社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2007－OS－104 (8)
2007／1／31

Privacy-Aware Data Object Containerによる細粒度データアクセス制御方式

鈴来 和久† 毛利 公一†† 大久保 英嗣††
†立命館大学大学院理工学研究科 ††立命館大学情報理工学部

プライバシ保護では，データの機密度やデータの提供者の意図に適応可能なデータ保護方式を実現する必要がある．このような要求を満たすためには，データの提供者の意図を越えたデータの伝搬を防ぐ必要がある．現在，主流となっているデータ保護手法は，計算機におけるデータ管理の基本単位であるファイルを保護の粒度とするアクセス制御方式である．提案手法は，ファイルより細粒度のアクセス制御を実現するために，保護すべきおのおののデータへのアクセスごとに，その可否を制御可能とするデータ保護方式である．本論文では，細粒度データアクセス制御を実現するPADOC（Privacy-Aware Data Object Container）の概要，プロトタイプシステムの実装，PADOCを適用したアプリケーションの実装例について述べる．

A Design of Privacy-Aware Data Object Container for An Adaptive and Fine-Grained Data Access Control

KAZUHISA SUZUKI† KOICHI MOURI†† EIJI OKUBO††
†Graduate School of Science and Engineering, Ritsumeikan University
††College of Information Science and Engineering, Ritsumeikan University

In the research area of the protection of private data, it is necessary to realize a special data protection method which satisfies confidentiality of sensitive data or intentions of owners of the data. In order to satisfy these requirements, it is necessary to prevent propagation of sensitive data against the owners' intentions. Several data protection methods generally used in today's computer systems guard each of a file which is a basic unit of the data management, from illegal accesses. Our proposal method is that it is able to control data access processings by each of sensitive data in order to realize a fine-grained data access control. In this paper, the overall goals of this reseach topic, details of Privacy-Aware Data Object Container (PADOC), a prototype implementation of PADOC, and an application example using PADOC are described.

1 はじめに

近年，個人情報や企業の機密情報が漏洩する事故が多発している．また，これらの情報漏洩に起因する悪質な勧誘行為や架空請求などの被害が発生している．このような背景から，電子化されたデータを保護するセキュリティ技術の開発や，計算機を利用するユーザに適用するガイドライン（セキュリティポリシ）の策定が進められている．

現在，我々が開発している *Salvia* [1] は，計算機で処理されるデータの漏洩を防ぐデータ保護機構を持つオペレーティングシステム（以下，OSと記す）である．特に，*Salvia* は，データの所有者（提供者）によって作成されたデータ保護ポリシに基づき，コンテキストに適応可能なアクセス制御を実現している点が特徴的である．これにより，プライバシを考慮する必要がある個人情報などの保護に適したデータ保護を実現している．

Salvia のようなデータ漏洩の防止を実現する計算機システムは，一般的に秘匿性（confidentiality）や完全性（integrity）に着目したセキュリティモデルに基づくデータ保護技術を適用している．秘匿性とは，データへのアクセス

権限を持たないユーザにそのデータが伝搬することを防ぐことである．すなわち，データにアクセスする主体とアクセスの対象にセキュリティレベルを与え，セキュリティレベル間の情報フローを制御する．これにより，高位のセキュリティレベルに属するデータが低位のセキュリティレベルに属するユーザに伝搬することを防ぐ．完全性とは，計算機システムの状態が正しいことを保証することである．すなわち，データや計算機システムが提供するサービスの破壊や不正な変更を防ぐために，完全性の低いデータによって完全性の高いデータが汚染されないように情報フローを制御する．

本論文では，秘匿性を考慮した細粒度データ管理手法 PADOC（Privacy-Aware Data Object Container）について述べる．PADOC は，*Secure Object* [2] の着想を発展させたものである．PADOC は，ファイル中のおのおののデータへのアクセスを制御可能とするためのデータの管理方式を実現する．これにより，データごとにその利用目的や提供可能な範囲が異なる特徴を持つプライバシデータに適したデータ保護を実現する．以下，本論文では，2 章で本研究の課題，および情報フローの制御を実現する関連技術と PADOC の比較について述べる．3 章で，PADOC の概要について，特に，データが持つセマンティクスに適応可能なアクセス制御手法について述べる．4 章では，PADOC のプロトタイプシステムの実装について述べ，5 章で PADOC を適用したアプリケーションの構築例を示す．

2 プライバシ保護に適した情報フロー制御の考察

2.1 本研究の課題と目標

経済協力開発機構（OECD）で定義されたプライバシ保護の原則 [3] では，個人情報などのプライバシを考慮すべきデータをその所有者の意思を反映して利用，管理することを義務づけている．すなわち，データ保護のためのアクセス制御の粒度は，データ提供者から提供されるおのおののデータを単位とし，また，データ提供者の意思に反するデータの利用を禁止するために，適切なアクセス制御を適用する必要がある．

一方，現在の計算機システムでは，ファイルを単位としてデータの移動や複製を行うことが多い．例えば，第 3 者にデータを提供する場合，フラッシュメモリなどの可搬性のある記憶媒体にファイルを複製したり，電子メールにファイルを添付するなどの手段で提供されることが多い．このような背景から，ファイルを保護の単位とする機密データの漏洩を防ぐ技術が多く提案され，製品化されている（例えば，[4, 5, 6] など）．また，*Salvia* もファイルを単位とするデータ保護を実現する OS である．*Salvia* は，ファイルごとにデータ保護ポリシを定義し，既存のアプリケーションを変更することなく，データ保護ポリシに基づくアクセス制御を適用することができる．

しかし，文献 [3] の要求を満たすためには，ファイル単位のデータ保護では不十分な場合がある．具体例を挙げると，複数の顧客の個人情報を 1 つのファイルで一元管理する場合を想定する．このとき，すべての顧客が個人情報の取扱いに関するポリシに同意し，そのポリシを共有できる場合は，ファイル単位のデータ保護手法を適用することにより，文献 [3] の要求を満たすことができる．しかし，顧客ごとに異なるポリシの適用を要求される場合，すべての顧客の要求を包括するポリシを記述することは困難である．特に，ファイル単位のデータ保護手法では，ファイルより粒度の細かいおのおののデータに対して，それを保護するためのポリシを適用することができない．

そこで，本研究では，(1) 伝搬範囲や保護強度が異なるおのおののデータに対して適切なアクセス制御を適用するデータ保護方式の実現，(2) アプリケーションに依存しないデータ保護方式の実現を目標とする．これらを実現することにより，保護を必要とするデータが複数の（異なる種類の）アプリケーションの間を伝搬する場合でも，データに関連づけられたポリシに基づき，適切なデータ保護が適用される．

2.2 情報フロー制御モデルに基づくデータ保護

情報フロー制御モデルは，データの秘匿性に着目したデータアクセス制御モデルである．情報フロー制御モデルの 1 つである Bell-LaPadula モデル [7] は，データの参照，更新，生成，削除，実行，追加などのデータに対する操作（アクセス）を実行する主体と対象にセキュリティクラスを与え，セキュリティクラス間の情報フローの許可・不許可を制御する．これにより，システム全体の情報フローの制御を実現する．Bell-LaPadula モデルは，TrustedBSD などの Trusted OS における強制アクセス制御を実現するモデルとして適用されている．Trusted OS では，アクセスの主体となるプロセスと，アクセスの対象となるファイル・ソケット・デバイスなどにセキュリティクラスを定義するためのセキュリティラベルを設定する．このため，アクセス制御の粒度の視点では，ファイルをアクセス制御の単位とするデータ保護を実現していると言える．

しかし，前節で述べたように，ファイルを単位とするアクセス制御では，ファイル中のおのおののデータに異なるアクセス制御ポリシを適用できない．このため，データごとに異なるアクセス制御を課すことができない．

2.3 Decentralized Label Model

分散ラベルモデル（Decentralized Label Model）[8] は，プライバシを考慮したデータ保護モデルである．特に，情報の公開や伝搬の制御を実現することを目的として，end-to-end のセキュリティポリシを適用するための手法を提案している．具体的には，Java を拡張した新しいプログラミング言語 Jif（Java Information Flow）を導入し，プログラムのコンパイル時と実行時の両方で実行する情報フロー

解析とアクセス制御により，プログラム中の変数に代入された値を保護する．保護対象のデータが代入される変数には，その変数に対するアクセス制御ポリシを定義するラベルが付与される．これらにより，変数から変数への値のコピーや通信チャネルへのデータの送信が，変数に付与されたラベルによって制御される．

しかし，Jif で扱う変数のアクセス制御ポリシはプログラマによって記述される．また，おのおのの変数に付与されるラベルにポリシが定義される．これらにより，データ自身が持つデータ保護の強度（データの伝搬範囲やアクセスの可否の制御）と異なるポリシがラベルによって強制される可能性がある．

2.4 サンドボックス技術を応用したデータ保護

信頼性が高くないプログラムを安全に実行するための手法として，サンドボックスがある．サンドボックスで実行されるプログラム（プロセス）は，他のプロセスから隔離された環境で実行され，計算機資源に対するアクセスも制限される．この特徴を応用することにより，データの伝搬範囲を制御することが可能となる．

SoftwarePot [9] は，ユーザが実行するプロセスを仮想的なファイルシステム内で実行するサンドボックスを実現する．プロセスは，仮想的なファイルシステムを介して実計算機上に存在するファイルやネットワーク資源にアクセスする．このとき，ファイルやネットワーク資源へのアクセスの可否は，仮想的なファイルシステムを構築する際に定義されるセキュリティポリシに基づいて決定される．

ただし，SoftwarePot では，複数のユーザによってセキュリティポリシが作成・修正される場合を想定している．例えば，仮想的なファイルシステムの構築者とその利用者が異なる場合に，利用者の計算機環境にあわせてセキュリティポリシの修正が行われることを容認している．このことは，データの提供者が意図していないポリシに書き換えられてしまう可能性があることを意味する．

そこで，サンドボックスによってプロセスが利用可能な計算機資源を制限し，保護すべきデータが出力先となりうる計算機資源に対するデータの出力を防ぐ．その際，プロセスが持つメモリ空間に読み出される（入力される）データの制御を本論文で提案する手法で実現する．すなわち，プロセスメモリ空間へ読み出されたデータに設定されたポリシに基づき，当該データの出力を制御する．これにより，プロセスメモリ空間に伝搬範囲を制限すべきデータが読み出された場合に，その伝搬範囲に応じてデータの出力を制限する．

2.5 暗号化技術によるファイル保護

今日の企業活動において，重要な情報を含むようなファイルをすべて持出し禁止にすることは，日常の業務などに支障をきたす場合が考えられるため，そのようなポリシを強制することは困難である．このため，保護すべきデータを含むファイルを安全に持ち出すことを可能にするシステムが必要となる．ここで，安全とは，文献 [10] で述べられているように，正当な権限を持つユーザのみが（ア）ファイルを移動させることができること，（イ）ファイルを読み出すことができること，および，（ウ）ファイルが第 3 者によって勝手に 2 次配布されないことを意味する．

これらの機能要件を満たす既存技術の 1 つに，公開鍵暗号がある．ファイルを他者に提供する際に，そのファイルの受信者の公開鍵を用いてファイルを暗号化した後に送信する．したがって，公開鍵を用いて暗号化されたファイルは，それを復号するための鍵を持つユーザのみがファイルを読み出すことができる．すなわち，鍵を持つユーザをファイルを閲覧する権限を持つ正当なユーザとみなすことができる．特に，文献 [10] では，公開鍵の配布に PKI（公開鍵基盤 : Public Key Infrastructure）を適用し，さらにファイルの 2 次配布を制御する手法を提案している．

文献 [10] の提案システムは，ファイルの伝搬範囲を制御するためのソフトウェア基盤として位置づけられる．ただし，ファイル単位の保護を実現するものであるため，2.1 節で示した要件を満たさない．そこで，文献 [10] の提案システムによってファイルの伝搬範囲を制御し，さらに本論文で提案する手法と組み合わせることにより，ファイルが複数の計算機を伝搬する場合でもファイルより細粒度のデータ保護が実現できると考えられる．

3 Privacy-Aware Data Object Container の概要

3.1 データのセマンティクスに適応可能なデータ保護

ファイルに保存されるおのおののデータをアクセス制御の単位とする場合，それらを保護方法を定義する保護ポリシをデータごとに設定可能とする必要がある．また，おのおののデータに対するアクセスを他のデータへのアクセスと区別し，そのアクセスの可否を制御できるデータ管理方式を実現する必要がある．

従来の OS が提供するファイルアクセスのためのインタフェースである VFS （Virtual File System）において，読出しや書込みなどのファイルアクセスは，1 バイト単位のデータブロックの集合（以下，バイト列と記す）を単位として実行される．これは，ファイルが VFS によってバイト列に抽象化されていることによる．しかし，この抽象化により，例えば読出しアクセスによって読み出されたデータが保護すべきデータか否かを OS が判定することはできない．これは，抽象化によってデータが持つセマンティクスが失われることに起因する．

ここで，本論文では，セマンティクスを入力データが個人情報などの保護すべきデータか否か，出力データがデータ漏洩の原因となりうる計算機資源に対する書込みか否か，データの演算によってデータの内容が変更されるか否かを意味するものと定義する．セマンティクスが保持

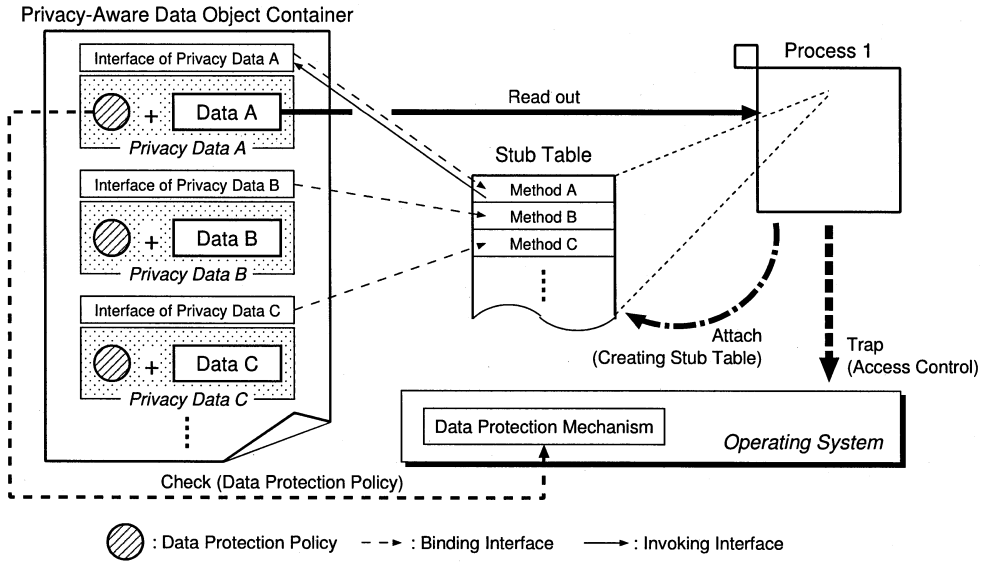

図 1　PADOC におけるデータアクセスの概要

されている場合，データの入力，演算，出力といったデータのライフサイクルにおいて，当該データが保護対象か否か，データの内容が変更されるか否か，データの出力が許可されているか否かを OS が判定することができる．しかし，先に述べたように，VFS が提供する汎用的なインタフェースを利用したアクセスではセマンティクスが失われるため，データフローが情報漏洩の要因となるか否かを判定することができない．

3.2　アプリケーション固有の擬似データストレージ

前節の議論から，データアクセス用のインタフェースの汎用性とデータが持つセマンティクスは，トレードオフの関係にあると言える．本研究の目的は，ファイルより細粒度のデータアクセス制御手法を実現することにある．このため，データが持つセマンティクスが維持できない場合，おのおののデータに対して個別のデータアクセス制御を課すことが困難となる．

データのセマンティクスを維持したデータアクセスの実現は，アプリケーションの実装方法とアプリケーションがアクセスするデータ（ファイル）の内部構造に依存する．前者は，アプリケーションを実行するプロセスのメモリ空間に読み出したデータをどのように処理するかは，アプリケーション開発者によって決定される．後者は，アプリケーションの実装にあわせて，データをファイルに格納する際のデータ構造を決定することが多い．このため，データアクセスに関する処理は，アプリケーションで実装されるべき機能と言える．ただし，これまでに述べたように，VFS が提供するインタフェースはデータが持つセマンティクスを保持しない．また，標準ライブラリが提供するデータアクセス用の API（Application Programing Interface）も，VFS が提供するインタフェースを用いて実現されているため，セマンティクスを保持しない．

そこで，PADOC では，データごとにアクセス用のインタフェースを定義し，データと当該データにアクセスするためのインタフェースを 1 対 1 に対応させる．これにより，PADOC を適用したアプリケーションでは，PADOC をアプリケーション固有の擬似データストレージとみなすことができる．すなわち，アプリケーション開発者は，データが持つセマンティクスを考慮した上で，アプリケーションが利用するおのおののデータが必要とする保護強度に応じたデータアクセス用のインタフェースを実装することができる．アプリケーションの実行環境であるプロセスは，インタフェースを介してデータへのアクセスを要求する．カーネルは，インタフェースの呼出しをトラップし，データへのアクセスの可否をデータ保護ポリシに基づいて制御する（図 1 参照）．

3.3　階層的なプロテクションドメイン

Salvia では，プロセスはファイルごとに定義されたデータ保護ポリシに従い，利用可能な計算機資源や実行可能なシステムコールが制限される．この制約は，*Salvia* のカーネルによってプロセスに強制的に課される．すなわち，プロセスに課される制約（プロテクションドメイン）は，データ保護ポリシが定義されているファイルによって

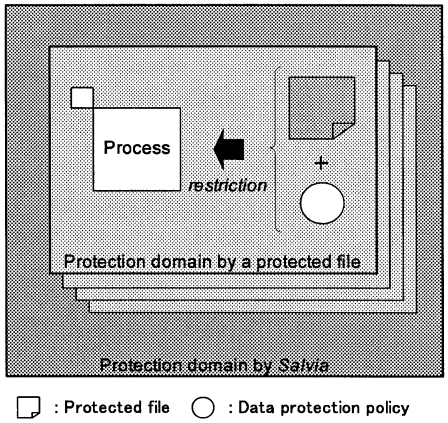

図2　ファイル単位のプロテクションドメイン

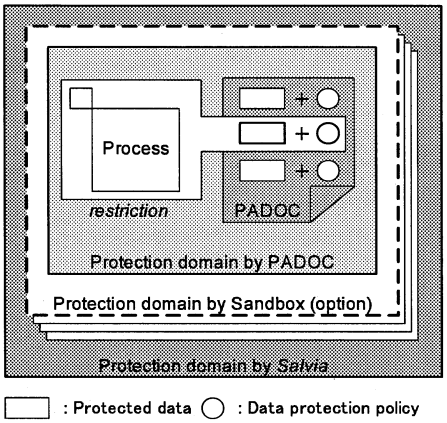

図3　階層的なプロテクションドメイン

決定される．また，カーネルは，その制約が課されることを保証するためのメカニズムを提供する（図2参照）．

PADOCでは，プロセスに課される制約は，プロセスがアクセスするおのおののデータに定義されたデータ保護ポリシによって決定される．従来の*Salvia*の場合，プロテクションドメインは，プロセスが別の保護ファイルにアクセスするまで固定される．一方，PADOCの場合，おのおののデータに対するアクセスが発生するごとに，プロテクションドメインの遷移の有無が検査される．このとき，従来の*Salvia*と同様にFail-safeの原則に基づき，もっとも制約が厳しいデータ保護ポリシに従ってプロテクションドメインを決定する．

PADOCに実装されるデータアクセス用のインタフェースは，データ保護ポリシに従ってアクセスの可否を制御するための機能を持たない．前節で述べたように，PADOCではデータへのアクセス要求はカーネルによってトラップされ，アクセスの可否が制御される．すなわち，PADOCを適用したデータ保護方式を実現する場合も，*Salvia*のようなOSのカーネルによってデータ保護ポリシに基づくアクセス制御が保証されるメカニズムが必要となる．

さらに，PADOCにおいて，サンドボックス技術を適用することにより，プロテクションドメインを階層化することが考えられる（図3参照）．サンドボックスによりプロセスが利用可能な計算機資源を予め制限し，その制限の下でデータ保護ポリシに従ってアクセスの可否を制御する．これにより，PADOCを適用したアプリケーションに悪意のある（データ漏洩を発生させる）実行コードや不具合が含まれていた場合でも，サンドボックス環境の外部にデータが漏洩することを防止できる．

3.4　PADOCによるデータアクセスの制御

例えば，図1において，Data A，B，Cはデータ提供者から提供されたデータであり，おのおのに対して（データ提供者が定義した）データ保護ポリシが設定されているものとする．このとき，Data A，B，Cのデータ保護ポリシは，次のように設定されていると仮定する．

- Data Aのデータ保護ポリシ
  すべての計算機資源に対する出力を**許可**する．
- Data Bのデータ保護ポリシ
  すべての計算機資源に対する出力を**禁止**する．
- Data Cのデータ保護ポリシ
  ファイル，ネットワークへの出力を**禁止**する．

ファイルをアクセス制御の単位とするデータ保護方式の場合，2.1節で述べたように，図1のData A，B，Cが保存されたファイルにアクセスしたプロセスに課すデータ保護ポリシの記述は困難である．例えば，保護ファイルにアクセスしたプロセスに対して他のファイルへの書込みアクセスを許可する場合，このポリシはData B，Cの提供者の意図に反する．また，プロセスに対してネットワークへの出力を禁止する場合，Data Aの提供者の意図に反する．

PADOCでは，おのおののデータに設定されたデータ保護ポリシに基づくアクセス制御を実現するために，データの読出し要求を契機としてプロセスのアクセス制御の状態を遷移させる（図4参照）．この状態遷移は，OS（カーネル）で管理する．特に，**CONTROLLED状態**に遷移したプロセスは，カーネルによってデータ保護ポリシに基づくアクセス制御が適用される．これにより，データのライフサイクルの途中で，データがその提供者が意図していない利用目的に利用されたり，意図していない提供範囲に伝搬することを防ぐ．

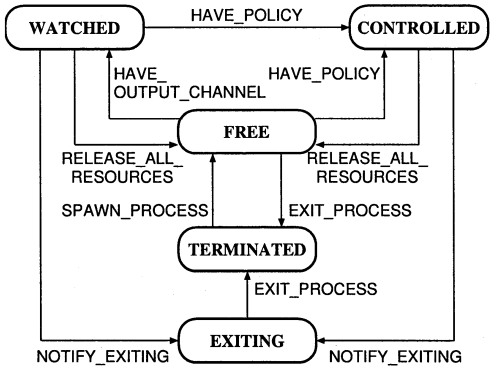

図 4　プロセスのアクセス制御の状態遷移

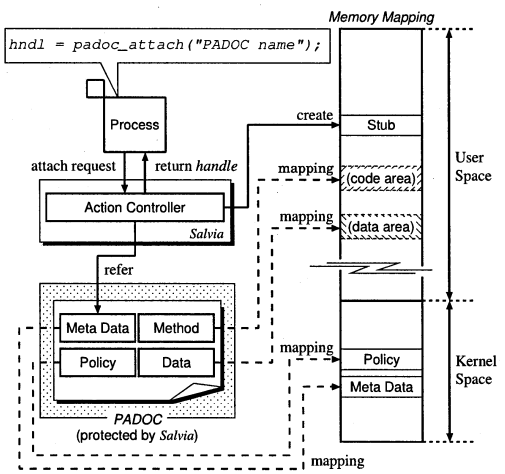

図 5　PADOC をアタッチした際のプロセスのメモリマップ

4　プロトタイプシステムの実装

4.1　データの管理方式

PADOC は，本論文で提案するファイルより細粒度のデータアクセス制御を実現するためのデータ管理方式の呼称である．PADOC は，プロセスによってアクセスされる対象となるデータ，データにアクセスするインタフェースとなるメソッド，データフローに課すアクセス制御を定義するデータ保護ポリシ，PADOC の管理情報を保持するメタデータから構成される．プロトタイプシステムでは，実装を簡易にするために，PADOC を通常のファイルと同様に，既存のファイルシステム上のファイルとして管理する．PADOC を構成する各要素の概要を以下に示す．

データ　保護対象となるデータである．また，UNIX 系の OS におけるファイルと同様に，2 次記憶装置上ではバイト列のデータブロックの集合として保存される．

メソッド　データへのアクセスを実行するためのプログラム（実行可能コード）である．プロセスは，メソッドをインタフェースとしてデータへのアクセスを要求する．メソッドは，データと同様に，2 次記憶装置上ではバイト列のデータブロックの集合として保存される．メソッドの呼出し方法は 4.4 節で述べる．

データ保護ポリシ　メソッドの呼出し条件が記述されている．*Salvia* におけるファイル単位のデータ保護と同様に，*Salvia* で取得可能なコンテキストをメソッド呼出しの制御条件に記述することができる．このポリシは，Action Controller によって解釈される．また，データと同様に，2 次記憶装置上ではバイト列のデータブロックの集合として保存される．

メタデータ　PADOC の管理情報や属性を示すデータである．PADOC の管理情報として，データ，メソッド，データ保護ポリシのデータサイズとそれらを保存するために消費している 2 次記憶装置のブロック数，ブロック番号がある．また，PADOC の属性として，通常のファイルにおける所有者やグループを示す情報，最終更新時刻などのタイムスタンプがある．すなわち，メタデータは，既存のファイルシステムにおける i ノードに相当する情報を管理する．

プロトタイプシステムでは，PADOC をファイルとして管理するため，PADOC はファイル名を持つことができる．このため，システムコールなどのファイルアクセスのための既存のインタフェースを用いたアクセスが可能となる．しかし，PADOC が管理するデータへのアクセスは，PADOC が公開するメソッドを介してのみ実行されるように制御される必要がある．すなわち，システムコールなどによるアクセスを禁止する必要がある．プロトタイプシステムでは，open システムコールに変更を加えて，PADOC への直接アクセスを禁止する．

4.2　プロセス空間へのアタッチ処理

プロセスが PADOC にアクセスするためには，通常ファイルの場合と同様に，アクセス可能な計算機資源として当該プロセスと関連づける必要がある．PADOC では，この処理をアタッチ処理と呼ぶ．アタッチ処理を実行した際のプロセスのメモリマップを図 5 に示す．アタッチ処理では，プロセスが PADOC にアクセスするために必要となるメモリの確保と，当該メモリ空間へのデータの読出しを実行する．具体的には，データへのアクセスを実現するメソッドの実体である実行可能プログラムと，アクセス対象のデータをマップするためのメモリ領域の確保と初期化処理を実行する．さらに，メソッドを呼び出すためのインタ

フェースとなるスタブを生成する．これらは，プロセスの仮想アドレス空間のうち，ユーザ空間におのおのの領域が確保される．

カーネルが上記の処理を実行する場合，マップすべきデータとその保護ポリシのデータサイズ，定義されているメソッドの個数，個々のメソッドのコードサイズといった情報が必要となる．カーネルは，これらの情報をメタデータから取得し，メモリを確保する際に利用する．また，データ保護ポリシは，メソッドの呼出しの可否を制御するために，カーネルが必要とするものである．これらの理由により，メタデータとデータ保護ポリシはカーネル空間にマップする．

データをマップするメモリ領域は，プロセスに直接アクセスされないように制御する必要がある．これを実現するために，細粒度保護ドメイン [11] で提案されている Intel x86 アーキテクチャのプロセッサのセグメンテーション機構を用いて制御する．すなわち，通常とは異なるデータセグメントに，データをマップするメモリ領域を配置する．セグメントの切替えは，メソッド呼出しの際にカーネルによって制御される．

アタッチ処理を正常に終えた場合，カーネルはプロセスにアタッチされた PADOC に対応するハンドルを返す．このハンドルは，open システムコールにおけるファイルディスクリプタと同様に，非負の整数値である．プロセスは，これ以降の処理において，カーネルから渡されたハンドルを用いて PADOC のメソッド呼出し処理を発行する．

4.3 プロセスとデータアクセス用のメソッドの関連づけ

メソッド呼出しは，プログラムの実行中に共有ライブラリ関数を呼び出す手法と同様に，メソッドの関連づけを行う必要がある．この処理は，PADOC を適用するアプケーションに提供する共有ライブラリ libpadoc.so が提供する API の 1 つである padoc_bind_method 関数を呼び出すことにより実現する．

PADOC では，データへのアクセスを実現するメソッドは，位置独立コード（PIC: Position Independent Code）としてコンパイルされた実行コードと，メソッドの関連づけに必要となるメソッドのシンボルの集合として構成される．各シンボルに対応するメソッドの実行コードのサイズは，前節で述べたようにメタデータに保存されている．padoc_bind_method 関数は，PADOC のハンドルと関連づけるシンボルを引数に持つ．padoc_bind_method 関数が実行されると，関数の内部でシステムコールが発行され，カーネルに制御が移行する．カーネルは，メタデータを参照し，プロセスが関連づけを要求したシンボルに対応する実行コードを特定する．ただし，実行コードのロード処理は，実行コードをマップするためのメモリの確保を除き，メソッド呼出しが発生するまで遅延させる．

メソッドの関連づけ処理が正常に終了すると，プロセスはスタブを介してメソッドの呼出し要求を発行できるようになる．スタブでは，メソッド呼出しの可否を判定するためのシステムコールを発行し，カーネルに制御を移行する．このシステムコールは，*Salvia* の Action Controller で処理する．

4.4 メソッド呼出しの制御方式

前節で述べたように，メソッド呼出しは Action Controller によって制御される．メソッド呼出しの処理手順を以下に示す．

1. アクセス対象のデータを特定する．
   Action Controller は，実行が要求されたメソッドの種類から，どのデータに対するアクセス要求なのかを特定する．メソッド呼出しが許可された場合，メソッドの実行コードをプロセスのメモリ空間にロードし，実行を開始する．
2. データ保護ポリシとコンテキストを比較し，メソッドの実行の可否を判定する．
   データ保護ポリシに記述されたメソッドの実行を制御するための条件文を検査する．*Salvia* におけるシステムコールの制御と同様に，コンテキストを用いてメソッドの実行の可否を制御する条件文を記述することができる．
3. 実行コードをプロセスのメモリ空間にロードする．
   メソッドが 1 度も実行されていない場合，メソッドの実行コードをロードする．メソッドを 1 度以上実行したことがある場合は，実行コードをロードしたページフレームの存在フラグ（Present bit）に 1 を設定する．
4. データセグメントレジスタを PADOC のデータセグメントに切り替える．
   メソッドの実行中に PADOC 内のデータにアクセスできるようにするため，プロセッサのデータセグメントレジスタに，データがマップされたメモリ領域のセグメントセレクタ値を設定する．
5. メソッドを実行する．
   CPU の走行モードをユーザモードに戻し，実行コードをロードしたアドレスからプログラムの実行を開始する．

メソッドの実行が完了すると，プロセスは，呼び出したメソッドによってアクセス可能なデータの取得を完了した状態となる．ただし，データセグメントレジスタが変更されているため，プロセスがそれまでに利用していたメモリ領域（ヒープなど）にアクセスできない．このため，スタブにおいてメソッド呼出しの後処理を要求するシステムコールを発行する．Action Controller は，このシステムコールをトリガとして次の処理を実行する．

1. データセグメントレジスタを元のデータセグメントに設定する.
   データセグメントレジスタをメソッドの実行前のセレクタ値に戻す. これにより, プロセスに対してPADOC内のデータに直接アクセスさせないこと, およびプロセスが所有しているヒープ・大域変数へのアクセスを可能とする.
2. 実行コードをロードしたページフレームを無効化する. Action Controllerによるメソッド呼出しの制御を回避されないようにするため, 実行コードをロードしたページフレームの存在フラグに0を設定する. これにより, プロセスが直接メソッドの呼出しを試みた場合でも, ページフォルト例外が発生し, メソッドを実行することができない.

5 PADOCを適用したアプリケーションの構築

アプリケーションの構築例として, 個人の氏名 (苗字と名前)・性別・年齢・生年月日・住所・メールアドレス・電話番号をXMLで書式化し, 管理するプログラムを考える. データを表示する機能は, 一般的なアプリケーションの作成方法と同様に, GUIアプリケーションのためのツールキットが利用できる. 今回は, GTK+を用いてC言語によるデータ表示プログラムを作成した.

アクセス制御が必要となるおのおののデータへのアクセスを実現するメソッドとして, `getFirstName()`, `getFamilyName()`, `getGender()`, `getAge()`, `getBirthday()`, `getPostalAddress()`, `getEmailAddress()`, `getTelephoneNumber()` をC言語で実装した. これらのメソッドは, 1名分の情報を画面に表示する要求が発生する際に呼び出される. メソッドの実行の可否は, データ保護ポリシに従い, *Salvia* によって制御される. また, 今回は, データ保護ポリシとしてデータの読出しの許可・不許可のみを設定した. すなわち,「読出し許可」という設定がされたデータのみが, 上記のメソッドによってデータを取得することができる. 例えば, 苗字・名前・性別のみが読出し許可となっているユーザの情報を表示する場合, 苗字・名前・性別以外のすべての表示項目は空白となる.

6 おわりに

本論文では, 秘匿性を考慮した細粒度データ管理手法PADOCについて述べた. 特に, データが持つセマンティクスに適応可能なアクセス制御を実現するために, おのおののデータに対してデータ保護ポリシを設定し, データへのアクセス要求が発生した際にその可否をOS (*Salvia*) によって制御する手法を提案した. 今後は, 提案手法によるアクセス制御にかかる性能評価を行う予定である.

参考文献

[1] 鈴来和久, 一柳淑美, 毛利公一, 大久保英嗣: Privacy-Aware OS *Salvia* におけるデータアクセス時のコンテキストに基づく適応的データ保護方式, 情報処理学会論文誌: コンピューティングシステム, Vol. 47, No. SIG 3 (ACS13), pp. 1 – 15 (2006).

[2] 鈴来和久, 毛利公一, 大久保英嗣: データの拡散防止を実現するコンテキスト適応型ソフトウェア基盤, 情報処理学会研究報告 2004-OS-95, Vol. 2004, No. 17, pp. 57 – 64 (2004).

[3] Organisation for Economic Co-operation and Development: OECD Guidelines on the Protection of Privacy and Transborder Flows of Personal Data, *http://www.oecd.org/document/18/0,2340,en_2649_201185_1815186_1_1_1_1,00.html* (2004).

[4] 日立ソフト: 情報漏洩防止ソリューション「秘文」, *http://hitachisoft.co.jp/hibun/*

[5] NEC: InfoCage, *http://www.nec.co.jp/cced/infocage/*

[6] 富士通: Systemwalker Desktop, *http://systemwalker.fujitsu.com/jp/solution/solution_31.html*

[7] Bell, D. E. and LaPadula, L. J.: Secure Computer System: Unified Exposition and Multics Interpretation, MTR-2997, MITRE Corporation; ESD-TR-75-306 (1976).

[8] Myers, A. C. and Liskov, B.: Protecting Privacy using the Decentralized Label Model, *ACM Transactions on Software Engineering and Methodology* (*TOSEM*), Vol. 9, No. 4, pp. 410 – 442 (2000).

[9] 大山恵弘, 神田勝規, 加藤和彦: 安全なソフトウェア実行システムSoftwarePotの設計と実装, コンピュータソフトウェア, Vol. 19, No. 6, pp. 2 – 12 (2002).

[10] 齋藤彰一: プロセスの通信制御と公開鍵暗号によるファイルの移動制御方式の提案, 情報処理学会研究報告 2006-OS-103, Vol. 2006, No. 86, pp. 79 – 86 (2006).

[11] 品川高廣, 河野健二, 高橋雅彦, 益田隆司: 拡張コンポーネントのためのカーネルによる細粒度保護ドメインの実現, 情報処理学会論文誌, Vol. 40, No. 6, pp. 2596 – 2606 (1999).