Alteration Method of Schedule Information on Public Cloud for Preserving privacy

Tatsuya Miyagami[1], Atsushi Kanai[1], Noriaki Saito[2], Shigeaki Tanimoto[3], Hiroyuki Sato[4]

[1] Graduate School of Engineering Hosei University, Tokyo, Japan
tatsuya.miyagami.9t@stu.hosei.ac.jp, yoikana@hosei.ac.jp
[2] NTT Information Platform Laboratories, Tokyo, Japan
saito.noriaki@lab.ntt.co.jp
[3] Chiba Institute of Technology, Chiba, Japan
shigeaki.tanimoto@it-chiba.ac.jp
[4] The University of Tokyo, Tokyo, Japan
schuko@satolab.itc.u-tokyo.ac.jp

***Abstract*— We are currently experiencing an explosion of cloud technologies. However, a cloud service administrator may be an untrustworthy third party. Therefore, companies dealing with confidential information cannot use a public cloud. In this paper, we propose a method for preventing the leakage of private information on a cloud schedule service. In this method, even a cloud administrator or a hacker who steals a cloud service login key cannot read the true schedule because the schedule date is altered and schedule content is encrypted. Consequently we can safely use a public cloud schedule service with this method. We also evaluated the method's performance using an actual alteration program on Google server. We implement the proposed method and show the performance is practical by evaluating actually.**

Keywords-cloud computing; Internet security; privacy; Google Calendar; schedule service; date alteration.

I. INTRODUCTION

We are currently experiencing an explosion of cloud technologies. However, by considering "cloud" as a social infrastructure, we must also consider security [1]. For example, a cloud system is not managed by a user; therefore, cloud users cannot be certain that their critical information is actually safe [2].

Since schedule services are useful for a variety of fields, many people use them to manage their personal scheduling information. However, companies are unable to use scheduling services because information may include private or confidential information [3]. If there is a malicious administrator in the cloud, he might steal a user's privacy information or confidential information. If a malicious administrator exploits a company's confidential information, such as an important meeting schedule or customer information, its finances and reputation may be seriously damaged [4].

To solve this problem, it is necessary to protect private or confidential information from third party tapping. It is easy to preserve privacy or confidential information by encryption with respect to documents. The contents of a schedule can be encrypted on the schedule server. However, the schedule dates cannot be encrypted on the schedule server because if the value of the encrypted data becomes binary, the value cannot be saved in the schedule server as a schedule date. Therefore, the encryption of schedule dates cannot be used with a calendar service without changing the calendar interface.

In this paper, we propose a method for altering dates. The original schedule dates are completely changed to different dates. By using both alteration and encryption, a schedule can be protected from a third party. This is a good solution for managing both security and convenience of existing cloud schedule services at the same time. Note that we are not concerned here with encryption of schedule contents, which is rather simple, but with the alteration of schedule dates.

The organization of this paper is as follows. Section II describes related work. Section III introduces our alteration method for schedule services. Section IV describes the saving and reading algorithms created with our method. Section V discusses the evaluation of our alteration method. Section VI summarizes this paper.

II. RELATED WORK

Techniques of preserving privacy have been discussed for On-Line Analytical Processing (OLAP). Furthermore, Database-As-a-Service (DAS), which provides data management services for cloud computing, has become familiar.

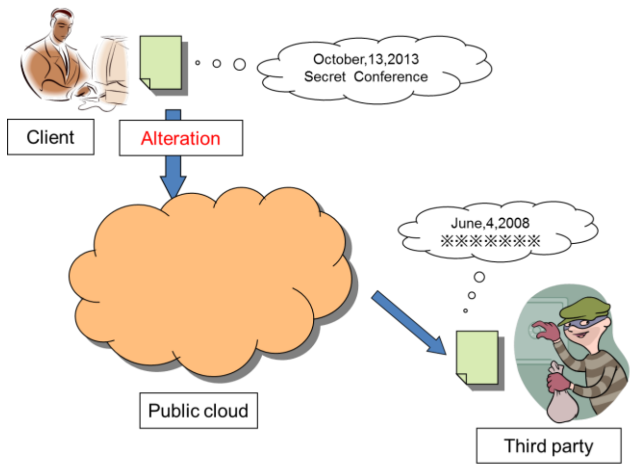


Figure 1. Alteration of schedule date in public cloud

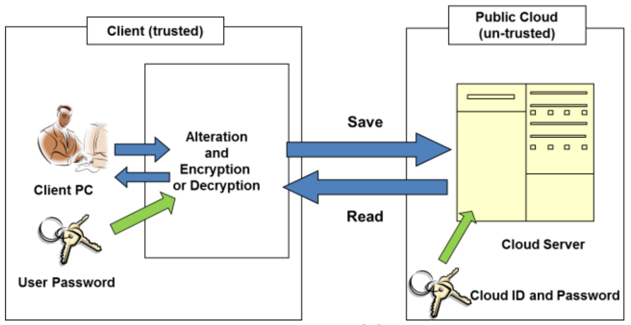


Figure 2. Trust model

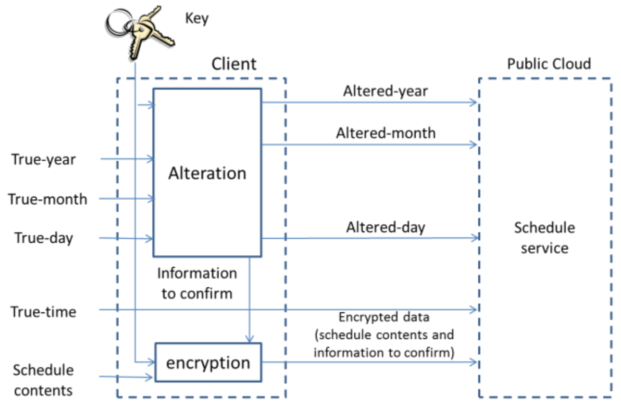


Figure 3. Saving schedule

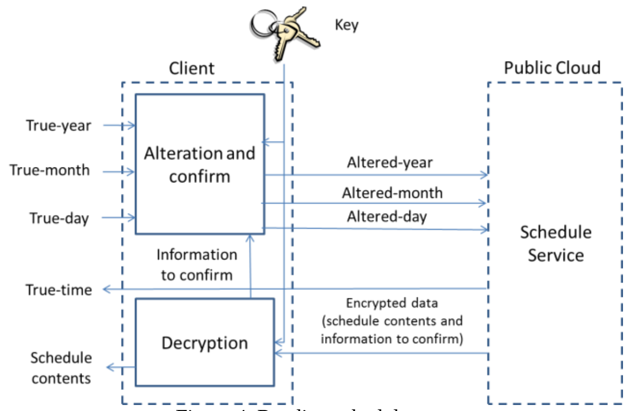


Figure 4. Reading schedule

With these technologies, server administrators may be untrustworthy third parties; therefore, privacy-preserving technologies have been necessary. In OLAP, perturbation techniques have been investigated for privacy-preserving data mining [5] [6] [7]. Using this technique, original values are perturbed and stored in a database; however, results of statistical queries remain correct. Consequently, privacy as original values is preserved. In DAS, cryptography has been commonly used to perform queries on encrypted data stored on a database [1] [8] [9].

The above techniques need to be applied to the basic functions of database systems, and it is necessary to replace or develop a new server system to use these technologies. On the other hand, we assume a current schedule server in the cloud. In this case, data types not treated on the schedule server cannot be used. This means dates need to be stored as dates in the schedule server through APIs. Therefore, dates cannot be encrypted because the binary value as an encrypted result cannot be stored in the date field in schedule databases. Furthermore, an altered date must be able to be decoded back to the original date but we need to maintain data mining results to be proper. For this reason, we developed a date alteration method.

III. CONCEPT OF ALTERATION METHOD OF SCHEDULE DATES

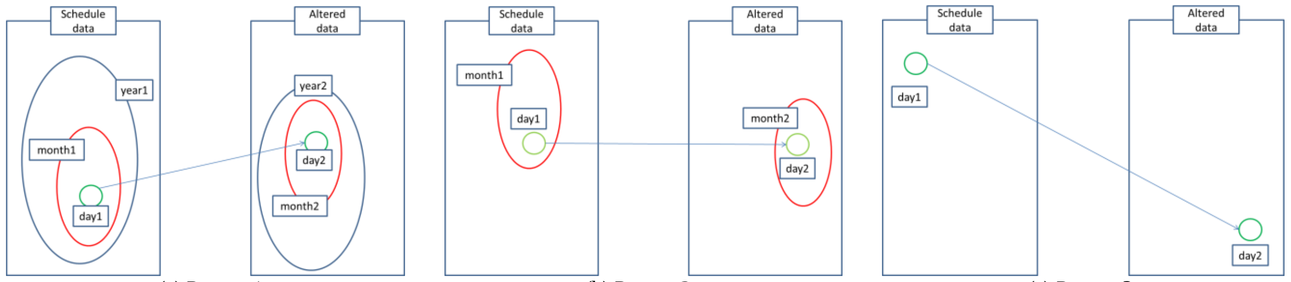
In this section, we describe the methodology and reading algorithms for altering dates.

A. *Overview of alteration method*

The process of altering schedule dates is shown in Figure 1. First, the original schedule date is prepared on the client side. Next, the original schedule is converted to a different date by using a password, which is inputted and stored on the client side. Finally, the altered date is transferred to the cloud server.

B. *Trust model*

A trust model is shown in Figure 2. We assume a public cloud is not trusted; consequently, a password of a public cloud service for account authentication is also not trusted. For example, a security aware cloud [10][11] has been proposed with this kind of trust model. For this reason, another password (Key), which is different from the original password, and alteration of the original schedule date have to be prepared. This password must be kept on the client side, and there must be alteration and encryption modules on the client PC because the PC is assumed as trusted.



(a) Pattern 1 (b) Pattern 2 (c) Pattern 3

Figure 5. Three patterns of schedule date alteration

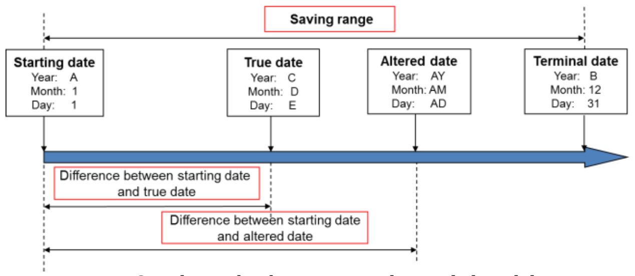


Figure 6. Relationship between true date and altered date

C. *Alteration*

A schematic of saving a schedule is shown in Figure 3. First, the original schedule data is divided into the elements of "true-year (TY)", "true-month (TM)", "true-day (TD)", "true-time", and "schedule-content" (which is the scheduled event). When the original date is altered, a Key, which is prepared by the client, is used. The TY, TM and TD are converted into “altered-year (AY)”, “altered-month (AM)” and “altered-day (AD)” by using an alteration algorithm and the Key. We only altered the TY, TM, and TD and true-time was not. Note that when the true date is altered, the information necessary to confirm the true date is created. A detailed explanation of this information is described in Section IV.

A schematic of reading a schedule is shown in Figure 4. It is assumed that the TY, TM, and TD are known by the client, and true-time and schedule-content is unknown. The TY, TM and TD are altered again using the same Key. A client accesses the altered date, the encrypted schedule contents, and true-time. The schedule is decrypted with a Key, and the schedule contents and information to confirm the true date are divided. Finally, the true date is output using the information for confirming it.

D. *Date alteration pattern*

We divided our date alteration method into three patterns, and examined the efficiency of saving and reading a schedule. These three patterns are shown in Figure 5.

In Pattern 1, the day is converted to another within the same month, the month is converted to another within the same year, and the year is converted to another in the entire range of years contained in the system. In Pattern 2, the day is converted to another within the same month and the month is converted to another in the entire range of months contained in the system. In Pattern 3, the day is converted into another day within the entire range contained in the system.

The performance and degree of vulnerability differ depending on each pattern. This is discussed in more detail in Section V.

IV. SAVING AND READIG ALGORITHMS

In this section, we describe the saving and reading algorithms of dates.

These algorithms were created using our alteration method. The relationship between true date and altered date is shown in Figure 6. The user’s password is input and converted into a series of numbers. These numbers are defined as a Key. The length of the Key should be 16 bits when using the exclusive-OR function. Moreover, when using a block cipher, the length of the key depends on the block cipher algorithm. Both algorithms are described as follows.

A. *Saving algorithm*

The saving algorithm of our alteration method is described as follows. The service’s range means the entire period of the calendar in the specific service. Note that “A”, “B”, “C”, “D”, and “E” are defined as “year of starting date”, “year of terminal date”, “year of true date”, “month of true date”, and “day of true date”.

(1) The difference between the true date and starting date of a particular service’s range is calculated.

a) Pattern 1

$$dif\_year = C - A \qquad (1)$$

b) Pattern 2

$$dif\_month2 = (C - A) \times 12 + (D - 1) \qquad (2)$$

c) Pattern 3

The number of days from the starting date to schedule date are calculated using the function $F_1(x)$. Here, $F_1(x)$ is the number of days.

$$dif\_day3 = F_1(A,1,1,C,D,E) \quad (3)$$

(2) The temporal values are calculated using the exclusive-OR function or a block cipher.

a) Pattern 1

$$EY = dif\_year \oplus Key \quad (4)$$

$$EM = (D-1) \oplus Key \quad (5)$$

$$ED = (E-1) \oplus Key \quad (6)$$

The following equations are used if a block cipher is used.

$$EY = E_{Key}(dif\_year) \quad (7)$$

$$EM = E_{Key}(D-1) \quad (8)$$

$$ED = E_{Key}(E-1) \quad (9)$$

b) Pattern 2

$$EM2 = dif\_month2 \oplus Key \quad (10)$$

$$ED2 = (E-1) \oplus Key \quad (11)$$

The following equations are used if a block cipher is used.

$$EM2 = E_{Key}(dif\_month2) \quad (12)$$

$$ED2 = E_{Key}(E-1) \quad (13)$$

c) Pattern 3

$$ED3 = dif\_day3 \oplus Key \quad (14)$$

The following equations are used if a block cipher is used.

$$ED3 = E_{Key}(dif\_day3) \quad (15)$$

(3) The calculated values in A-(2) are calculated using the "modulo function" to interpose between the service's ranges. Note that the function is to provide the remainder of the division. The value of reminders using the mod function is the altered date. The value of the quotient using the "division function" is the information to confirm the true date mentioned in Section II-C. Note that the function is to provide the quotient of the division. This information is called the Element of Read Data (ERD). ND means the number of days in one month.

a) Pattern 1

A quotient is calculated with the altered date as a result of the respective divisions. The altered date is then determined to be AY1, AM1, and AD1.

$$AY1 = \mathrm{mod}(EY, B-A) + A \quad (16)$$

$$AM1 = \mathrm{mod}(EM, 12) \quad (17)$$

$$AD1 = \mathrm{mod}(ED, ND) \quad (18)$$

$$ERD_{year} = div(ED, B-A) \quad (19)$$

$$ERD_{month} = div(EM, 12) \quad (20)$$

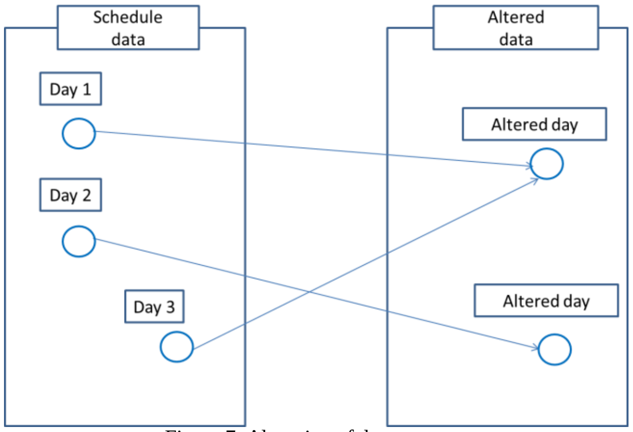


Figure 7. Alteration of dates

$$ERD_{day} = div(ED, ND) \quad (21)$$

b) Pattern 2

The year and month are combined and calculated as the number of months. SRM is the number of months in a service's range.

$$SRM = (B-A) \times 12 + 12 \quad (22)$$

$$AMN = \mathrm{mod}(EM2, SRM) \quad (23)$$

$$AD2 = \mathrm{mod}(ED2, ND) \quad (24)$$

$$ERD_{month\_number} = div(EM2, E-B) \quad (25)$$

$$ERD_{day2} = div(ED2, ND) \quad (26)$$

The altered date is determined to be AY2, AM2, and AD2.

$$AY2 = div(AMN, 12) + A \quad (27)$$

$$AM2 = \mathrm{mod}(AMN, 12) + 1 \quad (28)$$

c) Pattern 3

SRD is the number of days in a service's range. The days of the system range using $F_1(x)$ is calculated.

$$SRD = F_1(A,1,1,B,12,31) \quad (29)$$

$$ADN = \mathrm{mod}(ED3, SRD) + 1 \quad (30)$$

$$ERD_{day\_number} = div(ED3, SRD) \quad (31)$$

Using $F_2(x)$, the altered date is determined as AY3, AM3, and AD3. Here, $F_2(x)$ provides the altered-year, altered-month, and altered-day.

$$(AY3, AM3, AD3) = F_2(A,1,1,ADN) \quad (32)$$

(4) The contents of the schedule and ERD are concatenate.

(5) The above data is saved to the cloud server as the schedule contents. If necessary, the schedule contents are encrypted using the Key.

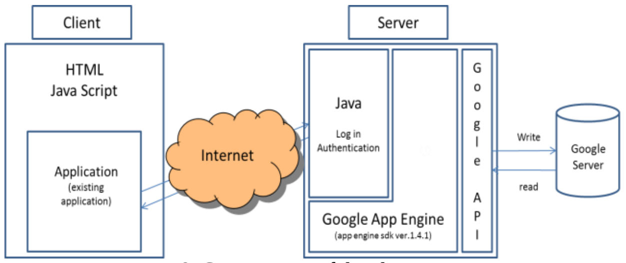


Figure 8. Composition of development setting

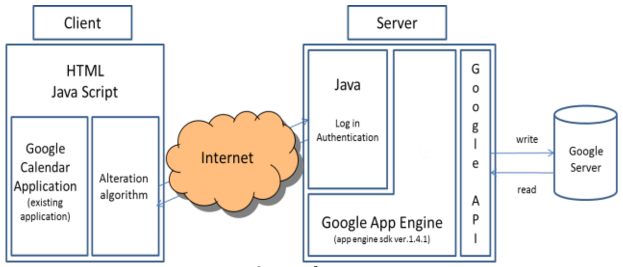


Figure 9. Final environment

B. Reading algorithm

The reading algorithm of our alteration method is described as follows.

(1) The altered date is calculated using the requested schedule date, and the saving algorithm is used to obtain the altered date.

(2) The schedule contents are read from the altered schedule date using the calendar interface.

(3) The ERD is extracted from the schedule contents and a true date is calculated. It is necessary to check the date because a true date can be altered many schedule data to one altered data, as shown in Figure 7. If this calculated date is equal to the requested date, the schedule contents become available; otherwise, they are rejected.

a) Pattern 1

The reverse order of the saving algorithm described in Section III-A is performed. The original date is calculated from the ERD.

$$TY = ((ERD_{year} \times (B - A) + AY) \oplus Key) + A \quad (33)$$

$$TM = ((ERD_{month} \times 12 + AM) \oplus Key) + 1 \quad (34)$$

$$TD = ((ERD_{day} \times ND + AD) \oplus Key) + 1 \quad (35)$$

When the block cipher is used, the true date is calculated as follows.

$$TY = (E_{Key}(ERD_{year} \times (B - A) + AY)) + A \quad (36)$$

$$TM = (E_{Key}(ERD_{month} \times 12 + AM)) + 1 \quad (37)$$

$$TD = (E_{Key}(ERD_{day} \times ND + AD)) + 1 \quad (38)$$

b) Pattern 2

The number of months from the starting date is computed using the altered year and altered month.

$$AMN = AY2 \times 12 + AM2 \quad (39)$$

The original date is calculated from the ERD.

$$TMN = ((ERD_{month\_number} \times SRM + AMN) \oplus Key) \quad (40)$$

$$TY2 = div(TMN, 12) + A \quad (41)$$

$$TM2 = \bmod(TMN, 12) + 1 \quad (42)$$

$$TD2 = ((ERD_{day2} \times ND + AD2) \oplus Key) + 1 \quad (43)$$

When the block cipher is used, the true date is calculated as follows.

$$TMN = (E_{Key}(ERD_{month\_number} \times SRM + AMN)) \quad (44)$$

$$TD2 = (E_{Key}(ERD_{day2} \times ND + AD2)) + 1 \quad (45)$$

Note that if “(44)” is applied, “(41)” and “(42)” are applied, and TY2, TM2 are calculated.

c) Pattern 3

The number of days from the starting date is calculated using the AY and AM, and the difference in the days from starting date to the altered schedule date is calculated using $F_1(x)$.

$$ADN = F_1(A, 1, 1, AY3, AM3, AD3) \quad (46)$$

The true date is calculated from the ERD.

$$(TY3, TM3, TD3) = F_2(A, 1, 1, ((ERD_{day\_number} \times SRD + ADN) \oplus Key)) \quad (47)$$

When the block cipher is used, the original date is calculated as follows.

$$(TY3, TM3, TD3) = F_2(A, B, C, (E_{Key}(ERD_{day\_number} \times SRD + ADN))) \quad (48)$$

(4) The computed TY, TM, and TD are compared with C, D, and E. If the two values are equal, the calculated schedule becomes available.

V. EVALUATION AND DISCUSSION

A. Implementation

We developed alteration modules on "Google Calendar" [12], which is a type of software as a service (SaaS). We did not use an existing calendar application, but we used the Google Calendar API [13] to evaluate the algorithm. The environment of this module is shown as Figure 8.

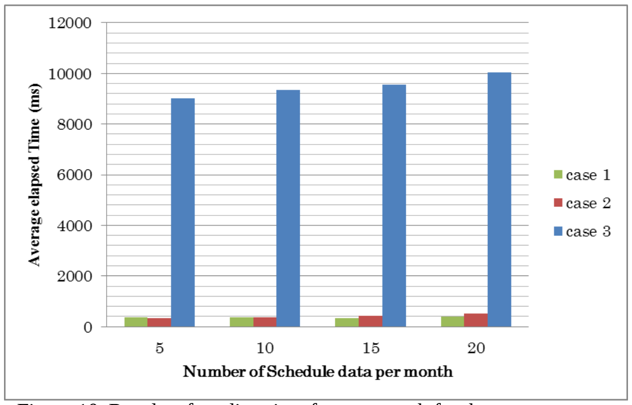


Figure 10. Results of reading time for one month for three patterns

TABLE I. RESULTS OF READING TIME FOR ONE MONTH FOR THREE PATTERNS

Number of schedule dates per month	*Pattern 1 (ms)*	*Pattern 2 (ms)*	*Pattern 3 (ms)*
5	353	334	9000
10	370	359	9333
15	340	418	9590
20	409	523	10049

In actual use, Google Calendar applications should be used for building the proposed algorithm into the application for user convenience, as shown in Figure 9.

B. Performance Evaluation

We evaluated the performance of the proposed method under the three patterns mentioned in the previous section, using the implemented modules discussed in the previous subsection.

B-1) Reading time for a month in all three cases.

B-2) Reading time for a year in Patterns 1 and 2.

We did not evaluate the performance of saving schedule data because there was no difference in performance among the three patterns. Therefore, we only evaluated performance of reading schedule data. When the schedule data is read, it is usually appropriate to read the data for one month or one week. In other words, it is not practical to use reading data for one year. However, by comparing the performances of the algorithm, the schedule data was read for a year.

For B-1, all schedules within the specified month given by a user are read, and the elapsed time of reading the schedules was evaluated in the three alteration patterns.
The measured results for B-1 are shown in Figure 10. The elapsed time of Pattern 3 was much longer than those of the other two patterns. Patterns 1 and 2 produce the schedule for a month when making only one API call. On the other hand, Pattern 3 produce the schedule for a month by making an API call which is based on the number of days in a month.

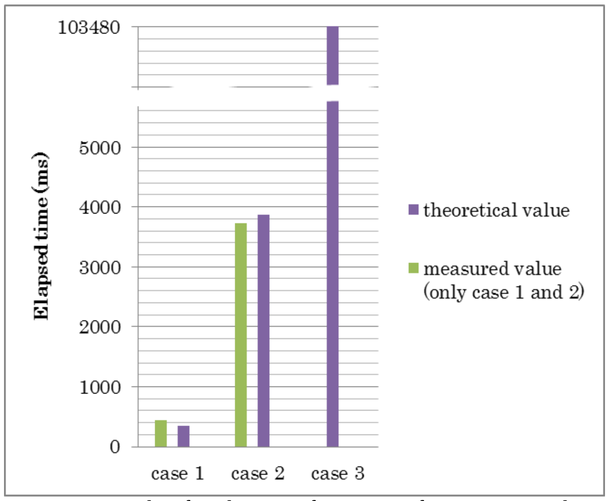


Figure 11. Results of reading time for one year for Patterns 1 and 2, and calculated theoritical value for Pattern 3

TABLE II. RESULTS OF READING TIME FOR ONE YEAR FOR PATTERN 1 AND 2

Number of schedule dates Per month	*pattern 1 (ms)*	*pattern 2 (ms)*
20	442	3731

There were not many differences in the elapsed time for Pattern 3 when the number of schedule dates included in one month increased. As a result, the elapsed time did not much depend on the number of schedule dates in B-1 because the number of API calls increased during the elapsed time.

Equation "(49)" is derived from Figure 10. Note that "T" is the elapsed time.

$$T = \quad t_n \times (A - x) + t_e \times x \qquad (49)$$

t_n : The elapsed time when there are no date for one API call.

t_e : The elapsed time when there is a date for one API call.

x : The number of API calls

A : The maximum number of API calls

Here, t_n and t_e are calculated from the elapsed time, which is measured for a month (maximum number of days is 31). Note that t_n and t_e are assumed to be constant numbers.

For calling schedule dates of a month, we substitute $A = 31$, $x = 5$, and $T = 9000$ ms into "(49)".

$$t_n \times (31 - 5) + t_e \times 5 = 9000 \qquad (50)$$

When $x = 10$,

$$t_n \times (31 - 10) + t_e \times 10 = 9300 \qquad (51)$$

From "(50)" and "(51)", t_n and t_e are calculated as

$$t_n = 280\,(\mathrm{ms}) \quad (52)$$

$$t_e = 344\ (\mathrm{ms}) \quad (53)$$

By substituting “(52)” and “(53)” into “(49)”, T is represented as

$$T = 280 \times (A - x) + 344 \times x \quad (54)$$

When substituting $x = 20$ to confirm the elapsed time,

$$T = 280 \times (31 - 20) + 344 \times 20 = 9960\,(\mathrm{ms}) \quad (55)$$

The calculated value was close to the measured value.

For B-2, we compared the elapsed time of reading schedule data for a year. The measured results for B-2 are shown in Figure 11. The elapsed time of Pattern 2 was longer than that of Pattern 1. This pattern obtained the schedule for a year with one API call. On the other hand, Pattern 2 obtained the schedule for a year with twenty API calls. Only the theoretical value of Pattern 3 was published in Figure 11 at this time because this pattern was not able to make API calls for a year in the experimental environment.

The theoretical value of the elapsed time was calculated. Using “(54)”.

The elapsed time of pattern 1 is

$$T = 280 \times (1 - 1) + 344 \times 1 = 344\,(\mathrm{ms}) \quad (56)$$

The elapsed time of pattern 2 is

$$T = 280 \times (12 - 8) + 344 \times 8 = 3872\,(\mathrm{ms}) \quad (57)$$

The elapsed time of pattern 3 is

$$T = 280 \times (365 - 20) + 344 \times 20 = 103480 \quad (\mathrm{ms}) \quad (58)$$

When “(56)”, “(57)” and Table II were compared, the actual measurement and theoretical values were close.

According to Tables I and II, the elapsed time was at most 10 seconds. In a cloud environment, it is thought that the processing time increases. Therefore, an elapsed time of 10 seconds is appropriate for practical use. However, since the theoretical value of Pattern 3 was at most about 103 seconds, it is not practical for reading schedule dates for a year.

The elapsed time is proportional to the number of API calls. Therefore, it can be said that Pattern 1 with API calls is superior to the others patterns in terms of performance. Therefore, the degree of module performance is higher in reverse order, Pattern 1 > Pattern 2 > Pattern 3.

To improve the elapsed time, it is necessary to develop an algorithm for reducing the number of API calls.

C. *Security of alteration*

A possible attack is described as follows.

First, the original schedule date may be predicted by a third party. As mentioned in Section II, Pattern 3 is the safest pattern because a date is mapped throughout the schedule range. Alteration of Pattern 2 is performed day to day within a month. Therefore, altered date distribution does not change month to month compared to the true distribution. Therefore, if a hacker observes a newly added schedule, he may know what month the current month is because the current month must be the month that the number of added schedules is largest. Therefore, it is difficult to guess the true date.

A user prepares a password (Key) beforehand for altering the date. However, if the same key is used for a long time, the schedule date range will not be large; therefore, a hacker can predict the key by brute force. To prevent this kind of attack, a user should periodically change the key. When the key is changed, it is necessary to simultaneously calculate the ERD again.

Second, the original schedule might be guessed from the altered schedule. A schematic of guessing the true schedule from an altered one is shown in Figure 12. For instance, there are many schedules in 2011, and there were few in 2010. First, 2010 and 2011 are altered to 2030 and 2005, respectively. It is assumed that the server administrator knows the schedule frequency in 2010 and 2011 of a client. The server administrator investigates the altered schedule frequency in 2030 and 2005 and compares each year. If it turns out that 2005 had many schedules, it may turn out the 2005’s altered schedules are equal to those of 2011. This is the same not only for the combination of "year and month" but also "month and day." Therefore, Pattern 3 is safest, and Pattern 2 is safer than Pattern 1.

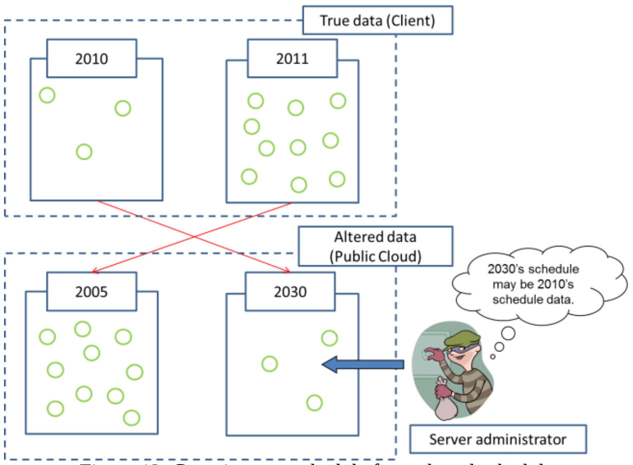

True data (Client)
2010
2011
Altered data (Public Cloud)
2005
2030
2030's schedule may be 2010's schedule data.
Server administrator

Figure 12. Guessing true schedule from altered schedule

TABLE III. AVAILABLE FUNCTIONS AND APIS

Function of current schedule APIs	***Availability of alteration***
Create an event	○
Create a new calendar	○
Repeat an event	○
Privacy settings for individual events	○
Edit or view event details	○
Delete or remove an event	○
Delete a calendar	○
Events that last all day	○
Color Code an Event	○
Edit your calendar name	○
Notifications (Daily Agenda)	×
Notifications (Event Reminders)	×

D. Available APIs

Schedule dates are all altered on the Google Calendar server, so some APIs might not perform properly. Existing Google Calendar APIs are listed in Table III. Existing schedule services have many functions. For instance, a notification API will be set at the date after alteration. As a result, an alarm will be activated on the wrong date.

VI. CONCLUSION AND REMARKS

There are various advantages in cloud computing; however, there are still many security problems. We proposed an alteration method to protect private or confidential information from third party tapping. We also implemented two alteration modules and evaluated the proposed method's performance in three patterns on Google calendar. We found that the method's performance was lower than usage without alteration, but it is still useful. We plan to develop a better performing algorithm in the future.

For actual use, Google Calendar applications should be used with the proposed modules built into the application for user convenience. This is for future work.

REFERENCES

[1] C. Almond, "A Practical Guide to Cloud Computing Security", Accenture and Microsoft, August 27, 2009

[2] "Public Cloud Computing Security Issues". [Online] Available:http://www.thebunker.net/managed-hosting/cloud/public-cloud-computing-security-issues/, <retrieved: November, 2011>

[3] D. Yuefa, W. Bo, G. Yaqiang, Z. Quan, and T. Chaojing, "Data Security Model for Cloud Computing", ISBN 978-952-5726-06-0, Proceeding of the 2009 International Workshop on Information Security and Application (IWISA 2009) Qingdao, China, November 21-22, 2009

[4] BalaBit IT security, "Cloud Security Risks and Solutions", First Edition, July 1, 2010

[5] R. Agrawal, R. Strikant, and D. Thomas, "Privacy Preserving OLAP" Proc. 25th ACM SIGMOD Int'1 Conf. Management of Data, ACM Press, 2005, pp. 251-262

[6] N. Zbang and W. Zbao, "Privacy-Preserving Data Mining Systems", IEE Computer Society Computer, April 2007, pp. 52-58

[7] J. Vaidya and C. Clifton, "Privacy-Preserving Data Mining: Why, How, and When", IEEE Security&Privacy Building Confidence in a Networked World, November/December, 2004

[8] D. Boneh, G. Crescenzo, R. Ostrovsky, and G. Persiano, "Public Key Encryption with Keyword Search", Proceedings of EUROCRYPT '04, vol. 3027

[9] Z. Yang, S. Zhong, and N. Wright, "Privacy-Preserving Queries on Encrypted Data", Proceedings of the 11th European Symposium On Research In Computer Security (Esorics), LNCS4189, pp. 479-495, 2006.

[10] H. Sato, A. Kanai, and S. Tanimoto, "A Cloud Trust Model in Security Aware Cloud", Proceedings of 10th International Symposium on Applications and the Internet (SAINT 2010), pp. 121

[11] H. Sato, A. Kanai, and S. Tanimoto, "Bulding a Security Aware Cloud by Extending Internal Control to Cloud", Proceedings of 10th International Symposium on Autonomous Decentrakized Systems (ISADS 2011), 2011.

[12] Google, "Date API Developer's Guide".[Online] Available:http://code.google.com/intl/ja/apis/calendar/data/2.0/developers_guide.html <retrieved: November, 2011>

[13] Google, "Calendar help". [Online] Available: http://www.google.com/support/calendar/ <retrieved: November, 2011>