Vol.2014-IOT-26 No.3
2014/6/28

教育用WindowsPCにおけるデジタル証明書を用いた柔軟かつ堅牢なアプリケーション実行制御システムの設計

岡本大輔[1,a)]　河野圭太[1]　山井成良[2]　横平徳美[1]

概要：我々は先行研究として，アプリケーションの実行可否を制御することにより，教育用 PC における個別のアプリケーション環境を提供するシステム（従来システム）を開発してきた．しかし，従来システムではアプリケーションの制御に実行ファイルのハッシュ値を用いており，実行ファイルの改ざんにより設定済みのルールが適用されなくなる問題があった．また，複数のアプリケーションを一括制御するグループ化機能はサポートしていたが，複数のグループを同時に制御することはできなかった．そこで本論文では，デジタル証明書を用いることにより厳密な実行制御をするとともに，証明書の階層構造を用いて柔軟なグループ運用を実現する手法を提案する．

キーワード：アプリケーション実行制御，デジタル証明書，教育用 PC

Design of Flexible and Strict Application Execution Control using Digital Certification on Educational Windows PCs

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

Abstract: We have developed an Application Execution Control System (traditional system) which provides individual application environments on each educational PC by controlling its execution. The traditional system uses a hash value of the execution file to control target application. If the execution file is falsified, a rule already set gets invalid. In addition, the traditional system has a function to set rules for multiple application at once. However, it cannot handle multiple groups at once. In order to solve these problems, this paper proposes a control method using digital certificates. It supports strict control and flexible group management by using hierarchical structure of digital certificates.

Keywords: Application Execution Control，Digital Certificate，Educational PC

1. まえがき

学生が教育目的で使用する PC（以下，教育用 PC）は，学生の利便性を考慮して複数の施設に分散的に配置され，どの PC からでも同一のユーザ環境が利用できるように設定される．この実現のために，教育用 PC は共通の OS イメージ（ひな形）により運用されることが一般的である．これは，ひな形を情報センターなどの管理者が作成し，そのイメージを全ての教育用 PC で利用する方式である．

この方式を用いることで，全ての PC に共通の変更を加える際にはひな形を修正しそれを反映させるだけでよく，管理者の負担削減を図ることができる．一方で，学生ごとや教室ごと等の個別制御をおこなうためにはそれぞれに対応したひな形を準備する必要があるため，ひな形の数に比例して管理者の負担も多くなる．

そこで我々の研究グループでは先行研究として，管理者負担軽減を図りつつ，教育用 PC 上のアプリケーション環境を個別かつリアルタイムに制御する“アプリケーション

[1] 岡山大学
Okayama University，Okayama，700-8530 JAPAN
[2] 東京農工大学
Tokyo University of Agriculture and Technology, Tokyo, 184-8588 JAPAN
[a)] daisuke.net@s.okayama-u.ac.jp

実行制御システム”（以下，従来システム）を開発してきた [1], [2], [3]．これは，教育用 PC が利用する OS イメージ上で，学生や教室等の個別条件に応じてアプリケーションの実行を許可あるいは禁止するシステムである．個別条件に応じて教員がリアルタイムに制御できるため，授業時間中など即時的な反映が求められる場面においても，個別のユーザ環境の構築を容易に実現できる．

従来システムでは，それぞれのアプリケーションの識別にハッシュ値を用いていた．ここでのハッシュ値とは，それぞれのアプリケーションの実行ファイルからハッシュアルゴリズムによって計算される 32 ビットの文字列である．このハッシュ値は入力データが 1 ビットでも書き換わると出力される値が変化する性質を持つ．このため，あるアプリケーションに対して禁止のルールが設定された状態で，悪意を持った学生によって実行ファイルのバイナリデータが改ざんされると禁止ルールが回避されるという問題があった．

また，従来システムは複数個の制御対象アプリケーションをグループとして登録し，そのグループに含まれたアプリケーションに対して一括してルールを設定・反映するグループ制御をサポートしていた．しかしこのグループのメンバにはアプリケーションのみ指定でき，グループの中にさらにグループを含めることができなかった．複数のグループに対してルールを設定したいときはそれぞれのグループ一つ一つにルールを設定する必要があるため，制御したいグループ数に応じて教員の負担が増大する問題があった．

そこでこれらの問題を解決するために，本論文では実行ファイルの改ざんを検知し，柔軟なグループ制御を可能にすることを目的として，アプリケーション制御にデジタル証明書を用いる手法を提案する．教育用 PC において学生がアプリケーションを起動しようとするときに実行ファイルに付加されたデジタル署名を検証し，有効性が確認できた場合のみ起動を許可するようにする．

また，階層的な証明書の認証関係をアプリケーション制御にも応用する．具体的には，ある証明書に対する信頼の可否（以下，有効性）を切り替えることで，その証明書が認証している下位の証明書の有効性もまとめて切り替える．有効性が無効化された，すなわち信頼されていない証明書で署名されたアプリケーションは起動できないようにする．さらに，証明書の認証関係は従来のグループ化機能同様に管理者が任意に構築できるようにする．これにより，上位の証明書の有効性を操作することで，下位の証明書で署名された複数のアプリケーションをまとめて制御できるようになるので，制御対象をまとめて制御するグループ制御の自由度を高くできる．

以下，第 2 章では従来のアプリケーション実行制御システムの概要および問題点について述べる．そして第 3 章では提案手法の概要と拡張機能について説明し，その後第 4 章で今回の提案手法の実現可能性を確認するための検証実験について述べる．最後に，第 5 章で本論文をまとめ，今後の課題について述べる．

2. 従来のアプリケーション実行制御システム

本章では，我々のグループが開発してきたアプリケーション実行制御システム（従来システム）の仕組みについて述べる．その後，従来システムの問題点について説明する．

2.1 従来システムの概要

前述のように，教育用 PC は一般的にイメージを用いて一括管理される．しかしこの方式では，イメージ数に応じて管理者の負担が増大するうえに，変更をリアルタイムに反映することができなかった．そこで我々の研究グループでは，学生や教室等の個別条件に応じて教育用 PC 上のアプリケーション利用を制限するルールを設定し，教員がリアルタイムに教育用 PC を制御するようなアプリケーション実行制御システム（以下，従来システム）を開発してきた．このシステムでは，学生のアプリケーション利用を制限するルールを教員用 PC から設定して，動的に教育用 PC に変更を反映し制御する．

この動作を実現するために，従来システムは 4 つのプログラムで構成されている．まず 1 つ目は，教員が制御ルールを設定するための機能が集約された“設定ツール”である．そして，教育用 PC に常駐してアプリケーションの実行を制御・監視する“実行制御プログラム”である．さらに，教員や学生の PC からの要求を受けるとそれぞれの制御情報などの応答を返す“サーバプログラム”である．またサーバは，制御ルールやアプリケーションの情報を保持するデータベース（以下，ポリシー DB）を併せ持ち，制御情報の格納や取得のために利用する．そして最後に，管理者が制御対象のアプリケーションをまとめて制御するためのグループを作成したり，アップデート等の変化がないか検査したりするための“管理ツール”である．

学生用 PC では実行制御プログラム，教員用 PC では設定ツール，管理者用 PC では管理ツールが動作している．教育用 PC が起動したとき，実行制御プログラムはサーバに問合せを送る．サーバがそれを受け取ると，ポリシー DB のテーブルを探索し結果を応答する．実行制御プログラムは結果にしたがってアプリケーションの実行を制御する．この状態で，教員が制御ルールを変更したときの動作を**図 1** に，管理者が制御対象の更新を検知した時の動作を**図 2** に示す．

図 1 のように，教員が制御ルールを変更すると，そのルールはポリシテーブルに登録される．その後，ルールの変更は対応する学生の実行制御プログラムに通知され，反

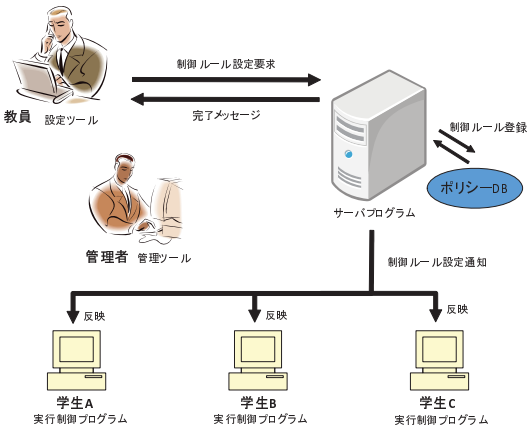

図 1 制御ルール設定時の動作

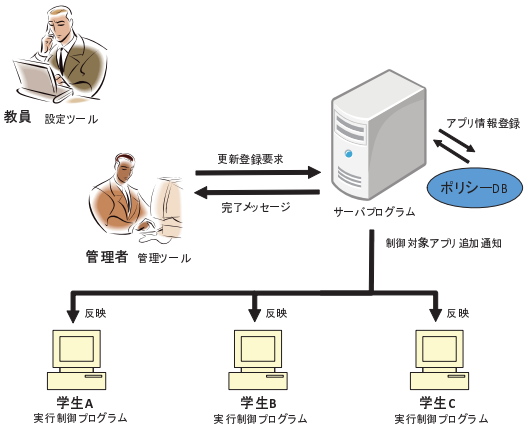

図 2 制御対象の更新検知時の動作

映される．

また図 2 のように，管理者が管理ツールを用いてあるソフトウェアの更新を検知すると，そのソフトウェアのデータがサーバに送信される．そしてサーバは受信データをポリシー DB に登録し，対応する実行制御プログラムに通知，反映させる．

従来システムではアプリケーションの実行を制御するために Windows のグループポリシーと呼ばれる機能を用いていた．グループポリシーではレジストリ情報を操作することでアプリケーションの起動を制御する機能をサポートしているが，このときアプリケーションの識別のためにハッシュ値が必要となる．このハッシュ値は，それぞれのアプリケーションの実行ファイル（exe ファイル）にハッシュアルゴリズムである MD5 を適用して得ることができる．これは exe ファイルのバイナリデータを入力とすることで，32 ビットのユニークな 16 進文字列を出力する．入力値が 1 ビットでも変化すると出力値が全く異なるうえに，違う入力値から同じ結果が出力されるハッシュ衝突の可能性が限りなく低いという特徴を持つ [4]．

また，従来システムは複数個の制御対象のアプリケーションをまとめて制御するために“デフォルト設定機能”，および“グループ化機能”をサポートしている．デフォルト設定機能では，制御対象のアプリケーション全ての起動を許可する状態（以下，デフォルト許可），あるいは禁止にする状態（以下，デフォルト禁止）に設定することができる．特に指定しなければ，標準の設定としてデフォルト許可としてすべてのアプリケーションの起動が許可される．デフォルト禁止にする場合は，システムがパスの規則としてワイルドカードを指定した禁止ルールを設定することで，すべてのアプリケーションの実行を禁止にする．このとき，任意のアプリケーションの起動を許可するルールを設定することで，利用させたいアプリケーション以外の使用を禁止することができる．

また，グループ化機能は任意のアプリケーションをグループとして登録し，そのグループに対して制御ルールを設定することで複数のアプリケーションを一括して制御する機能である．既に管理者が管理ツールを用いて制御対象アプリケーションのグループを作成していると，教員はそのグループを用いて複数のアプリケーションを同時に指定することが出来る．管理者は管理ツールを用いて，アプリケーションメンバを指定してグループを作成する．このグループ情報は作成時にポリシー DB の対応するテーブルへ登録される．

2.2 従来システムの問題点

2.2.1 実行ファイルの改ざんによるルール回避

教育機関においては，学生による不正利用を想定しておく必要がある．例えば，実行ファイルのバイナリ改ざんによる設定済みルールの適用回避がある．前節で述べたとおり，従来システムではアプリケーションの識別にハッシュ値を用いており，学生によって exe ファイルが書き換えられると計算されるハッシュ値が変化してしまう．これによって，既に設定されているルールが無効になる問題点がある．

以下に実際に改ざんを行った結果を示す．ここではあらかじめ，対象のアプリケーションの実行禁止ルールが設定されている．この状態で，そのアプリケーションを起動しようとすると実行制御プログラムによって**図 3** のようなメッセージが表示され，起動することが出来ない．その後，実行ファイルのバイナリデータの未使用部を改変 [*1] して，そのアプリケーションを起動してみると，起動禁止メッセージはなく通常通り起動できてしまう．これはバイナリデータが書き換えられたことで，実行ファイルから計算されるハッシュ値が変化し，設定されたルールに一致しなくなったためである．

2.2.2 制御対象グループ化機能の問題点

前節にある通り，従来システムは複数の制御対象をグループとして登録し，1 つの制御ルールをグループ内のメ

[*1] ここでは，バイナリエディタを用いて exe ファイルの未使用ビットを 00 から 01 に変更した．

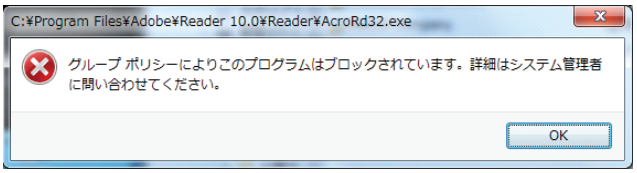

図 3 起動禁止メッセージ

ンバに適用できるグループ化機能をサポートしていた．この機能では，どのアプリケーションがどのグループに属するかのメンバ情報をポリシー DB で管理していた．グループごと，アプリケーションごとに一意な ID であるグループ ID，およびアプリケーション ID を割り当て，それぞれを対応付けてテーブルに格納していた．しかしながら，それぞれの ID を一対一で割り当てるこの手法では，グループ自体をメンバに指定して別のグループに含める動作ができなかった．複数のグループを同時に制御したいときには，1 つ 1 つのグループに対してルールの設定を行わなければならず，対象のグループ数に応じて教員の負担が増加する．**表 1** に従来のグループ化の例を示す．

表 1 グループの情報

グループ名	メンバ
IE	Internet Explorer 11.0
	Internet Explorer 11.1
	Internet Explorer 11.2
Firefox	Firefox 29.0
	Firefox 29.1

表 1 のように，グループ “IE” にはアプリケーション “Internet Explorer 11.0”，“Internet Explorer 11.1”，“Internet Explorer 11.2” が含まれており，グループ “Firefox” にはアプリケーション “Firefox 29.0”，“Firefox 29.1” が含まれているとする．このときブラウザアプリケーションのグループ “browser” を作成しようとすると，メンバとしてグループ “IE” “Firefox” を指定できないので，5 つのアプリケーションを 1 つずつ選択する必要がある．または，この 2 つのグループに対して 1 つずつルールを設定する必要がある．このように，グループをメンバに指定してより大きなグループを作成するとき，管理者に新たにグループを作成する負担，あるいは教員にルールを 1 つずつ設定する負担が生じる．

3. デジタル証明書を用いた制御手法

第 2 章で述べた問題点を解決するために，この論文ではデジタル証明書を用いた制御方法を提案する．この方法では，実行ファイルに付加されたデジタル証明書の有効性をアプリケーション実行時に確認することにより，実行ファイルに改ざんがあるか否かを判断する．また，証明書の階層構造を用いてグループを表現することにより，複数グループの一括制御を柔軟に実現する．

3.1 概要

第 2 章で述べた問題を解決するためには，exe ファイルのハッシュ値が変化したことを検出して起動を制御する動作，および任意の異なるグループをまとめて制御するような動作が必要となる．そこで本論文では，デジタル証明書を用いた制御手法を提案する．exe ファイルの改ざんを検知して起動制御する改ざん検知機能，および任意の制御対象に付与する証明書を階層的に構築して柔軟に制御するための階層機能を従来システムに追加する．

まず，改ざん検知機能は，学生がアプリケーションを起動したときに exe ファイルに付加されたデジタル証明書を検証して，有効性が確認された場合のみ起動を許可する機能である．これにより，証明書に不正があれば起動禁止を維持できるので，学生の改ざんによるルール回避を防ぐことができる．

次に，階層機能は，デジタル証明書の階層的な認証関係を制御に応用する．上位の証明書が信頼できるか否かの有効性を切り替えることにより，その証明書が認証している複数の証明書の有効性操作を可能にする．これによって，複数の証明書付きアプリケーションの実行を柔軟に制御することができる．

これらの機能を実現するため，従来システムに証明書を扱う処理を加える必要がある．具体的には，証明書をアプリケーションに付加する機能を設定ツールに追加し，実行制御プログラムを証明書による制御が行えるように拡張する必要がある．また，ポリシー DB に証明書のルールを格納できるようテーブルを拡張する必要がある．

3.2 提案手法の設計

前述のように従来システムでは，アプリケーションの制御にグループポリシーを採用し，アプリケーションの識別に実行ファイルから計算されるハッシュ値を用いていた．本論文では従来のハッシュ値制御に加えて，デジタル証明書を用いた制御を提案する．

グループポリシーでは，ハッシュ値だけでなくデジタル証明書による制御もサポートしている [5]．一般的にデジタル証明書はデータの改ざんが無いことを保証するために用いられる．このデジタル署名を実行ファイルに付加することができ，市販のソフトウェアにもデジタル証明書の付いた製品がある [6]．アプリケーション実行時に付加された証明書を検証し，結果が一致しない場合はデータの書き換えがあったと判断することができる．あらかじめ信頼できる証明書が付加されたアプリケーションの起動のみ許可されていると，不正な証明書では実行することができないため，前述のような不正利用に対応できる．

また，先に述べたとおり従来システムでは “すべてのアプリケーションの実行を禁止（許可）” するモードの切り替え（デフォルト設定）を行うことができる．証明書の改ざ

んに対応しながらデフォルト許可の状態を実現するには，あらかじめすべてのアプリケーションの起動を禁止するデフォルト禁止に設定し，アプリケーションに付与された全ての証明書を起動許可する動作が必要になる．これによって，証明書の改ざんが行われた場合でも，その証明書が起動を許可された証明書リストのデータと合致しないので起動禁止を維持することができる．

さらに，提案手法ではアプリケーションをまとめて制御するためのグループの指定に，証明書の階層的な認証関係を利用する．制御に用いる証明書として，それぞれのアプリケーションに署名するためのエンド証明書と，エンド証明書の有効性を認証するための中間証明書がある．グループポリシーでは，エンド証明書に対してルールを設定することで個別のアプリケーションを制御できるようになる．そのエンド証明書を認証する中間証明書の有効性を操作することにより，エンド証明書の有効性も変更され，結果として複数のアプリケーションを同時に制御することができる．

証明書が付与されたアプリケーションの起動許可／禁止を切り替えるための証明書の有効性操作について以下に示す．証明書を用いたルールを設定するために，名前や発行者・有効期間といった証明書情報をレジストリ内の“証明書ストア”[7] にインポートする必要がある．証明書が信頼できるものか，あるいは信頼されないものかは，証明書情報を証明書ストアのどの場所にインポートするかで決定される．**表 2** に，提案手法で特に重要な項目を示す．

表 2 証明書ストア内の主要なインポート先

項目名	インポートされる証明書
信頼されたルート証明機関	ルート証明書
中間証明機関	中間証明書
ほかの人	エンド証明書
信頼された発行元	許可ルールが設定された証明書
信頼されていない証明書	禁止ルールが設定された証明書

表 2 に示す通り，証明書の種類によってインポート先の項目が異なる．まず，ルート証明書として作成された証明書は“信頼されたルート証明機関”にインポートして有効化する．次に，中間証明書は“中間証明機関”にインポートする．最後に，エンド証明書は“ほかの人”にインポートすることで，グループポリシーで制御できる証明書として扱われるようになる．さらに，グループポリシーにおいて証明書の許可ルールを設定すると自動的に“信頼された発行元”にインポートされる．反対に，禁止ルールを設定すると“信頼されていない証明書”にインポートされ，それぞれのルールに応じて制御される．

先に述べたとおり，グループポリシーでは，エンド証明書を用いてそのアプリケーションの起動可否を制御することが出来る．しかしながら，証明書のルールを設定すると，同じ証明書で署名された他のアプリケーションも起動を制御されてしまう．たとえば，Microsoft Office 製品である Excel，Word，Powerpoint などでは同じ証明書が用いられているので，証明書のルールを一つ設定するといずれのアプリケーションに対してもそのルールが適用される．これを考慮して，提案手法では証明書を上書きするような機能も追加する．新たに付与された証明書に対してルールを設定することで，個別のアプリケーション制御に対応する．

次に，中間証明書はエンド証明書が信頼されていることを認証するが，中間証明書の有効性は下位のエンド証明書に継承される．前述のような証明書によるデフォルト許可状態においては，中間証明書を“信頼されていない証明書”にインポートすることで下位のエンド証明書も無効化され，結果として起動を禁止することができるのでグループ単位の禁止制御が実現できる．上位の中間証明書が無効化されると，そのエンド証明書が許可のルールを設定されて（“信頼された発行元”にインポートされて）いてもアプリケーションを起動出来ない．また，再び許可に設定しなおす場合は，その中間証明書を“信頼されていない証明書”から削除して，“中間証明機関”にインポートし直せばよい．

しかしながら，デフォルト禁止状態においては挙動が異なる．先ほどとは逆に中間証明書を“信頼された発行元”にインポートしても，その下位のエンド証明書が付加されたアプリケーションの起動が許可されるわけではない．デフォルト禁止のルールが適用され，そのアプリケーションは起動禁止を維持される．このように，デフォルト禁止においてグループを許可する動作については，現段階では階層構造だけで実現することは困難である．証明書の階層構造を用いた有効性操作によるグループ単位での許可機能の実現については，今後さらなる検討が必要である．

また，エンド証明書および中間証明書の認証関係は従来のグループ化機能と同じように，管理者が自由に構築できるようにする．これにより，従来法よりもさらに自由度の高いグループ制御が行えるようになる．

さらに，実行制御に使用する証明書を準備する必要があるため，設定ツールや管理ツールには証明書の作成や署名というような操作を行う機能を追加する．そこで，Microsoft 社が公開している Windows で動作するアプリケーション作成のためのソフトウェア開発キットである Windows SDK を利用する．このキットには，証明書作成のための makecert や，ファイルに署名するための signtool などの exe ファイルが含まれている．これらを従来プログラムに組み合わせることにより，証明書を作成したり，exe ファイルに署名したりするような動作を新たに実装する必要なく，制御に必要な拡張機能を実現することができる．

図 4 にこの機能を用いた全体の流れを示す．図 4 では，管理者によって証明書が作成および署名され，教員が証明書のルールを用いてアプリケーションを禁止した後，学生

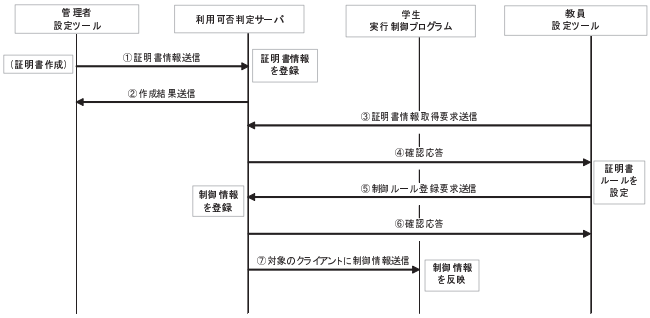

図 4 提案手法の動作

の教育用 PC に反映されるまでの手順を表している．反映後に改ざんが行われ，アプリケーションを起動しようとした場合でも，証明書によってそれを検知することにより起動禁止が維持される．

図 4 における動作手順を以下にまとめる．

（管理者が管理ツールで証明書を作成する）

(1) 証明書情報をサーバプログラムへ送信する
(2) サーバプログラム上で，証明書情報を対応するテーブルに登録し，作成完了メッセージを送信する

（教員が設定ツールを起動する）

(3) 設定ツールが制御情報取得要求をサーバプログラムへ送信する
(4) サーバプログラムは制御情報を DB から取得して応答する

（教員が設定ツールで証明書の禁止ルールを設定する）

(5) 設定ツールがルール情報をサーバプログラムへ送信する
(6) サーバプログラム上で，ルールを対応するテーブルへ登録，設定ツールへ完了メッセージを送信する
(7) 変更に該当する実行制御プログラムへ制御情報変更を通知，反映する

4. 検証実験

第 3 章で述べた，デジタル証明書を用いた制御手法の有効性および実現可能性を確認するための実験を行った．

4.1 (実験 1) 改ざん検知

実際に証明書を用いて禁止のルールを設定したアプリケーションの実行ファイルを改ざんし，起動時に証明書の改ざんがあったことを検知させ，起動禁止を維持できることを確認する．その際，実行ファイル中のどの部分を書き換えるかによって挙動が変わる．今回は，改変してもアプリケーションの動作に支障をきたさない未使用バイト，および付加された署名の部分にあたるバイトの 2 通りの改変を行った．

- (実験 1.1) 未使用バイトの改ざん

  [初期状態]

  証明書のデフォルト許可状態で，実行ファイル“testapp1.exe”にはエンド証明書“End1”（発行者：“InterCA1”）が署名してある．証明書 End1 が付加されたアプリケーションの実行を禁止するルールをグループポリシーで設定しているため，End1 は「信頼されていない証明書」にインポートされている．

  [実験手順]

  (1) testapp1.exe の起動ができないことを確認する．
  (2) 実行ファイルの未使用バイトを書き換える．
  (3) 先ほどと同様に testapp1.exe を実行し，起動禁止を維持していることを確認する．
  (4) 付加された証明書の改ざんを検知できていることを確認する．

  [実験結果]

  実行ファイルの改ざんを行った後，起動しようとすると起動禁止を示すメッセージが表示され，禁止ルールを維持できていることを確認した．これは，起動禁止のルールをハッシュ値ではなく証明書を用いて設定しており，バイナリデータ中の署名部分には変更がなかったためである．その後，実行ファイルに付与されたデジタル署名の状態をプロパティから確認すると，**図 5** のように署名が有効でないことを示すメッセージが表示された．これはバイナリデータから計算されるハッシュ値と，署名に含まれるハッシュ値が一致しないことを検出したためである．

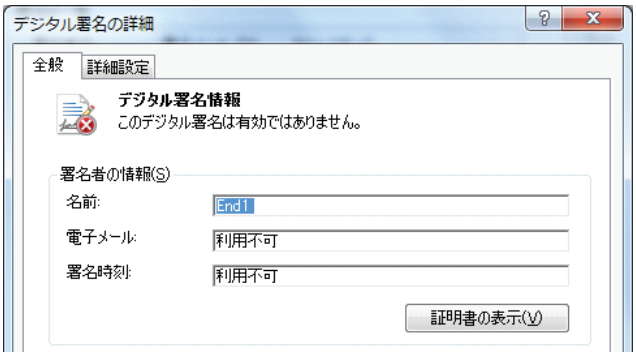

図 5 未使用バイト変更後の証明書情報

- (実験 1.2) 署名部分の改ざん

  [初期状態]

  実験 1.1 と同様に，証明書のデフォルト許可状態で，実行ファイル“testapp1.exe”には証明書 End1 が署名されている．また，証明書 End1 が付加されたアプリケーションの実行を禁止するルールをグループポリシーで設定しているため，End1 は「信頼されてい

ない証明書」にインポートされている．

[実験手順]

(1) testapp1.exe の起動ができないことを確認する．

(2) 実行ファイルの署名部分のビットを書き換える．

(3) 先ほどと同様に testapp1.exe を実行し，起動禁止を維持していることを確認する．

(4) 付加された証明書の改ざんを検知できていることを確認する．

[実験結果]

実行ファイルの改ざんを行った後，起動しようとすると実験 1.1 と同様に起動禁止を示すメッセージが表示され，禁止ルールを維持できていることを確認した．この実験では，署名部分の変更により証明書 End1 の禁止ルールは適用されなくなったが，デフォルト禁止ルールが適用される．デフォルト禁止ルールはパスの規則によりワイルドカードで指定され，すべてのアプリケーションに適用されるため，testapp1.exe の起動禁止が維持できる．その後，実行ファイルに付与されたデジタル署名の状態をプロパティから確認すると，**図 6** のように証明書の変更が行われたことを示すメッセージが表示された．

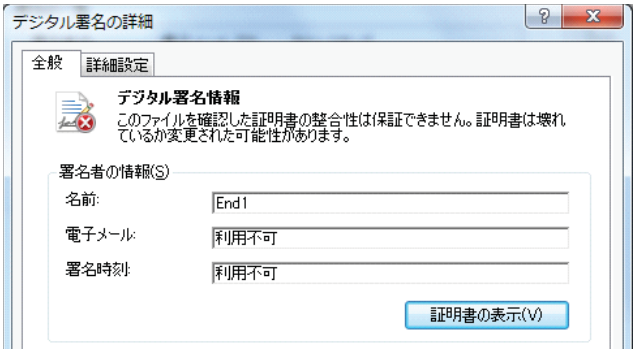

図 6 署名部分変更後の証明書情報

4.2 (実験 2) 階層的な制御

中間証明書とエンド証明書を用いて，証明書の認証関係を階層的に構築する．その状態で中間証明書を無効化することで対応するエンド証明書が署名されたアプリケーションが起動できないことを確認する．

[初期状態]

デフォルト禁止で全てのアプリケーションの起動を禁止している．エンド証明書“End1”(発行者：“InterCA1”) および“End2”(発行者：“InterCA1”) は，中間証明書“InterCA1”(発行者：“RootCA”) によって認証されている．また，実行ファイル“testapp1.exe”には証明書 End1 が，“testapp2.exe”には証明書 End2 が付加されている．中間証明書 InterCA1 は“中間証明機関”にインポートされており有効である．また，それぞれの証明書を用いて起動を許可するルールをグループポリシーで設定しているため，End1 および End2 は「信頼された発行元」にインポートされている．

[実験手順]

(1) testapp1.exe および testapp2.exe が起動できることを確認する．

(2) 証明書 InterCA1 を“信頼されていない発行元”にインポートして証明書を無効化する．

(3) testapp1.exe と testapp2.exe を実行し，それぞれ起動できないことを確認する．

[実験結果]

中間証明書 InterCA1 を“信頼されていない発行元”にインポートしたのち，testapp1.exe を実行すると起動禁止メッセージが表示されて起動できなくなっていた．testapp2.exe についても同様であった．また，証明書の認証関係をプロパティから確認すると，End1 を認証していた InterCA1 が失効しているメッセージが表示された．以上のことから，中間証明書を無効化することにより，下位のエンド証明書も同時に無効化でき，制御に反映できることが確認できた．

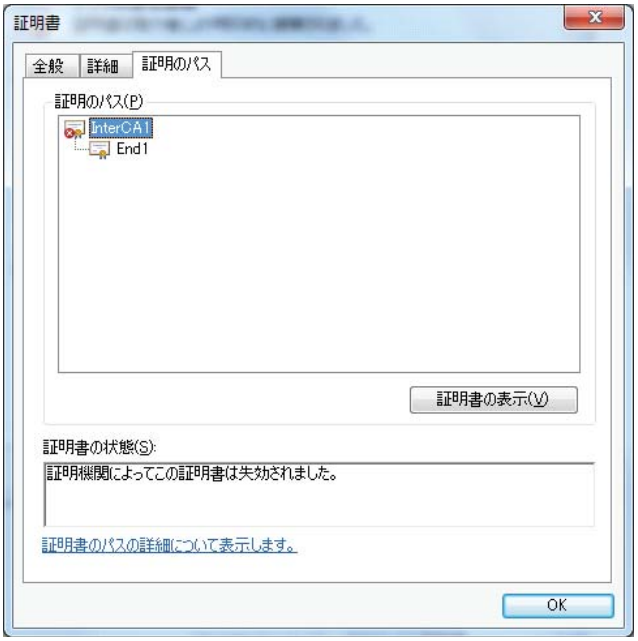

図 7 中間証明書 InterCA1 の無効化

4.3 (実験 3) 証明書の検証時間

アプリケーション起動時において，証明書の有効性を検証する時間を測定した結果を示す．中間証明書の階層数別にも検証し，運用に支障がない範囲で動作することを確認する．

[初期状態]

デジタル署名なしのもの，**図 8** のようにルート証明書と中間証明書合わせて 4 つに認証された 5 階層のエ

ンド証明書“End3”(発行者：“InterCA3”)，および**図 9** のようにルート証明書と中間証明書合わせて 9 つに認証された 10 階層のエンド証明書“End4”(発行者：“InterCA8”) をそれぞれ同じ内容の exe ファイルに署名している．

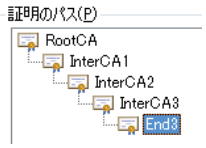

図 8 End3 の認証関係

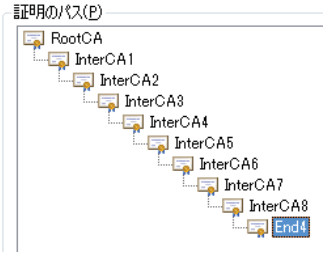

図 9 End4 の認証関係

[実験手順]

(1) デジタル署名なしのものについて，起動するコマンドを入力してから画面が表示されるまでの時間を 5 回測定する．

(2) 同様に 5 階層の証明書，および 10 階層の証明書についてもそれぞれ 5 回ずつ測定する．

[実験結果]

それぞれの実験結果を以下の**表 3** に示す．

表 3 起動時間の比較

	起動時間 [ms]		
	署名なし	5 階層	10 階層
1 回目	107.42	562.71	1136.72
2 回目	117.19	281.25	178.16
3 回目	158.20	166.02	227.54
4 回目	111.33	169.92	215.19
5 回目	103.52	187.50	174.80
平均	119.53	273.48	386.48

表 3 より，5 階層の証明書を用いた場合は，署名なしの場合に比べて起動時間が最大で 500ms 程度増加することがわかる．また，10 階層の場合はこの増加時間は 1s 程度になる．我々が想定する階層数は高々 5 階層程度であり，起動時間に与える影響は軽微である．たとえば，先述のような“ブラウザ”グループでは，ルート証明書と中間証明書“ブラウザ”，および各エンド証明書の 3 階層である．基本方針としては，この例のように 3 階層での使用を想定している．使用目的により階層数は異なるが，基本方針に中間証明書 2 つを加えた 5 階層程度あればグループ表現には十分である．よって，起動時間上の問題はないといえる．

また，それぞれの証明書において 2 回目以降は 1 回目の起動に比べて起動時間が削減されていることがわかる．これは 1 度目に検証した証明書情報を，該当する検証機構にキャッシュしているためであると考えられる．

5. むすび

本論文では，アプリケーション実行制御システムにおけるデジタル証明書を用いた制御法を提案し，設計および検証実験を行った．従来のアプリケーション実行制御システムでは，アプリケーションの制御にハッシュ値を用いていたため，実行ファイルの改ざんに対応することができなかった．また，複数のアプリケーションをまとめて制御ルールを設定するグループ化機能をサポートしていたが，グループの構築には制約があり，柔軟な制御対象の設定が困難であった．

しかし，今回の研究において，デジタル証明書を用いてデフォルト状態や証明書のルールを適切に設計することによって，データファイルの改ざんにも対応できることが確認できた．また，証明書の階層構造をアプリケーション制御に応用し，証明書の有効性を切り替えることで，柔軟にグループ制御が行えることも確認した．

今後の課題として，アプリケーション実行制御プログラムへの提案手法の実装，およびデフォルト禁止時のグループ許可設定への証明書の適用が挙げられる．

参考文献

[1] 関谷章仁，川上崇，河野圭太，山井成良：教育用 Windows PC における同時起動数を考慮したアプリケーション利用制御システム，情報処理学会研究報告，Vol.2010-IOT-8, No.1，pp.1–6 (2010).

[2] 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).

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

[4] Rivest R., The MD5 Message-Digest Algorithm, RFC1321 (1992).

[5] Microsoft Developer Network：ソフトウェア制限ポリシーの概要 (online)，入手先 ⟨http://msdn.microsoft.com/ja-jp/library/cc759106⟩ (2014.06.04).

[6] シマンテック・ウェブサイトセキュリティ：導入事例 | シマンテック コードサイニング証明書 (online)，入手先 ⟨https://www.jp.websecurity.symantec.com/codesign/resources/cp_sc.html⟩ (2014.06.04).

[7] Microsoft Developer Network：証明書ストア (online)，入手先 ⟨http://msdn.microsoft.com/ja-jp/library/cc757138⟩ (2014.06.04).