SCIS 2002 The 2002 Symposium on Cryptography and Information Security
Shirahama,Japan, Jan.29-Feb.1,2002
The Institute of Electronics,
Information and Communication Engineers

多機能セキュアデータベースのアクセス制御ポリシー設定問題

Access Control Policy Information Assignment in Database with Multiple Security Functions

安東 学* 松浦 幹太* 工藤 道治† 馬場 章*
Manabu ANDO Kanta MATSUURA Michiharu KUDO Akira BABA

あらまし 著者らは多様なセキュリティ操作に適したオブジェクト指向データベースの構造について考察し，そのプロトタイプを提示した．本稿では，このような構造を持つデータベースを多機能セキュアデータベースと呼ぶ．これは，データの形態に応じたセキュリティ操作を行うために，クラスの内容を最小限に抑え，多数のクラスを定義するものである．また，セキュリティ操作のためのルールを一つのクラスで定義する．さらに，セキュリティ操作を数種類組み合わせることによってセキュリティ強度を向上させる．著者らは多機能セキュアデータベースを実現させる一手法として，必須処理付きアクセス制御方式の適用を考えている．これにより，セキュリティ操作の組み合わせが容易になる．しかしながら，従来の実現方法では，ポリシー設定によって様々な問題が発生すると予想される．本稿では，予想される問題点を整理し，解決方法の一部を提示する．さらに，定性的評価を行い，提示する解決方法の有用性を検討する．

キーワード 多機能セキュアデータベース，必須処理付きアクセス制御方式，アクセス制御ポリシー

1 はじめに

データベースのコンテンツ形態が多様化し，従来のデータベース構造では安全な保存・運用が難しくなっている．また，アプリケーションによっては著作権などの問題が生じることもあり，権利関連のトラブルもまた増加・多様化する傾向にある．著者らは，[1] において多様なセキュリティ操作に適したオブジェクト指向データベースの構造について考察した．本稿ではこれを多機能セキュアデータベースと呼ぶ．

このデータベースの重要な特徴は，非常に細かいクラス定義である．柔軟なセキュリティ操作を実現するには，クラスの内容を最小限にする必要がある．クラス数は増加するが，セキュリティ管理面から考えると，この構造が適しているという結論を得た．また，各種セキュリティ操作はクラスのメソッドとして定義し，セキュリティ操作実行に関するルールは独立したクラスに記述する ([1])．

著者らは [1] において，多機能セキュアデータベースには数種類のセキュリティ操作の関連付けが必要である，という見解を示した．例えば，文書型のデータであれば，秘匿性と真正性の保証が必要となる．これは異なるセキュリティ操作によって実現されるが，それぞれ独立に保証するのでは十分なセキュリティ操作とはいえない．セキュリティ操作を関連付けることにより，要求を同時に満たす必要がある．

多機能セキュアデータベースにおいて，セキュリティ操作の関連付けを実現させる一手法として，必須処理付きアクセス制御方式 (PBAC, Provision-based Access Control)([2][3][4][5]) の適用を考えている．PBAC 方式は，アクセスの許可/拒否に加えて必須処理を指定するためのモデルであり，アクセス制御規則とセキュリティ処理とを結びつける仕組みをもつ．PBAC 方式を利用することにより，著者らが主張するセキュリティ操作の関連付けも容易に実現できる．

しかし，多機能セキュアデータベースに従来の PBAC 方式を適用すると，記述の自由度が大きいために，様々な問題が発生する．特に，アクセス制御規則 (ACPI, Access Control Policy Information) の設定によっては，ユーザビリティや秘匿性，システム稼動性などの問題が発生すると予想される．

本稿では，アクセス制御ポリシー設定によって生じる問題点を分類し，解決方法の一部を提示する．さらに，それぞれの解決方法について定性的評価を行う．

* 東京大学大学院情報学環・学際情報学府，〒 113-0033 東京都文京区本郷 7-3-1，Interfaculty Initiative in Information Studies, Graduate School of Interdisciplinary Information Studies, Univ. of Tokyo, 7-3-1, Hongo, Bunkyo-ku, Tokyo, 113-0033, Japan
† 日本アイ・ビー・エム 東京基礎研究所，〒 242-8502 神奈川県大和市下鶴間 1624-14，Tokyo Research Laboratory, IBM Japan Ltd., 1623-14, ShimoTsuruma, Yamato-shi, Kanagawa, 242-8502, Japan

2　多機能セキュアデータベースの概要

従来のデータベースセキュリティの研究では，秘匿性を重視したアクセス制御関連の研究が多数を占めていた ([6][7][8])．しかし，多様な形態をもつデータベースを保護するためには，単純なアクセス制御だけでは不十分である．著者らは多様なセキュリティ操作に適したデータベース構造を考察し，プロトタイプを提示した ([1])．本稿ではこのようなデータベースを多機能セキュアデータベースと呼ぶ．

2.1　多機能セキュアデータベースの構造

多機能セキュアデータベースの具体例（知識共有データベース）を図 1 に示す．図において，Project_Information, Text_Data, TextAuthenticated_Data, Image_Data, ImageAuthenticated_Data, Status はクラスである．

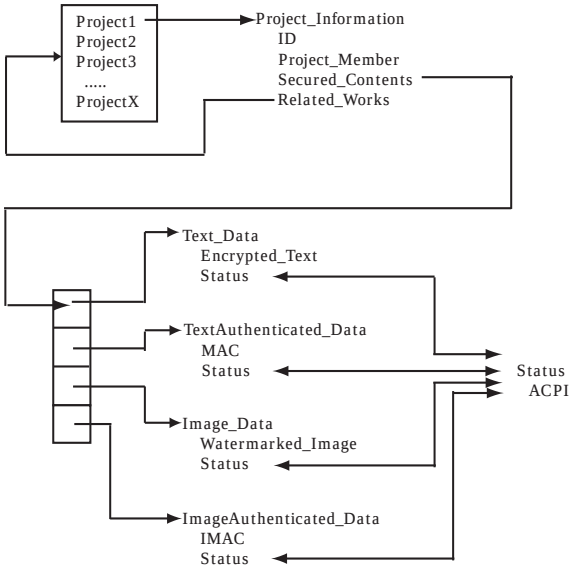

図 1: 多機能セキュアデータベース

2.1.1　構成要素

多機能セキュアデータベースの構成要素を以下に挙げる．

- 主体: 客体に対して操作を行うエンティティを主体と呼ぶ．
- 客体: 主体に操作される要素を客体と呼ぶ．ここでは，各クラスやクラス内の属性が客体となる．
- 関連: クラス間の結びつきを関連という．図においては矢印で示されている．
- メソッド: 通常のメソッド (read, write など) の他に，セキュリティ操作を定義する．例えば，encrypt などをメソッドとして定義する．

2.1.2　各クラスの詳細

図 1 の例を利用し，多機能セキュアデータベースの詳細を述べる．

- Project_Information クラス: 4 種類の属性をもつ．ここでは，多機能セキュアデータベースに特徴的な Secured_Contents について述べる．
  - Secured_Contents: 各種セキュリティ操作を実行したデータを格納するクラスへの参照をもつ．ここでは，Text 型と Image 型のデータ形態を仮定している．
- Text_Data クラス，Image_Data クラス: それぞれのデータ形態に適したセキュリティ操作を実行し，独立したクラスに定義する．
- TextAuthenticated_Data クラス，ImageAuthenticated_Data クラス: 各データの真正性を保証するためのデータを格納するクラスである．図 1 では，MAC と表記されているが，これは特定の認証子生成方法に限定するものではない．データの真正性を保証する意味を持つ認証子一般のことを示している．
- Status クラス: セキュリティ操作に必要となるルールを格納する．メソッドによってセキュリティ操作が実行されるとき，このクラスが必ず参照される．

2.2　多機能セキュアデータベースの特徴

多機能セキュアデータベースの特徴を以下に列挙する．

- セキュリティ操作ごとにクラスを定義する．従って，クラスの内容は最小限のものとなる．クラス数は増加するが，柔軟なセキュリティ操作を実行することができる．さらに，主体に与える権限を最小限度に設定することが容易になる．更新作業について考慮すると，主体の権限はでき得るかぎり小さいほうが良いと考えられるため，このような構造を持つ．
- Status クラスは全ての ACPI を格納し，一元的に管理する．大規模なデータベースになれば ACPI の設定はより複雑になり，不適切な設定が増加すると予想される．Status クラスのように一元管理にしておけば，そのような設定を早期に発見することができる．
- 数種類のセキュリティ操作を関連付ける必要がある．例えば，図 1 において，Text_Data クラスの内容と TextAuthenticated_Data クラスの内容は連動していなければならない．そのために，セキュリティ操作を関連付ける．

- セキュリティ操作は各クラスのメソッドによって定義される．メソッド実行時には必ず Status クラスを参照し，ACPI の記述に従った操作を行わなければならない．

2.3 セキュリティ操作の関連付け

多機能セキュアデータベースにおいて，データを安全に管理・運用するにはセキュリティ操作を関連付ける必要がある．例えば，Text_Data クラスの Encrypted_Text を更新するとき，更新データの暗号化処理に加えて認証子の更新も実行しなければならない．両処理により，秘匿性と真正性が保証される．これらは個別に保証されても不十分である．両者が同時に保証される必要がある．そこで，セキュリティ操作の関連付けが必要となる．著者らは，セキュリティ操作の関連付けを実現する一手法として，PBAC 方式の適用を考えている．

3 PBAC アクセス制御方式の概要

PBAC アクセス制御方式は，従来のアクセス制御方式における出力値である許可/拒否のニ値に加え，必須処理を出力し，より柔軟なアクセス制御を可能にする方式である．詳細については [2][4] などを参照されたい．本稿では，PBAC 方式の概要について述べる ([3][4]).

3.1 PBAC 方式のプリミティブとモデル

PBAC 方式で使用されるプリミティブは，アクセス要求者である主体 (SBJ)，対象となる客体 (OBJ)，アクセスモード (ACT)，アクセス許可/拒否を示す符号 (PRM)，必須処理 (OPS)，アクセス要求時のコンテキスト (CXT) である．このうち，CXT はオプションである．以下に，アクセス制御規則 (ACPI, Access Control Policy Information)，アクセス要求 (AR, Access Request)，アクセス決定 (AD, Access Decision) の構成を示す．

- ACPI は (OBJ, SBJ, ACT, PRM, OPS, CXT, SBJ2) の 7 つ組で構成される．PRM が grant を示すとき，SBJ は OBJ に対し CXT 下で ACT を行うことができる．ただし，OPS を全て実行しなければならない．SBJ2 は PRM や OPS の設定者とする．なお，本稿では簡素化のために CXT と SBJ2 を略した 5 つ組で記述する．
- AR は (OBJ, SBJ, ACT, CXT) の 4 つ組で構成される．SBJ が OBJ に対し，CXT 下で ACT を行いたいという要求である．なお，本稿では CXT を略した 3 つ組で記述する．
- AD は (OBJ, SBJ, ACT, PRM, OPS, CXT) の 6 つ組で構成される．PRM が grant のとき，SBJ が OBJ に対し CXT 下で ACT を実行できるが，OPS で示された必須処理を全て行わなければならない．PRM が deny のとき，アクセスはできないが，OPS で示された必須処理は実行しなければならない．なお，本稿では CXT を略した 5 つ組で記述する．

モデルの構成は，AR が発生したとき，システムは ACPI に基づいた評価を行い，AD を出力する．AD には，アクセスの許可/拒否と必須処理が含まれる．PRM が許可を示し，かつ出力された必須処理 OPS を実行すればアクセスが許可される．PRM が拒否を示した場合も，出力された必須処理 OPS を実行しなければならない．

3.2 PBAC 方式の関数

必須処理を導き出すための関数を以下に列挙する．なお，ここに示されている関数は本稿と関係のあるもののみであって，PBAC 方式で使用される全ての関数ではない．

- *eval_pvn*: AR を入力し、評価後 ACPI に基づいて OPS を出力する。
- *eval_prm*: AR を入力し、PRM を出力する。
- *provisions*: AR を入力し、AD の (OBJ, SBJ, ACT, CXT) を出力する。
$state\Big(\{\bigcup_{\{eval\_pvn(AR)\}} provision(eval\_pvn(AR))\}\Big)$
- *provision*:AR を入力し、OPS を出力する。

$$\begin{cases} \phi : if\ AR = \phi \\ err(AR) : if\ AR \neq \phi\ \wedge\ prov\_prm(AR) = deny \\ \{\bigcup_{\{eval\_pvn(AR)\}} provision(eval\_pvn(AR)), AR\} : \\ \quad if\ AR \neq \phi \wedge prov\_prm(AR) = grant \end{cases}$$

- *prov_prm*: AR を入力し、PRM を出力する。

$$\begin{cases} grant : if\ eval\_prm(AR) = grant \wedge \\ \quad (eval\_pvn(AR) = \phi \vee \\ \quad (\forall_{eval\_pvn(AR)} prov\_prm(eval\_pvn(AR)) = grant)) \\ deny : if\ eval\_prm(q) = deny \vee \\ \quad \exists_{eval\_pvn(q)} prov\_prm(eval\_pvn(q)) = deny \end{cases}$$

3.3 具体的動作例

多機能セキュアデータベースに PBAC 方式を適用したときの具体的動作例を示す．PBAC 方式の適用により，セキュリティ操作の関連付けが容易になることがわかる．さらに，主体に付与する権限を詳細に設定できることもわかる．なお，$Px(x = 1, 2, 3)$ は ACPI を示し，$R1$ は AR を示す．主体 *Alice* はロール $S1$ を有する．図 1 の例において，以下のような ACPI が設定されている．

$P1(Encrypted\_Text, S1, write, grant,$
$\quad (Encrypted\_Text, S1, encrypt),$
$\quad (MAC_{TextAuthenticated\_Data}, S1, write))$
$P2(Encrypted\_Text, S1, encrypt, grant)$
$P3(MAC_{TextAuthenticated\_Data}, S1, write, grant)$

このACPI設定では，Encrypted_Textへの書き込み時に，データの暗号化と認証子の更新を必須処理として規定している ($P1$)．更に，メソッドによって実行されるデータの暗号化と認証子の更新に関する権限を設定する ($P2, P3$)．ここで，$R1(Encrypted\_Text, Alice, write)$ が発生する．このとき，PBAC方式のアルゴリズムは以下のように動作する．

1. $eval\_prm$ は $R1$ に適用するべきACPIが $P1$ であると判定し，$eval\_prm(R1) = grant$ を出力する．
2. $provisions(R1)$ は $provision$ を呼び出す．このとき，$eval\_pvn(R1)$ からの出力が引数となる．ここでは，
$$\begin{cases} (Encrypted\_Text, S1, encrypt) \\ (MAC_{TextAuthenticated\_Data}, S1, write) \end{cases}$$
が引数となり，二つの新たなARが発生する．
3. $provision$ は再帰的に呼び出され，$P2$ と $P3$ が適用するべきACPIとして選択される．
4. 最終的に $provisions$ がPRM($= grant$) と必須処理OPS($= encrypt\ and\ write$) をADとして出力する．
5. $Alice$ がEncrypted_Textに書き込むとき，データの暗号化と認証子の更新が課せられる．

$R1$ により，encrypt必須処理とwrite必須処理が $Alice$ に求められ，セキュリティ操作の関連付けを実現することができる．また，$Alice$ が要求される必須処理を実行する権限を持つということも同時に検証される．

以上のように，多機能セキュアデータベースに求められるセキュリティ操作の関連付けはPBAC方式によって容易に記述することができる．

3.4 Rule Verifier

[4]では，ポリシー設定によって発生する問題を解決する要素として，“Rule Verifier” という概念が示されている．これは記述段階からポリシーの内容をモニターし，誤りを検知するという仕組みである．これから本稿で扱うポリシー設定問題は，Rule Verifierによって解決することができる．ただし，[4]においては，Rule Verifierの具体例は示されていない．従って，本稿で示すポリシー設定問題の解決方法はRule Verifierの具体的実装例ということになる．

4 アクセス制御ポリシー設定問題

多機能セキュアデータベースにPBAC方式を適用すると，ポリシー設定によっては様々な問題が発生する．特に，人手によってポリシーを設定すると仮定したとき，設定者の不注意などによって本来意図したものとは異なるポリシーが記述される可能性がある．本稿では，このようなポリシー設定問題を議論する．

4.1 ポリシー設定の分類

以下に問題を発生させるポリシー設定を分類する．

1. ARが正当なものであるにも関わらず，ポリシー設定ミスのためにdenyフラグが設定されているとき．
2. 1と同様の場合において，denyフラグは設定されていないが，適用するべきポリシー設定が記述されていないとき．
3. 2回以上の必須処理呼び出しのために，処理が無限ループになってしまうとき．

4.2 ポリシー設定問題が発生する原因

これらは手作業によるポリシー設定ミスを主な原因としている．データベースが大規模になれば，このようなポリシー設定問題はさらに深刻になる．従って，このような問題をシステムが自動的に解決するような仕組みが必要である．以下，分類した問題それぞれについて解決方法の一部を提示する．なお，具体例は全て図1を使用している．主体は $Alice, Bob$ を設定し，それぞれ $S1, S2$ というロールを有する．両者はほぼ同等の権限をもつとする．また，Px はACPIを,Rx はARを示すこととする．

5 不当なdenyフラグ設定

5.1 具体例

以下のようにACPIを設定する．

$P4(Encrypted\_Text, S1, write, grant,$
$\quad (Encrypted\_Text, S1, encrypt),$
$\quad (MAC_{TextAuthenticated\_Data}, S1, write))$
$P5(Encrypted\_Text, S1, encrypt, grant)$
$P6(MAC_{TextAuthenticated\_Data}, S1, write, grant,$
$\quad (Log, S1, write))$
$P7(Log, S1, write, deny)$

$S1$ のEncrypted_Textへの書き込みは正当なアクセスであると仮定すると，ポリシー設定問題は $P7$ の設定にみられる．本来は $P7'(Log, S1, write, grant)$ とするべきであったところで，設定ミスが発生し，$P7$ の設定になってしまった．ここで $R2(Encrypted\_Text, Alice, write)$ が起こると，$prov\_prm$ によりdenyフラグが検出され，$provision$ はエラーを出力する．しかし，仮定から $Alice$ が要求したアクセスは正当なものと認められるため，$P7$ の設定ミスである．エラー出力はシステムの動作面からは正当なものであるが，ポリシー設定問題の面からは解決とはならない．

5.2 解決方法

$P7$ はログに関する ACPI であり，ここに示されている ACPI だけをみて修正することはできない．$S1$ が真にログ書き込み権を所有しているかどうかが問題となる．そこで，Rule Verifier に次のような機能を持たせる．(1)$S1$ とほぼ同じ権限を持つ主体 (例えば $S2$ など) がログに関してどのような権限をもっているかを探索する，(2) 探索された ACPI に設定されている権限のうち多数を占める権限設定を数種類抽出する，(3) 抽出されたものに関して優先順位を付与する，(4) 優先順位の高いものを一時的に $P7$ として記述する．この方法は一時的なエラー回避方法であるため，ポリシー設定の責任者へ警告メールを送信するなど，ACPI の完全修正を促す処理を行うことが望ましい．このとき，(2) で付与された優先順位を参照できるようにしておけば，設定者にとっても便利である．

5.3 評価

5.2 の方法を評価する際に問題となる点は，一時的に与えられた権限のために意図しなかった主体が権限を所有してしまうことである．5.2 の例では，$S1$ はログ書き込み権を所有していないが $S2$ らは所有していた，という状況であると $S1$ は一時的に本来意図されていない権限を有してしまう．この問題を踏まえ 5.2 の方法を評価する．(1)5.2 の方法は一時的解決法であり，ポリシー設定者には設定上の問題があることを知らせる仕組みになっている．そのため，意図されていない権限が与えられたまま放置されるということはない（ただし，ポリシー設定者を全面的に信用する場合のみ）．(2) 探索に関するパラメータによって権限が拡大するかどうか決定されるため，システムの方針によって調整することができる．つまり，5.2 における「ほぼ同じ権限を持つ主体」の選定方法や抽出基準となる「多数」の割合設定などによって安全性が変化する．従って，パラメータを変化させることによって意図しない権限設定の回避になると考えられる．なお，著者らはこの評価を定量的に示しているわけではない．実装と定量的評価は今後の課題としたい．

6 必要なポリシーが欠如している設定

6.1 具体例

以下のような ACPI が設定されている．

$P8(Encrypted\_Text, S1, write, grant,$
$\quad (Encrypted\_Text, S1, encrypt),$
$\quad (MAC_{TextAuthenticated\_Data}, S1, write))$
$P9(Encrypted\_Text, S1, encrypt, grant)$
$P10(MAC_{TextAuthenticated\_Data}, S1, write, grant,$
$\quad (Log, S1, write))$
$P11(Log, S1, read, grant)$

$S1$ の Encrypted_Text への書き込みは正当なものであるとする．ここでは $P11$ の設定に問題がみられる．本来あるべき $P11'$ は $S1$ のログ書き込みに関する権限の設定である．しかし，必要な ACPI が記述されていない．ここで，$R3(Encrypted\_Text, Alice, write)$ が発生すると，エラーが出力されずに処理が行われる．もし，デフォルトポリシーを “Default Denied” に設定しておけば，ACPI に記述がないアクセスについては全て拒否することになる．しかし，上記した仮定のもとでは，このデフォルトポリシーは問題を解決しているとはいえない．

6.2 解決方法

この問題を解決する方法は二種類考えられる．

6.2.1 一時的に ACPI を修正する方法

この問題も 5.2 と同様，$S1$ がログに書き込む権限を所有しているかどうかによって対応が変わる．従って，5.2 のように Rule Verifier に ACPI の探索と優先度付けの機能を持たせ，一時的に修正することで解決する．さらに，ポリシー設定者に連絡し，人手による修正を要求する．

6.2.2 システムを異常停止させる方法

デフォルトポリシーが “Default Denied” のとき，アクセスが真に拒否されたのか，ACPI の設定問題なのかが明確ではない．そこで，デフォルトポリシーを “Default Stop” とし，必要な ACPI が記述されていない場合，システムを異常停止させて，ポリシー設定者に警告メールを送信することにより人力的解決を求める．このデフォルトポリシーであれば，ACPI の設定ミスであることが明確になるため，問題の早期解決に役立つ．

6.3 評価

6.2.1 の方法に関しては，5.3 の評価と同様である．6.2.2 の方法では，デフォルトポリシーが “Default Denied” のときより ACPI 設定問題の発生が明確になる．従って，ポリシー設定者を全面的に信頼するのであれば，ACPI 設定ミスの早期検知と解消につながると考えられる．

7 無限ループになる設定

7.1 具体例

無限ループを発生させるような ACPI を以下のように設定する．

$P12(Encrypted\_Text, S1, read, grant,$
$\quad (MAC_{TextAuthenticated\_Data}, S1, authenticate))$
$P13(MAC_{TextAuthenticated\_Data}, S1,$
$\quad authenticate, grant, (Encrypted\_Text, S1, read))$

ここで，$R4(Encrypted\_Text, Alice, read)$ が発生する．Encrypted_Text 読み出しのための必須処理は MAC に

よる認証である．しかし，MAC による認証の必須処理が Encrypted_Text 読み出しとなっているため，ループから抜け出すことができない．

7.2 解決方法

この問題は [4] においても議論されている．Rule Verifier にカウンター機能を持たせることで解決することができる．例えば，AR にカウンター領域を組み込み，PBAC 方式の関数には必須処理を呼び出すたびにカウンターを減らしていく処理を付け加える．これにより，無限ループを回避することができる．図 2 にカウンター機能を付与したアルゴリズムの一例を示す．ここでのデフォルトポリシーは “Counter 0 Denied” となる．

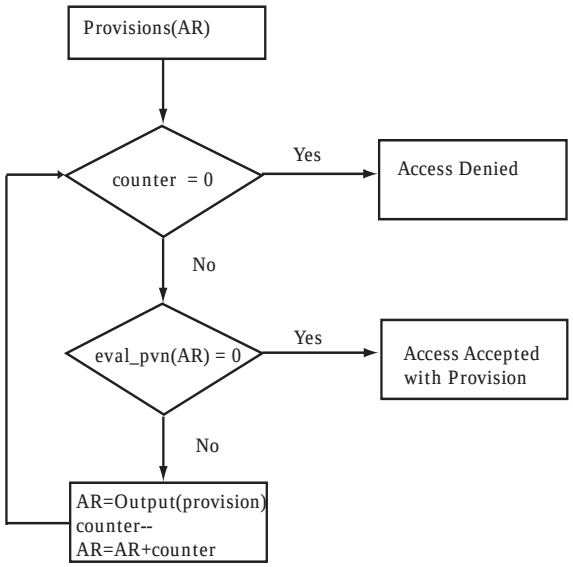

図 2: カウンター機能の一例

7.3 評価

7.2 の方法では，カウンター数の設定が小さいとき正当なアクセスが実行できない可能性がある（全ての ACPI にカウンター機能を付与すると仮定する）．これについては，システムの処理速度などが影響するため一般的な評価はできないが，システムの諸条件を無視できるのであれば，カウンター数を十分に大きくすることで解決できる．

カウンター機能は無限ループを発生させるような ACPI 設定の早期発見にも役立つ．無限ループが発生したとき，そのアクセスは拒否されてしまうが，ログに記録が残る．ログから原因となる ACPI を探し，解決することができる．

8 おわりに

本稿では，多機能セキュアデータベースにおいてセキュリティ操作の関連付けを行うための一手法として PBAC 方式を適用し，アクセス制御ポリシー設定によって発生することが予想される問題について整理・分析した．また，その解決方法の一部を提案し，評価を行った．本稿では触れなかったが，データベースに特有の問題である間接的情報フローについても PBAC 方式で解決することができると考えられる．今後の課題は，本稿で提案した方法の実装と定量的評価である．

参考文献

[1] 安東学，松浦幹太，馬場章: “多様なセキュリティ操作を考慮したオブジェクト指向データベースの構造,” コンピュータセキュリティシンポジウム (CSS2001) 論文集，情報処理学会，pp19-24, Oct. 2001.

[2] 工藤道治，平山唯樹，羽田知史，Alexander Vollschwitz: “ポリシー評価と執行に基づくアクセス制御モデルと記述言語,” 2000 年暗号と情報セキュリティシンポジウム予稿集, 電子情報通信学会, SCIS2000-A33, Jan. 2000.

[3] 工藤道治，羽田知史: “PBAC アクセス制御方式に基づくセキュリティポリシー,” 2001 年暗号と情報セキュリティシンポジウム予稿集，電子情報通信学会，pp171-176. Jan. 2001.

[4] Michiharu Kudo and Satoshi Hada: “Access Control Model with Provisional Actions,” IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, E84-A, pp295-302, Jan. 2001.

[5] Michiharu Kudo: “PBAC: Provision-based Access Control Model,” International Journal for Information Security, Springer Verlag (to appear).

[6] Eduardo B. Fernandez, Ehud Gudes, and Haiyan Song: “A Model for Evaluation and Administration of Security in Object-Oriented Databases,” IEEE Transactions on Knowledge and Data Engineering, Vol.6, No.2, pp275-292. IEEE, April, 1994.

[7] Sushil Jajodia, Pierangela Samarati, V. S. Subrahmanian, and Elisa Bertino: “A Unified Framework for Enforcing Multiple Access Control Policies,” Proceedings of International Conference on Management of Data, pp.474-486. ACM SIGMOD, May, 1997.

[8] Dirk Jonscher and Klaus R. Dittrich: “Argos–A Configurable Access Control System for Interoperable Environments,” Database security IX : status and prospects : proceedings of the Ninth Annual IFIP TC11 Working Conference on Database Security, pp.43-60. Chapman & Hall, 1996.