Electronic Preprint for Journal of Information Processing Vol.23 No.4

Regular Paper

Strict Application Execution Control with Hierarchical Group Management Using Digital Certificates on Educational Windows PCs

Daisuke Okamoto[1,†1,a)] Keita Kawano[1] Nariyoshi Yamai[2] Tokumi Yokohira[1]

Received: September 16, 2014, Accepted: March 4, 2015

Abstract: We have developed a system (traditional system) to flexibly provide the requested applications environment on educational Windows PCs. The traditional system dynamically controls the execution of applications installed on each educational PC depending on the rules defined by teachers as well as by administrators. The traditional system, however, has a low tolerance for malicious attacks. If the execution file of a certain application is falsified, the corresponding rules already applied become invalid. In addition, though the traditional system has a function to define groups of controlled applications, it does not support hierarchical groups. This reduces the usability of the traditional system. In order to address these issues, this paper proposes a control method of application execution using digital certificates. The proposed method has a high tolerance for the falsification of execution files by controlling their executions based on the reliability of the corresponding digital certificates. It also improves its usability by introducing hierarchical group management utilizing hierarchical structure for digital certificates.

Keywords: application execution control, digital certificate, educational PC

1. Introduction

In many educational institutions, PCs used by students for education are arranged into several rooms of scattered facilities to make usage more convenient for students [1], [2], [3]. Those PCs are called "educational PCs" and provide a common user environment to the students [3], [4]. To achieve this efficiently, educational PCs are usually managed using one (or some) common disk image.

Since all the administrators have to do is to maintain the common disk image and reflect it into all the educational PCs, their burden of management has been reduced. This management, however, has a drawback of lacking the flexibility to provide an individual applications environment depending on the requests from teachers.

Therefore, we have developed a system (traditional system) to control the execution of applications on educational Windows PCs, as previous research [5], [6], [7]. The traditional system allows or prohibits the execution of applications on each educational PC depending on individual requests. Even when immediacy is required, teachers can change the applications environment in their classroom in real time, with a configuration tool for teachers.

However, the execution rules already set become invalid if the falsification of execution files occurs in the traditional system. The traditional system uses hash values of the corresponding execution files to identify each application. The hash value here, is a 32-bit unique value calculated from each execution file. If a certain execution file is falsified, its hash value changes, resulting in the invalidation of the corresponding rules.

Moreover, although the traditional system has a function to define groups of controlled applications, it does not support hierarchical groups [6]. A group cannot include other groups in itself. If a teacher wants to control the execution of applications in multiple groups, the teacher needs to set the rules to the corresponding groups one by one. This reduces the usability of the traditional system.

To solve these problems, this paper proposes a control method of application execution using digital certificates. The proposed method can defeat the falsification of execution files and realize hierarchical applications group management. When a student starts up an application on their educational PC, a program of the proposed method verifies the reliability of the certificate used to certify the application. If its reliability is ensured, the student can use the application.

Furthermore, the proposed method applies hierarchical structure for certificates to application execution control. The proposed method can control the execution of applications in an arbitrary hierarchy of groups by trusting or distrusting the corresponding certificates.

The rest of this paper is organized as follows. Section 2 discusses related work. Section 3 describes an outline and problems of the traditional system. Details of the proposed control method using digital certificates are described in Section 4. The results of several experimentations are shown in Section 5 to confirm the

[1] Okayama University, Okayama 700–8530, Japan
[2] Tokyo University of Agriculture and Technology, Koganei, Tokyo 184–8588, Japan
[†1] Presently with KDDI CORPORATION
[a)] daisuke.net@s.okayama-u.ac.jp

feasibility of the proposed method. Section 6 summarizes this paper and describes some future work.

2. Related Work

The administration of large educational systems is one of the important missions of the information centers in educational institutions. They have constructed their educational systems by employing various solutions to overcome their individual problems and satisfy their individual needs [8].

Providing a common user environment on all the educational PCs is important to increase convenience for students [3], [4]. To achieve this efficiently, most educational systems have some kind of image-based centralized management system [2], [3]. Using such a system, one common disk image can be easily reflected into all the educational PCs. Some universities have maintained multiple common disk images to satisfy individual requirements from different departments [2], [9]. This method, however, increases the burden of system management while increasing user satisfaction.

As an older but simple way to reflect disk images, there is a method to distribute the latest disk image and replace disk images of educational PCs with it at night. Our current system employs this type of centralized management. Some universities employ a newer way to distribute the latest disk image, called NetBoot [2], [9]. This type of centralized management system allows educational PCs to get the latest disk image with high-speed network when they boot up. Some universities employ the NetBoot system with diskless thin clients in consideration of fault tolerance [4]. Other universities employ the NetBoot system with local hard disks [2], [9]. In educational systems, a large number of PCs start up almost simultaneously at the beginning of classes. To reduce the start-up time, local hard disks are used to cache the present disk image.

In addition, some progressive universities have introduced a method called Virtual Desktop Infrastructure (VDI) [3], [4], [10]. Educational PCs run on a high-spec virtualized infrastructure, and students use them through local PCs like thin clients. Basically, VDI system has no restriction of the type of clients. Students can use their own devices to access the virtual educational PCs. Utilizing this feature, some universities implement the "Bring Your Own Computers" policy to reduce system introduction cost and to promote studies at home or elsewhere [11].

As a method to construct a virtual educational PCs environment with cloud computing platform, Apache Virtual Computing Lab (VCL) has developed [12], [13]. Some universities are trying to share their computing resources among multiple universities to reduce system introduction and management cost [13], [14]. Other universities are trying to allow teachers, or perhaps students, to create their own disk images and maintain them [15], [16], [17]. The latter provides management flexibility when the teachers, or the students, have a certain amount of skill and knowledge about computers.

Moreover, in many universities having a variety of departments, it is necessary to use multiple OSs (including different versions of the same OS) on educational PCs, such as the Microsoft Windows family, Apple OS X, and Linux [18], [19], [20], [21]. To address this issue, some universities employ the multiboot feature while other universities introduce the virtualization feature in which one OS runs on another OS [1], [2], [4], [22]. This paper, however, focuses on educational PCs having only a single version of Windows, for implementation reason and for simplicity [*1].

As described above, Educational systems have various needs and many solutions for administering them have been introduced. Even with those solutions, however, an application environment on each educational PC cannot be changed after its start-up. Our solution is applicable in many educational systems employing the above solutions.

On the other hand, application streaming is another method to realize individual application environments [23], [24], [25]. In this method, some streaming servers have virtualized applications, and educational PCs get each application from the servers and use it when they need. This method, however, needs powerful streaming servers to handle simultaneous requests from educational PCs at the same class.

There are some vendor solutions to control the execution of application installed on Windows PCs [26], [27], [28], [29]. As far as our investigation, however, no solutions use digital certificates to identify each application. For instance, from their web site, we can guess one solution uses hash values like our traditional system [26].

3. Traditional System

In this chapter, an outline of the traditional application execution control system we have developed is described. After that, two problems of the traditional system are shown.

3.1 Outline of Traditional System

As mentioned above, educational PCs are typically managed using a centralized management system with one (or some) common disk image [7]. However, this system cannot reflect the modification of the image in real time. Moreover, the administrative burden increases in proportion to the number of images.

In order to solve these problems, we have developed an application execution control system (traditional system) [5], [6], [7]. The traditional system controls user's application execution by setting control policies. The settings of the policy can be done in real time even in a class. If an application which is tried to be used by a student has a limitation on the number of license, the traditional system determines the application usage depending on the number of remaining license. This system can immediately and flexibly construct different application environments.

The traditional system consists of four programs. The first is the Configuration Tool. It is a program for teachers to set the policy of application usage. The second is the Execution Control Program. It is a program to actually control the use of specified application on each educational PC. In addition, there is a program for administrators to configure group control, called Management Tool. The last is a server program to store the rules and direct whole operation, called Policy Decision Server. It has a database (Policy Database) which retains execution rules of ap-

[*1] Actually, the OS on all of our current educational PCs is Windows 7.

plication.

The operation of the traditional system when a teacher sets a control rule is shown in **Fig. 1**.

As shown in Fig. 1, each student PC has Execution Control Program, teacher PC has Configuration Tool, and administrator PC has Management Tool. When a student logs in to an educational PC, the Execution Control Program receives initial control rules from the Policy Decision Server and reflects them. Then, if the teacher asks the server to modify the execution rules, the modification is notified to the Execution Control Program on the corresponding student PCs and is reflected.

The traditional system uses a function of Windows called "Group Policy," to control the execution of applications. With a feature of the Group Policy, application start-up can be controlled by changing a registry value [30]. A hash value is used to identify each application. The hash value is a hexadecimal unique value calculated from each execution file. This value has a feature that if the input data is changed, the output data is also changed [31].

Moreover, the traditional system has default-state configuration function and grouping function to control multiple application at once. With the default configuration function, a teacher can set default state which is a rule for all non-specific applications. If no default-state is set, "default-permission" which allows all non-specific application execution is chosen as standard configuration. To achieve "default-prohibition," a prohibition rule using wildcard of "path rule" [32] is set. At this time, by setting individual permission rule, the teacher can allow the use of application that the teacher wants.

Furthermore, the grouping function can control multiple applications at once by registering some applications as a group and setting a control rule to the group. If an administrator has created a group of control target applications using the Management Tool, teachers can control the execution of multiple applications simultaneously using the group definition.

3.2 Problems of Traditional System

This section describes two problems of the traditional system.

3.2.1 Low Tolerance for Falsification of Execution Files

As mentioned above, the hash value is used to identify each application in the traditional system. Students having expert knowledge, however, can change the hash value of the execution file of a certain application by rewriting its binary data using a kind of binary editor. If a calculated hash value is changed, control rules already set are no longer valid.

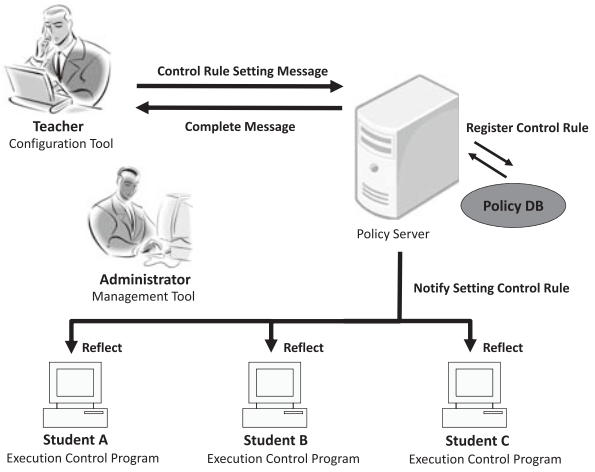

Fig. 1 The operation of the traditional system when a control rule is set.

Under the default-permission state, individual prohibition rules for specific applications are usually defined. If a student rewrites the binary data of the execution file of a corresponding application, the hash value of the defined prohibition rule and the hash value of the execution file can be different. Then the rule becomes invalid and the student can execute the application that the student wants to use.

Under the default-prohibition state, individual permission rules for specific applications are usually defined. If a student rewrites the binary data of the execution file of a corresponding application, the hash value of the defined permission rule and the hash value of the execution file can be different. Then the rule becomes invalid and the student can prevent the execution of the application that is inconvenient for the student but is needed for the administration. Normally, this type of falsification should be restricted with proper permission management. More of this is described in Section 4.1.

An example of actual falsification is shown below. The execution prohibition rule of the corresponding application has been set in advance. When a user attempts to start up the application, the message shown in **Fig. 2** is displayed and it cannot be executed. After that, assume that the user falsifies an unused part in binary data of the execution file, and attempts to start it up again. This time, no message is displayed, and the application can be executed. This is because the hash value has changed and the rule has become invalid.

3.2.2 Lack of Hierarchical Group Management

As described in previous chapter, the traditional system has a grouping function. This function can reflect a single rule into multiple applications at once. However, it was designed for the purpose of simple grouping (e.g. different versions of applications). A group cannot include other groups into itself. With the current Configuration Tool, if a teacher wants to control multiple groups at once, the teacher has to set control rules to the groups one by one. The burden on the teacher increases in proportion to the number of the target groups. An example of traditional grouping is shown in **Table 1**.

As in Table 1, the group "IE" contains the application "Internet

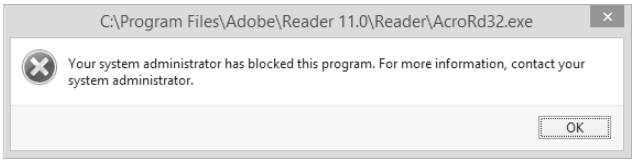

Fig. 2 The execution prohibition message.

Table 1 Group information.

Group Name	Member
IE	Internet Explorer 11.0
	Internet Explorer 11.1
	Internet Explorer 11.2
Firefox	Firefox 29.0
	Firefox 29.1

Explorer 11.0," "Internet Explorer 11.1," and "Internet Explorer 11.2." The group "Firefox" contains the application "Firefox 29.0" and "Firefox 29.1." At this time, suppose that the administrator attempts to create a group "browser." The administrator needs to select 5 applications one by one since the administrator cannot specify the groups "IE" and "Firefox" as the group members. In addition, a burden of management increases because the administrator has to maintain these duplicate information. If the application "Firefox 29.2" is newly registered as target application at this time, the administrator needs to register it to the group "browser" in addition to the group "Firefox" as each group member.

4. Proposed Control Method Using Digital Certificates

In order to solve the problems described in Section 3, this paper proposes a new control method of application execution using digital certificates. An outline of the proposed method is shown in Section 4.1. Then, Sections 4.2 and 4.3 describes the detailed functions of the proposed method. The operation flow of the proposed method is shown in Section 4.4.

4.1 Outline

To solve the problems described in Section 3, new functions to defeat the falsification of execution files and to control the groups flexibly are needed.

To defeat the falsification of execution files, their permissions have to be managed properly in educational systems. Except for personal areas, no files should be changed by students. This method is always effective to protect applications installed by administrators. Our proposal also assumes this setting. However, the administrators cannot control permissions of execution files of applications installed on the personal area by students. This allows students to falsify these execution files and cheat on the system.

As for hierarchical group management, there is a way to simply extend the traditional way of management. Hierarchical group information can be managed together with associated application information on the Policy Database. However, when actual modification of control policies for a certain group is enforced with the Execution Control Program, the setting for individual application in the group has to be changed iteratively with this method. The number of iteration and the amount of associated control messages increase linearly as the number of group members grows.

This paper introduces two functions to address these problems. A function called "Defeat Falsification Function" controls the start-up of each execution file based on the reliability of the corresponding digital certificate. In addition, a function called "Hierarchical Control Function" constructs the hierarchical structure for certificates, and controls multiple groups at once.

First, the Defeat Falsification Function verifies the reliability of the digital certificate used to certify each execution file each time the associated application starts up. If its reliability is not ensured, the execution is to be prohibited. In this way, the proposed system can prevent malicious student from avoiding the rules.

Next, the Hierarchical Control Function applies hierarchical structure for digital certificates to application execution control. By switching a reliability, which indicates whether the certificate can be trusted or not, of an upper layer certificate, the execution of multiple applications with lower layer certificates can be controlled at once.

In order to achieve these functions, we have to add some features to handle digital certificates to the traditional system. First, a function which modifies digital signatures, which include the information of the digital certificates of their signers, on execution files of applications has to be added to the Management Tool. Next, we have to allow the Execution Control Program to control application execution based on the digital certificates. In addition, we have to modify the Policy DB to store certificate rules.

4.2 Defeat Falsification

In general, digital signatures are used for the purpose of ensuring that the data contains no falsification. The digital signature is able to be appended to execution files of applications, and there are many such products [33]. If a digital signature appended on an application is verified and the result is invalid, it determines that there is a data rewriting. However, as far as we have tested, even if the falsification occurs, any error messages are not displayed for local applications unless we check the information of the digital signatures from their properties. The proposed system aims to prohibit the use of such falsified applications. Only the applications certified by the signers with reliable certificates can be executed.

Note here that, in the Group Policy, not only hash values ("hash rules" [34]) but digital certificates ("certificate rules" [35]) are also supported as the identifier for execution control [36]. The identification is based on the digital certificate of the signer of the digital signature. If a rule for an application is defined using the corresponding certificate, other applications which are certified by the signer with the same certificate are affected by the rule at the same time. For instance, because signers of the Microsoft Office suite (e.g., Excel, Word and PowerPoint) are the same, single certificate rule is applied for these applications. Thus, for individual application execution control, we add a function which overwrites the digital signature in the proposed method. By setting rules using different certificates of different signers, strict application control can be supported.

The Group Policy actually controls application execution by verifying the reliability of the corresponding certificates. The reliability of the certificates is switched depending on the control rules. If the execution of applications associated with the certificate is configured as permission, the certificate is set to valid. On the contrary, if it is configured as prohibition, the certificate is set to invalid. In fact, the information of the certificate is registered to a special part of the registry, named "Certificate Stores" [37]. The important items to switch the reliability of the digital certificates are shown in **Table 2**.

Table 2 The import destination to switch the reliability in the certificate stores.

item's name	which certificate will be imported
Trusted Publishers	certificates with permission rule
Disallowed Certificates	certificates with prohibition rule

As shown in Table 2, if a permission rule using certificates is set in the Group Policy, the certificate information is automatically imported into "Trusted Publishers" to set the certificate valid. On the contrary, if a prohibition rule is set, it is imported into "Disallowed Certificates" automatically to set the certificate invalid. Thus, we let the proposed method configure the Certificate Stores properly to control the application execution.

As mentioned above, the traditional system can switch a mode (default state) which prohibits or permits the execution of all non-specified application. Under the state of the default-prohibition, the execution of the falsified application is not allowed because its digital signature is invalid [*2]. On the other hand, the behavior varies with falsification parts of the execution file of the application when the state is the default-permission and a prohibition rule for that application using the digital certificate is set. If the unused parts of the execution file are falsified, the prohibition rule is applied because the signature parts has no changes. However, if the signature parts are falsified, the certificate information is not matched with the prohibition rule which is already registered. Therefore, a user can execute the application even if the prohibition rule using the digital certificates is set under the state of the default-permission.

For preventing the execution of the falsified applications like that, we modify the traditional default-states. We rebuild the default-permission state by allowing the execution of all the applications by certifying them with trusted certificates. For realizing this state, a prohibition rule using wildcard of "path rule" (all-prohibition rule) is set and the execution of all applications is set to permit using the permission rules using certificates [32]. Herewith, even if the signature parts of the execution file are falsified, the system can prohibit the execution of the application because the above all-prohibition rule is applied.

Actually, this way of rebuilding can be also applied to the traditional execution control using hash values. We, however, still use digital certificates instead of hash values mainly because we utilize the hierarchy of digital certificates for hierarchical control to be described in the following section. In addition, using the above described feature, i.e. rules are defined with the signer of digital signatures, we can reduce the total number of rules under the rebuilt default-permission state. Some applications that are required for the administration of educational systems but changing their execution policies are not needed during classes can be signed by the same signer and can be controlled at once.

4.3 Hierarchical Control

There are some certificates to use for application execution control. First, the certificates which are used to sign applications called "End certificate." Next, the "Intermediate certificate" is for certifying the End certificates or the lower layer Intermediate certificates. At last, the "Root certificate" locates in the top of the structure and certifies the reliabilities of lower layer certificates.

Note here that, the system cannot control application execution using the digital certificates only if these certificates are just created. Administrators have to import these certificates into the proper part of the Certificate Stores in advance. The important items for the proposed method to validate each certificate are shown in **Table 3**.

As shown in Table 3, the items of import destination differ depending on the kinds of certificates. First, the certificates created as the Root certificates becomes valid by importing into "Trusted Root Certification Authorities." Next, the Intermediate certificates are imported into "Intermediate Certification Authorities." Finally, the End certificate can be handled as the certificate which can be controlled in the Group Policy, by importing them into "Other People."

As mentioned above, the Intermediate certificate ensures that the End certificate is reliable. If the Intermediate certificates are invalidated (imported into "Disallowed Certificates"), the lower layer End certificates are also invalidated at the same time. And then, if the Intermediate certificates are re-validated (deleted from "Disallowed Certificates"), its End certificates are also re-validated. In other words, the reliability of the Intermediate certificate is inherited to the lower layer End certificates.

The proposed system applies this feature to the application group control. We achieve the prohibition of application group, by invalidating the certificates of the hierarchy level corresponding to the application groups.

As described above, under the default-permission state, if the upper Intermediate certificate is invalidated, application cannot start up even if the End certificate is set a permission-rule (imported into "Trusted Publishers"). The rule is able to re-set to permission by deleting the Intermediate certificate from "Disallowed Certificates."

Under the default-prohibition state, however, the behavior is different from previous case. In the Group Policy, even if the Intermediate certificate is just imported into "Trusted Publishers," an execution of application certified by the lower layer End certificate is not allowed. Unless each End certificate is set a permission-rule (imported into "Trusted Publishers"), the application execution is kept as prohibition. As shown in here, the process that permits the group in the default-prohibition state is difficult to realize only using hierarchical structure at this stage. We need more considerations of realizing a function that permits the application execution for the group by reliability operation using hierarchical structure for certificates.

To achieve above execution control, the proposed system allows the administrator to construct a certification relation of the End certificates and the Intermediate certificates flexibly. By using this way, the proposed method can support group control more freely than the traditional method, since the number of the Intermediate certificates is able to be configured as administrators like. By changing the certification relation between the Intermediate certificates and the End certificates, the proposed method

Table 3 The import destination to validate certificates in the Certificate Stores.

item's name	which certificate will be imported
Trusted Root Certification Authorities	Root certificates
Intermediate Certification Authorities	Intermediate certificates
Other People	End certificates

[*2] The execution of applications with invalid digital signature is prohibited by the default-prohibition rule even if the permission rule using the corresponding certificate is defined.

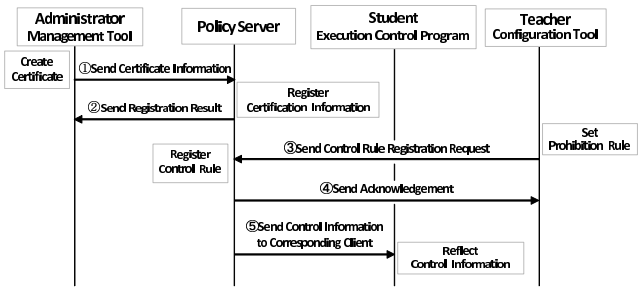

Fig. 3 The operation flow of the proposed method.

can change a range of applications controlled execution.

We have to add some function to create and sign the certificates to the Configuration Tool and the Management Tool, for preparing the certificates used to control execution. We are thinking of using the Windows SDK published by the Microsoft Corporation. This is a Software Development Kit to create applications operated on the Windows PC. This kit includes some execution files such as "makecert" for creating the certificate, "signtool" for signing to each file, and so on [38].

4.4 Operation Flow

The operation flow of the proposed method is shown in **Fig. 3**. In advance, an administrator creates certificates and appends them to the execution control target applications. Then, a teacher prohibits a certain application. After this, the Educational PC reflects its rule.

An operating procedure is concluded as follows.

(an administrator creates certificates by the Management Tool)

(1) The Management Tool sends these certificate information to the Server Program.

(2) The Server program registers the certificate information into the corresponding tables, and sends a message indicating completing the registration.

(a teacher sets a prohibition rule by the Configuration Tool)

(3) The Configuration Tool sends the rule to the Server Program.

(4) The Server Program registers the rule to the corresponding table and sends a message indicating the registration completed to the Configuration Tool.

(5) The Server Program notifies the control information modification to the corresponding Execution Control Program. Then, the Execution Control Program reflects it.

5. Experiment

We experimented to verify the efficacy and feasibility of the control method using digital certificates described in Section 4.

5.1 Defeat Falsification

We examined the behavior when the execution file of a control target application is falsified. First, we falsify the execution file of the application which is set a prohibition rule of the certificate. Then, we confirm keeping the execution prohibited. Here, the behavior varies depending on where is replaced in the execution file. Thus, we experimented to falsify two parts in the file. The first is the unused bytes which have no impact to the operation of the application even if it was re-written. And the other is the appended certificate part.

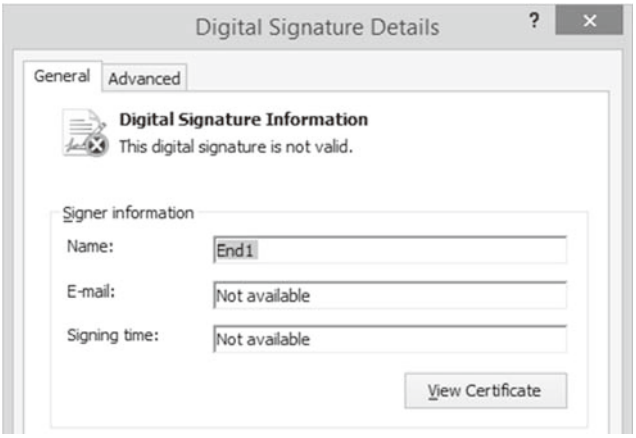

Fig. 4 Certification information after falsification of unused bytes.

- (Experiment 1.1) Falsification of the unused bytes

  [Initial State]

  All-prohibition rule using wildcard is set. An execution file "testapp1.exe" is signed with an End certificate "End1" (publisher: "InterCA1"). A rule which prohibits execution of application signed with the "End1" is set by using the Group Policy.

  [Procedure]

  (1) Confirm that testapp1.exe cannot be started up.

  (2) Replace unused bytes in the execution file.

  (3) Just the same as above, confirm that testapp1.exe cannot be started up.

  (4) Check the state of digital signature.

  [Result]

  The message indicating execution prohibition was displayed, when we attempt to start up the application after falsifying the execution file. The prohibition rule had been retained, since the rule was set by using a certificate instead of a hash value as the identifier. There were no changes in the certificate part of the binary data.

  After that, we confirmed the state of the digital signature from properties of the execution file, and a message indicating that the signature is not valid as shown in **Fig. 4** was displayed. This is because the hash value calculated from the binary data and the value included in the signature were not matched.

- (Experiment 1.2) Falsification of certificate part

  [Initial State]

  As same as the Experiment 1.1, an execution file "testapp1.exe" is signed with an End certificate "End1" under the state of all-prohibition rule using wildcard is set. A rule which prohibits execution of application signed with the "End1" is set by using the Group Policy.

  [Procedure]

  (1) Confirm that testapp1.exe cannot be started up.

  (2) Rewrite the bits of certificate part in the execution file.

  (3) As same as previous, confirm that testapp1.exe cannot be started up.

  (4) Check the state of digital signature.

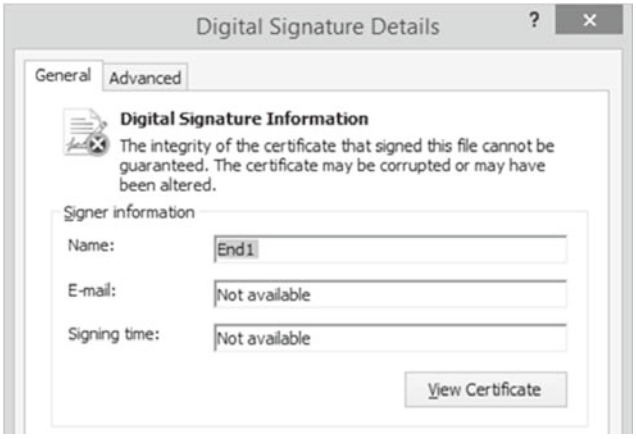

Fig. 5 Certification information after falsification of certificate part.

[Result]

After the falsification of the execution file, we could confirm that the prohibition rule had been retained by seeing the message indicating execution prohibition. In this experiment, the all-prohibition rule using wildcard was applied instead of the End1 prohibition rule. That is, the execution prohibition of "testapp1.exe" had been retained, since the prohibition rule using wildcard of the "path rules" worked. After that, we verified the state of the digital signature appended to the execution file from properties of the execution file, and confirmed that a message like **Fig. 5** indicating that the certificate was modified was shown.

5.2 Hierarchical Control

[Initial State]

All-prohibition rule using wildcard is set. The End certificate "End1" and "End2" (publisher of both: "InterCA1") are certified by the Intermediate certificate "InterCA1" (publisher: "RootCA"). Moreover, an execution file "testapp1.exe" is signed with the End1. In the same way, "testapp2.exe" is signed with the End2. The InterCA1 is imported into "Intermediate Certification Authorities," and thus it is valid. In addition, the execution of applications signed with the End1 and End2 are permitted by the Group Policy.

[Procedure]

(1) Confirm that "testapp1.exe" and "testapp2.exe" can be started up.

(2) Invalidate the InterCA1 by importing it into "Disallowed Certificates."

(3) Confirm that "testapp1.exe" and "testapp2.exe" cannot be started up.

[Result]

After importing the InterCA1 into "Disallowed Certificates," we attempted to execute the "testapp1.exe." Then, a prohibition message was displayed, and we were not able to start up the application. The case of "testapp2.exe" was the same as this. Further, when we confirmed certificate relation from their properties, a message indicating "the interCA1 is revoked" as shown in **Fig. 6** is displayed. According to the above results, we confirm that we can prohibits all execution of application signed with the lower layer End certificate by invalidating the Intermediate certificates.

Fig. 6 Invalidation of the Intermediate certificate "InterCA1."

Fig. 7 End3's certification path.

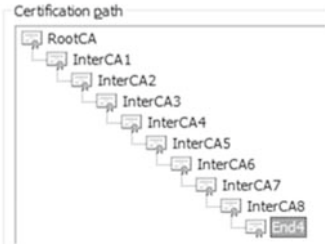

Fig. 8 End4's certification path.

5.3 Verification Time of Digital Certificate

The results of measurement time of verifying certificate reliabilities when starting up an application are shown here. The experiments were conducted with changing the number of hierarchies of Intermediate certificates to confirm the proposed method has short response time to meet actual use.

[Initial State]

For the three pattern experiments, the application which is not appended the digital certificate, the End certificate "End3" (publisher: "InterCA3"), and the End certificate "End4" (publisher: "InterCA8") are prepared. The "End3" is the 5 levels certificate whose certification path consists of the 3 Intermediate certificates and the one Root certificate as shown in **Fig. 7**. Moreover, the "End4" is 10 levels certificate whose certification path has the 8 Intermediate certificates and the one Root certificate as shown in **Fig. 8**.

[Procedure]

(1) Measure time until the start-up completion from com-

Table 4 The comparison of start-up time.

	start-up time[ms]		
	without signature	5 levels	10 levels
1st time	107.42	562.71	1136.72
2nd time	117.19	281.25	178.16
3rd time	158.20	166.02	227.54
4th time	111.33	169.92	215.19
5th time	103.52	187.50	174.80
average (from 2nd to 5th)	122.56	201.18	198.92

mand input about application without digital signature 5 times.

(2) Append the End certificate "End3" to the application.

(3) In the same way, measure the time about the 5 levels.

(4) Overwrite the End certificate "End4" to the application.

(5) In the same way, measure the time about the 10 levels.

[Result]

The result of each experiment is shown in **Table 4**.

With the Table 4, we can find out that the start-up time of the execution file signed with 5 levels certificate is increasing about 500ms at most, compared with the case of no-signature. Moreover, the time difference is about 1s in case of 10 levels. The impact on the application start-up is tiny, because the number of hierarchies we assumed is about 5 at most. For instance, the group "Browser" as described above has 4 levels such as Root certificate, Intermediate certificate "Browser," Intermediate certificate "IE" or "Firefox," and each End certificate. Though the number of hierarchies differs depending on the usage, it is sufficient to represent the application group unless the number of hierarchies gets so large.

Furthermore, the start-up time from the second time to fifth time is reduced compared to the first time for each certificate. It is considered that the certificate information that is verified in the first time has cached in the system.

6. Conclusion

In this paper, we proposed and designed a control method of the execution of applications using digital certificates. Since the traditional system used hash values to identify each application, it could not prevent falsification of execution files. In addition, flexible control target setting was difficult because the traditional group construction had restrictions.

However, by using the digital certificates, we can defeat the falsification of execution file and prohibit the execution. Moreover, by applying the hierarchical structure for the digital certificates to the application control, we can control the application group flexibly.

In future work, we will implement a prototype of the proposed method, and will consider how to make the group permission settings under the state of default-prohibition using the certificates.

References

[1] Fujimura, N., Inoue, H. and Hashikura, S.: Experience with the educational ICT environment in Kyushu University, *Proc. SIGUCCS 2009*, pp.167–172 (2009).

[2] Ueda, H., Kita, H., Mori, M., Ishii, Y., Tonomura, K., Ueki, T., Uehara, T. and Kajita, S.: Kyoto University Educational Computer System with Network Boot and Desktop Virtualization for Reduction in Total Cost of Ownership, *Proc. IOTS 2012*, pp.47–54 (2012) (in Japanese).

[3] Hamamoto, N., Ida, H., Saitoh, T., Sakai, H., Otagiri, T. and Kumehara, S.: Construction of computer system for PC rooms for education using Virtualization Technology, *Journal for Academic Computing and Networking*, No.17, pp.33–41 (2013) (in Japanese).

[4] Tadaki, S., Tanaka, Y., Matsubara, Y., Hieida, Y., Eto, H. and Watanabe, K.: Virtual user desktops through server-hosted thin-clients (The new educational terminal system in Saga University), *IPSJ SIG Technical Report*, Vol.2010-IOT-11, No.3, pp.1–5 (2010) (in Japanese).

[5] Fujiwara, M., Kawano, K. and Yamai, N.: An Execution Control System for Application Software Reducing Administrative Burden of Educational PCs, *Proc. C3NET 2012*, pp.375–380 (2012).

[6] Okamoto, D., Fujiwara, M., Kawano, K. and Yamai, N.: Target Application Grouping Function Considering Software Updates on Application Execution Control System, *Proc. ADMNET 2013*, pp.627–632 (2013).

[7] Kawano, K., Okamoto, D., Fujiwara, M. and Yamai, N.: A Flexible Execution Control Method of Application Software for Educational Windows PCs, *J. Inf. Process.*, Vol.22, No.2, pp.161–174 (2014).

[8] Masuda, H.: The Large Scale Educational Computer Systems, *IPSJ Magazine*, Vol.45, No.3, pp.225–226 (2004) (in Japanese).

[9] Sugiura, T.: Development and Evaluation of Netboot Computer Room System Based on Virtual Server Infrastructure, *Journal for Academic Computing and Networking*, No.17, pp.43–50 (2013) (in Japanese).

[10] Segawa, H., Tsujisawa, T. and Tatsumi, T.: Consolidation of Servers and Development of Educational Terminal System by Virtualization Technique, *Journal for Academic Computing and Networking*, No.11, pp.134–141 (2011) (in Japanese).

[11] Fujimura, N.: Bring Your Own Computers Project in Kyushu University *Proc. SIGUCCS 2013*, pp.43–50 (2013).

[12] The Apache Software Foundation: Apache VCL (online), available from ⟨https://vcl.apache.org/⟩ (accessed 2014-12-31).

[13] Schaffer, H.E., Averitt, S.F., Hoit, M.I., Peeler, A., Sills, E.D. and Vouk, M.A.: NCSU's Virtual Computing Lab: A Cloud Computing Solution, *IEEE Computer Society*, Vol.42, No.7, pp.94–97 (2009).

[14] Kajita, S.: Investigation on A Next-generation Student Terminal Service for Teaching and Learning on Cloud Environment, *Proc. SSS 2012*, pp.213–215 (2012) (in Japanese).

[15] Rindos, A., Vouk, M., Vandenberg, A., Pitt, S., Harris, R., Gendron, D. and Danford, T.: The Transformation of Education through State Education Clouds, *IBM Global Education* (2010).

[16] Li, P.: Provisioning Virtualized Datacenters through Virtual Computing Lab, *Proc. FIE 2010, T3C*, pp.1–6 (2010).

[17] Vouk, M., Averitt, S., Bugaev, M., Kurth, A., Peeler, A., Shaffer, H., Sills, E., Stein, S. and Thompson, J.: "Powered by VCL" - Using Virtual Computing Laboratory (VCL) Technology to Power Cloud Computing, *Proc. ICVCI 2008*, pp.1–10 (2008).

[18] Microsoft: Windows (online), available from ⟨http://windows.microsoft.com/en-us/windows/home⟩ (accessed 2014-12-29).

[19] Apple: OS X Yosemite (online), available from ⟨http://www.apple.com/osx/⟩ (accessed 2014-12-29).

[20] The CentOS Project: CentOS Project (online), available from ⟨http://www.centos.org/⟩ (accessed 2014-12-29).

[21] Canonical: Ubuntu: The leading OS for PC, tablet, phone and cloud (online), available from ⟨http://www.ubuntu.com/⟩ (accessed 2014-12-29).

[22] Maruyama, S., Saita, K., Kozuka, M., Ishibashi, Y., Ikeda, K., Mori, M. and Kita, H.: A Virtual Machine Solution for Large Scale Educational Computer Systems, *J. Inf. Process.*, pp.949–964 (2005) (in Japanese).

[23] Microsoft: Application Virtualization (online), available from ⟨http://technet.microsoft.com/en-us/appvirtualization/⟩ (accessed 2014-12-27).

[24] CITRIX: XenApp (online), available from ⟨http://www.citrix.com/products/xenapp/overview.html⟩ (accessed 2014-12-27).

[25] VMware: ThinApp (online), available from ⟨http://www.vmware.com/products/thinapp/⟩ (accessed 2014-12-27).

[26] Sky: SKYSEA Client View (online), available from ⟨http://www.skyseaclientview.net/⟩ (accessed 2014-12-27).

[27] Kaspersky Lab: Endpoint Security Select (online), available from ⟨http://usa.kaspersky.com/business-security/endpoint-select⟩ (accessed 2014-12-31).

[28] McAfee: McAfee Application Control (online), available from ⟨http://www.mcafee.com/us/products/application-control.aspx⟩ (accessed 2014-12-31).

[29] Computer Wing: Wingnet (online), available from

⟨http://www.cwg.co.jp/?page_id=141⟩ (accessed 2014-12-31).
[30] Microsoft TechNet: Group Policy (online), available from ⟨http://technet.microsoft.com/en-us/library/cc725828(WS.10).aspx⟩ (accessed 2014-09-04).
[31] Rivest, R.: The MD5 Message-Digest Algorithm, RFC1321 (1992).
[32] Microsoft Developer Network: Create a path rule (online), available from ⟨http://msdn.microsoft.com/en-us/library/cc781337⟩ (accessed 2014-09-12).
[33] Symantec: Symantec Code Signing Certificates for Microsoft Authenticode (online), available from ⟨https://www.symantec.com/code-signing/microsoft-authenticode⟩ (accessed 2014-09-04).
[34] Microsoft Developer Network: Create a hash rule (online), available from ⟨http://msdn.microsoft.com/en-us/library/cc781507⟩ (accessed 2014-09-12).
[35] Microsoft Developer Network: Create a certificate rule (online), available from ⟨http://msdn.microsoft.com/en-us/library/cc757067⟩ (accessed 2014-09-12).
[36] Microsoft Developer Network: Software restriction policies overview (online), available from ⟨http://msdn.microsoft.com/en-us/library/cc759106⟩ (accessed 2014-09-04).
[37] Microsoft Developer Network: Certificate stores (online), available from ⟨http://msdn.microsoft.com/en-us/library/cc757138⟩ (accessed 2014-09-04).
[38] Microsoft Developer Network: CryptoAPI Tools Reference (online), available from ⟨http://msdn.microsoft.com/en-us/library/windows/desktop/aa380240(v=vs.85).aspx⟩ (accessed 2014-09-10).

Daisuke Okamoto received his B.E. and M.E. degrees in engineering from Okayama University, Okayama, Japan, in 2013 and 2015. He is currently with KDDI CORPORATION. His research interests include distributed systems.

Keita Kawano received his B.E., M.E., and Ph.D. degrees from Osaka University, Osaka, Japan, in 2000, 2002, and 2004, respectively. From October 2004 to March 2010, he was an assistant professor of the Information Technology Center, Okayama University, Okayama, Japan. From April 2010 to March 2011, he was an assistant professor of the Center for Information Technology and Management, Okayama University. Since April 2011, he has been an associate professor of the same center. His research interests include mobile communication networks and distributed systems. He is a member of IEEE and IEICE.

Nariyoshi Yamai received his B.E. and M.E. degrees in electronic engineering and his Ph.D. degree in information and computer science from Osaka University, Osaka, Japan, in 1984, 1986 and 1993, respectively. In April 1988, he joined the Department of Information Engineering, Nara National College of Technology, as a research associate. From April 1990 to March 1994, he was an assistant professor in the same department. In April 1994, he joined the Education Center for Information Processing, Osaka University, as a research associate. In April 1995, he joined the Computation Center, Osaka University, as an assistant professor. From November 1997 to March 2006, he joined the Computer Center, Okayama University, as an associate professor. From April 2006 to March 2014, he was a professor in the Information Technology Center (at present, the Center for Information Technology and Management), Okayama University. Since April 2014, he has been a professor in the Institute of Engineering, Tokyo University of Agriculture and Technology. His research interests include distributed system, network architecture and Internet. He is a member of IEICE and IEEE.

Tokumi Yokohira received his B.E., M.E. and Ph.D. degrees in information and computer sciences from Osaka University, Osaka, Japan, in 1984, 1986 and 1989, respectively. From April 1989 to May 1990, he was an assistant professor in the Department of Information Technology, the Faculty of Engineering, Okayama University, Okayama, Japan. From May 1990 to December 1994 and from December 1994 to March 2000, he was a lecturer and an associate professor, respectively, in the same department. From April 2000 to June 2003, he was an associate professor in the Department of Communication Network Engineering of the same faculty. From July 2003 to March 2005, he was a professor in the same department. Since April 2005, he has been a professor of the Department of Information and Communication Systems, the Graduate School of Natural Science and Technology, Okayama University. His present research interests include performance evaluation and improvement of computer networks and communication protocols, design algorithm of optical networks and network securities. He is a member of IEEE Communication Society, IEICE and IPSJ.