Vol. 47 No. 12 IPSJ Journal Dec. 2006

Regular Paper

Design of τ-Gradual Key-Management Schemes for Mobile Content Distribution

Kazuhide Fukushima,† Shinsaku Kiyomoto† and Toshiaki Tanaka†

Copyright protection is a major issue in online content-distribution services and many key-management schemes have been proposed for protecting content. Key-distribution processes impose large burdens even though the communications bandwidth itself is restricted in the distribution of mobile content provided to millions of users. Mobile devices also have low computational capacities. Thus, a new scheme of key management, where the load on the key-distribution server is optimal and loads on clients are practical, is required for services. Tree-based schemes aim at reducing the load on the server and do not take reducing the load on clients into account. The load on clients is minimized in a star-based scheme, on the other hand, while the load on the server increases in proportion to the number of clients. These structures are far from being scalable. We first discuss a relaxation of conventional security requirements for key-management schemes in this paper and define new requirements to improve the efficiency of the schemes. We next propose the τ-gradual key-management scheme. Our scheme satisfies the new security requirements and loads on the server, and it has far fewer clients than conventional schemes. It uses an intermediate configuration between that of a star- and a tree-structure that allows us to continuously change it by controlling the number of clients in a group, $m_{\max}$. The scheme can be classified as τ-star-based, τ-tree-based, or τ-intermediate depending on the parameter, $m_{\max}$. We then present a quantitative evaluation of the load on the server and clients using all our schemes based on practical assumptions. The load on the server and that on clients involves a trade-off with the τ-intermediate scheme. We can construct an optimal key-management structure according to system requirements using our schemes, while maintaining security. We describe a concrete strategy for setting parameter $m_{\max}$. Finally, we present general parameter settings by which loads on both the server and clients using the τ-intermediate scheme are lower than those using the τ-tree-based scheme.

1. Introduction

The high-speed mobile Internet has recently been expanded to include 3G mobile services. Digital-content delivery that provides music, movies, and games for mobile phones has become a major service for mobile users. Pay-broadcasting is expected to become a new mobile service of particular importance in the near future. Copyright protection is a major issue with these services. It is easy to duplicate digital content and because it requires little effort to do so illegal content is being widely circulated. Encrypting digital content is one solution to prevent illegal copying and generate income from clients. Many schemes for managing encryption keys have been proposed. These schemes are called *key-management schemes*. However, existing key-management schemes are not ideal for the content-distribution services required by mobile devices.

The load on the client is low in the star-based scheme [6], which is an existing key-management configuration using a logical star-structure. The load on the server, however, increases in proportion to the number of clients (see Section 2.2.1) and this scheme is far from scalable. The tree-based scheme [16], on the other hand, which is an existing configuration using a logical tree-structure, reduces the load on the server to a logarithmic order for the number of clients.

However, the tree-based scheme is still not ideal. First, it only reduces the load on the key-management server while the star-based one only reduces that on clients. We cannot construct an optimal key-management structure for a changing mobile environment using these existing schemes. Thus, new key-management schemes based on an intermediate structure between that of the star and tree configurations are required. Second, the current security requirements are too strict. The key-management server must update keys whenever

† KDDI R&D Laboratories Inc.

a client joins or leaves to satisfy security requirements (Section 2.2.2). The tree-based and star-based schemes are designed to satisfy two security requirements i.e., Backward Security (BS) and Forward Security (FS). BS involves joining clients not obtaining any old shared keys, and FS involves leaving clients not obtaining any shared keys, which are distributed after the client leaves. The load on the server and client increases in proportion to the number of joins and leaves in both key-management schemes. The load will be too high for mobile devices with low computational capacities when there is an extremely large number of clients. For example, a client must receive 39,900 messages per second in the worst case assuming that the total number of clients is 1,000,000, the degree of the tree is 2, and 1,000 clients join and leave every second. Thus, relaxed security requirements are required to improve efficiency.

We first discuss relaxing conventional security requirements in this paper and then define new security requirements to improve efficiency. We then propose our τ-gradual key-management scheme, which satisfies the new security requirements and imposes far fewer loads on the server and the client than conventional schemes. The scheme has a hybrid star- and tree-structure, and can be continuously changed by controlling the number of clients in a group, $m_{\max}$. Finally, we evaluate the loads on the server and the clients in these schemes and discuss the trade-off that has to be made between these loads with the τ-intermediate scheme.

We can construct an optimal key-management structure that satisfies system requirements using a hybrid scheme while maintaining security. Furthermore, the loads imposed on both clients and servers by the τ-intermediate scheme are lower than those by the τ-tree-based configuration under certain conditions.

2. Related Work

Broadcast encryption was first proposed by Berkovits [5]. Fiat, et al. [10] formalized the basic definition of broadcast encryption. Since then, many key-management schemes have been proposed. Existing key-management schemes can be classified into two types, .i.e., *stateless* and *stateful*.

As the clients' keys in a stateless scheme are never updated, extremely secure devices for storing each client's keys can be configured inexpensively. Numerous stateless schemes have been proposed for this reason. However, generating a pre-shared key in stateless schemes depends on the maximum number of clients and the size of the message increases as the number of clients leaving the service increases. Thus, stateless schemes are not suitable for long-term or large-scale services.

We do not need to consider the maximum number of clients in stateful schemes, on the other hand. The size of the message also does not depend on the number of clients leaving the service. As stateful schemes are suitable for long-term and large-scale services, we investigated these for key management.

2.1 Stateless schemes

The content-encryption key is only based on the content-distribution message and the pre-shared key for each client. That is, keys for each client are distributed when they join the service and are never updated.

Many stateless schemes have been proposed [1)~4),8),9),11),13),14)].

2.2 Stateful schemes

The content provider uses a shared key as the content-encryption key. Each client has a shared key; key-encryption keys necessary for encrypting the shared key and other key-encryption keys. The shared key and key-encryption keys are upgraded. Therefore, the state of a client during key updates will affect his or her ability to decrypt future keys.

Many stateful schemes have been proposed [6),7),12),15)~17)].

The simplest one is the star-based scheme [6)]. All clients use a shared key, and the key is updated when a client joins or leaves. Wong, et al. [17)] and Wallner, et al. [16)] proposed a primary tree-based scheme. The shared key was assigned to the root node and the key-encryption keys to the interior nodes. Individual client keys were also assigned to leaf nodes. Each client had keys assigned to all nodes along the shortest path from the root node to the leaf node. Chang, et al. [7)] and Kim, et al. [12)] improved their schemes using Boolean Function Minimization and Diffie-Hellman key exchange. Finally, Pinkas [15)] proposed a scheme for reducing the number of update messages needed by a previously off-line member.

Details on the star-based and tree-based schemes are set out below. Each client shares an individual key with the key-management server and trusts the server.

2.2.1 Star-based Scheme

A client has two keys, i.e., a shared key and an individual key.

Join Process The server updates the current shared key. It then encrypts the updated shared key with the old shared key and sends it to existing clients. The server also encrypts the key with the individual key of a new client, and sends it to this client.

Leave Process The server updates the current shared key. It then encrypts the updated shared key with each individual key and sends it to each client.

2.2.2 Tree-based Scheme

This scheme uses a logical k-ary tree called the key-management tree. Each node of the tree corresponds to a key for clients, and does not correspond to a real entity such as a router. The shared key is assigned to the root node of the tree, the individual keys of the client are assigned to leaf nodes, and key-encryption keys are assigned to intermediate nodes. A client has $\log_k N$ keys assigned to the ancestor nodes of the node where its individual key is assigned. (N is the total number of clients and k is the degree of the tree.) In this scheme, the server directly sends messages (encrypted key) to clients.

Join Process The server updates the shared key and all the key-encryption keys assigned to the ancestor nodes of the node where the individual key of a new client is assigned. It then encrypts each updated key with the old keys and sends them to existing clients who have the old keys. Finally, it encrypts these keys with the individual key of the new client, and sends the encrypted keys to this client.

Leave Process The server updates the shared key and all the key-encryption keys assigned to the ancestor nodes. It then encrypts all the updated keys with keys assigned to the child nodes, and sends them to existing clients who have the child keys. It finally encrypts these keys with the individual key of the new client, and sends the encrypted keys to this client.

3. Issues

3.1 Requirements for Mobile Services

We assumed content-distribution services for mobile devices where charges were levied on usage-based rates. Clients usually use services in their free time such as when commuting and when the service utility time is short. Generally, these services have the following characteristics:

- Their number of clients is extremely large.
- Their communication bandwidth is restricted.
- The computational capacities of their clients are low.
- Their clients frequently join and leave since the service utility time is short.

Thus, a new key-management scheme for distributing mobile content should satisfy the following requirements.

- The load on the key-management server needs to be optimal.
- The load on the client needs to be practical.

The first requirement is necessary to minimize the cost of communication. This is mandatory for mobile-content distribution where the communication bandwidth is restricted. The second requirement is needed for mobile devices with low computational capacities. However, we can reduce the total cost of communication by sharing various processes with clients if mobile devices have sufficient margin for computational capacities. We should thus minimize the load on the server while considering the load on the client.

3.2 Problems with Existing Schemes

The existing schemes suffer from two problems.

First, they are either based on the star or the tree structure. The load on the client is low in the former. However, the server must issue N types of messages (encrypted keys) when a client leaves in the star-based scheme, where N is the number of clients (mobile devices).

The load on the server is extremely high, and this scheme is far from scalable. The server only needs to issue about $\log_k N$ types of messages when a client joins or leaves in tree-based schemes, where k is the degree of the tree. These schemes are more efficient than those that are star-based. However, the shared key and the key-encryption keys are updated every time a client joins or leaves. Some schemes [7),12),15)] reduce the size of the messages used for updating keys. However, these schemes do not reduce the frequency with which the keys need updating and do not take the load on clients into consideration. If we apply the scheme to large-scale services, clients frequently join or leave and keys must be updated often. Therefore, a large load is imposed on clients, and these schemes are not ideal for devices such as mobile phones that have low computational

capacities.

Second, current security requirements are too strict. A valid client in content-distribution services can obtain common content-encryption keys from key-management messages at the same time as being charged a fee for the content. An illegal client could try to obtain an old content-encryption key from an old key-management message sent before he or she joins, or try to obtain a new key from a message, which will be sent after he leaves. It should be noted that anyone can obtain encrypted content but only valid paying clients can obtain the content-encryption key.

Many existing key-management schemes, including those that are star- and tree-based, satisfy following two requirements:

Backward Security (BS) No joining client can obtain any shared keys that were distributed in the past. More formally, a client that joined at time t_0 cannot obtain any shared keys at time $t < t_0$.

Forward Security (FS) No leaving client can obtain shared keys that will be distributed in the future. More formally, a client that left at time t_0 cannot obtain any shared keys at time $t > t_0$.

These requirements are impractical for large-scale mobile-broadcasting services. We assumed a tree-based key-management scheme where there were $N = 1{,}000{,}000$ clients, and the degree of the tree was $k = 2$ with 1,000 clients joining and 1,000 clients leaving every second. Under this assumption, the server must issue 79,700 messages, and a client must receive at most 39,900 messages per second in the worst case. When a client joins, the server updates all the $\log_k N$ keys, which are assigned to the ancestors of the node where the individual key of the new client will be assigned. The server then encrypts each updated key with two keys, i.e., the key before update and the individual key of the new client. Thus, the server must issue $2\log_k N$ encrypted keys. The new client and siblings of the new client, whose individual key is assigned to the sibling nodes, must also receive $\log_k N$ keys. When, a client leaves, the server updates all the $\log_k N$ keys, which the leaving client has. The server then encrypt each updated key with k keys assigned to child nodes. The server must thus issue $k\log_k N$ encrypted keys and sibling clients of the leaving clients receive $\log_k N$ keys. Thus, the server must issue $(2\log_k N + k\log_k N) \cdot 1{,}000 = 79{,}700$ messages. A client must receive $(\log_k N + \log_k N) \cdot 1{,}000 = 39{,}900$ messages in the worst case. The load on the server is heavy and the load on the clients in particular is too heavy for mobile devices with low computational capacities.

4. Overview of τ-Gradual Key-Management Scheme

We propose the τ-gradual key-management scheme. However, the current requirements are too impractical for large-scale mobile-content distribution services as discussed in Section 3.2. Therefore, we relaxed the BS and FS requirements and defined new ones, i.e., ε_1-gradual Backward Security and ε_2-gradual Forward Security.

ε_1-gradual Backward Security (ε_1-gBS) No client that joined at time t_0 can obtain any shared keys at time $t < t_0 - \varepsilon_1$ ($\varepsilon_1 \geq 0$). We have denoted this by ε_1-gBS throughout the rest of this paper.

ε_2-gradual Forward Security (ε_2-gFS) No client that left at time t_0 can obtain any shared keys at time $t > t_0 + \varepsilon_2$ ($\varepsilon_2 \geq 0$). We have denoted this by ε_2-gFS throughout the rest of this paper.

ε_1 and ε_2 are parameters and these can be determined by service providers. ε_1-gBS is identical to BS and ε_2-gFS is identical to FS if $\varepsilon_1 = 0$ and $\varepsilon_2 = 0$. It is both realistic and necessary to relax these requirements. We can use the broadcasting service for mobile phones as an example. A service provider is not overly concerned if a client watches free content for periods exceeding a few tens of seconds for a service that is charged for every hour.

The τ-gradual key-management scheme satisfies the new requirements. This scheme also uses an intermediate structure between that of a star and a tree, i.e., group keys are proposed using the star-based scheme, and shared key and key-encryption keys are managed using the tree-based scheme. The structure is controlled by the maximum number in a group, $m_{\max}$. The scheme can also be classified as τ-star-based, τ-tree-based, or τ-intermediate depending on $m_{\max}$. The structure is identical to the conventional star-structure when $m_{\max} = 1$, and is identical to the conventional tree-structure when $m_{\max} = N$ (N is the total number of clients). We called the new configuration the *τ-star-based* and the *τ-tree-based scheme* in these cases. Finally, we called it the

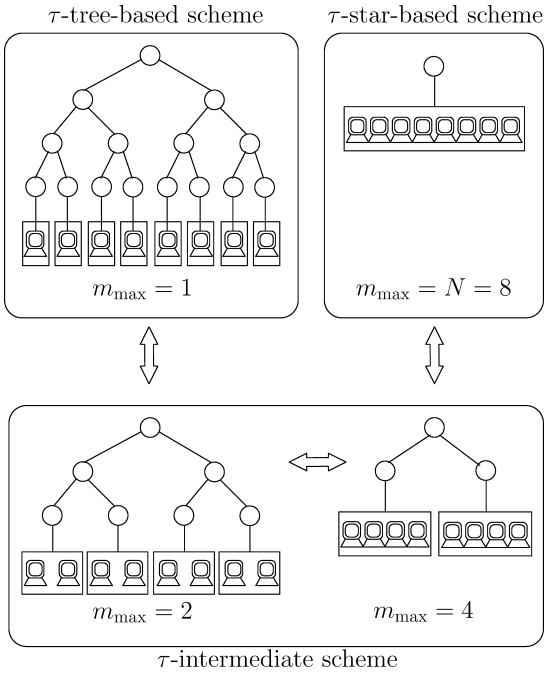

Fig. 1 τ-gradual key-management scheme.

τ-intermediate scheme when $1 < m_{\max} < N$. **Figure 1** shows the relation between these three schemes.

5. Detailed Description of τ-Gradual Key-Management Schemes

We will now give a detailed description of the proposed schemes. They use a logical key-management structure as in the existing configurations. Additionally, each client shares an individual key with the key-management server and trusts this server.

5.1 τ-Star-based Scheme

The key-management server updates the shared key every τ seconds, and it does not update the key when a client joins or leaves.

(1) **Join Process**
The server encrypts the current shared key, K_1, with the individual key of a joining client, and sends it to the client.

(2) **Key Update Process**
The server updates the shared key. The server then encrypts the shared key with each individual key and sends it to each client. (Note that the shared key is not updated when a client joins or leaves.)

5.2 τ-Tree-based Scheme

The key-management server updates the shared key and key-encryption keys every τ seconds, and it does not update the keys when a client joins or leaves.

(1) **Join Process**
The server encrypts the current shared key, K_1, with the individual key of a joining client and sends it to the client.

(2) **Keys Update Process**
The server updates the shared keys and key-encryption keys assigned to the ancestors of the nodes, where the individual keys of clients that have left are assigned.

5.3 τ-Intermediate Scheme

The τ-intermediate scheme consists of the following five processes: preparing, joining, leaving, distributing content, and updating keys.

(1) **Preparation Process** We will now explain the preparation process. The key-management server executes this process only once, i.e., when the key-management scheme is started.

Step 1 Create Groups
The server creates $m_{\max}$-client-groups G_1, G_2, ..., and G_n, where $n = \lceil N/m_{\max} \rceil$.

Step 2 Generate Group Keys
The server generates group keys K_{G_1}, K_{G_2}, ..., and K_{G_n}, which are shared among all the clients in each group. The server encrypts the group keys with each individual key, and sends it to each client.

Step 3 Construct Tree
The server constructs a complete k-ary tree with height $h = \lceil \log_k n \rceil$, then removes $n - k^h$ leaf nodes where $n \neq k^n$. This tree is the key-management tree for the proposed scheme. The server then assigns group keys K_{G_1}, K_{G_2}, ..., and K_{G_n} to the leaf nodes of the tree. The server next generates a shared key, K_1, and key encryption keys K_2, K_3, ..., and K_i. The server assigns the shared key to the root node (node 1), and the key-encryption keys to the interior nodes (nodes 2, 3, ..., i). The shared key and key-encryption key are shared among clients belonging to groups whose group key is assigned to the descendant leaf nodes. The shared key, K_1, is used as the content-encryption key since all the clients share this key.

Step 4 Distribute Keys
The server sends the shared key and the key-encryption key generated in Step 3 to the clients belonging to the groups, G_1, G_2, ..., and G_n. The keys are encrypted with the key-encryption keys or the group keys assigned to the child nodes. The server

then sends them to clients who have the encryption key.

(2) **Joining Process** The server does not update keys but encrypts the shared key, K_1, with the individual key of a joining client, and sends it to the client.

(3) **Leaving Process** The server does not update keys, but sets a revocation flag on the group from which the client left. The flag indicates that the group is going to be revoked in the next key-updating process.

Figure 2 has an example. Let the maximum number of clients in a group, $m_{\max}$, be eight. The server sets the flag on groups G_3, G_5, and G_6 since the numbers of clients in these groups is smaller than $m_{\max}$.

(4) **Content Distribution Process** The server encrypts content with the shared key, K_1, and sends it to all the clients.

(5) **Key Updating Process** The server updates keys every τ ($\tau > 0$) seconds. The server updates the keys according to the following steps:

Step 1 Revoke Groups

The server revokes groups on which flags have been set. The server then discards the group keys of the revoked groups.

Groups G_3, G_5, and G_6 in our example are revoked (see Fig. 2).

Step 2 Reconstruct Groups

The server creates new $m_{\max}$-client-groups from joining clients and clients in the revoked groups. It then generates group keys for the new groups. The server encrypts the group keys with an individual key for each client, and sends it to each client.

There are 21 clients in revoked groups G_3, G_5, and G_6 and two joining clients in Fig. 2. The server creates two 8-client groups, G_9 and G_{10}, and a 7-client group, G_{11}.

Step 3 Reconstruct Tree

The server assigns new group keys to the leaf node of the tree. The new group keys are assigned to nodes where the group keys of the revoked groups were assigned. If no groups were revoked in Step 1, the server adds new leaf nodes to the tree, and assigns the new group key to them. In this case, the server transforms the leaf node with the least depth into an intermediate node. The server then adds two leaf nodes to this intermediate node for the new group and existing groups whose group key was assigned to the group.

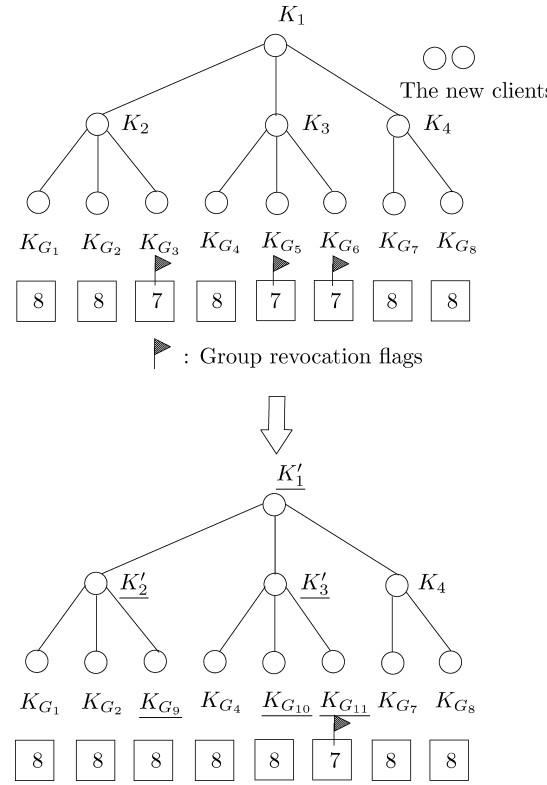

Fig. 2 Key updating process.

In the example in Fig. 2, the server assigns group keys K_{G_9}, $K_{G_{10}}$, and $K_{G_{11}}$ to nodes to which group keys K_{G_3}, K_{G_5}, and K_{G_6} were respectively assigned.

Step 4 Updating Keys

Keys assigned to all the ancestors of the nodes, to which the new group keys were assigned, are updated.

In the example in Fig. 2, the server updates the shared key, K_1, and key encryption keys, K_2 and K_3, since they were assigned to the ancestors of the nodes for the new group keys.

Step 5 Redistributing Keys

The server encrypts the updated keys, and sends them to clients. The updated keys are encrypted with the keys assigned to the child nodes, as in Step 4 of the preparation process.

In the example in Fig. 2, the server encrypts the shared key, K_1', with key-encryption keys, K_2' and K_3', and K_4 and sends them to the clients belonging to groups in sets $\{G_1, G_2, G_9\}$, $\{G_4, G_{10}, G_{11}\}$, and $\{G_7, G_8\}$, respectively.

6. Analysis

We will now present a quantitative method of evaluation for determining the efficiency of the schemes. We will also give an analysis of the security of the τ-gradual key-management scheme.

6.1 Efficiency

We compared the existing and new schemes under the following assumptions:

- The total number of clients was N.
- The average watching time was T.
- N/T clients joined and random N/T clients left per second.
- The degree of the key-management tree was k.
- The maximum number of clients in a group was $m_{\max}$.

We defined the load on the key-management server L_s by the average number of messages that the server issued per second. Next, we defined the load on the clients, L_c, by the maximum number of messages that a client received per second. Each message was an encrypted shared key, an encrypted key-encryption key, or an encrypted group key.

6.1.1 τ-Star-Based Scheme

The key-management server issues messages when a joining client joins and the server updates the shared key.

(1) **Joining Process**

The server issues N/T messages for joining clients per second, and the joining clients receive one message.

(2) **Key Updating Process**

The server issues N messages, and a client receives one message in this process. Thus, the server issues N/τ messages per second, and a client receives $1/\tau$ messages.

Therefore, we have,

$$L_{s,\tau-\mathrm{star}}(N;\tau) = N\left(\frac{1}{T}+\frac{1}{\tau}\right),$$

and,

$$L_{c,\tau-\mathrm{star}}(N;\tau) = \frac{2}{\tau}.$$

6.1.2 τ-Tree-Based Scheme

The key-management server issues messages when a joining client joins or when it updates the shared keys.

(1) **Joining Process**

The server issues N/T messages per second to joining clients, and the joining clients receive one message.

(2) **Leaving Process**

The height of the tree is $\log_k N$. The server must issue updates about $\tau \log_k N$ keys, since $N\tau/T$ clients leave in τ seconds. The server then encrypts each updated key with the k group keys or the k key-encryption keys assigned to the child nodes. The server issues $f(N\tau/T, k, \log_k N)$ messages. Function $f(n;k,h)$ indicates the number of messages that the server issues when n clients leave, where k is the degree of the key-management tree and h is the height of the tree. f is given by

$$f(n;k,h) = \frac{kn-1}{k-1} + nk(h-\log_k n).$$

(See Theorem 1 in Appendix.) That is, the server issues $f(N\tau/T, k, \log_k N)/\tau$ messages per second, and a client whose individual key is assigned to sibling nodes receives $\log_k N$ messages.

Therefore, we have:

$$L_{s,\tau-\mathrm{tree}}(N;k,\tau) = \frac{N}{T} + \frac{f\left(\frac{N\tau}{T}, k, \log_k N\right)}{\tau},$$

and

$$L_{c,\tau-\mathrm{tree}}(N;k,\tau) = \frac{1+\log_k N}{\tau}.$$

6.1.3 τ-Intermediate Scheme

The server issues messages when (1) the joining process and (2) the key updating process (Steps 2 and 4) are executed.

(1) **Joining Process**

The server issues N/T messages to joining clients per second and the joining clients receive one message.

(2) **Key Updating Process**

(a) Reconstruct Groups (Step 2)

Approximately $N\tau/T$ groups are revoked since this process is executed every τ seconds. That is, there are almost $\min(N, Nm_{\max}\tau/T)$ clients in the revoked groups. The server sends new group keys to the clients. Thus, the server issues $\min(N/\tau, Nm_{\max}/T)$ messages per second and the clients in the revoked groups receive one message.

(b) Redistribute Keys (Step 4)

The height of the tree is $\log_k(N/m_{\max})$. The server must issue $f\left(\frac{N\tau}{T}, k, \log_k \frac{N}{m_{\max}}\right)$ messages. Thus, the server issues $\frac{f\left(\frac{N\tau}{T}, k, \log_k \frac{N}{m_{\max}}\right)}{\tau}$ messages per second. Clients whose group keys are assigned to the sibling nodes of the nodes where the group keys of the revoked groups are as-

Table 1 Loads on key-management server and client in key-management schemes.

	Server	Client
Star-based	$\frac{N(N+1)}{T}$	$\frac{2N}{T}$
τ-Star-based scheme	$N\left(\frac{1}{T}+\frac{1}{\tau}\right)$	$\frac{2}{\tau}$
τ-Intermediate	$\frac{N}{T}\left\{1+\min\left(\frac{T}{\tau}, m_{\max}\right)\right\}+\frac{f\left(\frac{N\tau}{T},k,\log_k \frac{N}{m_{\max}}\right)}{\tau}$	$\frac{1+\log_k(N/m_{\max})}{\tau}$
τ-Tree-based	$\frac{N}{T}+\frac{f\left(\frac{N\tau}{T},k,\log_k N\right)}{\tau}$	$\frac{1+\log_k N}{\tau}$
Tree-based	$\frac{N(k+1)\log_k N}{T}$	$\frac{2N\log_k N}{T}$

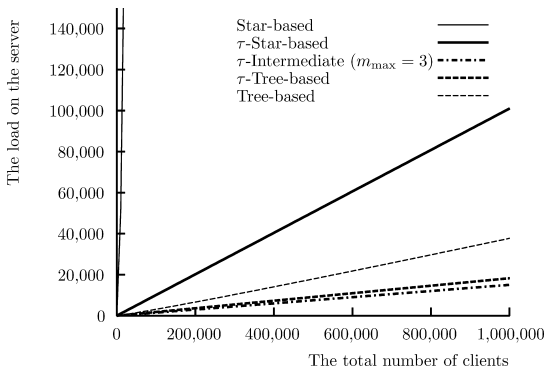

Fig. 3 Load on key-management server (where $k = 2$ and τ=10).

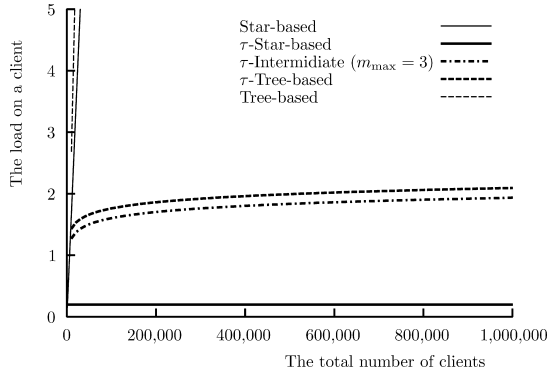

Fig. 4 Load on client (where $k = 2$ and τ=10).

signed, receive $\log_k(N/m_{\max})$ messages. Therefore, we have

$$L_{s,\mathrm{inter}}(N; m_{\max}, k, \tau) = \frac{N}{T}\left\{1+\min\left(\frac{T}{\tau}, m_{\max}\right)\right\} + \frac{f(\frac{N\tau}{T}, k, \log_k \frac{N}{m_{\max}})}{\tau}$$

and

$$L_{c,\mathrm{inter}}(N; m_{\max}, k, \tau) = \frac{1+\log_k(N/m_{\max})}{\tau}.$$

We have listed our results in **Table 1**; **Fig. 3** and **Fig. 4** show the loads on the server and client.

The loads in the τ-intermediate scheme are almost identical to those in the star-based one where $m_{\max} = N$, and are almost identical to those in the tree-based scheme where $m_{\max} = 1$. A trade-off has to be made in the τ-intermediate scheme between imposing a load on the server or the clients. We can flexibly adjust the load on the server and the clients with the τ-intermediate scheme while maintaining security. For example, we can reduce the load on clients by assigning a large value to $m_{\max}$, where the server has high capacity. We can reduce the load on the server by assigning a small value to $m_{\max}$, where the capacity of the clients is high. Furthermore, the loads on both the server and clients with the τ-intermediate scheme are lower than those with the tree-based scheme if:

$$\begin{cases} m_{\max} = 3 & (k = 2, 4) \\ m_{\max} = 2, 3, \ldots, k-1 & (5 \le k \le T/\tau) \end{cases};$$

(see Theorem 2). We have listed an example in **Table 2**. The parameters N, τ, k, and T were set based on actual content-distribution services for mobile content (see **Table 3**).

6.2 Security

A joining client receives the current shared key and the current key-encryption keys from the server. Since the keys are updated every τ seconds, the worst-case scenario is that the joining client can decrypt a message at τ seconds before he or she joins. However, the client cannot obtain an older key from the old message since the current key is encrypted with a newer key. However, the client cannot obtain an older key from the old message, since this message only contains the current key. Thus, the new schemes satisfy τ-gBS.

The worst-case scenario is that a leaving client can decrypt a message at τ seconds after he leaves. However, the group that the leaving client belonged to is revoked during the next

Table 2 Trade-off between load on key-management server and client.

	Server	Client
Star-based	1,000,000,000	2,000
τ-Star-based	101,000	0.20
τ-Intermediate ($m_{\max} = 1{,}000$)	98,400	1.10
τ-Intermediate ($m_{\max} = 500$)	100,000	1.20
τ-Intermediate ($m_{\max} = 200$)	103,000	1.33
τ-Intermediate ($m_{\max} = 100$)	105,000	1.43
τ-Intermediate ($m_{\max} = 50$)	57,000	1.53
τ-Intermediate ($m_{\max} = 20$)	29,600	1.66
τ-Intermediate ($m_{\max} = 10$)	21,600	1.76
τ-Intermediate ($m_{\max} = 5$)	18,600	1.86
τ-Intermediate ($m_{\max} = 3$)	18,100	1.93
τ-Intermediate ($m_{\max} = 2$)	18,300	1.99
τ-Tree-based	18,300	2.09
Tree-based	79,700	39,900

Table 3 Parameter setting based on mobile content-distribution services.

N (total number of clients)	1,000,000
τ (Security parameter [s])	10
T (Average watching time [s])	1,000
k (Degree of key-management tree)	2

key updating process and the group key is updated. The leaving client cannot decrypt the message that includes the new shared key and the new key-encryption keys. Thus, the new schemes satisfy τ-gFS.

As a result, the three schemes satisfy τ-gBS and τ-gFS defined in Section 3.

7. Discussion

We will now explain how to construct an optimal key structure.

The parameter τ must be set by the content provider, since this indicates the period that a client can watch content free of charge. For example, the content provider can set τ to a few seconds or a few tens of seconds. The content provider will not be overly concerned if clients watch free content for periods exceeding a few tens of seconds for a service that charges for every minute.

Thus, the key distributor can choose an optimal key-management scheme according to system requirements while maintaining security. The security of the τ-gradual key-management scheme is only dependent on parameter τ. We will describe a concrete strategy. The load on clients can be reduced by assigning a large value to $m_{\max}$, and the load on the server can be reduced by assigning a small value to $m_{\max}$. Furthermore, the loads imposed on both the server and client by the τ-intermediate scheme are lower than those by the τ-tree-based scheme if

$$\begin{cases} m_{\max} = 3 & (k = 2, 4) \\ m_{\max} = 2, 3, \ldots, k-1 & (5 \le k \le T/\tau) \end{cases}$$

8. Conclusion

We considered efficient key-management schemes for mobile devices. The conventional star-based scheme is far from being scalable and the conventional tree-based one does not take the load on clients into account. The latter is not ideal for devices with low computational capacity. We relaxed the conventional security requirements for key-management schemes, and defined new requirements, ε_1-gBS and ε_2-gFS, to improve their efficiency. We then proposed a τ-gradual key-management scheme. Our scheme satisfies the new security requirements, τ-gBS and τ-gFS, and the loads imposed on the server and the clients by this are much lower than those imposed by conventional star-based and tree-base schemes. It uses an intermediate configuration between that of a tree- and star-structure, and can be controlled by the number of clients in a group, $m_{\max}$. The new scheme can be classified as τ-star-based, τ-tree-based, or τ-intermediate by changing the parameter, $m_{\max}$. We presented a quantitative evaluation of the load imposed on the server and clients by our schemes based on practical assumptions. The load on the server and that on clients involves a trade-off using the τ-intermediate scheme. We can construct an optimal key-management structure according to the system requirements using our schemes, while maintaining security. We described a concrete strategy for setting parameter $m_{\max}$. We also presented general parameter settings in which the loads imposed on both the server and clients

by the τ-intermediate scheme were lower than those by the τ-tree-based one.

References

1) Asano, T.: A Revocation Scheme with Minimal Storage at Receivers, *Advances in Cryptology — ASIACRYPT 2002, Lecture Notes in Computer Science*, Vol.2501, pp.433–450 (2002).
2) Asano, T.: Reducing Storage at Receivers in SD and LSD Broadcast Encryption Schemes, *Information Security Applications, 4th International Workshop, WISA 2003, Lecture Notes in Computer Science*, Vol.2908, pp.317–332 (2003).
3) Asano, T.: Secure and Insecure Modifications of the Subset Difference Broadcast Encryption Scheme, *Information Security and Privacy: 9th Australasian Conference, ACISP 2004, Lecture Notes in Computer Science*, Vol.3108, pp.12–23 (2004).
4) Asano, T.: Reducing Receiver's Storage in CS, SD, and LSD Broadcast Encryption Schemes, *IEICE Trans.*, Vol.E88-A, No.1, pp.203–210 (2005).
5) Berkovits, S.: How To Broadcast A Secret, *Advances in Cryptology — EUROCRYPT '91, Lecture Notes in Computer Science*, Vol.547, pp.535–541 (1991).
6) Burmester, M.: On the Risk of Opening Distributed Keys, *Advances in Cryptology — CRYPTO '94, Lecture Notes in Computer Science*, Vol.839, pp.308–317 (1994).
7) Chang, I., Engel, R., Kandlur, D., Pendarakis, D. and Saha, D.: Key Management for Secure Internet Multicast using Boolean Function Minimization Techniques, *Proc. IEEE Infocomm '99*, Vol.2, pp.689–698 (1999).
8) Cheon, J.H., Jho, N., Kim, M.-H. and Yoo, E.S.: Skipping, Cascade, and Combined Chain Schemes for Broadcast Encryption, Cryptology ePrint Archive, Report 2005/136 (2005). http://eprint.iacr.org/
9) Dodis, Y. and Fazio, N.: Public-Key Broadcast Encryption for Statless Receivers, *Digital Rights Management* (*DRM '02*), *Lecture Notes in Computer Science*, Vol.2696 (61–80).
10) Fiat, A. and Naor, M.: Broadcast Encryption, *Advances in Cryptology — CRYPTO '93, Lecture Notes in Computer Science*, Vol.773, pp.480–491 (1994).
11) Halevy, D. and Shamir, A.: The LSD Broadcast Encryption Scheme, *Advances in Cryptology — CRYPTO 2002, Lecture Notes in Computer Science*, Vol.2442, pp.47–60 (2002).
12) Kim, Y., Perrig, A. and Tsudik, G.: Simple and fault-tolerant key agreement for dynamic collaborative groups, *ACM Conference on Computer and Communications Security*, pp.235–244 (2000).
13) Naor, D., Naor, M. and Lotspiech, J.B.: Revocation and Tracing Schemes for Stateless Receivers, *Advances in Cryptology — CRYPTO 2001, Lecture Note in Computer Science*, Vol.2139, pp.41–62, Springer-Verlag, London, UK (2001).
14) Naor, M. and Pinkas, B.: Efficient Trace and Revoke Schemes, *Financial Cryptography 2000, Lecture Notes in Computer Science*, Vol.1962, pp.1–20 (2000).
15) Pinkas, B.: Efficient State Updates for Key Management, *Workshop on Security and Privacy in Digital Rights Management 2001, Lecture Notes in Computer Science*, Vol.2320, pp.40–56 (2001).
16) Wallner, D., Harder, E. and Agee, R.: Key Management for Multicast: Issues and Architectures, RFC 2627 (Informational) (1999). http://www.ietf.org/rfc/rfc2627.txt
17) Wong, C.K., Gouda, M.G. and Lam, S.S.: Secure Group Communications Using Key Graphs, *Proc. ACM SIGCOMM '98 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pp.68–79 (1998).

Appendix

Theorem 1. *The server issues at most $f(n;k;h)$ messages in the key updating process when n keys assigned to leaf nodes are updated. k is the degree of the key-management tree and h is the height of the tree. $f(n;k,h)$ is given by:*

$$f(n;k,h) = \frac{k(kn-1)}{k-1} + nk(h - \log_k N).$$

Proof. The locations of the revoked groups are random since clients randomly leave the service. The shared key and all the key encryption keys assigned to the node in the upper $\log_k n+1$ layers of the tree can be updated. At most, n keys are updated in each of the last layers. **Figure 5** shows how many keys can be updated. In the upper $\log_k n + 1$ layers, at most

$$1 + k + \ldots + n = \frac{kn-1}{k-1}$$

keys are updated, and in the last $h - \log_k n$ layers, at most $n(h - \log_k n)$ keys are updated. The server encrypts the updated keys with the k keys assigned to the child nodes. Therefore, the server issues at most

$f(n;k,h)$

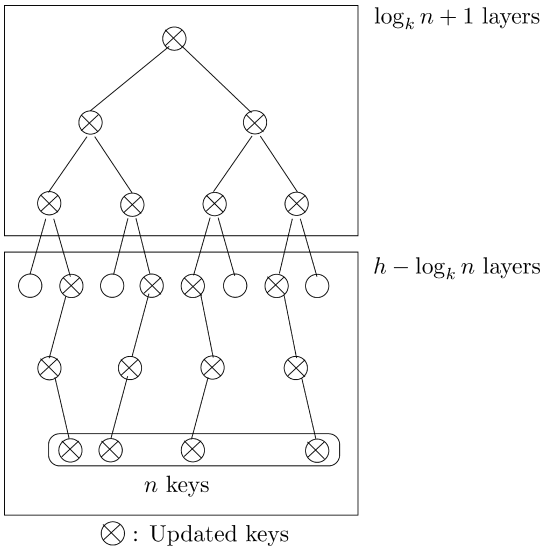

Fig. 5 Keys that can be updated in key-management tree.

$$= k\left\{\frac{kn-1}{k-1} + n(h - \log_k n)\right\}$$
$$= \frac{k(kn-1)}{k-1} + nk(h - \log_k n)$$

messages. □

Theorem 2. *The loads imposed on both the server and a client by the τ-intermediate scheme are lower than those by the τ-tree-based scheme if*

$$\begin{cases} m_{\max} = 3 & (k = 2, 4) \\ m_{\max} = 2, 3, \ldots, k-1 & (k \geq 5) \end{cases}.$$

Proof. The load imposed on a client by the τ-intermediate scheme is always lower than that by the τ-tree-based scheme since:

$$D_c(m_{\max}; k)$$
$$= L_{c,\tau-\text{tree}} - L_{c,\text{inter}}$$
$$= \frac{1 + \log_k N}{\tau} - \frac{1 + \log_k \frac{N}{m_{\max}}}{\tau}$$
$$= \frac{\log_k m_{\max}}{\tau} > 0.$$

We then investigate the load on the server. The difference between the loads imposed by the τ-tree-based scheme and the τ-intermediate is given by:

$$D_s(m_{\max}; k)$$
$$= L_{c,\tau-\text{tree}} - L_{c,\text{inter}}$$
$$= \left[\frac{N}{T}\{1 + m_{\max}\} + \frac{f(\frac{N\tau}{T}, k, \log_k \frac{N}{m_{\max}})}{\tau}\right]$$
$$- \left[\frac{N}{T} + \frac{f(\frac{N\tau}{T}, k, \log_k N)}{\tau}\right]$$
$$= \frac{N}{T}(k \log_k m_{\max} - m_{\max}).$$

$D_s(3; k) = \log_2 \frac{9}{8} > 0$, where $k = 2, 4$, and, $D_s(2; k) = \log_k \frac{2^k}{k^2} > 0$, $D(3; k) > 0$, ..., and, $D_s(k-1; k) = \log_k \frac{(k-1)^k}{k^{k-1}} > 0$, where $k \geq 5$. Therefore, the load imposed on the server by the τ-intermediate scheme is lower than by the τ-tree-based scheme, if:

$$\begin{cases} m_{\max} = 3 & (k = 2, 4) \\ m_{\max} = 2, 3, \ldots, k-1 & (5 \leq k \leq T/\tau) \end{cases}.$$

We assumed $m_{\max} < k \leq T/\tau$ in this theorem. The assumption is not valid if $k > T/\tau$. However, this case is not realistic, since T is much greater than τ. Note that T is the average time for watching, for example 1000 s, and τ is the period that the content provider allows a client to watch content for free, e.g., a few tens of seconds. □

(Received March 3, 2006)
(Accepted October 3, 2006)
(Online version of this article can be found in the IPSJ Digital Courier, Vol.2, pp.792–803.)

Kazuhide Fukushima received his M.E. in Information Engineering from Kyushu University, Japan, in 2004. He joined KDDI and has been engaged in research on digital rights management technologies, including software obfuscation and key-management schemes. He is currently a researcher at the Information Security Lab. of KDDI R & D Laboratories Inc. He is a member of IEICE and ACM.

Shinsaku Kiyomoto received his B.E. in Engineering Sciences and his M.E. in Materials Science from Tsukuba University, Japan, in 1998 and 2000. He joined KDD (now KDDI) and has been engaged in research on stream ciphers, cryptographic protocols, and mobile security. He is currently a research engineer at the Information Security Lab. of KDDI R & D Laboratories Inc. He received his Doctorate in Engineering from Kyushu University in 2006. He received the Young Engineer Award from IEICE in 2004. He is a member of JPS and IEICE.

Toshiaki Tanaka received his B.E. and M.E. in Communication Engineering from Osaka University, Japan, in 1984 and 1986. He joined KDD (now KDDI) and has been engaged in research on cryptographic protocols, mobile security, digital rights management, and intrusion detection. He is currently a senior manager at the Information Security Lab. of KDDI R & D Laboratories Inc. He is a member of IEICE.