IEEE INFOCOM*. New York, NY.

SHI, C. AND BHARGAVA, B. 1998. A fast mpeg video encryption algorithm. In *Proceedings of the 6th ACM Multimedia Conference*. Bristol, UK.

STALLINGS, W. 1998. *Cryptography and Network Security: Principles and Practice*. Prentice Hall.

TOSUN, A. S. AND FENG, W. C. 2002. Secure video transmission using proxies. Tech. rep., Computer and Information Science, Ohio State University.

WEN, J., SEVERA, M., ZENG, W., LUTTRELL, M., AND JIN, W. 2002. A format compliant configurable encryption framework for access control of video. In *IEEE Trans. Circ. Syst. Video Tech. (Special Issue on Wireless Video)*, 545–557.

WINSBOROUGH, W. AND LI, N. 2002a. Protecting sensitive atributes in automated trust negotiation. In *Proceedings of the ACM Workshop on Privacy in the Electronic Society*. Washington, DC.

WINSBOROUGH, W. AND LI, N. 2002b. Towards practical autmated trust negeotiation. In *Proceedings of the 3rd Workshop on Policies for Distributed Systems and Networks*. Monterey, CA.

WU, K., YU, P. S., AND WOLF, J. 2001. Segment-based proxy caching of multimedia streams. In *Proceedings of WWW*. Hongkong, China.

WU, Y. AND BAO, F. 2004. Collusion attack on a multi-key secure video proxy scheme. In *Proceedings of the ACM Multimedia Conference*. New York, NY.

YANG, H. AND LU, S. 2004. Commutative cipher based en-route filtering in wireless sensor networks. In *Proceedings of IEEE VTC Wireless Security Symposium*. Los Angeles, CA.

YEUNG, S., LUI, J., AND YAU, D. 2002. A case for multi-key secure video proxy: Theory, design, and implementation. In *Proceedings of the ACM Multimedia Conference*. Juan-les-Pins, France.

YEUNG, S., LUI, J. C., AND YAU, D. K. 2005. A multi-key secure multimedia proxy using asymmetric reversible parametric sequences: Theory, design, and implementation. In *IEEE Trans. Multimedia*. 7.

ZHANG, Z., WANG, Y., DU, D., AND SU, D. 2000. Video staging: A proxy-server based approach to end-to-end video delivery over wide-area networks. *IEEE Trans. Netw*. *8*. 429–442.

Received January 2007; revised May 2007, August 2007; accepted September 2007