IPSJ Journal Vol. 49 No. 9 2963–2974 (Sep. 2008)

Regular Paper

Optimization of Group Key Management Structure with a Client Join-Leave Mechanism

Kazuhide Fukushima,[†1] Shinsaku Kiyomoto,[†1]
Toshiaki Tanaka[†1] and Kouichi Sakurai[†2]

Many group key management schemes that reduce the total communication cost and/or the computational cost imposed on client devices have been proposed. However, optimizations of the key-management structure have not been studied. This paper proposes ways to optimize the key-management structure in a hybrid group key management scheme. The proposed method is able to minimize both the total communication cost and the computational cost imposed on client devices. First, we propose a probabilistic client join/leave model in order to evaluate the communication and computational costs of group key management schemes. This model idealizes client actions generally and considers the existence of the peaks of the joining/leaving frequency. Thus, we can analyze not only the average case scenario but also the worst case scenario using this model. Then, we formalize the total computation cost and the computational cost imposed on client devices in group key management schemes under the model. We present both an average case analysis and a worst case analysis. Finally, we show the parameters that minimize the total communication cost and the computational cost imposed on clients under the model. Our results should be useful in designing a secure group communication system for large and dynamic groups.

1. Introduction

1.1 Background

Recently, high-speed Internet has expanded to include 3G mobile services and a FTTH service for personal computers. Multicast services such as content distribution, pay-per-view, online auction, internet relay chat, and video conferencing have become common, and these services may require some form of secure communication. *Group key management schemes* provide a secure communication by sharing the group key between client devices. The center must update the group key and securely distribute it when a client joins or leaves in order to maintain the security. We call this process *rekeying*. In large-scale services, joins and leaves of clients occur frequently and the total number of managed keys is enormous while the network bandwidth and the computational capacity on client devices are restricted. Therefore, optimization of rekeying is an important and challenging issue.

Broadcast encryption [1),8),9)] is another approach whereby the same key (the session key) can be shared for secure communication. The session key is based only on the message from the center and the pre-shared keys of each client device that are never updated. Thus, only revocation is possible, and a client cannot join the system dynamically. On the other hand, keys are updated in group key management schemes [3),10),13),14)]. The updated group key is derived using the encrypted keys received from the center and the current keys. Both joining and leaving of client devices are realizable by updating the group key. Thus, group key management schemes are suitable for dynamic systems in which clients become alternately active and inactive.

1.2 Related Work

Group Key Management Schemes

The simplest group key management scheme is the star-based scheme [11),14)]. In this scheme, the computational cost imposed on client devices is low. Each client device receives only one encrypted key when a client joins or leaves. However, the total communication cost is too high. The center issues M (the total number of active clients) distinct encrypted keys when a client leaves, while it issues two encrypted keys when a client joins.

Wong, et al. [14)] and Wallner, et al. [13)] proposed a tree-based scheme which uses a logical k-ary tree for the key allocation to the client devices. In the scheme, key encryption keys, which are shared by multiple client devices, reduce the total communication cost when a client leaves. The center issues $2\log_k M$ and $k\log_k M$ distinct encrypted keys when a client joins and leaves, respectively. On the other hand, the computational cost imposed on a client is higher than that in the star-based scheme. Each client device receives $\log_k M$ encrypted keys when a client joins or leaves.

†1 KDDI R&D Laboratories, Inc.
†2 Kyushu University

We proposed a hybrid group key management scheme (τ-gradual key-management scheme) [5),6)] in order to solve the problems associated with the star-based and tree-based schemes. The proposed scheme uses a hybrid structure derived from the star and tree schemes that can be controlled by several parameters and satisfies the relaxed security requirement. The hybrid structure generalizes key-management structures in existing schemes. Both the total communication cost and the computational cost imposed on the client are much lower than those imposed by the star-based and tree-based schemes. However, ways of minimizing the communication cost and the computational cost based on various system restrictions (such as network bandwidth and computational capacity of client devices) are desirable in real services. We did not discuss the optimization problem, which remains an open issue.

Optimization Schemes

Chang, et al. [3)] and Pinkas, et al. [10)] proposed an improved tree-based scheme. They reduced the number of updated keys when multiple clients join or leave. Chang, et al. used a Boolean minimization technique and Pinkas, et al. used a pseudo-random generator. Zhu, et al. [16)] proposed a partitioned key-management tree scheme that leverages the temporal patterns of client joinings/leavings in order to reduce the overhead of rekeying. Aurisch [2)] proposed an optimization method based on batched rekeying and probabilistic tree construction. Eltoweissy, et al. [4)] proposed an optimization method based on a combinational approach. Zhang [15)] proposed a key agreement scheme for dynamic collaborative groups. They reduce rekeying cost by using a batched rekeying and a tree balancing algorithm. Nevertheless, this scheme is based on a distributed approach with public key cryptography, in which clients must agree on a common key without the center by repeating Diffie-Hellman key exchange. Thus, a key management structure is difficult to be optimized since the structure is dynamically constructed by each client device. This approach is suitable for distributed and collaborative network applications such as in internet relay chat and video conferencing. On the other hand, our notation of group key management scheme (see Section 2.1) assumes the existence of the key management center. Thus, the center can determine the key management structure in order to optimize the key management cost. This is suitable for a centralized network application such as in content distribution services and pay-per-view services.

However, optimization methods that construct a key-management structure based on boundary conditions (e.g., the network bandwidth and computational capacities on client devices) have not been proposed. An optimization method based on key-management structure, which is the most basic component in a group key management scheme, is required for a further reduction of the total communication cost and the computational cost imposed on client devices. Furthermore, a client join/leave model is required in order to evaluate the efficiency of a group key management scheme in a worst case scenario.

1.3 Our Contribution

We propose a probabilistic client join/leave model in order to evaluate the communication and computational costs of group key management schemes. This model idealizes client actions generally and considers the existence of the peaks of the joining/leaving frequency. We can analyze not only the average case scenario but also the worst case scenario using this model; thus, our model is more generic than existing probabilistic models. We then propose an optimization method for our proposed group key management scheme, namely, the hybrid scheme. The optimization method can minimize both the total communication cost and the computational cost imposed on client devices by changing the key-management structure of the hybrid scheme. These methods are useful for designing a secure communication system for multicast services. For example, the minimization method for reducing the total communication cost is suited to services that require a high network bandwidth (e.g., pay-per-view). Furthermore, the minimization method for reducing the computational cost is suited to services for devices with low computational capacity (e.g., content distribution services for mobile devices).

First, we introduce a probabilistic client join/leave model in order to quantitatively evaluate group key management schemes. Then, we formalize the total computation cost and the computational cost imposed on client devices in group key management schemes under the model. We provide both an average case and worst case analysis. Finally, we present optimal key-management structures that minimize the total communication cost and computational cost in the hybrid scheme based on an analysis of the worst case scenario.

Our optimization method focuses on the key-management structure, which is the most basic component in a group key management scheme. Thus, other optimization schemes in Section 1.2 are available for further optimization. For this reason, we discuss only the key-management structure.

2. Group Key Management Scheme

2.1 Preliminary Notations

Let $\mathcal{N}$ be the set of all the client devices, $|\mathcal{N}| = N$. Each client device shares an individual key with the center in advance. $\mathcal{M}(\subset \mathcal{N})$ denotes the set of active client devices, $|\mathcal{M}| = M$. Group key management schemes enable the center to transmit a message to all the client devices, such that all devices in $\mathcal{M}$ can decrypt the message correctly and any coalition of client devices in $\mathcal{N}\backslash\mathcal{M}$ cannot decrypt it. Client devices in $\mathcal{M}$ share the group key that is used to encrypt the message. An active client device may dynamically change to being inactive, and vice versa. When a client device joins or leaves, the center updates the group key to prevent the joining device from obtaining the former messages and the leaving device from obtaining the latter messages.

2.2 Hybrid Scheme

The scheme uses a hybrid structure derived from star and tree schemes. Each client device belongs to a subgroup and shares the subgroup key with other client devices in the subgroup. While the subgroup key is managed using a star-based scheme, the group key is managed using a tree-based scheme. A key-management tree, where individual keys are replaced with subgroup keys, is used. The structure used in the hybrid scheme is controlled by the degree of the tree, k, and the maximum number in a subgroup, m, and we find the optimal setting for these. The structure is identical to the star-structure when $m = M$, and identical to the tree-structure when $m = 1$. **Figure 1** shows the relation between the three schemes.

Furthermore, the hybrid scheme uses batched rekeying. The center updates the group key and distributes it for every τ seconds instead of when a client joins or leaves.

A detailed description of the hybrid scheme is given in the appendix.

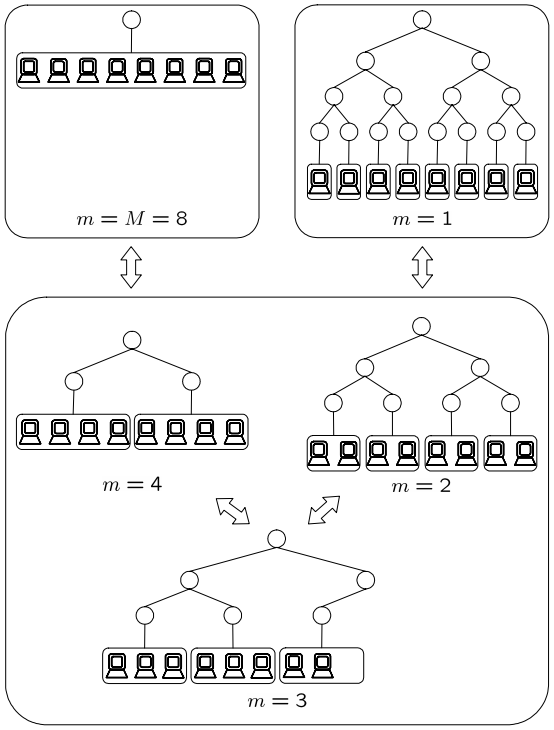

Fig. 1 Structure of the hybrid scheme.

3. Optimization Method

We optimize the hybrid scheme. We assume that the total communication cost and the computational cost imposed on client devices in group key management schemes depend on the following five factors:

- The total number of clients M
- The average service usage time T
- The duration of batched rekeying τ
- The degree of the key-management tree k
- The maximum number of client devices in a subgroup m

We need to minimize the total communication cost or the computational cost

imposed on client devices since the communication bandwidth and the computational capacity are limited. For example, the computational capacity of client devices is highly limited in a content service for mobile devices. In this case, we should minimize the computational cost. On the other hand, client devices have extra computational capacity in a content service for personal computers. In this case, we can minimize the total communication cost by sharing some processes between the center and client devices. Thus, we should reduce the total communication cost while maintaining a feasible computational cost imposed on client devices, or reduce the computational cost while maintaining a feasible total communication cost. Note that the change in storage data size is much smaller than the total communication cost and the computational cost imposed on clients (see Table 3).

The key distributor cannot control the parameters M, T and τ. M and T depend on a client's action, and the parameter τ depends on the policies of the content provider. The distributor can control the parameters k, m. Here, we consider the following three factors:

(1) **The duration of batched rekeying τ**
Since this parameter indicates the period that a client can watch content without paying, it should be controlled by the content provider. For example, the content provider can set τ to a few seconds or a few tens of seconds. The content provider allows clients to watch content for periods without paying for a few tens of seconds in a service that charges for every minute.

(2) **The degree of the tree k**
A key distributor can select the parameter when the key-management tree is constructed.

(3) **The number of client devices m**
A key distributor can control this parameter at any time. The key-management structure does not change. The center has only to change the number of client devices in a new subgroup that is generated after this parameter is changed.

First, we define a probabilistic client join/leave model in Section 4 in order to quantitatively evaluate the total communication cost and the computational cost imposed on client devices in group key management schemes. The evaluations are shown in Section 5. Then, we optimize the key-management structure in the hybrid scheme based on the result of the analysis of a worst case scenario in Section 6.

4. A Probabilistic Client Join/Leave Model

We introduce a probabilistic client join/leave model in order to quantitatively evaluate group key management schemes under the same model. The model is that of general broadcasting services where clients randomly join or leave and satisfies the following conditions:

- The total number of active clients is constant, namely M,
- A join of a client is according to Poisson process with mean $\lambda_{ave} = M/T$,
- A leave of a client is according to Poisson process with mean $\lambda_{ave} = M/T$.

Our model is based on the Poisson process which is widely used to model random events. Tainaka, et al.[12] also evaluated their group key management scheme using a join-leave model based on the Poisson process. The Poisson process has a parameter that denotes the expected number of events (i.e., joins or leaves) that occur per second. We determine this parameter by Little's law[7]. The law says that the average number of clients in a stable service M equals their average arrival rate λ multiplied by the average service usage time T, or $M = \lambda T$. Little's law can be applied to various systems where the number of client devices is stable. For example, it is used to estimate the average waiting time of customers and determine the number of tellers in a bank. We consider our model is reasonable if our assumption is satisfied.

Note that our model cannot be applied to cases where client actions are not random or the number of client devices is not stable. The Poisson process is not applicable in the former case, and Little's law does not work in the latter case. We need another evaluation model in these cases.

5. The Efficiency of the Hybrid Scheme

We evaluate the total communication cost and the computational and storage data size imposed on client devices in the hybrid scheme in order to determine the optimal settings.

5.1 Definitions

We define the total communication cost *Comm* [keys/second] by the average

number of distinct encrypted keys that the center issues per second. The computational cost imposed on client devices *Comp* [keys/second] is defined by the maximum number of keys that a client device receives per second. Additionally, the storage data size *Stor* [keys] is defined by the number of keys that a client device must hold.

For simplicity, we evaluate the total communication cost and the computational cost imposed on client devices under the following deterministic system that satisfies the following conditions:

- A join of a client occurs λ times per second,
- A leave of a client occurs λ times per second,
- The total number of active client devices increases from M to $M+1$ when a client joins,
- The total number of active client devices decrease from $M+1$ to M when a client leaves.

The deterministic system approximates our probabilistic model. We then analyze the average case where $\lambda = \lambda_{ave}$ and the worst case where $\lambda = \lambda_{max}$.

Average Case Analysis

The frequency of joining and leaving of clients is $\lambda_{ave} = M/T$ in the average case. Therefore, we can obtain the communication cost and the computational cost in the average case by replacing λ in the following results with M/T.

Worst Case Analysis

We evaluate the total communication cost and the computational cost in the worst case using a probabilistic method. We define the peak of a probabilistic variable X at confidence p by the infimum of x such that

$\Pr[X \leq x] \geq p$.

The peak of X at confidence p is the minimal x such that

$$\sum_{k=0}^{x} \frac{e^{-\lambda_{ave}} \lambda_{ave}^{k}}{k!} \geq p,$$

if X corresponds to the Poisson distribution with mean λ_{ave}. We denote the peak by λ_{max}. **Table 1** shows λ_{max} (at confidence 0.999) corresponding to each λ_{ave}. We can obtain the communication cost and the computational cost in the worst case by replacing λ in the following results with $\lambda_{\max}$.

Table 1 The peaks at confidence 0.999 corresponding to each mean.

λ_{ave}	λ_{max}
10	21
20	35
50	73
100	132
200	245
500	571
1,000	1,099
2,000	2,140
5,000	5,221

The Poisson distribution with mean λ_{ave} is approximated by the normal distribution with mean λ_{ave} and standard deviation $\sqrt{\lambda_{ave}}$. Thus, the peak at confidence p is approximated by:

$$\lambda_{max} \sim \lambda_{ave} + \alpha\sqrt{\lambda_{ave}} \quad (\text{for } \lambda_{ave} \gg 1),$$

where α satisfies

$$\int_{-\infty}^{\alpha} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right) dx = p$$

and can be found from the standard normal distribution table.

5.2 Evaluation of the Hybrid Scheme

We analyze the communication cost, the computational cost and the storage size of the hybrid scheme for each case $m = M$, $1 < m < M$ and $m = M$.

Case 1: Where $m = M$

The center issues encrypted keys when a client joins and the center updates the group key. A client device must hold the group key and the individual key, thus $Stor = 2$.

(1) **Join Process**

The center issues the encrypted group key, and the joining client device receives the key. Thus, the center issues λ keys per second.

(2) **Rekeying Process**

The center issues M keys, and a client device receives a key in this process. Thus, the center issues M/τ distinct encrypted keys, and a client device receives $1/\tau$ keys per second since this process is executed for every τ seconds.

Therefore, we have

$$Comm = \lambda + \frac{M}{\tau},$$

and

$$Comp = 1 + \frac{1}{\tau}.$$

Case 2: Where $1 < m < M$

The center issues encrypted keys when (1) the join process and (2) the rekeying process (Step 2 and Step 4) are executed. A subgroup from which a client device leaves is revoked, and new m-client-device-subgroups are reconstructed. Thus, the number of subgroups is M/m in the steady state of the model. A client device must hold all the keys assigned to the ancestor nodes (including the subgroup key assigned to the leaf nodes) and the individual key, thus $Stor = \lceil \log_k(M/m) \rceil + 1$. For real x, $\lceil x \rceil$ denotes the minimal integer that is not less than x.

(1) **Join Process**

The center issues the encrypted group key, and the joining client device receives the key. Thus, the center issues λ keys per second.

(2) **Rekeying Process**

The keys are updated in Step 2 and Step 4.

(a) **Reconstruct Subgroups (Step 2)**

At most $\lambda\tau$ subgroups are revoked since this process is executed every τ seconds. That is, there are at most $\min(m\lambda\tau, M)$ client devices in the revoked subgroups. The center sends new subgroup keys to the client devices. Thus, the center issues $\min(m\lambda, M/\tau)$ distinct encrypted keys per second. The client devices in the revoked subgroups receive one key. That is, client devices receive at most $1/\tau$ keys per second.

(b) **Redistribute Keys (Step 4)**

The height of the tree is $\log_k(M/m)$. The center must issue $k \log_k(M/m) \cdot \lambda\tau$ distinct encrypted keys. Thus, the center issues $\lambda k \log_k(M/m)$ distinct encrypted keys per second.

Some client devices receive $\log_k(M/m)$ keys. The subgroup keys of these devices are assigned to the sibling nodes of the nodes where the subgroup keys of the revoked subgroups are assigned. That is, each client device receives at most $\log_k(M/m)/\tau$ keys per second.

Therefore, we have

$$Comm = \lambda[1 + \min(m, M/(\lambda\tau)) + k \log_k(M/m)] \quad (1)$$

and,

$$Comp = 1 + \frac{1 + \log_k(M/m)}{\tau}. \quad (2)$$

Case 3: Where $m = 1$

The center issues encrypted keys when a client joins or when it updates the group keys. A client device must hold all the keys assigned to the ancestor nodes (including the individual key), thus

$Stor = \lceil \log_k M \rceil$.

(1) **Join Process**

The center issues the encrypted group key, and the joining client device receives the key. Thus, the center issues λ keys per second.

(2) **Rekeying Process**

$\lambda\tau$ clients leave since this process is executed every τ seconds. Additionally, the height of the tree is $\log_k M$. The center issues at most $k \log_k M \cdot \lambda\tau$ distinct encrypted keys, and the client devices whose individual key is assigned to the sibling nodes receive $\log_k M$ keys. That is, the center issues at most $\lambda k \log_k M$ distinct encrypted keys per second, and a client device receives at most $(\log_k M)/\tau$ keys per second.

Therefore, we have

$$Comm = \lambda(1 + k \log_k M),$$

and,

$$Comp = 1 + \frac{\log_k M}{\tau}.$$

The results are shown in **Table 2**. The τ-intermediate scheme represents a trade-off between the total communication cost and the computational and storage costs imposed on client devices. We show an example in **Table 3**.

In the hybrid scheme, we can flexibly adjust the total communication cost along with the computational and storage costs while maintaining security. For example, we can reduce the total communication cost imposed on client devices by assigning a small value to m, where the computational capacity of client devices is high. We provide details in Section 6.

Table 2 The total communication cost and computational and storage data size imposed on client devices.

m	*Comm*	*Comp*	*Stor*
$m = M$	$\lambda + M/\tau$	$1 + 1/\tau$	2
$1 < m < M$	$\lambda[1 + \min(m, M/(\lambda\tau)) + k \log_k(M/m)]$	$1 + [1 + \log_k(M/m)]/\tau$	$\log_k(M/m) + 1$
$m = 1$	$\lambda(1 + k \log_k M)$	$1 + \log_k M/\tau$	$\log_k M$

Table 3 The total communication cost and computational and storage costs imposed on client devices (where $M = 10{,}000$, $T = 100$, $\tau = 1$, and $k = 2$).

m	*Comm*	*Comp*	*Stor*
10,000	10,100	2.00	2
1,000	10,800	5.32	5
500	11,000	6.32	6
200	11,300	7.64	7
100	11,400	8.64	8
50	6,630	9.64	9
20	3,890	11.0	10
10	3,090	12.0	11
5	2,790	13.0	12
1	2,760	14.3	14

6. Optimization of the Hybrid Scheme

We optimize the hybrid scheme based on results of the worst case analysis. First, the total communication cost is minimal while the computational cost imposed on client devices is lower than their computational capacity C [keys/second] (We assume all the client devices have computational capacities greater than C). Then, the computational cost is minimal while the total communication cost is lower than the network bandwidth Bd [keys/second]. Optimal settings for the degree of the key-management tree, k, and the maximum number of client devices in a subgroup, m, are demonstrated.

6.1 Optimization of the Communication Cost

The key distributor should select the smallest m such that the computational cost imposed on each client device is lower than C. The optimization method is shown below:

Case 1: Where the computational capacity is too low

In this case, the equation

$$C < \frac{1}{\tau}$$

holds. The group key management scheme cannot be applied since the computational capacity of client devices is too low. The key distributor must negotiate with the content provider to set a larger value for τ.

Case 2: Where the computational capacity is moderate

In this case, the equation

$$\frac{1}{\tau} \leq C < 1 + \frac{\log_2 M}{\tau}$$

holds. The total communication cost is lowest when the computational cost imposed on client devices equals their computation capacity, C, i.e.

$$C = 1 + \frac{1 + \log_k(M/m)}{\tau}. \tag{3}$$

Then, we have

$$m(k) = \frac{M}{k^{(C-1)\tau - 1}} \tag{4}$$

from Eq. (3). Next, we substitute Eq. (4) for Eq. (1) and obtain

$$Comm(k) = \lambda\{1 + m(k) + k \log_k[M/m(k)]\}. \tag{5}$$

The key distributor should select k such that *Comm* is minimal.

Case 3: Where the computational capacity is high enough

In this case, the equation

$$C \geq 1 + \frac{\log_2 M}{\tau}$$

holds. The computational capacity of client devices is high enough in this case. The key distributor should set $k = 3$ and $m = 1$; i.e. the distributor should use the ternary tree structure (Note that the communication cost is minimal in the ternary tree structure i.e., in the case where $k = 3$).

6.2 Optimization of the Computational Cost

The key distributor should select the largest value for m such that the total communication cost is lower than Bd. The optimization method is shown below:

Case 1: Where the network bandwidth is too low

In this case, the equation

$$Bd < \lambda_{max}(1 + 3\log_3 M)$$

holds. The group key management scheme cannot be applied since the network bandwidth is too narrow.

Case 2: Where the network bandwidth is moderate

In this case, the equation

$$\lambda_{max}(1 + 3\log_3 M) \leq Bd < \lambda_{max} + \frac{M}{\tau}$$

holds. The computational cost imposed on client devices is lowest when the total communication cost equals the network bandwidth, Bd, i.e.,

$$Bd = \lambda_{max}[1 + m + k\log_k(M/m)]. \quad (6)$$

The key distributor should select m and k such that Eq. (2):

$$Comp(m, k) = 1 + \frac{1 + \log_k(M/m)}{\tau}$$

is minimal while Eq. (6) holds.

Case 3: Where the network bandwidth is high enough

In this case, the equation

$$Bd \geq \lambda_{max} + \frac{M}{\tau}$$

holds. The network bandwidth is wide enough in this case. That is, the distributor should use the star structure where the computational cost of client devices is minimal.

6.3 Examples

We optimize the hybrid scheme for two sample services, 1) a content delivery service for mobile phones and 2) a pay-per-view service for PCs. Parameters for these services are shown in **Table 4**.

The first service is an example of a content distribution service for mobile devices with low computational capacity ($C = 2.50$) and a narrow network bandwidth ($Bd = 1{,}000$). The number of client devices is large and the average service usage time is short; thus, joins and leaves of client devices occur frequently. The optimal parameters to optimize the communication cost are $k = 40$ and $m = 40$ and the parameters to optimize the computational cost are $k = 19$ and $m = 18$.

Table 4 The optimal parameters for the two sample services.

Parameter	Service (1)	Service (2)
M	10,000	1,000,000
T	600	5,000
τ	1	0.1
Optimization of *Comm*		
C	2.50	200
k	40	3
m	40	1
Comm	1,680	7,950
Comp	2.50	157
Optimization of *Comp*		
Bd	1,000	20,000
k	19	19
m	18	23
Comm	996	19,900
Comp	3.15	52.0

The second service is an example of a pay-per-view service for PCs with high computational capacity ($C = 200$) and wide network bandwidth ($Bd = 20{,}000$). The optimal parameters to optimize the communication cost are $k = 3$ and $m = 1$ and the parameters to optimize the computational cost are $k = 19$ and $m = 23$.

7. Conclusion

We discussed minimization of the total communication cost and the computational cost imposed on client devices in the hybrid scheme. First, we introduced a probabilistic client join/leave model in order to evaluate group key management schemes. This model idealizes client actions generally and considers the existence of the peaks of the joining/leaving frequency. Thus, the model can be used to analyze both average case and worst case scenarios. We then formalized the total computation cost and the computational cost imposed on client devices in group key management schemes under the model. We provided both average case and worst case analyses. Finally, we studied optimal key-management structures that minimize the total communication cost and computational cost in the

hybrid scheme based on the worst case analysis. A key distributor can select parameters that minimize the total communication cost while the computational cost imposed on client devices is lower than the computational capacity, or that minimize the computational cost while the communication cost is lower than the network bandwidth. Our results should prove to be useful when designing a secure communication system for multicast services.

References

1) Asano, T.: Reducing Receiver's Storage in CS, SD and LSD Broadcast Encryption Schemes, *IEICE Trans. Fundamentals*, Vol.E88-A, No.1, pp.203–210 (2005).
2) Aurisch, T.: Optimization Techniques for Military Multicast Key Management, *Proc. Military Communications Conference, 2005* (*MILCOM 2005*), Vol.4, pp.2570–2576 (2005).
3) 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).
4) Eltoweissy, M., Heydari, M.H., Morales, L. and Sudborough, I.H.: Combinatorial Optimization of Group Key Management, *Journal of Network and Systems Management*, Vol.12, No.1, pp.33–50 (2004).
5) Fukushima, K., Kiyomoto, S. and Tanaka, T.: Design of τ-Gradual Key-Management Schemes for Mobile Content Distribution, *IPSJ Journal*, Vol.47, No.12, pp.3137–3148 (2006).
6) Fukushima, K., Kiyomoto, S. and Tanaka, T.: Evaluation of Dual-Structure Key-management Scheme suitable for Mobile Services, *Proc. 7th International Conference on Mobile Data Management* (*MDM'06*) (2006).
7) Little, J.D.C.: A Proof of the Queueing Formula $L = \lambda W$, *Operations Research*, Vol.9, No.3, pp.383–387 (1961).
8) Naor, D., Naor, M. and Lotspiech, J.B.: Revocation and Tracing Schemes for Stateless Receivers, *Advances in Cryptology — CRYPTO 2001, Lecture Notes in Computer Science 2139*, London, UK, pp.41–62, Springer-Verlag (2001).
9) Naor, M. and Pinkas, B.: Efficient Trace and Revoke Schemes, *Financial Cryptography 2000, Lecture Notes in Computer Science 1962*, pp.1–20 (2000).
10) Pinkas, B.: Efficient State Updates for Key Management, *Workshop on Security and Privacy in Digital Rights Management 2001, Lecture Notes in Computer Science 2320*, pp.40–56 (2001).
11) Rafaeli, S. and Hutchison, D.: A Survey of Key Management for Secure Group Communication, *ACM Computing Surveys*, Vol.35, pp.309–329 (2003).
12) Tainaka, Y. and Yamamoto, M.: A Key Management Protocol with Active Network Technology for Secure Multicast, *Proc. 2nd International Workshop on Active Network Technologies and Applications* (*ANTA 2003*), pp.63–74 (2003).
13) Wallner, D., Harder, E. and Agee, R.: Key Management for Multicast: Issues and Architectures, RFC 2627 (Informational) (1999).
14) Wong, C.K., Gouda, M. 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).
15) Zhang, J., Sun, L.-F., Tang, Y. and Yang, S.-Q.: D-VKT: A Scalable Distributed Key Agreement Scheme for Dynamic Collaborate Groups, *IEICE Trans. Comm.*, Vol.E90-B, No.4, pp.750–760 (2007).
16) Zhu, S., Setia, S. and Jajodia, S.: Performance Optimizations for Group Key Management Schemes, *Proc. 23rd International Conference on Distributed Computing Systems* (*ICDCS'03*), pp.163–171 (2003).

Appendix

A.1 Detailed Description of the Hybrid Scheme

The hybrid scheme [5),6)] uses the logical key-management structure shown in Fig. 1. The center executes the rekeying process for every τ seconds, and it does not update the group key when a client joins or leaves. The key-management structure of the scheme depends on the parameter m that denotes the number of client devices in a subgroup. We show the process for the cases $m = M$, $1 < m < M$ and $m = 1$.

Case 1: Where $m = M$

In this case, the key-management structure is identical to the conventional star structure.

(1) **Join Process**

The center encrypts the group key, K_G, with the individual key of a joining client, and sends it to the client.

(2) **Rekeying Process**

The center updates the group key. Then, the center encrypts the updated key with each individual key and sends it to each client.

Case 2: Where $1 < m < M$

In this case, the key-management structure is hybrid. The subgroup key is managed by a star structure, and the group key is managed by a tree structure. A preparation process is needed to distribute subgroup keys to client devices.

(1) **Preparation Process**

The center executes this process only once, i.e., when the group key management scheme is started.

Step 1 Create Subgroups

The center creates m-client-subgroups SG_1, SG_2, ..., SG_l, where $l = \lceil M/m \rceil$.

Step 2 Generate Subgroup Keys

The center generates subgroup keys $K_{SG_1}, K_{SG_2}, \ldots, K_{SG_l}$, which are shared among all the clients in each subgroup. The center encrypts the subgroup keys with each individual key, and sends it to each client.

Step 3 Construct Tree

The center constructs a complete k-ary tree with height $h = \lceil \log_k l \rceil$, then removes $k^h - l$ leaf nodes if $l \neq k^h$. This tree is the key-management tree for the proposed scheme. The center then assigns subgroup keys K_{SG_1}, K_{SG_2}, ..., K_{SG_l} to the leaf nodes of the tree. The center next generates a group key, K_G, and key-encryption keys K_2, K_3, ..., K_i. The center assigns the group key to the root node (node 1), and the key-encryption keys to the interior nodes (nodes 2, 3, ..., i). The group key and key-encryption key are shared among clients belonging to subgroups whose subgroup key is assigned to the descendant leaf nodes. The group key, K_G, is used as the content-encryption key since all the clients share this key.

Step 4 Distribute Keys

The server sends the group key and the key-encryption key generated in Step 3 to the clients belonging to the subgroups, SG_1, SG_2, ..., SG_l. The keys are encrypted with the key-encryption keys or the subgroup keys assigned to the child nodes. The server then sends them to clients who have the encryption key.

(2) **Joining Process**

The center encrypts the group key, K_G, 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 subgroup which the client left from. The flag indicates that the subgroup is going to be revoked in the next key-updating process.

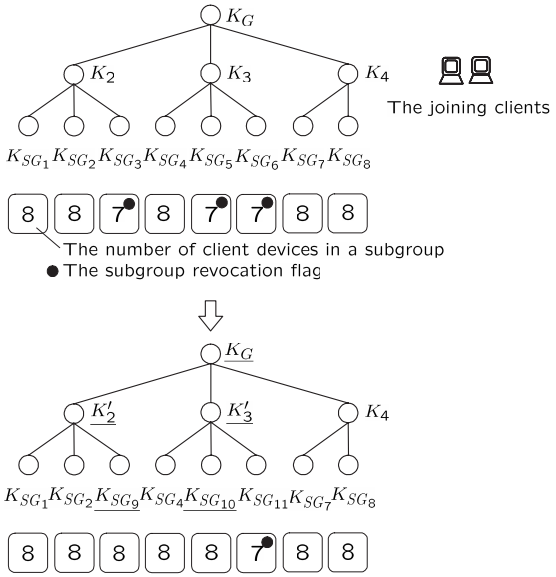

Fig. 2 Key updating process.

Figure 2 shows an example. Let the maximum number of clients in a subgroup, m, be eight. The center sets the flag on subgroups SG_3, SG_5, and SG_6 since the numbers of clients in these subgroups is smaller than m.

(4) **Rekeying Process**

The center updates keys every τ ($\tau > 0$) seconds. The center updates the keys according to the following steps:

Step 1 Revoke Subgroups

The center revokes subgroups on which flags have been set. The center then discards the subgroup keys of the revoked subgroups.

Groups SG_3, SG_5, and SG_6 in our example are revoked (see Fig. 2).

Step 2 Reconstruct Subgroups

The center creates new m-client-groups from joining clients and clients in the revoked subgroups. It then generates subgroup keys for the new subgroups. The center encrypts the subgroup keys with an individual key for each client,

and sends it to each client.

There are 21 clients in revoked subgroups SG_3, SG_5, and SG_6 and two joining clients in Fig. 2. The center creates two 8-client subgroups, SG_9 and SG_{10}, and a 7-client subgroup, SG_{11}.

Step 3 Reconstruct Tree

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

In the example in Fig. 2, the center assigns subgroup keys K_{SG_9}, $K_{SG_{10}}$, and $K_{SG_{11}}$ to nodes to which subgroup keys K_{SG_3}, K_{SG_5}, and K_{SG_6} were respectively assigned.

Step 4 Updating Keys

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

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

Step 5 Redistributing Keys

The center 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 center encrypts the shared key, K'_G, with key-encryption keys, K'_2 and K'_3, and K_4 and sends them to the clients belonging to subgroups in sets $\{SG_1, SG_2, SG_9\}$, $\{SG_4, SG_{10}, SG_{11}\}$, and $\{SG_7, SG_8\}$, respectively.

Case 3: Where $m = 1$

In this case, the key-management structure is identical to the conventional tree structure.

(1) **Join Process**

The center encrypts the current group key, K_G, with the individual key of a joining client and sends it to the client.

(2) **Rekeying Process**

The center updates the group keys and key-encryption keys assigned to the ancestors of the nodes, to which the individual keys of clients that have left are assigned.

(Received November 30, 2007)
(Accepted June 3, 2008)
(Original version of this article can be found in the Journal of Information Processing Vol.16, pp.130–141.)

Kazuhide Fukushima received his M.E. in 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 received the Dr. degree in engineering from Kyushu University in 2007. He is a member of IEICE.

Kouichi Sakurai received the B.S. degree in mathematics from Faculty of Science, Kyushu University and the M.S. degree in applied science from the Faculty of Engineering, Kyushu University in 1986 and 1988, respectively. He was engaged in research and development on cryptography and information security at the Computer and Information Systems Laboratory at Mitsubishi Electric Corporation from 1988 to 1994. He received his doctorate in engineering from the Faculty of Engineering, Kyushu University in 1993. From 1994, he worked for the Department of Computer Science of Kyushu University in the capacity of associate professor, and became a full professor in 2002. His current research interests are in cryptography and information security. Dr. Sakurai is a member of the IEICE, the Mathematical Society of Japan, ACM, IEEE and the International Association for Cryptologic Research.