A Batch Rekeying Time Decision Algorithm for IPTV Systems

Sangho Lee
Dept. of Computer Science and Engineering
Pohang Univ. of Science and Technology (POSTECH)
Pohang, Republic of Korea
Email: sangho2@postech.ac.kr

Jong Kim
Div. of IT Convergence Engineering
Pohang Univ. of Science and Technology (POSTECH)
Pohang, Republic of Korea
Email: jkim@postech.ac.kr

***Abstract*—This paper proposes an algorithm to decide on the batch rekeying time for group key management schemes for Internet protocol television (IPTV) systems. Batch rekeying schemes can reduce the effect of membership changes in group key management schemes by collecting and processing several join and leave events at once. Because membership of IPTV systems frequently changes due to the frequent changes of TV channels viewed by subscribers, batch rekeying schemes are suitable to IPTV systems. Our algorithm considers one more factor; programs broadcasted on IPTV channels can have various values. By computing the expected loss from the value of programs, and the number and time of join and leave events, our algorithm decides on the next rekeying time where the expected loss becomes larger than the predefined loss value given by the service provider.**

***Index Terms*—IPTV, group key management, batch rekeying, rekeying policy**

I. Introduction

Internet protocol television (IPTV) is a technology to deliver TV programs to subscribers via the Internet. IPTV systems usually use IP multicast technology to send data to a group of receivers at once. However, the data sent via the IP multicast are delivered to every receiver whether it is authorized or not. Thus, we need an access control scheme to protect IPTV programs from unauthorized receivers.

As an access control scheme for IPTV systems, we can use a group key management scheme [1]. For instance, by grouping the authorized subscribers for a specific TV channel and then assigning a group key for the group, IPTV systems can securely deliver TV programs to the group by encrypting the TV programs with the group key. The performance of the group key management scheme heavily depends on the number of group members and the frequency of membership changes. Specifically, the individual rekeying which performs rekeying whenever membership changes occur introduces a large overhead to the entire systems [2]. Because subscribers frequently change the TV channels they view, individual rekeying is not suitable for IPTV systems.

This research was supported by the MKE (The Ministry of Knowledge Economy), Korea, under the ITRC (Information Technology Research Center) support program supervised by the NIPA (National IT Industry Promotion Agency) (NIPA-2010-C1090-1031-0009) and WCU (World Class University) program through the National Research Foundation of Korea funded by the Ministry of Education, Science and Technology (R31-2008-000-10100-0).

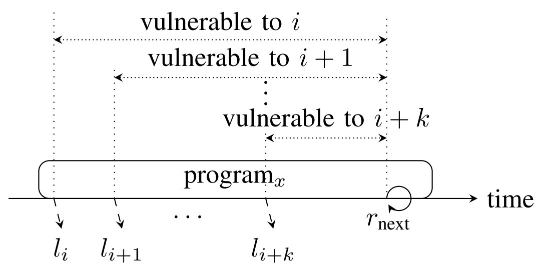
To reduce the overhead of group key management due to the frequent membership changes, periodic rekeying [2], batch rekeying [3]–[7], and exposure-oriented rekeying [8], [9] schemes have been proposed. In the periodic rekeying scheme [2], rekeying is performed at a predefined interval. In the batch rekeying schemes [3]–[7], rekeying is performed whenever a predefined number of membership changes occur. Both schemes can reduce the overhead due to the membership changes. However, they cannot consider the real loss because they ignore the value of data. Exposure-oriented rekeying schemes [8], [9] solve this problem by performing rekeying whenever the loss due to the exposure of data exceeds a predefined loss value. The exposure-oriented rekeying schemes assume that the data of a group have the same value. In IPTV systems, however, programs on the same channel can have different values. Therefore, exposure-oriented rekeying cannot be applied to IPTV systems directly.

In this paper we propose an algorithm which decides the batch rekeying time for IPTV systems with a group key management scheme. The proposed algorithm decides the next rekeying time where the expected loss exceeds a predefined loss value. The factors of this decision are the value of programs, the number of membership changes, and the time of the membership changes. Unlike the exposure-oriented rekeying [8], [9], the proposed algorithm can deal with programs with various values. Thus, the proposed algorithm is more suitable for IPTV systems than the previous schemes.

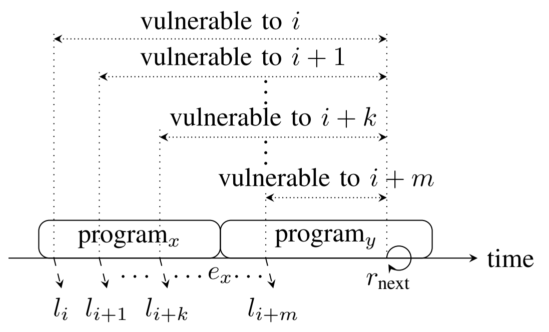
The remainder of this paper is organized as follows. In Section II we introduce our system model and assumptions. In Section III we propose a rekeying time decision algorithm. In Section IV we analyze the proposed algorithm. Finally, we conclude this paper in Section V.

II. System Model and Assumption

We assume an IPTV system which supports pay-per-view services for some channels. To support pay-per-view service, the system maintains the channel each subscriber has currently joined, the programs which are broadcasted on the channel, and the start and end time of the subscriber's watching. The channels for the pay-per-view service are secured by group key schemes. Whenever join and leave events occur on the channels, rekeying should be performed to assure



(a) When $k+1$ subscribers from i to $i+k$ leave from a program x



(b) When $m+1$ subscribers from i to $i+m$ leave from two programs x and y

Fig. 1. Rekeying against leave events

backwards and forwards secrecy. We also assume that the value of programs which are broadcasted on the channels can be different. This program value is predetermined based on the agreement between the content providers and service providers.

III. Proposed Algorithm

A. Rekeying against Leave Events

When a subscriber leaves a TV channel, we have to update the channel key to prevent access to later programs on the TV channel from the leaving subscriber. However, immediate rekeying is not desired because it will introduce a large overhead to the entire system. So, we have to postpone the next rekeying for as long as possible to reduce the rekeying overhead by using the batch rekeying schemes [3]–[7]. To determine the next rekeying time for the leave events, we suggest algorithms which use the value of programs as one of their inputs.

Let us consider a case where several leave events occur in a program x whose value is v_x. Assume that $k+1$ subscribers from i to $i+k$ leave from x at time from l_i to l_{i+k}. The next rekeying will occur at r_{next} before the end of x, e_x (Fig. 1a). Then, the expected loss at time r_{next} is

$$\tilde{L} = \sum_{a=i}^{i+k} v_x(r_{\text{next}} - l_a). \tag{1}$$

Our goal is to find the r_{next} which makes $\tilde{L}$ be the value L which is the predefined loss value given by the service provider. By rearranging the above equation according to r_{next}, we can represent r_{next} as a function of L:

$$r_{\text{next}}(L) = \frac{1}{k+1}\left(\sum_{a=i}^{i+k} l_a + \frac{L}{v_x}\right). \tag{2}$$

We have to check whether $r_{\text{next}}(L)$, Eq.(2), converges because if it does not then we cannot be assured of the security of the system. Lemma 1 is the result.

Lemma 1. *Even if many leave events occur in a program, rekeying will be eventually performed.*

Proof: Let the number of leave events k approaches infinity. Then,

$$\begin{aligned}\lim_{k\to\infty} r_{\text{next}}(L) &= \lim_{k\to\infty}\frac{1}{k+1}\left(\sum_{a=i}^{i+k} l_a + \frac{L}{v_x}\right)\\ &\le \lim_{k\to\infty}\frac{1}{k+1}\left((k+1)l_{i+k} + \frac{L}{v_x}\right)\\ &\le \lim_{k\to\infty} l_{i+k} + \frac{L}{(k+1)v_x}\\ &\le \lim_{k\to\infty} l_{i+k}.\end{aligned}$$

The above equation shows that even if infinite leave events occur in a program, rekeying will be performed before the last leave event. Therefore, even if an infinite amount of leave events occur in a program, rekeying will be performed eventually. ■

Let us consider another case where several leave events occur in two consecutive programs x and y, whose values are v_x and v_y, respectively. Assume that $k+1$ subscribers from i to $i+k$ leave from x at time from l_i to l_{i+k}, and then $m-k$ subscribers from $i+k+1$ to $i+m$ leave from y at time from l_{i+k+1} to l_{i+m}. The next rekeying will occur at r_{next} after the end of x, e_x (Fig. 1b). Then, the expected loss at time r_{next} is

$$\begin{aligned}\tilde{L} =& \sum_{a=i}^{i+k} v_x(e_x - l_a) + (k+1)v_y(r_{\text{next}} - e_x) + \\ &\sum_{a=i+k+1}^{i+m} v_y(r_{\text{next}} - l_a).\end{aligned} \tag{3}$$

By rearranging the above equation according to r_{next}, we can represent r_{next} as a function of L:

$$\begin{aligned}r_{\text{next}}(L) =& \frac{1}{(m+1)v_y}\Bigg(L - (k+1)v_x e_x + v_x\sum_{a=i}^{i+k} l_a + \\ &(k+1)v_y e_x + v_y \sum_{a=i+k+1}^{i+m} l_a\Bigg).\end{aligned} \tag{4}$$

We also have to check whether Eq.(4) converges. Lemma 2 is the result.

Lemma 2. *Even if many leave events occur across two programs, rekeying will be eventually performed.*

Proof: Let the number of leave events m approaches infinity. Then,

$$\begin{aligned}
&\lim_{m\to\infty} r_{\text{next}}(L) \\
&\quad = \lim_{m\to\infty} \frac{1}{(m+1)v_y}\Bigg(L-(k+1)v_x e_x + v_x \sum_{a=i}^{i+k} l_a + \\
&\quad (k+1)v_y e_x + v_y \sum_{a=i+k+1}^{i+m} l_a\Bigg) \\
&\quad = \frac{1}{\lim\limits_{m\to\infty}(m+1)v_y}\Bigg(L-(k+1)v_x e_x + v_x \sum_{a=i}^{i+k} l_a + \\
&\quad (k+1)v_y e_x + \lim_{m\to\infty} v_y \sum_{a=i+k+1}^{i+m} l_a\Bigg) \\
&\quad \le \frac{1}{\lim\limits_{m\to\infty}(m+1)v_y}\Bigg(L-(k+1)v_x e_x + v_x \sum_{a=i}^{i+k} l_a + \\
&\quad (k+1)v_y e_x + \lim_{m\to\infty}(m-k)v_y l_{i+m}\Bigg) \\
&\quad \le \lim_{m\to\infty} l_{i+m}
\end{aligned}$$

The above equation shows that even if infinite leave events occur across two programs, rekeying will be performed before the last leave event. Therefore, even if an infinite amount of leave events occur across two programs, rekeying will be performed eventually. ■

B. Rekeying against Join Events

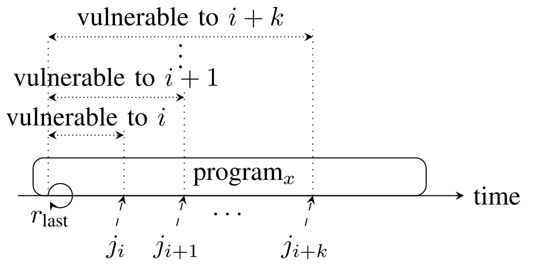
When a subscriber joins to a TV channel, we have to give a channel key to the joining subscriber. However, it is not desirable that the subscriber can access the programs which were broadcast on the channel before the subscriber joined. Therefore, the channel key should be updated before it is delivered to the subscriber. In a similar way to the leave events, immediate rekeying is not desired because of the system overhead. Therefore, we need algorithms to decide on the rekeying time for the join events.

Let us consider a case where several join events occur in a program x whose value is v_x. Assume that the last rekeying has occurred at r_{last}, and then $k+1$ subscribers from i to $i+k$ join to x at time from j_i to j_{i+k} before the end of x, e_x (Fig. 2a). Then, the expected loss at time j_{i+k} is

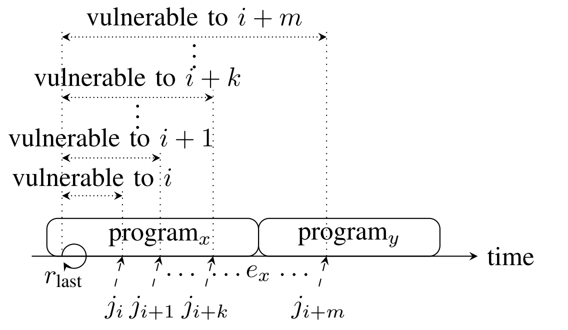
$$\tilde{L}_{i+k} = \sum_{a=i}^{i+k} v_x(j_a - r_{\text{last}}). \tag{5}$$

When $\tilde{L}_{i+k}$ is larger than or equal to the predefined loss value L, rekeying should occur immediately. Since L and r_{last} are fixed, and $\tilde{L}_{i+k}$ has monotonically increasing, rekeying will be performed eventually.

Let us consider another case where several join events occur in two consecutive programs x and y, whose values are v_x and v_y, respectively. Assume that the last rekeying has occurred at r_{last}. $k+1$ subscribers from i to $i+k$ join to x at time from j_i to j_{i+k} and then $m-k$ subscribers from $i+k+1$ to



(a) When $k+1$ subscribers from i to $i+k$ join to a program x



(b) When $m+1$ subscribers from i to $i+m$ join to two programs x and y

Fig. 2. Rekeying against join events

$i+m$ join to y at time from j_{i+k+1} to j_{i+m} (Fig. 2b). Then, the expected loss at time j_{i+m} is

$$\begin{aligned}
\tilde{L}_{i+m} = &\sum_{a=i}^{i+k} v_x(j_a - r_{\text{last}}) + (m-k)v_x(e_x - r_{\text{last}}) + \\
&\sum_{a=i+k+1}^{i+m} v_y(j_a - e_x).
\end{aligned} \tag{6}$$

If $\tilde{L}_{i+m} \ge L$ then rekeying should occur immediately.

C. Rekeying in Combined Cases

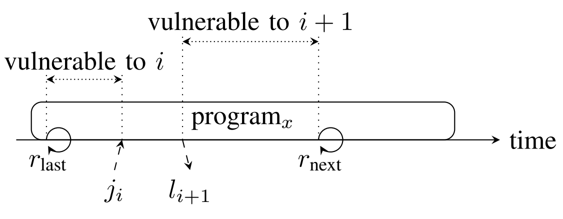
In this subsection, we consider cases where the leave and join events occur simultaneously. First, let us consider a case where the last event is a leave event. Assume that the last rekeying has occurred at r_{last} in a program x whose value is v_x. A subscriber i joins to x at time j_i and then another subscriber $i+1$ leaves from x at time l_{i+1}. The next rekeying will occur at r_{next} before the end of x, e_x (Fig. 3a). Then, the expected loss at time r_{next} is

$$\tilde{L} = v_x(r_{\text{next}} - l_{i+1}) + v_x(j_i - r_{\text{last}}). \tag{7}$$

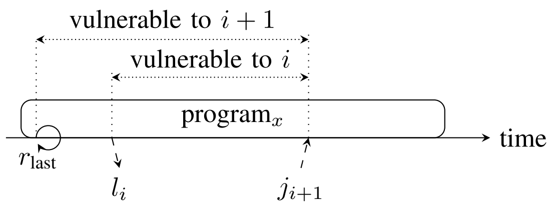
By rearranging the above equation according to r_{next}, we can represent r_{next} as a function of L:

$$r_{\text{next}}(L) = l_{i+1} + \frac{1}{v_x}(L - v_x(r_{\text{last}} - j_i)). \tag{8}$$

The above equation means that to compute the next rekeying time when a join event occurs before the leave event, we only need to extract the loss due to the join event from L, and then compute the next rekeying time with the proposed



(a) Join and then leave



(b) Leave and then join

Fig. 3. Rekeying against leave and join events

algorithm. This result can be applied to the case where several join and leave events occur before the last event which is leave occurring in two consecutive programs. In this case, r_{next} is

$$r_{\text{next}}(L) = \frac{1}{(|\mathcal{L}_x| + |\mathcal{L}_y|)v_y} \bigg(L - L_{\text{join}} + |\mathcal{L}_x|(v_y - v_x)e_x + v_x \sum_{l \in \mathcal{L}_x} l + v_y \sum_{l \in \mathcal{L}_y} l \bigg), \tag{9}$$

where L_{join} is the expected loss due to the join events, and $\mathcal{L}_x$ and $\mathcal{L}_y$ are the sets of leave events on programs x and y, respectively. Since L_{join} is smaller than L (if not, rekeying had already occurred), Eq.(9) will converge according to Lemma 2.

Let us consider the case where the last event is a join event. Assume that the last rekeying has occurred at r_{last} in a program x whose value is v_x. A subscriber i leaves from x at time l_i and then another subscriber $i+1$ joins to x at time j_{i+1}. The next rekeying will occur at r_{next} before the end of x, e_x (Fig. 3b). Then, the expected loss at time j_{i+1} is

$$\tilde{L}_{i+1} = v_x(j_{i+1} - r_{\text{last}}) + v_x(j_{i+1} - l_i). \tag{10}$$

In a similar way to the case where the last event is a leave event, the expected loss is the sum of the losses due to the leave and join events. This result can be applied to the case where several join and leave events have occurred before the last event which is join occurring in two consecutive programs. In this case, the expected loss due to the join events is

$$L_{\text{join}} = \sum_{j \in \mathcal{J}_x} v_x(j - r_{\text{last}}) + |\mathcal{J}_y| v_x(e_x - r_{\text{last}}) + \sum_{j \in \mathcal{J}_y} v_y(j - e_x), \tag{11}$$

where $\mathcal{J}_x$ and $\mathcal{J}_y$ are the sets of join events on program x and y, respectively. Also, the expected loss due to the leave events L_{leave} at the time of the last join event j_{last} is

$$L_{\text{leave}} = \sum_{l \in \mathcal{L}_x} v_x(e_x - l) + |\mathcal{L}_x| v_y(j_{\text{last}} - e_x) + \sum_{l \in \mathcal{L}_y} v_y(j_{\text{last}} - l). \tag{12}$$

If $L_{\text{join}} + L_{\text{leave}} \geq L$ then rekeying should occur immediately. If not, r_{next} needs to be updated by using Eq.(9).

Finally, by using the equations introduced in this section, we create an algorithm to decide on the next rekeying time which should be performed whenever leave or join events occur (Algorithm 1). The time complexity of the algorithm is $O(n)$ where n is related to the number of leave and join events. Because $O(n)$ is a relatively high overhead, we optimize the algorithm by expanding the equations and reusing previously computed values. Algorithm 2 is the result which has an $O(1)$ time complexity.

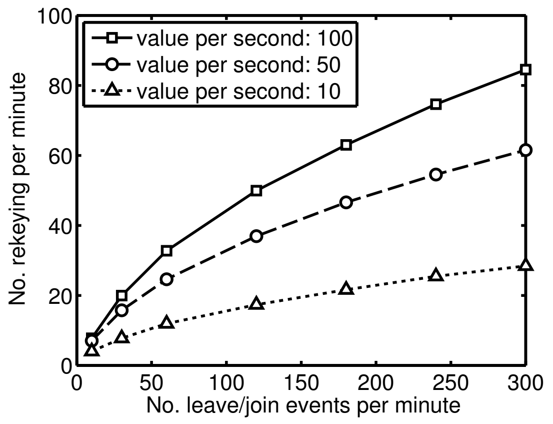


Fig. 4. The number of rekeying according to the number of events per a minute. The predefined loss is 100.

IV. Analysis

We check the number of rekeying according to the number of leave and join events, and the value of programs in MATLAB. Each interval between events follows an exponential distribution whose mean is $\frac{1}{\text{the number of events per second}}$. The results show that the number of rekeying increases when the number of events or the value of programs increases (Fig. 4).

V. Conclusions

In this paper we proposed a batch rekeying time decision algorithm for IPTV systems. The proposed algorithm uses the value of programs to decide the rekeying time according to the expected loss. It achieves access controls to programs and efficient rekeying at the same time. The analysis results show that the proposed algorithm adaptively changes the number of rekeying according to the number of leave and join events, and the value of programs.

```
input : Predefined loss $L$; Last rekeying time $r_{\text{last}}$;
        Value of the programs $x$ and $y$ $v_x, v_y$; End time of $x$ $e_x$;
        Sets of the previous leave and join events on $x$ and $y$
        $\mathcal{L}_x, \mathcal{L}_y, \mathcal{J}_x, \mathcal{J}_y$;
        Type of the current event $type$;
        Time that the current event occurs $c$
output: Updated $r_{\text{next}}, \mathcal{L}_x, \mathcal{L}_y, \mathcal{J}_x, \mathcal{J}_y, L_{\text{join}}$
if $t \le e_x$ then
    if $type = join$ then
        $\mathcal{J}_x \leftarrow \mathcal{J}_x \cup c$;
        $L_{\text{join}} \leftarrow \sum_{j \in \mathcal{J}_x} v_x(j - r_{\text{last}})$;
        $L_{\text{leave}} \leftarrow \sum_{l \in \mathcal{L}_x} v_x(c - l)$;
        if $L_{\text{join}} + L_{\text{leave}} \ge L$ then
            $r_{\text{next}} \leftarrow c + \epsilon$;
        end
        else
            $r_{\text{next}} \leftarrow (\mathsf{Sum}(\mathcal{L}_x)v_x + L - L_{\text{join}})/(|\mathcal{L}_x|v_x)$;
        end
    end
    else
        $\mathcal{L}_x \leftarrow \mathcal{L}_x \cup c$;
        $L_{\text{join}} \leftarrow \sum_{j \in \mathcal{J}_x} v_x(j - r_{\text{last}})$;
        $r_{\text{next}} \leftarrow (\mathsf{Sum}(\mathcal{L}_x)v_x + L - L_{\text{join}})/(|\mathcal{L}_x|v_x)$;
    end
end
else
    if $type = join$ then
        $\mathcal{J}_y \leftarrow \mathcal{J}_y \cup c$ ;
        $L_{\text{join}} \leftarrow \sum_{j \in \mathcal{J}_x} v_x(j - r_{\text{last}}) + |\mathcal{J}_y| v_x(e_x - r_{\text{last}}) +$
        $\sum_{j \in \mathcal{J}_y} v_y(j - e_x)$;
        $L_{\text{leave}} \leftarrow$
        $\sum_{l \in \mathcal{L}_x} v_x(e_x - l) + |\mathcal{L}_x| v_y(c - e_x) + \sum_{l \in \mathcal{L}_y} v_y(c - l)$;
        if $L_{\text{join}} + L_{\text{leave}} \ge L$ then
            $r_{\text{next}} \leftarrow c + \epsilon$;
        end
        else
            $r_{\text{next}} \leftarrow (|\mathcal{L}_x|(v_y - v_x)e_x + \mathsf{Sum}(\mathcal{L}_x)v_x +$
            $\mathsf{Sum}(\mathcal{L}_y)v_y + L - L_{\text{join}})/((|\mathcal{L}_x| + |\mathcal{L}_y|)v_y)$;
        end
    end
    else
        $\mathcal{L}_y \leftarrow \mathcal{L}_y \cup c$;
        $L_{\text{join}} \leftarrow \sum_{j \in \mathcal{J}_x} v_x(j - r_{\text{last}}) + |\mathcal{J}_y| v_x(e_x - r_{\text{last}}) +$
        $\sum_{j \in \mathcal{J}_y} v_y(j - e_x)$;
        $r_{\text{next}} \leftarrow (|\mathcal{L}_x|(v_y - v_x)e_x + \mathsf{Sum}(\mathcal{L}_x)v_x + \mathsf{Sum}(\mathcal{L}_y)v_y +$
        $L - L_{\text{join}})/((|\mathcal{L}_x| + |\mathcal{L}_y|)v_y)$;
    end
end
return $r_{\text{next}}, \mathcal{L}_x, \mathcal{L}_y, \mathcal{J}_x, \mathcal{J}_y$
```

Algorithm 1: Rekeying time decision algorithm

```
input : Predefined loss $L$; Last rekeying time $r_{\text{last}}$;
        Value of the programs $x$ and $y$ $v_x, v_y$; End time of $x$ $e_x$;
        Sums of the time of the previous leave and join events on
        $x$ and $y$ $S_{\mathcal{L}_x}, S_{\mathcal{L}_y}, S_{\mathcal{J}_x}, S_{\mathcal{J}_y}$;
        Numbers of the previous leave and join events on $x$ and $y$
        $N_{\mathcal{L}_x}, N_{\mathcal{L}_y}, N_{\mathcal{J}_x}, N_{\mathcal{J}_y}$;
        Expected loss due to the previous join events $L_{\text{join}}$;
        Type of the current event $type$;
        Time that the current event occurs $c$
output: Updated $r_{\text{next}}, S_{\mathcal{L}_x}, S_{\mathcal{L}_y}, S_{\mathcal{J}_x}, S_{\mathcal{J}_y}, N_{\mathcal{L}_x}, N_{\mathcal{L}_y}, N_{\mathcal{J}_x}, N_{\mathcal{J}_y}$
if $t \le e_x$ then
    if $type = join$ then
        $N_{\mathcal{J}_x} \leftarrow N_{\mathcal{J}_x} + 1$;
        $S_{\mathcal{J}_x} \leftarrow S_{\mathcal{J}_x} + c$;
        $L_{\text{join}} \leftarrow S_{\mathcal{J}_x} v_x - N_{\mathcal{J}_x} v_x r_{\text{last}}$;
        $L_{\text{leave}} \leftarrow N_{\mathcal{L}_x} v_x c - S_{\mathcal{L}_x} v_x$;
        if $L_{\text{join}} + L_{\text{leave}} \ge L$ then
            $r_{\text{next}} \leftarrow c + \epsilon$;
        end
        else
            $r_{\text{next}} \leftarrow (S_{\mathcal{L}_x} v_x + L - L_{\text{join}})/(N_{\mathcal{L}_x} v_x)$;
        end
    end
    else
        $N_{\mathcal{L}_x} \leftarrow N_{\mathcal{L}_x} + 1$;
        $S_{\mathcal{L}_x} \leftarrow S_{\mathcal{L}_x} + c$;
        $r_{\text{next}} \leftarrow (S_{\mathcal{L}_x} v_x + L - L_{\text{join}})/(N_{\mathcal{L}_x} v_x)$;
    end
end
else
    if $type = join$ then
        $N_{\mathcal{J}_y} \leftarrow N_{\mathcal{J}_y} + 1$;
        $S_{\mathcal{J}_y} \leftarrow S_{\mathcal{J}_y} + c$;
        $L_{\text{join}} \leftarrow S_{\mathcal{J}_x} v_x + S_{\mathcal{J}_y} v_y + N_{\mathcal{J}_y} v_x e_x - (N_{\mathcal{J}_x} +$
        $N_{\mathcal{J}_y}) v_x r_{\text{last}} - N_{\mathcal{J}_x} v_y e_x$;
        $L_{\text{leave}} \leftarrow$
        $N_{\mathcal{L}_x}(v_x - v_y)e_x + (N_{\mathcal{L}_x} + N_{\mathcal{L}_y}) v_y c - S_{\mathcal{L}_x} v_x - S_{\mathcal{L}_y} v_y$;
        if $L_{\text{join}} + L_{\text{leave}} \ge L$ then
            $r_{\text{next}} \leftarrow c + \epsilon$ ;
        end
        else
            $r_{\text{next}} \leftarrow (N_{\mathcal{L}_x}(v_y - v_x)e_x + S_{\mathcal{L}_x} v_x + S_{\mathcal{L}_y} v_y + L -$
            $L_{\text{join}})/((N_{\mathcal{L}_x} + N_{\mathcal{L}_y})v_y)$;
        end
    end
    else
        $N_{\mathcal{L}_y} \leftarrow N_{\mathcal{L}_y} + 1$;
        $S_{\mathcal{L}_y} \leftarrow S_{\mathcal{L}_y} + c$;
        $r_{\text{next}} \leftarrow (N_{\mathcal{L}_x}(v_y - v_x)e_x + S_{\mathcal{L}_x} v_x + S_{\mathcal{L}_y} v_y + L -$
        $L_{\text{join}})/((N_{\mathcal{L}_x} + N_{\mathcal{L}_y})v_y)$;
    end
end
return $r_{\text{next}}, S_{\mathcal{L}_x}, S_{\mathcal{L}_y}, S_{\mathcal{J}_x}, S_{\mathcal{J}_y}, N_{\mathcal{L}_x}, N_{\mathcal{L}_y}, N_{\mathcal{J}_x}, N_{\mathcal{J}_y}$
```

Algorithm 2: Rekeying time decision algorithm optimized

REFERENCES

[1] D. Wallner, E. Harder, and R. Agee, "Key management for multicast: Issues and architectures," RFC 2627, IETF, 1999.

[2] S. Setia, S. Koussih, S. Jajodia, and E. Harder, "Kronos: A scalable group re-keying approach for secure multicast," in *Proceedings of 2000 IEEE Symposium on Security and Privacy*, 2000, pp. 215–228.

[3] P. Lee, J. Lui, and D. Yau, "Distributed collaborative key agreement protocols for dynamic peer groups," in *Proceedings of the 10th IEEE International Conference on Network Protocols*, 2002, pp. 322–331.

[4] X. S. Li, Y. R. Yang, M. G. Gouda, and S. S. Lam, "Batch rekeying for secure group communications," in *Proceedings of the 10th International Conference on World Wide Web*, 2001, pp. 525–534.

[5] W. Ng, M. Howarth, Z. Sun, H. Cruickshank, and A. Singapore, "Dynamic balanced key tree management for secure multicast communications," *IEEE Transactions on Computers*, vol. 56, no. 5, pp. 590–605, 2007.

[6] J. Pegueroles and F. Rico-Novella, "Balanced batch LKH: new proposal, implementation and performance evaluation," in *Proceedings of the 8th IEEE International Symposium on Computers and Communication*, 2003, pp. 815–820.

[7] X. Zhang, S. S. Lam, D. Lee, and Y. Yang, "Protocol design for scalable and reliable group rekeying," *IEEE/ACM Transactions on Networking (TON)*, vol. 11, no. 6, pp. 908–922, 2003.

[8] Q. Zhang and K. Calvert, "On rekey policies for secure group applications," in *Proceedings of the 12th International Conference on Computer Communications and Networks*, 2003, pp. 559–564.

[9] G.-Y. Lee, C.-K. Jeong, Y. Lee, and H.-J. Kim, "Re-key interval optimization for secure group communications," *Journal of Information Science and Engineering*, vol. 23, no. 3, pp. 893–906, 2007.