情報処理学会論文誌　コンピューティングシステム　Vol.7　No.4　34–44 (Dec. 2014)

ディスク暗号化システムにおける プリブート認証OSを用いたユーザ認証方式

平兮 亮[1,a)]　関山 友輝[1]　峯 博史[1]　真矢 譲[1]　大島 訓[1]　猪口 修史[2]　河野 健二[3]

受付日 2014年4月4日, 採録日 2014年8月18日

概要：情報漏洩事故の防止のために，OSも含むディスク全体を暗号化してデータを保護するシステムが利用されている．従来のディスク暗号化システムでは，コンピュータ電源投入直後のOS起動前環境でユーザ認証処理を行い，暗号化されたOSを起動する．しかしOS起動前の環境では利用できるデバイスの機能に制限があり，GUIやチャレンジ・レスポンスなどを用いた多様な認証手段の提供が難しいという課題がある．本稿では，ディスク暗号化システムのユーザ認証方式として，認証専用のプリブート認証OSを設け，暗号化されたユーザOSを起動するプリブート認証OS方式を提案する．OSの機能を利用してユーザ認証を行うことにより，多様な認証手段を提供する．プリブート認証OSでの認証後は，コンピュータの再起動を不要とするリブートレス起動によりユーザOSへ切り替えることで，プリブート認証OSで生成したメモリ上の暗号鍵を消去することなくユーザOSの復号と起動を実現する．提案方式を実装したディスク暗号化システムのOS起動時間を評価し，多様な認証手段を実現しながらOSの起動時間を従来の30秒増に抑えられることを確認した．

キーワード：オペレーティングシステム，暗号化，セキュリティ

User Authentication Method Using Pre-boot Authentication OS on Full Disk Encryption Systems

Ryo Hirana[1,a)]　Tomoki Sekiyama[1]　Hiroshi Mine[1]　Yuzuru Maya[1]　Satoshi Oshima[1]
Shuji Inokuchi[2]　Kenji Kono[3]

Received: April 4, 2014, Accepted: August 18, 2014

Abstract: Full disk encryption (FDE) systems are used to prevent the information leakage accident. The existing FDE systems perform the user authentication for booting encrypted OS in the pre-boot environment just after that the computer is powered on. In the pre-boot environment, some device features are restricted. Therefore it is difficult to provide the various user authentication features such as GUI and challenge-response authentication. To solve this problem, we propose the user authentication method using the pre-boot authentication OS, which is the OS for exclusive use of user authentication and booting the encrypted user OS. In the proposed method, the FDE system performs the various user authentication features in the pre-boot authentication OS environment. After that the pre-boot authentication OS creates the encryption key onto the main memory and switches processing to the user OS without rebooting computer. Then encryption key is not erased from main memory, the FDE system is able to decrypt user OS and boot it. We evaluate the proposed method, and confirm that it realizes the various user authentication features and suppress the OS boot time to 30 seconds increase.

Keywords: Operating System, encryption, security

[1] 株式会社日立製作所横浜研究所
Yokohama Research Laboratory, Hitachi, Ltd., Yokohama, Kanagawa 244–0817, Japan
[2] 株式会社日立製作所
Hitachi, Ltd., Chiyoda, Tokyo 101–8608, Japan
[3] 慶應義塾大学理工学部情報工学科
Department of Information and Computer Science, Keio University, Yokohama, Kanagawa 223–8522, Japan
[a)] ryo.hirana.ax@hitachi.com

1. はじめに

近年，企業や政府機関において，従業員の所有するノートパソコンなどの端末の紛失や不正使用が情報漏洩事故につながる事例が発生し，社会問題となっている [2], [18]．こうした事故を防止するために，端末の情報セキュリティの強化が要求されている．端末に格納された情報を保護する手法の 1 つにデータ暗号化がある [12]．これは，重要データを暗号化してディスクに保存することで，端末を紛失しても重要データの取り出しを困難とし，情報漏洩を防ぐものである．

データ暗号化では，保護すべき情報の種類や重要度に応じて，個々のファイル単位の暗号化（以下，ファイル暗号化方式）や，ディスク全体の暗号化（以下，ディスク暗号化方式）など暗号化の範囲を選択する．ディスク暗号化方式は，暗号化の有無をファイル単位で指定する必要のあるファイル暗号化方式と比べて，データをディスクに書き込む際に自動的に暗号化されるため，暗号化の実施忘れが発生しないなど，高いセキュリティを提供できる利点がある．

実際，高いセキュリティが必要となる環境ではディスク暗号化方式を採用することが多い．たとえば，NASA（National Aeronautics and Space Administration）ではノートパソコンの盗難にともなう情報漏洩をきっかけに，NASA 所有の全ノートパソコンにディスク暗号化ソフトウェアを導入する方針を示している [7]．

また，データ暗号化では，所有者のみが暗号化されたデータを利用できるようにするため，正規ユーザか否かを判別するユーザ認証が行われる．ユーザ認証では，ユーザ名とパスワードによる単純な認証のほかに，多様な認証手段を提供することが重要である．トークンデバイスや生体情報のようなパスワード以外の情報を併用した二要素認証 [8] は，パスワードのみを用いる認証よりも強固なセキュリティをユーザに提供できる．また，チャレンジ・レスポンス認証は，ユーザがパスワードを忘れた場合に，画面に表示されたチャレンジコードを電話などで管理者に伝えると，管理者から返答されたレスポンスコードを使って端末に一時的にログインできる認証手段である．企業や政府機関のサポートデスクなどで管理者が遠隔地にいる場合でも電話による会話のみで認証を行えるため，ユーザと管理者の運用負荷を低減させることができる．このように，ユーザや管理者は多様な認証手段を使い分けることで，高いセキュリティや容易な運用を実現できる．さらに，ユーザ認証では，正規ユーザが多様な認証手段を使い分け，暗号化データにアクセスするまでの手間を削減するため，GUI（Graphical User Interface）を利用したユーザ認証画面などの利便性の高いインタフェースを提供することも重要である．

しかし，ディスク暗号化方式では，暗号化された OS を起動するために OS 起動以前にユーザ認証を行い，ディスクに格納されたデータを復号しながら OS 起動処理を進める仕組みが必要である．これらの処理はプリブート認証 [9] と呼ばれ，OS 起動前の BIOS（Basic Input Output System）環境で実行される．BIOS 環境では OS が起動していないため，OS が提供する GUI 表示や，デバイスドライバによるデバイスアクセスの機能も利用できない．このような環境で多様な認証手段，および使いやすいユーザインタフェースを実現するには，OS と同等の機能を独自に実装しなければならないという課題がある．

本稿では，前述の課題を解決するため，ディスク暗号化システムのプリブート認証に認証専用の OS を用いる方式を提案する．プリブート認証 OS 方式では，暗号化された OS の起動前にプリブート認証 OS を起動してユーザ認証を行う．その後，ユーザ認証により生成された暗号鍵を用いて，暗号化された OS を復号しながら起動する．認証の際にプリブート認証 OS が提供する機能を利用することで，多様な認証手段を提供できる．

以下，2 章では本稿が対象とする脅威を，3 章では先行研究として従来のデータ暗号化方式を，4 章では提案方式の概要を説明する．さらに，5 章では提案方式の実装を示し，6 章では提案方式の OS 起動時間を評価し，最後に考察を加える．

2. 対象とする脅威

本章では，本稿が対象とする情報セキュリティ上の脅威を示し，提案方式において保護する範囲を明らかにする．本稿では，ノートパソコンなどの端末の盗難や紛失にともなう情報漏洩の脅威を対象とする．端末を入手した第三者によるデータの読み出し行為を防ぐことを目的に，暗号化によってディスク内のデータを保護する方式を提案する．また，暗号化データに正規ユーザがアクセスするための認証とデータ復号の手法も述べる．

一方で，暗号化を行うプログラムを狙ったマルウェアや脆弱性の悪用による攻撃や，ユーザ認証に成功して暗号化データにアクセスできる正規ユーザの悪意ある行動，正規ユーザからのパスワードなどの認証情報の不正取得など，ディスク内のデータを直接狙ったものでない脅威は，提案方式による保護の範囲外とする．

次章以降で，暗号化を行うプログラムがこれらの脅威にさらされていないことを前提としたうえで，データ暗号化の従来方式および提案方式を比較する．

3. 従来のデータ暗号化方式

本章では，従来のデータ暗号化方式を分類し，各方式をセキュリティや認証手段などの利便性の観点で比較する．企業や政府機関で利用されるデータ暗号化システムでは，高いセキュリティや，コンピュータの専門家ではないユー

表 1　データ暗号化方式の比較
Table 1　Comparison of the data encryption methods.

方式名	ファイル暗号化方式	ディスク暗号化方式					
		ブートローダ方式	ハイパーバイザ方式	ネットワークブート方式	専用ディスク方式	暗号化ボリュームマウント方式	提案方式
暗号化の範囲	ファイル単位	OS 含むディスク全体				OS 除くディスク全体	OS 含むディスク全体
多様な認証手段	○	× OS なし	○	× OS なし	× OS なし	○	○
ユーザインタフェース	CUI / GUI	CUI	CUI / GUI	CUI	CUI	CUI / GUI	CUI / GUI
構成要件	—	—	仮想化機構	ネットワーク接続	専用ディスク	—	—

ザでも容易に使える利便性が求められる．特に，多様な認証手段を使い分けられることや，ノートパソコンのようなモバイル端末に導入し，外出先での盗難や紛失の際に情報漏洩を防止できることは重要である．

従来のデータ暗号化方式は，暗号化の範囲により，ディスクに格納された特定のファイルのみを暗号化するファイル暗号化方式と，ディスク全体を暗号化するディスク暗号化方式に分類される．各方式の違いを**表 1** に示す．

3.1　ファイル暗号化方式

ファイル暗号化方式は，ユーザが指定した特定のファイルを暗号化する方式である [3]．暗号化の対象をファイル単位で細かく指定できる反面，ユーザの運用により重要なデータの暗号化忘れが起こりうるなど，セキュリティ面に課題が残る．

たとえば，重要なファイルを暗号化すると，そのファイルを参照するたびに復号し，参照が終わると再び暗号化しなければならない．PDF など一部のドキュメントでは対応ソフトウェアに暗号化機能が組み込まれ，復号と再暗号化は不要になるが，代わりにソフトウェアごとに異なる操作を理解する必要がある．このような操作をユーザが独自の判断で正確に行うことは難しい．NASA における情報漏洩の事例 [7] でも，重要なファイルの暗号化が推奨されていたにもかかわらず，紛失したノートパソコン内のファイルは暗号化されていなかった．情報漏洩の発生後は，すべての PC にディスク暗号化ソフトウェアを導入する方針が示されている．

一方，ファイル暗号化ソフトウェアは OS 上で動作するアプリケーションとして提供される．ユーザが使い慣れたインタフェースを提供できるため，利便性は高い．

3.2　ディスク暗号化方式

ディスク暗号化方式は，ファイルを格納するディスク全体を暗号化する．ファイル暗号化方式に比べデータ暗号化の範囲が広く，データをディスクに書き込む際に自動的に暗号化されることで暗号化忘れも起こらないため高いセキュリティを実現できる．また，OS も含めたディスク全体を暗号化すると，OS が作成するシステムファイルに残るデータも暗号化できる．これにより多くのデータを保護できるが，暗号化された OS を復号して起動するための仕組みを新たに設ける必要があり，利便性を損なう場合がある．

ディスク暗号化方式は，以下の 5 方式に分類される．(a) から (d) が OS も含めて暗号化する方式，(e) がデータのみを暗号化する方式である．

(a) ブートローダ方式

OS を起動するブートローダにおいて暗号化処理を行い，OS の復号と起動を行う方式である．ディスク暗号化ソフトウェアがコンピュータの電源投入後に呼び出される OS のブートローダを置き換え，OS を復号し起動する．OS の標準機能として搭載されている Microsoft 社の BitLocker [5] [*1]や Apple 社の File Vault 2 [1] [*2]，オープンソースで開発が進む TrueCrypt [14] などがある．ブートローダ方式では，ディスク暗号化ソフトウェアが OS 起動前の環境で動作するため，インタフェースがユーザにとって使い慣れないものとなる．また，トークンデバイスなどの特殊なデバイスを用いた認証手段を提供する場合は，OS 内のデバイスドライバ相当の機能を別途開発する必要がある．

(b) ハイパーバイザ方式

コンピュータを仮想化して OS を仮想マシン上で実行し，ハイパーバイザにおいて暗号化処理を行う方式である [4], [13]．ディスク暗号化ソフトウェアはハイパーバイザ上に導入され，仮想マシンが発行するディスク I/O 命令を取得し，暗号化されたデータを復号する．ハイパーバイザ方式では仮想化にともなう性能面のオーバヘッドが存在

[*1] BitLocker は，米国 Microsoft Corp. の米国およびその他の国における登録商標です．
[*2] File Vault は，米国 Apple Inc. の米国およびその他の国における登録商標です．

する．このオーバヘッドはプロセッサの仮想化支援機能や様々な最適化手法により低減させることができるものの，省電力性が重視されるモバイル端末においては，必ずしも仮想化支援機能が搭載されているとは限らず，ハイパーバイザの存在を前提とすることは難しい．また，ブートローダ方式と同様に，ユーザインタフェースや，特殊デバイス利用のためには，OS 相当の機能の開発が必要となる．

(c) ネットワークブート方式

データが格納されたディスクを端末ではなく，ネットワーク接続されたサーバ側に保持し，サーバ上で暗号化処理を行う方式である [11]．端末にデータを格納しないため高いセキュリティを実現できる反面，ネットワークに接続できない環境では OS を起動できないため，端末を持ち運ぶ場合，外出先での利用に制限がある．

(d) 専用ディスク方式

Opal [15] のような規格に則り，暗号化機能を搭載した専用のディスクを用いる方式である [19]．暗号化処理をハードウェアで行うことにより，ソフトウェアで行う他方式と比べて性能面に優れている．しかし，専用のディスクを用いるため，それらを未搭載の端末ではディスクの交換と OS の再インストールの作業が必要となる．そのため，ユーザが利用している既存の端末への導入が難しい．

(e) 暗号化ボリュームマウント方式

複数のディスクやボリュームを用意して OS は非暗号化，OS 以外のデータは暗号化されたボリュームに格納し，OS の起動後に暗号化ボリュームをマウントする方式であり，Linux [*3] の dm-crypt [10] などが開発されている．OS をそのまま利用できるため，ユーザが使い慣れたインタフェースを提供できる．一方で，OS は平文で保存されるため，一部のデータを保護できないという問題がある．

たとえば，OS はスワップファイルを仮想メモリとして利用し，メモリの内容をディスクに保存することがある．また，ユーザが過去に開いたファイルの履歴やキャッシュなどを独自に保存する場合がある．このように，OS が自動的に作成するシステムファイル内に重要なデータが格納される可能性があるが，そのすべてを個々のユーザが把握し対処することは難しい．したがって，重要データを保護するには OS も含めて暗号化することが望ましい．

3.3 従来方式の課題

ファイル暗号化方式では，ファイル単位で暗号化をするため暗号化忘れの問題がある．OS を含めないディスク暗号化方式は，システムファイルに重要データが残る恐れがあるため同様に問題である．一方，OS も含むディスク暗号化方式では，ディスク全体を暗号化するため，暗号化忘れの問題はないものの，暗号化された OS を起動する前にユーザ認証を行うプリブート認証が必要となる．しかし，プリブート認証では多様な認証手段や使いやすいインタフェースの提供が難しい．

プリブート認証を行う環境では OS が起動していないため，デバイスドライバを用いたデバイス固有の機能の制御や，マウス操作やウィンドウシステムなどの GUI 制御など，OS が提供する様々な機能を利用することができない．したがって，トークンデバイスなどの特殊なデバイスを用いた認証や，GUI などによるユーザインタフェースの提供が困難となる．これを実現するには OS 相当の機能を独自に実装しなければならない．

4. 提案方式の概要

本章では，ディスク暗号化方式において多様な認証手段を提供するプリブート認証 OS 方式の概要を述べる．

4.1 プリブート認証 OS 方式の提案

前述の課題を解決するために，本稿ではプリブート認証 OS 方式を提案する．プリブート認証 OS 方式では，暗号化された OS（以下，**ユーザ OS**）の起動前に認証専用の OS（以下，**プリブート認証 OS**）を起動し，プリブート認証 OS 上でユーザ認証を行う．

プリブート認証 OS を用いることで，ユーザ認証の機能を OS 上のアプリケーションとして実装し，OS が持つ様々な機能を利用できるようになる．これにより，ディスク暗号化方式の特徴である高いセキュリティを確保しながら，従来方式では困難であった，多様な認証手段の提供を可能にする．

プリブート認証 OS 方式は，ブートローダ方式におけるディスク暗号化ソフトウェアをプリブート認証 OS に置き換えたものに相当する．このような構成をとることで，ハイパーバイザ方式のような仮想化によるオーバヘッドがなく，ネットワークブート方式におけるネットワーク接続や，ハードウェア方式における専用ディスクなども不要となる．したがって，モバイル端末も含む様々な端末に導入することができる．

4.2 システム構成

プリブート認証 OS 方式のシステム構成を**図 1** に示す．プリブート認証 OS 方式は，(1) 暗号化されたユーザ OS と，(2) ユーザ認証専用のプリブート認証 OS からなる．

(1) ユーザ OS

ユーザ OS は，ユーザが業務などの用途で利用する OS であり，情報漏洩防止のため暗号化された状態でディスクに格納される．暗号化されたデータを読み書きするため，ユーザ OS は暗号化フィルタを有する．

(2) プリブート認証 OS

プリブート認証 OS は，暗号化されたユーザ OS を起動

[*3] Linux は，Linus Torvalds 氏の米国およびその他の国における登録商標です．

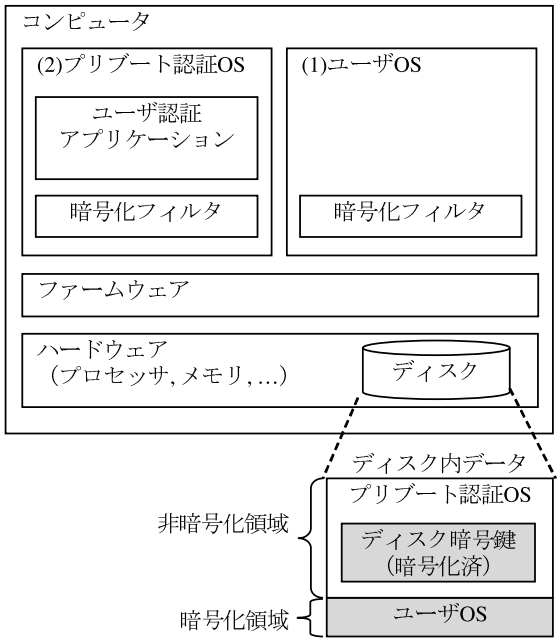


図 1　提案方式のシステム構成
Fig. 1　Configuration of the proposed method.

するためにユーザ認証処理を行う専用の OS であり，暗号化されずにディスクに格納されている．ユーザ認証を行うアプリケーションのほか，ユーザ OS と同様に暗号化されたデータを読み書きするための暗号化フィルタを有する．

4.3　暗号鍵の管理

ディスクの暗号化領域に格納されたデータの暗号鍵は，別に用意された暗号鍵で暗号化され，プリブート認証 OS と同様の非暗号化領域に格納される．ディスク暗号鍵の所在はユーザ認証アプリケーションが把握しており，パスワードなどによるユーザ認証に成功することで，ディスク暗号鍵が復号されてメモリ上に配置され，プリブート認証 OS から利用可能となる．

4.4　OS 起動処理

プリブート認証 OS 方式におけるユーザ OS 起動までの処理概要を**図 2** に示す．コンピュータの電源が投入されると，ファームウェアによるデバイス初期化処理に続いてプリブート認証 OS がユーザ OS よりも先に起動する．

プリブート認証 OS はユーザ認証アプリケーションを実行し，パスワードやトークンデバイスなどによるユーザ認証の実施をユーザに対して求める．ユーザ認証に成功した場合，ユーザ認証アプリケーションは暗号化された状態のディスク暗号鍵を復号し，暗号化フィルタとともにメモリ上に配置する．以上の操作を行った後，プリブート認証 OS は終了しユーザ OS に処理を切り替える．

ユーザ OS はメモリ上の暗号鍵と暗号化フィルタを用いてディスク内の暗号化領域からユーザ OS 自身を復号しながら起動する．これにより，暗号化されたユーザ OS の起動を実現する．

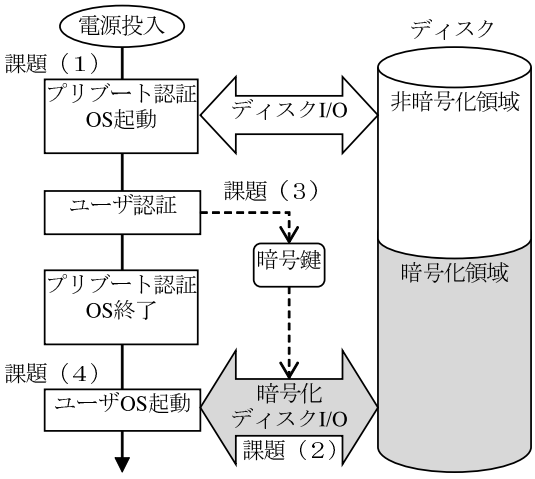


図 2　提案方式による OS 起動処理
Fig. 2　OS boot sequence of the proposed method.

4.5　プリブート認証 OS 方式の課題

プリブート認証 OS を用いたディスク暗号化を実現するには，OS 起動処理における以下の 4 つの課題を解決する必要がある．各課題は図 2 の課題（1）から（4）に対応する．

（1）プリブート認証 OS の起動

暗号化されたユーザ OS よりも前にプリブート認証 OS を起動し，ユーザ認証を行う．

（2）暗号化ディスク I/O 処理

暗号化されたディスクを読み書きするために，ディスク I/O を暗号化もしくは復号する．

（3）暗号鍵の共有と保護

プリブート認証 OS がディスクから読み出し，復号した暗号鍵をユーザ OS との間で共有し，さらに他のプログラムによって上書きされないように保護する．

（4）リブートレス起動

プリブート認証 OS を終了してユーザ OS を起動する際に，コンピュータを再起動せず，暗号鍵をメモリ上に保持したまま OS を切り替える．

次章において，これらの課題を解決するプリブート認証 OS 方式の実装を説明する．

5.　プリブート認証 OS 方式の実装

本章では，4.5 節で述べた課題（1）～（4）を解決するプリブート認証 OS 方式の実装を述べる．ここではユーザ OS として Windows[*4] 7 を，プリブート認証 OS として Linux カーネル 2.6.18 を利用する．また，コンピュータのファームウェアとして BIOS を，プロセッサとして Intel 社 x86 系を利用している．

[*4] Windows は，米国 Microsoft Corp. の米国およびその他の国における登録商標です．

提案方式ではOS固有の機能は使わないため，コンピュータ上で動作可能であれば任意のOSをユーザOSやプリブート認証OSとして利用できる．プリブート認証OSについては，次節以降の処理をOS起動・終了プロセスやデバイスドライバに実装する必要があるため，Linuxのように改変が容易なOSであることが望ましい．

5.1　プリブート認証OSの起動

本節では，4.5節の課題（1）を解決するため，暗号化されたユーザOSよりも前にプリブート認証OSを起動する方法について述べる．

OSは，コンピュータの電源投入後に呼び出されるBIOSによってディスクから読み込まれ，起動される．ディスクの先頭セクタはMBR（Master Boot Record）と呼ばれ，OSのブートローダのアドレスが格納されている．BIOSはそのアドレス情報をもとにブートローダを実行し，ブートローダがOS本体をディスクから読み出すことでOSの起動が行われる．

提案方式において，ユーザOSより前にプリブート認証OSを起動するには，MBRに格納されたアドレス情報をプリブート認証OSのブートローダのアドレス情報に書き換えればよい．このとき，MBRを書き換えるとともにユーザOSのブートローダのアドレス情報を保存しておき，プリブート認証OSの終了後にユーザOSのブートローダを実行することで，ユーザOSを起動することができる．

5.2　暗号化ディスクI/O処理

本節では，4.5節の課題（2）を解決するため，暗号化されたディスクに対してI/Oを行うために必要な暗号化フィルタの挿入処理について説明する．

ユーザOSは暗号化された状態でディスクに格納されている．そのため，それらのデータを読み書きするプログラムは，ディスクからデータを読み込む際にデータを復号する必要がある．また，ディスクに対しデータを書き込む場合には，反対に暗号化が必要である．ディスクの暗号化領域に対して行われるこのような処理を暗号化ディスクI/Oと呼ぶ．提案方式では，**図3**に示すようにディスクI/O処理の中途に暗号化フィルタを挿入することで，暗号化ディスクI/Oを実現する．

データの暗号化には，暗号化前後のサイズが一致するブロック暗号方式を用いる．ディスクI/Oを発行するプログラムはデータの所在をセクタ単位で指定しており，暗号化によりサイズが変わり異なるセクタにデータを格納すると齟齬が生じるためである．代表的なブロック暗号方式として，米国標準に採用されているAES（Advanced Encryption Standard）などがある．

暗号化フィルタはユーザOSの起動前後で異なる位置に挿入する必要がある．これは，それぞれの環境でディスクI/Oの方法が異なるためである．以下，5.2.1項でユーザOS起動前，5.2.2項でユーザOS起動後の環境について説明する．

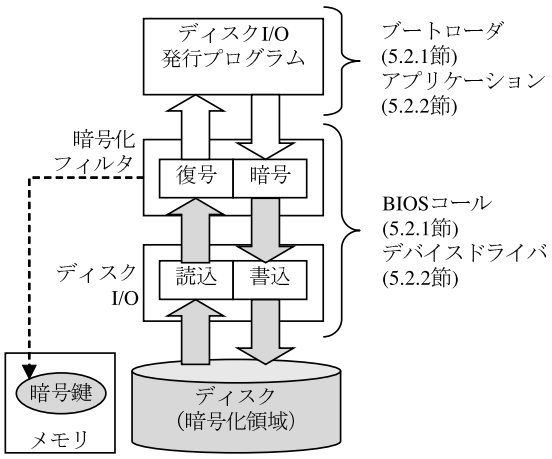


図3　暗号化ディスクI/O処理
Fig. 3　Processing of the encrypted disk I/O.

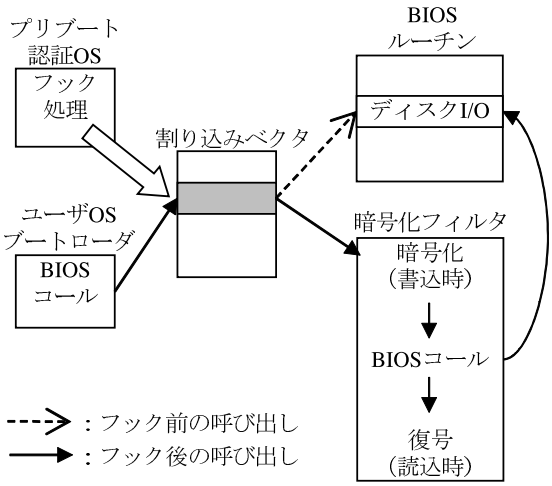


図4　ユーザOS起動前の暗号化フィルタの挿入
Fig. 4　Insertion of an encryption filter before User OS booted.

5.2.1　ユーザOS起動前の暗号化フィルタの挿入

ユーザOS起動前の環境では，ブートローダなどのプログラムはBIOSコールと呼ばれるソフトウェア割込み命令を利用してディスクI/Oを行う．BIOSはBIOSコールの発行を検知すると，割込みベクタを参照し，検知した割込みの番号に対応したルーチンを呼び出し，ディスクI/Oなどの処理を行う．

提案方式では，BIOSコールをフックすることで，ディスクI/O処理の中途に暗号化フィルタを挿入する．

フック処理の概要を**図4**に示す．プリブート認証OSは，BIOSの割込みベクタに格納された値を，ディスクI/O処理を行うルーチンのアドレスから暗号化フィルタのアドレスに書き換える．そして，暗号化フィルタからディスクI/O処理を行う本来のルーチンを呼び出すように設定する．

以上の処理により，ブートローダなどがBIOSコールを

発行した際に，暗号化フィルタを経由したディスク I/O を行う．BIOS コールをフックすることで，ユーザ OS やそのブートローダを改変することなく，暗号化ディスク I/O を実現する．

5.2.2　ユーザ OS 起動後の暗号化フィルタの挿入

ユーザ OS 起動後の環境では，アプリケーションなどのプログラムはシステムコールを発行し，それを受けた OS がディスク I/O ドライバを利用してディスク I/O を行っている．

提案方式では，暗号化フィルタをデバイスドライバとして作成し，あらかじめユーザ OS のディスク I/O ドライバの上位ドライバとして暗号化フィルタドライバを挿入しておく．これにより，ユーザ OS 起動前の環境と同様の暗号化ディスク I/O を実現する．

5.3　暗号鍵の共有と保護

本節では，4.5 節の課題（3）を解決するため，暗号鍵をプリブート認証 OS とユーザ OS との間で共有する方法について説明する．

データの暗号化および復号に用いる暗号鍵はプリブート認証 OS におけるユーザ認証で生成される．暗号鍵は，ディスクなどの不揮発性デバイスに格納するとコンピュータの紛失時に暗号鍵が流出する恐れがあるため，メモリなどの揮発性デバイスに格納すべきである．このとき暗号鍵は 5.2 節で述べた 2 カ所の暗号化フィルタから参照されるため，両者から参照できるアドレスに配置する必要がある．また，メモリ上の暗号鍵は，BIOS やユーザ OS などにより上書きされないように保護する必要がある．

提案方式において，5.2.1 項で述べた BIOS ルーチンはリアルモードと呼ばれる 16 bit の環境で実行され，メモリ空間が 1 [MB] に制限される．そこで，提案方式では暗号鍵をメモリの先頭から 1 [MB] 以内の空き領域に配置することで，ユーザ OS 起動前後の環境における暗号鍵の共有を実現する．また，BIOS などのファームウェアが備えるメモリ管理テーブルを変更し，暗号鍵を格納した領域を予約領域と設定することで，ユーザ OS のブートローダや BIOS のルーチンが暗号鍵を上書きすることを防ぐ．これにより，暗号鍵の保護を実現する．

5.4　リブートレス起動

本節では，4.5 節の課題（4）を解決するため，メモリ上に配置した暗号鍵を保持したままプリブート認証 OS からユーザ OS へ切り替えるためのリブートレス起動の処理について説明する．

5.4.1　OS 切替え処理

一般に，デュアルブートのような 1 つのコンピュータに複数の OS をインストールした構成では，OS を切り替える際はコンピュータを再起動する．このときメモリ上のデータは BIOS の再起動ルーチンによってすべて消去される．これによりプリブート認証 OS が生成した暗号鍵も消去され，ユーザ OS を復号できなくなる．そのため，プリブート認証 OS からユーザ OS へ切り替える際は，コンピュータの再起動を迂回する必要がある．このような処理をリブートレス起動と呼ぶ．

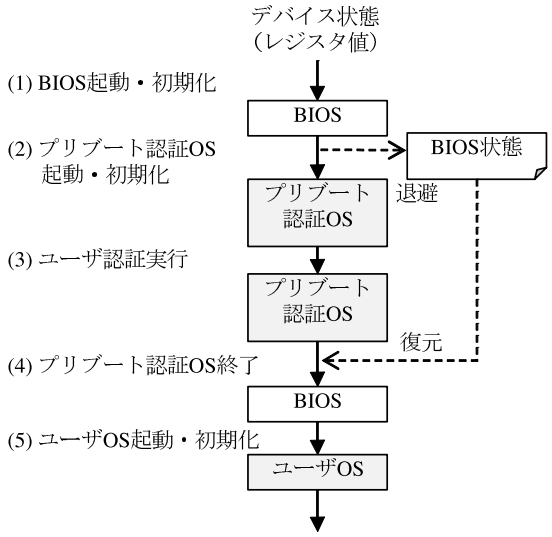


図 5　デバイス状態の退避と復元
Fig. 5　Saving and restoring the device status.

提案方式では，プリブート認証 OS の終了処理を変更し，BIOS の再起動ルーチンを実行する代わりに，5.1 節において MBR から取得したユーザ OS のブートローダを読み込み，起動する命令を実行している．これにより，コンピュータの再起動を迂回し，メモリ上の暗号鍵を保持したまま，プリブート認証 OS からユーザ OS へ処理を切り替えることができる．また，副次的な効果として BIOS の起動を省略できるため，ユーザ OS への切替えが高速となる．

5.4.2　デバイス状態の退避と復元

リブートレス起動を行うと，メモリ上のデータを消去しない代わりに，デバイスの状態を表すレジスタの値がプリブート認証 OS により変更されたまま残るという問題がある．

OS は，OS 起動処理の中でデバイスの初期化を行う．これにより各デバイスのレジスタ値は BIOS が初期化した値から変化し，OS がデバイスドライバを介してデバイスを利用できるようになる．その一方で，BIOS からはデバイスを利用できなくなる．これは，プリブート認証 OS が終了した後に，ユーザ OS のブートローダが BIOS コールを発行しても，BIOS がディスク I/O を実行できず，ユーザ OS を起動できないことを意味する．

提案方式では，この問題の解決のために，プリブート認証 OS においてデバイス状態の退避と復元を行う．処理の概要を**図 5** に示す．

（1）コンピュータの起動時，デバイスは BIOS により初期

化され，レジスタにはBIOSが定めた値が格納される．
(2) プリブート認証OSが起動すると，デバイスはプリブート認証OSにより初期化され，レジスタにはプリブート認証OSが定めた値が格納される．このとき，プリブート認証OSはデバイスの初期化処理よりも前にレジスタの値を退避させる．
(3) プリブート認証OSの実行中は，デバイス状態はプリブート認証OSが任意に変更できる．
(4) プリブート認証OSの終了時，退避させておいた値をレジスタに格納し，BIOSにより初期化されたデバイス状態を復元する．
(5) ユーザOSの起動時は，プリブート認証OSの終了時にデバイス状態を復元したことで，BIOSがデバイスを再び利用できるようになる．これにより，通常どおりBIOSを用いてディスクI/Oを実行できる．

以上の処理によりリブートレス起動にともなう問題を回避し，メモリ上の暗号鍵を保持したままプリブート認証OSからユーザOSへ処理を切り替える．

6. 評価

本章では，本稿で提案したプリブート認証OS方式について，ユーザOS起動までの時間や提案方式での実現機能を評価し，プリブート認証OSを利用しない従来方式と比較する．

6.1 測定環境

測定環境を**表 2** に示す．この環境において，従来方式と提案方式のそれぞれについて，コンピュータの電源投入からユーザOSのユーザ認証完了までの時間を3回測定し，ユーザOSの平均起動時間を算出した．起動時間はBIOS起動，プリブート認証OS起動，ユーザ認証，ユーザOS起動の4つに分解し，それぞれの処理時間を測定した．なお，ユーザ認証処理の時間は，ユーザ名とパスワードをキーボードから入力するのに要する時間として一律10秒とした．暗号化のアルゴリズムには，ブロック暗号の一種であるAES 128 bitを利用している．

従来方式は，提案方式がプリブート認証OS上で行うユーザ名とパスワードによる認証処理を，BIOS上で行うプログラムとして実装している．また，提案方式のプリブート認証OSとして利用するLinuxは，カーネルのコンフィグレーションを初期設定から変更しユーザ認証に不要な機能やデバイスドライバを無効化している．これにより，状態の退避と復元が必要なデバイス数の削減および起動時間の短縮を図る．

表 2　測定環境
Table 2　Measurement environment.

項目	諸元
CPU	Intel Core[*5] 2 Duo T7100 1.80 [GHz]
メモリ	3 [GB]
ハードディスク	80 [GB] 5400 [RPM]
BIOS	Lenovo 7NETC2WW 2.22 (Phoenix TrustedCore[*6])
ユーザOS	Microsoft Windows 7 SP1 32 bit
プリブート認証OS	CentOS 5.6 32 bit (Linux kernel 2.6.18)
GUI基盤	X Window System
暗号化方式	AES 128 bit

[*5] Coreは米国Intel Corp.の米国およびその他の国における登録商標です．
[*6] TrustedCoreは米国Phoenix Technologies Ltd.の米国およびその他の国における登録商標です．

6.2 考察

ユーザOSの平均起動時間を測定した結果を**図 6** に，従来方式と提案方式のそれぞれにおいて実現できる機能および各々のTCB（Trusted Computing Base）サイズを**表 3** に示す．

提案方式のTCBは，プリブート認証OSのブートローダ，カーネル，ユーザ認証アプリケーションおよびその実行基盤となるライブラリであり，コード行数は10^6のオーダであった．また，従来方式のTCBはBIOS上で動作するプログラムとそのブートローダであり，コード行数は

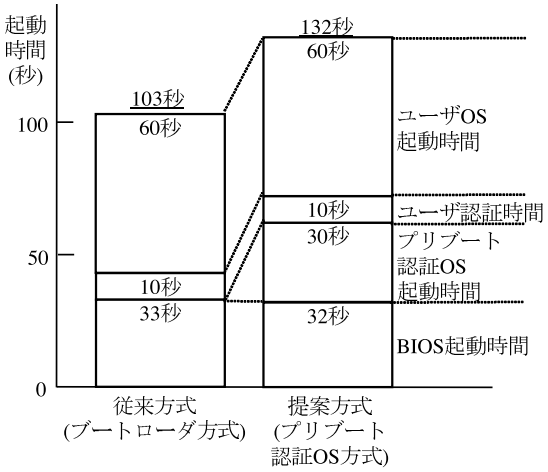


図 6　OS起動時間の評価
Fig. 6　Evaluation of OS boot time.

表 3　従来方式と提案方式の比較
Table 3　Comparison of existing methods and the proposed method.

比較項目		従来方式	提案方式
セキュリティ	OS含むディスク暗号化	○	○
	TCBサイズ [行数]	10^4	10^6
機能	パスワード認証	○	○
	チャレンジ・レスポンス認証	×	○
	シングルサインオン	×	○
	GUI表示	×	○

10^4 のオーダであり，2 桁の差異がある．

ユーザ OS の平均起動時間については，提案方式ではプリブート認証 OS の起動に約 30 秒を要し，従来方式と比べて起動時間が増加している．

提案方式の評価結果に関し，(1) OS 起動時間，(2) セキュリティ，(3) ユーザ認証機能の観点から考察を行う．

(1) OS 起動時間

提案方式において，ユーザ OS 起動完了までの時間は従来方式よりも約 30 秒増加している．この時間はプリブート認証 OS の起動時間に相当しており，プリブート認証 OS 上でのデバイス初期化，デバイスドライバ読み込み，GUI 環境起動に要する時間が含まれている．プリブート認証 OS ではカーネルのコンフィグレーションを変更して起動時間を短縮しているが，それに加えて GUI 基盤として利用している X Windows System の起動高速化などにより提案方式の OS 起動時間を改善できると考える．

(2) セキュリティ

(2 − 1) データの保護

従来方式と提案方式はともにディスク暗号化方式の一種であり，OS も含むディスク全体を暗号化するため，OS のシステムファイルも含めたデータを保護できる．これにより，端末の紛失や盗難にともなう情報漏洩事故を防ぐことができる．

(2 − 2) 暗号鍵の保護

ディスク暗号鍵は，ユーザ認証を行うまでは，別に用意された鍵で暗号化されてディスクに格納されている．そのため復号に用いるパスワードなどの情報が漏洩しない限り安全である．Apple 社の FileVault 2 など，ソフトウェアによってはディスク暗号鍵を外部のサーバに預けることができる．この方式ではディスクへのアクセスにネットワーク接続が必須となるため利用できる環境が限られるが，より高いセキュリティを実現できる．

また，4.3 節で述べたように，ディスク暗号鍵はユーザ認証後に平文の状態でメモリ上に配置される．このとき，悪意あるユーザが端末を操作できるとメモリダンプなどによりディスク暗号鍵を取得される恐れがある．不正取得を防止するには，データを配置するメモリアドレスを無作為に設定してデータの所在の特定を困難とする ASLR（Address Space Layout Randomization）[6] などの対策を併用する必要がある．

(2 − 3) プリブート認証 OS の保護

2 章で述べたように，提案方式はディスクに格納されたデータを保護するものであり，プリブート認証 OS 自身は保護の対象ではない．しかしながら，提案方式の TCB は従来方式と比べプリブート認証 OS の分だけ増加しており，攻撃される可能性が高くなっている．

攻撃の例として，非暗号化状態でディスクに格納されているプリブート認証 OS のファイル構造を解析し OS やアプリケーションを改ざんする，OS の脆弱性を悪用して任意のプログラムを実行させる，ブートローダを置き換えてブートキットに感染させる，などがあげられる．これらの攻撃により，ユーザのパスワードやディスク内のデータを復号するための暗号鍵を不正に取得される恐れがある．

対策として TPM（Trusted Platform Module）[16] を用いた Trusted Boot による改ざん検知や，プリブート認証 OS 内で指定したプログラム以外を起動させないホワイトリスト型の実行制御が有効である．これらの対策を行うことで，暗号化によるデータの保護に加え，プリブート認証 OS 自身の保護も可能となり，より高いセキュリティを実現できる．

(3) ユーザ認証機能

BIOS 上で実行される従来方式は，使用できるメモリ領域やデバイスアクセスに制限を受けるため，実現できる機能が限られる．一方，提案方式は，物理メモリ領域全体を使えることに加え，OS が持つ GUI 基盤を利用できるため，GUI による認証画面やマウスなどの入力デバイスによる操作も実現する．ユーザ認証の際はユーザ名とパスワードに基づく一般的な認証のほかに，チャレンジ・レスポンス認証や，プリブート認証 OS での認証に成功すれば再度パスワードを入力することなくユーザ OS にログイン可能となるシングルサインオンなどの機能を実現している．

また，提案方式を利用することで，認証トークンデバイスを用いた二要素認証のような，より強固なセキュリティを実現する見込みが得られた．提案方式において実現可能な認証手段は，認証に用いるデバイスのデバイスドライバの有無に依存する．例として，認証用トークンデバイスや指紋認証用リーダのデバイスドライバがプリブート認証 OS 上で動作しない場合，その装置を用いた認証手段は提供できない．しかしながら，プリブート認証 OS に Linux を採用することで，各デバイスの開発元が提供する既存のデバイスドライバを利用でき，多くの認証手段に対応可能となる．

7. おわりに

本稿では，ディスク暗号化システムにおける OS 起動およびユーザ認証の方式として，プリブート認証 OS 方式を提案した．プリブート認証 OS が持つデバイスドライバや GUI 基盤を利用することで，多様な認証手段と GUI によるユーザインタフェースを有する認証機能を実現する．

プリブート認証 OS 方式では，コンピュータの電源投入後にプリブート認証 OS を起動し，ユーザ認証を行って暗号鍵を生成した後，その暗号鍵を利用してユーザ OS を復号し起動する．プリブート認証 OS の終了時はリブートレス起動によりコンピュータの再起動を迂回することで，メモリ上の暗号鍵を消去することなく，ユーザ OS へと処理を切り替える．

提案方式を評価した結果，従来のブートローダ方式に比べてユーザ OS 起動完了までの時間を約 30 秒増に抑えたうえで，チャレンジ・レスポンス認証などの多様な認証手段や，GUI やシングルサインオンを用いたユーザインタフェースを提供できることを確認した．

今後の課題として，UEFI（Unified Extensible Firmware Interface）[17] の評価があげられる．本稿ではファームウェアとして BIOS を対象としたが，近年はより高機能な UEFI の採用が進んでいる．UEFI では GUI 表示やデバイス接続などの仕組みが規格化されており，プリブート認証 OS 相当の機能を UEFI アプリケーションとして実装できると期待される．OS ブートが不要となり起動時間が短縮される可能性があるが，既存のデバイスドライバ資産を利用できないため多様な認証手段の実現が難しいと考えられ，UEFI アプリケーションによる実装可否も含めた評価が必要である．

参考文献

[1] Apple: FileVault 2, Apple Inc. (online), available from ⟨http://support.apple.com/kb/HT4790⟩ (accessed 2014-03-10).

[2] Baker, W., Goudie, M., Hutton, A., Hylender, C.D., Niemantsverdriet, J., Novak, C., Ostertag, D., Porter, C., Rosen, M., Sartin, B. and Tippett, P.: 2010 Data Breach Investigations Report, United States Secret Service (online), available from ⟨http://www.wired.com/images_blogs/threatlevel/2011/04/Verizon-2011-DBIR_04-13-11.pdf⟩ (accessed 2013-06-12).

[3] Blaze, M.: A Cryptographic File System for UNIX, *Proc. 1st ACM Conference on Computer and Communications Security*, *CCS '93*, pp.9–16, ACM (online), DOI: 10.1145/168588.168590 (1993).

[4] Liang, M. and wen Chang, C.: Research and design of full disk encryption based on virtual machine, *2010 3rd IEEE International Conference on Computer Science and Information Technology* (*ICCSIT*), Vol.2, pp.642–646 (online), DOI: 10.1109/ICCSIT.2010.5565144 (2010).

[5] Microsoft: BitLocker Drive Encryption, Microsoft Corp. (online), available from ⟨http://windows.microsoft.com/en-US/windows7/products/features/bitlocker⟩ (accessed 2013-05-08).

[6] Microsoft: Windows ISV Software Security Defenses, Microsoft Corp. (online), available from ⟨http://msdn.microsoft.com/en-us/library/bb430720.aspx⟩ (accessed 2014-02-28).

[7] NASA: NASA's Efforts to Encrypt its Laptop Computers, NASA (online), available from ⟨http://oig.nasa.gov/specialReview.html⟩ (accessed 2013-06-10).

[8] O'Gorman, L.: Comparing passwords, tokens, and biometrics for user authentication, *Proc. IEEE*, Vol.91, No.12, pp.2021–2040 (online), DOI: 10.1109/JPROC.2003.819611 (2003).

[9] Phoenix: TrustedCore PreBoot Authentication, Phoenix Technologies Ltd. (online), available from ⟨http://www.phoenix.com/pages/pre-boot-authentication⟩ (accessed 2013-05-08).

[10] Saout, C.: dm-crypt: A device-mapper crypto target, dm-crypt (online), available from ⟨http://www.saout.de/misc/dm-crypt/⟩ (accessed 2014-06-14).

[11] Satran, J., Meth, K., Sapuntzakis, C., Chadalapaka, M. and Zeidner, E.: Internet Small Computer Systems Interface (iSCSI), Internet Engineering Task Force (online), available from ⟨http://tools.ietf.org/html/rfc3720⟩ (accessed 2013-04-09).

[12] Scarfone, K.A., Souppaya, M.P. and Sexton, M.: SP 800-111. Guide to Storage Encryption Technologies for End User Devices, Technical report, Gaithersburg, MD, United States (2007).

[13] Shinagawa, T., Eiraku, H., Tanimoto, K., Omote, K., Hasegawa, S., Horie, T., Hirano, M., Kourai, K., Oyama, Y., Kawai, E., Kono, K., Chiba, S., Shinjo, Y. and Kato, K.: BitVisor: A Thin Hypervisor for Enforcing I/O Device Security, *Proc. 2009 ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, *VEE '09*, pp.121–130, ACM (online), DOI: 10.1145/1508293.1508311 (2009).

[14] TrueCrypt: TrueCrypt, TrueCrypt Foundation (online), available from ⟨http://www.truecrypt.org/⟩ (accessed 2013-05-08).

[15] TrustedComputingGroup: Storage Application Note: Encrypting Drives Compliant with Opal SSC, Trusted Computing Group (online), available from ⟨http://www.trustedcomputinggroup.org/resources/storage_application_note_encrypting_drives_compliant_with_opal_ssc⟩ (accessed 2013-06-14).

[16] TrustedComputingGroup: Trusted Platform Module, Trusted Computing Group (online), available from ⟨http://www.trustedcomputinggroup.org/developers/trusted_platform_module⟩ (accessed 2013-05-08).

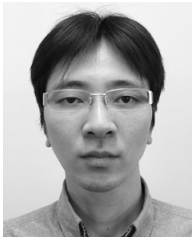
[17] UEFI: UEFI Specification, UEFI Forum (online), available from ⟨http://www.uefi.org/specifications⟩ (accessed 2012-05-23).

[18] 経済産業省：平成 23 年情報処理実態調査結果報告書 (2012).

[19] 二村和明，矢崎孝一，中村洋介，郭　兆功，山田　勇：自己消去を可能にする HDD 認証強化，コンピュータセキュリティシンポジウム 2009（CSS2009）論文集，Vol.2009, pp.1–6 (2011).

平兮 亮　（正会員）

2006 年九州大学工学部電気情報工学科卒業．2008 年九州大学大学院システム情報科学府知能システム学専攻修士課程修了．同年（株）日立製作所入社．現在，横浜研究所に勤務．OS，サーバ仮想化，セキュアプラットフォームの研究開発に従事．

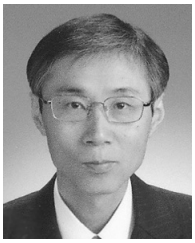


関山 友輝　（正会員）

2004 年京都大学工学部電気電子情報工学科卒業．2006 年京都大学大学院エネルギー科学研究科エネルギー社会・環境科学専攻修士課程修了．同年（株）日立製作所入社．現在，横浜研究所に勤務．OS，サーバ仮想化，セキュアプラットフォームの研究開発に従事．

峯 博史　（正会員）

2000 年東京大学工学部電子工学科卒業．2002 年東京大学大学院工学系研究科情報工学専攻修士課程修了．同年（株）日立製作所入社．現在，横浜研究所に勤務．OS，映像配信技術，セキュアプラットフォームの研究開発に従事．

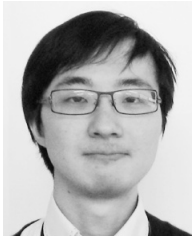


真矢 譲　（正会員）

1980 年愛媛大学工学部電子工学科卒業．同年（株）日立製作所に入社．戸塚工場，システム開発研究所，横浜研究所にて，電子交換機，フォールトトレラントコンピュータ，UNIX サーバ，大形計算機，NAS，サーバ仮想化，セキュアプラットフォームの研究に従事．博士（工学）．電子情報通信学会会員．

大島 訓　（正会員）

1996 年東京理科大学理工学部情報科学科卒業．1998 年東京理科大学大学院理工学研究科情報科学専攻博士課程前期修了．同年（株）日立製作所入社．現在，横浜研究所勤務．OS，サーバ仮想化の研究開発に従事．



猪口 修史

2009 年学習院大学理学部数学科卒業．同年（株）日立製作所入社．現在，セキュリティ関連ソフトウェア，情報システム向け OS・ミドルウェアの設計開発に従事．

河野 健二　（正会員）

1993 年東京大学理学部情報科学科卒業．1997 年東京大学大学院理学系研究科情報科学専攻博士課程中退，同専攻助手に就任．現在，慶應義塾大学理工学部情報工学科准教授．博士（理学）．1999 年度，2008 年度，2009 年度情報処理学会論文賞受賞．2000 年度山下記念研究賞受賞．オペレーティングシステム，システムソフトウェア，ディペンダブルシステム，インターネットセキュリティ等に興味を持つ．IEEE/CS，ACM，USENIX 各会員．