

Singapore Management University Institutional Knowledge at Singapore Management University

Research Collection School Of Information Systems

School of Information Systems

7-2005

How Much You Watch, How Much You Pay

Yongdong WU

Institute for Infocomm Research, Singapore

Hwee Hwa PANG


Singapore Management University, hhpang@smu.edu.sg

Robert H. DENG

Singapore Management University, robertdeng@smu.edu.sg

DOI: <https://doi.org/10.1117/12.632558>

Follow this and additional works at: https://ink.library.smu.edu.sg/sis_research

 Part of the [E-Commerce Commons](#), and the [Information Security Commons](#)

Citation

WU, Yongdong; PANG, Hwee Hwa; and DENG, Robert H.. How Much You Watch, How Much You Pay. (2005). *SPIE Proceedings: Visual Communications and Image Processing Conference 2005, July 12-15, Beijing, China*. 5960, 961-968. Research Collection School Of Information Systems.

Available at: https://ink.library.smu.edu.sg/sis_research/1135

This Conference Proceeding Article is brought to you for free and open access by the School of Information Systems at Institutional Knowledge at Singapore Management University. It has been accepted for inclusion in Research Collection School Of Information Systems by an authorized administrator of Institutional Knowledge at Singapore Management University. For more information, please email libIR@smu.edu.sg.

How Much You Watch, How Much You Pay

Yongdong Wu^a, HweeHwa Pang^a and Robert H. Deng^b

^aInformation Security Lab, Institute for Infocomm Research, Singapore

^bSchool of Information Systems, Singapore Management University, Singapore

ABSTRACT

This paper presents a pay-video scheme that manages video stream, key stream and payment stream efficiently. In our scheme, the owner segments a video into fragments and encrypts them with independent keys. The keys are generated with a novel concept called as hash interval, where each hash interval discloses a range of numbers without disclosing any information on numbers outside of the range. The video fragments are then broadcast on one or more channels. A buyer can purchase the keys to decrypt any fragments and, within each fragment, any desired quality level. The accompanying payment protocol is integrated with the key management protocol seamlessly and hence the computation cost is very low.

Keywords: Hash Interval, Micro-payment stream, Pay-video

1. INTRODUCTION

Media streaming is becoming more and more popular due to the explosive growth of Internet and multimedia processing technologies and applications.¹ To become a lucrative business of content delivery, a flexible access control strategy integrated with flexible payment scheme is in desire. Specifically, the access control approach must enable the buyer has more than one selection, e.g., time and quality, and hence forms different service groups. For example, Ma *et al.*² propose a dynamic access control scheme for group communications which support multiple service groups with different access privileges. The groups may be either independent or dependent. Their method allows dynamic formation of service groups and maintains forward/backward security when buyers switch service groups. On the other hand, a flexible payment mechanism accommodates different usage and charging models. Particularly useful are micro-payment schemes that offer the ability to make payments of small amounts. Although several micropayment schemes³⁻⁹ have been proposed, few are designed to support selective payments of video fragments. The challenge here is how to integrate the key management and payment scheme such that the buyer pays for what she is interested in.

One straightforward method is to divide the video into fragments and each fragment is protected with an independent key. The protected fragments are broadcast via CDN (content distribution Network) publicly, but the decryption keys will be disseminated secretly. If the buyer requests for some fragments, the seller sends the corresponding decryption keys to the buyer. Clearly, this naïve solution is flexible but inefficient.

Wu *et al.*¹⁰ presented a flexible access method for secure multicast streaming. In the scheme, an owner segments a video into fragments, and prepares the protected video by selectively encrypting the video fragments off line. Proxies store the protected videos and distribute them. To decrypt a protected video or fragment, a buyer sends to the seller a request for the decryption keys. The seller then generates an enabling block which enables the buyer to access video or fragments in a period. Therefore, the scheme enables the buyer to access a fragment or video for a restricted number of dissemination times, but the seller selectively encrypts the video only once, and seller performs no encryption at all. However, its capability of collusion resilience is low.

Perkins *et al.*¹¹ proposed a scheme which enabled to access the content for predefined times. In the scheme, a trust player should be installed such that the number of licenses can be decreased gradually. To defeat against license backup, an on-line license authority or registrar should be deployed. The scheme also suffers from both additional computational cost and storage.

To overcome the above challenge, we partition a video into fragments based on the video's utilization characteristics,^{12, 13} and protect each fragment with a unique key. The keys are produced with our novel hash interval

Send correspondence to Yongdong Wu: wudong@i2r.a-star.edu.sg, www.i2r.a-star.edu.sg/icsd/staff/yongdong/

as realized through two embodiments: two-way chain and down-tree. This hash interval scheme enables a buyer to generate the fragment keys from the seller’s authorization so as to reduce the network overhead. After the protected video is produced, it is transmitted in a broadcast channel. If the video is encoded into several layers, the video layers are distributed with different channels. When a buyer wishes to purchase a range of fragments, she pays for and obtains the keys specifically for those fragments. At the same time, the buyer cannot render any other fragment because she has no ways to generate their keys.

Section 2 introduces our proposed hash interval scheme and its embodiments, and Section 3 describes the possible business model. Section 4 addresses the pay-video scheme. Section 5 depicts the performance of the proposed scheme in terms of security and cost. Section 6 draws a conclusion.

2. HASH CHAIN AND HASH INTERVAL

In this section, we introduce some primitives which will be employed in the present scheme.

2.1. Hash Function

A hash function takes a variable-length input string and converts it to a fixed-length output string, called a hash value. A one-way hash function, denoted as $h(\cdot)$, is a hash function that works in one direction: it is easy to compute a hash value $h(m)$ from a pre-image m ; however, it is hard to find a pre-image that hashes to a particular hash value. There are many existing one-way hash functions, such as SHA-1.¹⁴

2.2. Hash Chain

Hash chain¹⁵ is widely used in many micro-payment schemes (e.g. PayWord³) due to its low computation cost. In the hash chain shown in Fig. 1, a seed K is used to create a sequence of numbers. A number k_i can be generated from any number $k_j (i < j)$ further down the chain, without disclosing any information on numbers beyond k_j . Formally, given a master seed K , the hash chain of length n is constructed as follow,

$$\begin{aligned} k_N &= K \\ k_i &= h(k_{i+1}) = h^{N-i}(K), \quad i = N - 1, \dots, 2, 1 \end{aligned}$$

where k_1 is a public number.

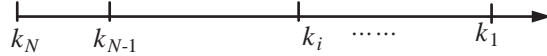


Figure 1. Hash Chain generated from a secret K .

2.3. Hash Interval

Since any number k_i in a hash chain can be generated from $k_j (j > i)$, the hash chain is not applicable for range access. For example, a hash chain cannot be used to merely expose the values $\Gamma = \{k_i, k_{i+1}, \dots, k_j\}$ for some fixed i and j . Instead, we propose the *hash interval* $HI(\cdot)$ which is used to disclose a range of numbers without disclosing any information on other numbers below or above the stipulated range. Here we propose two embodiments to realize the hash interval: two-way chain and down-tree.

2.3.1. Two-Way Chain

As an extension of the hash chain, we set up two hash chains in opposite directions. Given the seeds K_l and K_r , the hash interval of length N is constructed as

$$\begin{aligned} L_N &= K_l \\ L_i &= h(L_{i+1}) = h^{N-i}(L_N), \quad i = N - 1, \dots, 2, 1 \\ R_1 &= K_r \\ R_i &= h(R_{i-1}) = h^{i-1}(R_1), \quad i = 2, \dots, N \\ k_i &= h(L_i \parallel R_i) \end{aligned} \tag{1}$$

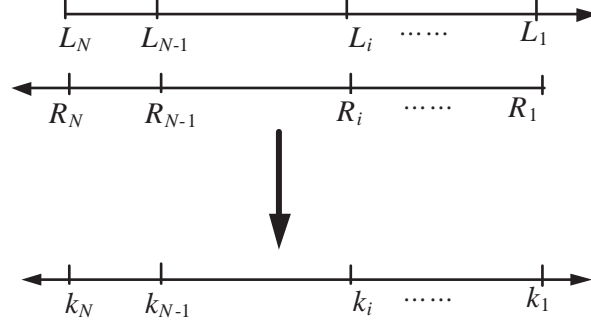


Figure 2. Hash Interval constructed with Two-way chains.

where “||” is the concatenation operator. Fig.2 illustrates the construction process.

Claim 1: With two-way chain, two numbers L_j and R_i are able to disclose the range $\Gamma = \{k_i, k_{i+1}, \dots, k_j\}$, and no number $k_t \notin \Gamma$ is disclosed if no collusion exists.

Proof: It is easy to know that any range can be deduced from its two end numbers based on the above construction method. Now we prove the second property: no extra number is disclosed.

Assume there is at least one extra number $k_t \notin \Gamma$ is disclosed, i.e., $t < i$ or $t > j$. Let’s consider $t < i$ first, $k_t = h(L_t || R_t)$. Since $h(\cdot)$ is a one-way function, L_t and R_t are known to the attacker if k_t is known to the attacker. That is to say, the attacker is assumed to obtain R_t from R_i according to $R_i = h^{i-t}(R_t)$. Therefore, the attacker can subvert the one-way function $h(\cdot)$. Thus, the attacker can not obtain any extra number k_t where $t < i$. Similarly, the attacker can not obtain any extra number k_t where $t > j$. Claim 1 holds.

2.3.2. Hash Interval Tree

The two-way chain is efficient for exposing a range since only two terminal numbers need to be disclosed. Unfortunately, it is vulnerable to collusion attack. Specifically, in some cases, if two ranges Γ_1 and Γ_2 are exposed, $\exists k, k \notin \Gamma_1 \cup \Gamma_2$, but k which is a leaf of HI-tree is leaked. To avoid this weakness, we propose a second embodiment which is a hash tree constructed in a top-down manner. In this Subsection, we seek to adapt Sandhu’s approach¹⁶ to arrive at a Hash Interval tree (HI-tree). The tree root is assumed to be known in advance, and the value of any non-root node is derived from its parent node. That is to say, given any non-leaf node with value v of an q -ary tree, the value of its child node i is $h_i = h(v || i)$, ($i = 1, 2, \dots, q$) assuming there are q child nodes. All the value of leaf node constitutes a universal set $\{k_i\}$. Fig.3 illustrates the construction process.

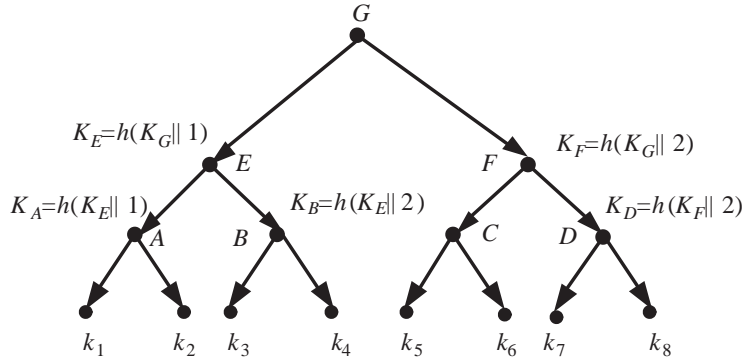


Figure 3. Hash Interval Tree ($q = 2$).

To release an interval of numbers, an enabling block * is issued. For example, if a buyer requires the numbers $\Gamma = \{k_2, k_3, \dots, k_6\}$ in Fig.3, the enabling block $\mathbf{S} = \{k_2, h_B, h_C\}$ is calculated with the method shown in Table 1. In Table 1, the element $x \in \mathbf{X}$ is the node position in the HI-tree, $sibling(x)$ is the sibling node of node x and $parent(x)$ is the parent node of node x . The enabling block will be sent to the buyer as well as their positions in the HI-tree.

Table 1. Generating enabling block given $q = 2$

input: $\mathbf{X} = \{a, a + 1, \dots, b\}$ output:enabling block \mathbf{S}
$\mathbf{S} = nil$ $\mathbf{Enblk}(\mathbf{X})$ end
function $\mathbf{Enblk}(\mathbf{X})$ $\mathbf{U} = nil$ $\forall x \in \mathbf{X}$ if $sibling(x) \notin \mathbf{X}$ then $\mathbf{S} \leftarrow \mathbf{S} \cup \{x\}$ else $\mathbf{U} \leftarrow \mathbf{U} \cup \{parent(x)\}$ $\mathbf{X} \leftarrow \mathbf{X} - \{sibling(x)\}$ endif $\mathbf{X} \leftarrow \mathbf{X} - \{x\}$ if $\mathbf{U} \neq nil$ then $\mathbf{Enblk}(\mathbf{U})$ endif

Claim 2: In an HI-tree, the enabling block merely discloses the specific range $\Gamma = \{k_i, k_{i+1}, \dots, k_j\}$, but no number $k_t \notin \Gamma$ is leaked.

Proof: It is easy to prove the correctness of claim 2 when there is no collusion attack. Now let's focus on collusion attack. That is to say, an attacker obtains a finite number of enabling blocks B_i . Denote the union $\mathbf{B} = \bigcup B_i$. Expanding all the non-leaf nodes in the enabling blocks to their leaf nodes as a set \mathbf{Y} . It is a one-to-one mapping between \mathbf{B} and \mathbf{Y} . Since all the leaves are independent, the attacker can not obtain any new leaf x with \mathbf{Y} . Therefore, no $k_t \notin \Gamma$ is disclosed.

3. A VIDEO BUSINESS MODEL

It is possible that a buyer may be interested in some period of the video, not only the start fragments, for example, the goal period of a football competition. To provide access to selected video fragments, the hash interval is a suitable tool for secure video delivery.

In the proposed scheme, the system includes the buyer, seller, owner and bank. Fig.4. The owner prepares the protected content, and the buyer will send the request and payment to seller, and render the authorized content. The bank will deal with the payment issue.

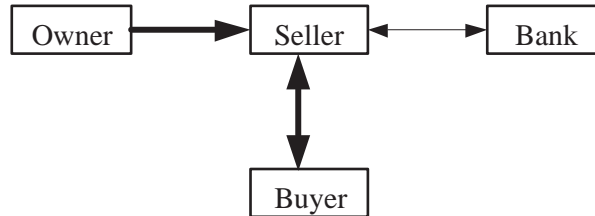


Figure 4. System Structure.

*Enabling block: Given a sequence of leaves, an enabling block consists of the minimal number of nodes of the HI-tree and those nodes cover the given leaves exactly.

In the video business, the content stream is protected with an encryption function and sent publicly, but the channel for key delivery is assumed to be secure. To obtain the content sent in a secure delivery channel, a non-authorized buyer may

- obtain the non-authorized content independently or in a collusion way, e.g., the buyer obtains a fragment of higher quality from several fragments of lower quality, or deduce a decryption key for a non-authorized fragment from several keys.
- pay less than what she should. In the payment stage, the buyer may send small number of coins to the seller, e.g., a coin is used twice.

4. PAY-VIDEO PROTOCOL

In the present video distribution application, there are four modules: protected video preparation, Enabling block generation, payment and rendering.

4.1. Video fragment

To enable buyers to purchase selected portions within a video stream, the video should be segmented into fragments that are encrypted with different keys. In general, a video can be segmented into fragments at random start and end numbers. However, it is more effective to base the segmentation on the video’s utilization characteristics.^{12,13} As found in experiments by Acharya and Smith,¹² only 55% of all requestors play the entire video, with most stoppages occurring during the first 5% of the movie playback period (see Figure 5). The short starting pattern suggests to distribute the first several minutes frequently. Thus, a video should be segmented based on the usage percentage so as to provide a multiple of services.

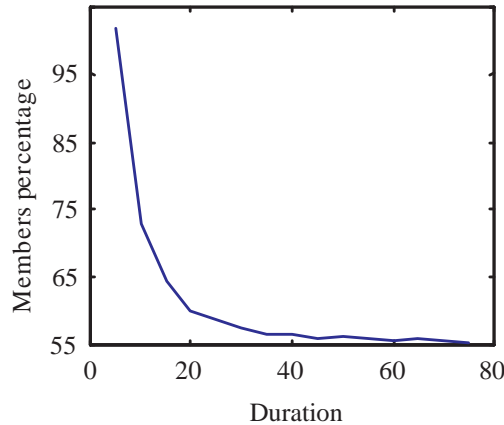


Figure 5. Partial Playback¹² – only 55% of all requestors play the entire video, and most stoppages occur during the first 5% of the movie playback period.

4.2. Protected fragment

To generate a protected video stream with an efficient key management, the owner creates an HI-tree with the method in Section 2, and encrypts the video fragments with the keys (leaves) in the HI-tree.

Furthermore, if a fragment is encoded with n layers (for example, an MPEG-4 stream comprises a basic layer and several enhancement layers), each layer is encrypted independently and distributed on a separate channel. In this case, the key structure includes a key chain and an HI-tree as shown in Fig.6 or Fig.7. Despite either method is applicable to access control for the fragments, their performance depend on the statistics of buyers’ requests. For example, the method in Fig.6 is suitable for a specific time period, while the method in Fig.7 is more efficient to the applications when buyers pays for all the fragments at a specific quality layer.

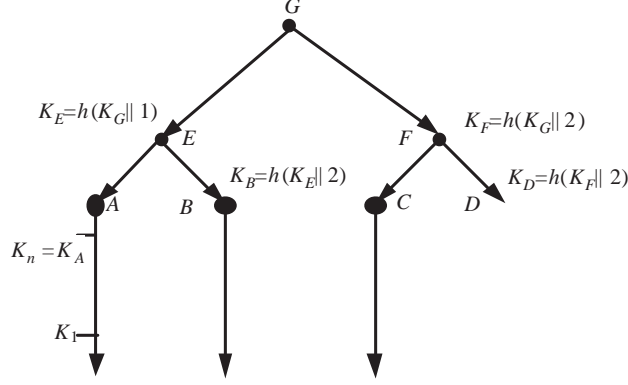


Figure 6. Key structure given time access is of higher priority than quality access. An HI-tree is used to generate the key for a certain requirement(e.g. time) and a hash chain is used to generate key for quality.

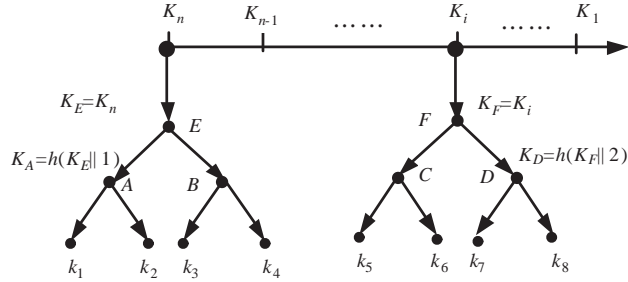


Figure 7. Key structure which is efficient for quality request, where K_n is used for highest quality and K_1 is used for lowest quality.

For simplicity, we merely discuss the key structure in Fig.7 hereinafter. At the owner side, each layer $l \leq n$ is segmented into N fragments denoted as $S_{l1}, S_{l2}, \dots, S_{lN}$. Then, the owner generates the layer keys with the hash chain with a master key K , denote the l th layer key $K_l = h^{n-l}(K)$. For each layer, the owner generates the fragment key $\{k_{li}, i = 1, 2, \dots, N\}$ with an HI-tree whose root is K_l . According to an encryption algorithm $enc(\cdot)$ such as AES, the owner creates the protected fragments as the ciphertext $enc(k_{li}, S_{li})$, $i = 1, 2, \dots, N$, $l = 1, 2, \dots, n$. The protected fragments are stored into the seller for delivery.

4.3. Enabling block

After receiving the request and payment from the buyer, the seller checks the validation of the payment, the seller searches the corresponding quality tree(s) with the method shown in Table 1. For example, if the buyer requests the quality l_0 and fragments between S_{li} and S_{lj} , $i < j$. The seller looks up all the quality trees for each quality $l \leq l_0$, and all the leaves (or keys) for the fragments S_{li} and S_{lj} , $i < j$ and constructs the enabling block B_l . Then, the seller sends $\bigcup_{l=1}^{l_0} B_l$ to the buyer.

4.4. Rendering

Generally, if the buyer requires l layers, she will receive l enabling blocks, and then generate all the fragment keys $\{k_{li}\}$. With the keys k_{li} for fragment i in a specific layer l , the protected video fragment will be decrypted correctly and further decoded with the target player.

4.5. Payment sub-protocol

In order to render selected video fragments, the buyer has to pay the seller based on the desired fragments and quality layers. An electronic payment protocol includes three sub-protocols: signing check, depositing check, and clearing check.

4.5.1. Signing Check

A buyer pays a seller for a transaction by digitally signing a piece of data that identifies the transaction. Initially, the buyer collects the message D : buyer name, seller name, the time, account and credit information. She also prepares a hash chain $C = \{c_0, c_1, \dots, c_N\}$, where $c_i = h(c_{i+1})$ represents a basic unit of payment, say 1 cent, and N is the upper boundary of her credit. The buyer sends the message $M_0 = \{D, c_0\}$, her signature on M_0 , and her credit certificate issued by the bank to the seller.

Subsequently, in the i th transaction T_i , if the buyer is interested in a range of video fragments $\{s_i, \dots, e_i\} \subseteq \{1, \dots, N\}$, she sends to the seller the payment as

$$M_i = \{s_i, e_i, c_{m_{i-1}+p_i}\} \quad (2)$$

where p_i is payment in the transaction T_i , and m_{i-1} represents the total payment in the last transactions T_1, \dots, T_{i-1} .

4.5.2. Depositing check

The seller first verifies the certificate and the message M_0 . If they are authentic, he verifies the authenticity of payment M_i based on the last transaction message M_{i-1} . If the verification result is also positive, the seller transmits to the buyer the corresponding enabling blocks constructed with the mechanism in Subsection 2.3.

4.5.3. Clearing check

After receiving the payment messages M_i , the bank will deduct the amount p_i from the buyer's account and credit the corresponding amount to the seller's account. Optionally, the seller may forward the payment messages to the bank in batches.

5. PERFORMANCE

5.1. Cost

In the present scheme, the owner will generate an HI-tree for each quality layer off line. Assume that there are n quality levels and N leaves in each HI-tree. The owner will calculate $2N \times n + n - 1$ hash values in total. In addition, the owner will encrypt the fragments with $N \times n$ encryption operations. At the seller side, the seller will generate the enabling block to each request as well as check verification. The computation cost for verification is almost constant. But the cost of generating enabling block varies with the request. Roughly, it is much less than the number of requested fragments. Similarly, the buyer will reconstruct the access keys for all the requested fragments and the computational cost is the same as the seller. Since the computational resource is mainly used for hash operations, the computation cost is very small. For example, based on experiment result by Wei,¹⁷ the hash computation speed is up to 68MB/s for SHA-1 with a Pentium 4 processor.

The network overhead consists of payment messages and enabling blocks. The payment overhead is constant for each transaction, while the communication overhead is $O(n \times \log_2 \Delta_i)$ where Δ_i is the requested fragments in the transaction T_i .

5.2. Security

5.2.1. Access violation

Our proposed scheme offers backward security and forward security due to the safeguards of the hash interval in Subsection 2.3. That is to say, a buyer who joins at time t_i cannot obtain the keys for any time $t < t_i$. Thus, the buyer cannot decrypt the "old" fragments even if she has recorded the network traffic. Furthermore, if the buyer's payment entitles her to only up to time t_j of the video, she would not be able to render any video fragments from $t > t_j$.

5.2.2. Payment violation

Our scheme adopts the payment scheme PayWord³ for providing payment stream which corresponds to the fragment stream. With PayWord, a buyer can pay exactly what she purchases and she is not able to double payment. Meanwhile, the seller can not charge the buyer more than what the buyer should pay.

6. CONCLUSION

This paper presents a pay-video scheme that manages video stream, key stream and payment stream efficiently. Our scheme proposes a new primitive called hash interval which is an extension of hash chain. To generate the protected video fragments, the owner fragments a video into fragments and encrypts them with independent keys. The keys are generated from hash interval. The video fragments are then broadcast on one or more channels. A buyer purchases the keys to decrypt any consecutive fragments and, any desired quality level. With the payment stream idea from PayWord, the accompanying payment protocol is integrated with the key management protocol seamlessly and hence the computation cost is very low.

REFERENCES

1. Z. Morley Mao, David Johnson, Oliver Spatscheck, Jacobus E. van der Merwe, and Jia Wang, "Efficient and Robust Streaming Provisioning in VPNs," WWW2003, pp.118-127.
2. Di Ma, Yongdong Wu, Robert Deng, and Tieyan Li, "Dynamic Access Control for Multi-privileged Group Communications," 6th International Conference on Information and Communications Security, Lecture Notes in Computer Science (LNCS) 3269, pp.508-519, 2004
3. R. Rivest and A. Shamir, "Payword and micromint: two simple micropayment schemes," International Workshop on Security Protocols, LNCS 1189, pp.69-87, 1997.
4. S. Micali and R. Rivest, "Micropayments revisited," CT-RSA 2002, LNCS 2271, pp.149-163, 2002.
5. DigiCash website, <http://digicash.com>.
6. Ronald L. Rivest. Electronic lottery tickets as micropayments. In Proceedings of Financial Cryptography 97, LNCS 1318, pp.307-314, 1997.
7. S. Glassman, M. Manasse, M. Abadi, P. Gauthier, and P. Sobalvarro, "The millicent protocol for inexpensive electronic commerce," WWW4, 1995. <http://www.research.digital.com/SRC/personal/MarkManasse/uncommon/ucom.html>.
8. Charanjit Jutla and Moti Yung. "PayTree: amortized-signature for flexible MicroPayments," USENIX Workshop on Electronic Commerce, pp.213-221, 1996.
9. W3C, "Micropayments overview," <http://www.w3.org/ECommerce/Micropayments/>.
10. Yongdong Wu, and Feng Bao, "Flexible Access to Video Streaming," IEEE Vehicular Technology Conference (VTC), Track 0401-05, CD-ROM, 2004.
11. G. Perkins, and P. Bhattacharya, "An Encryption Scheme for Limited K-time Access to Digital Media ", *IEEE Transactions on Consumer Electronics*, 49(1):171-176, 2003.
12. S. Acharya, and B. Smith, "An Experiment to Characterize Videos Stored on the Web," ACM/SPIE Multimedia Computing and Networking 1998.
13. M. Chesire, A. Wolman, G. Voelker, and H. Levy, "Measurement and Analysis of a Streaming Media Workload," USENIX Symposium on Internet Technologies and Systems (USITS), 2001.
14. National Institute of Standards and Technology, "Secure Hash Standard (SHS)", FIPS Publication 180-1, 1995.
15. L. Lamport, "Password Authentication with Insecure Communication," *Communication of ACM*, 24(11):770-772, 1981.
16. R. S. Sandhu, "Cryptographic implementation of a tree hierarchy for access control", *Information Processing Letters*, 27(2): 95-98, 1988.
17. Dai Wei, Crypto++ 5.2.1 Benchmarks, <http://www.eskimo.com/~weidai/benchmarks.html>