時間分割を用いた階層的アクセス権設定法

井上清一[†] 中盛友紀[†]
下川徳之[†] 永瀬宏[†]

ファイルシステムにおけるアクセス制御を強制的に実施する有望な手法として，BLP（Bell and LaPadula）モデルを原則とした階層的運用がある．この運用では，情報のフローを下位から上位に一方向化するため，機密性に優れたシステムの構築が可能である．また，多数のファイルを扱う状況のために，機械的に階層レベル情報を付加するSLA（Security Level Assignment）アルゴリズムが用意されている．しかし，BLPモデルを原則とした階層的運用では，各々のユーザからアクセス要求を収集した際，機密要求との間に矛盾が含まれる場合がある．SLAにおいてはこのようなアクセス要求は検出のみ可能であり，システム管理者がその後，矛盾を含むアクセス要求を提示したユーザに対してアクセス要求の再検討を依頼する必要があった．本稿では，各アクセス要求が恒常的に確保しておく必要がないケースに着目し，SLA時間分割アルゴリズムを提案する．アクセス要求は恒常的に必要なものばかりではないため，アクセス要求を満たす十分な時間を割り付けることができれば，生じた余剰時間を他のアクセス要求に割り当てることができる．これにより，ユーザに再検討させることなく，アクセス要求を実施できる可能性を生じる．

Assignment of Hierarchical Access Right Which Uses Time Division

Kiyokazu Inoue,[†] Yuki Nakamori,[†] Noriyuki Shimokawa[†]
and Hiroshi Nagase[†]

The hierarchical access control using the BLP (Bell and LaPadula) model is known as a effective technique of the mandatory access control in a file system. Since this access flow control restricts information flow to one-way from low level to high level, the system of high confidentiality can be built. Moreover, SLA (Security Level Assignment) algorithm is prepared in order to determine the hierarchical level information for many files. In the hierarchical access control design based on the BLP model, access demands are firstly collected from each user. Then, sometimes inconsistency may be included in those demands. As for SLA algorithm, only detection of such a demand is possible. The system administrator needed to request the user who presented inconsistent access demand. This paper proposes SLA time division algorithm. This algorithm is based on the case that each access demand does not always to be kept constantly. Because of the characteristics of temporal access demand, if sufficient time to fill a access demand of a user can be assigned, the produced surplus time can be assigned to other access demands. An access demand can be fullfilled by this algorithm, without any additional design examination taken by a user.

1. はじめに

階層的なレベル情報を持つファイルシステムを設計する上で，そのレベル情報を決定する手法にSLA（Security Level Assignment）アルゴリズムがある[1),2)]．システム管理者は，ユーザが望むアクセス要求（以下，要求と述べる）と，特定の情報フローを禁止する機密性条件を実現するために，このアルゴリズムを用いて機械的に階層レベルを割り当てる．SLAアルゴリズムはユーザから寄せられた要求を検査し，それらをすべて満たすことのできる階層レベル情報を，システム管理者へ返答することができる．

しかし，SLAアルゴリズムが要求を収集する過程において，各々のユーザから寄せられた要求の組合せには矛盾が含まれる場合がある．各々のユーザは通常，他のユーザの要求を意識することなく，システム管理者に自身の要求を伝える．SLAアルゴリズムはこのとき，矛盾が含まれる要求を検知してシステム管理者に通知する．システム管理者はこの通知を受け，矛盾を含む要求を提示したユーザに対して，要求の再検討を要請しなければならない．

† 金沢工業大学情報工学科
Department of Information and Computer Engineering, Kanazawa Institute of Technology

本稿はこのような状況に対して，SLA 時間分割アルゴリズムを提案する．このアルゴリズムは，各要求が恒常的に確保しておく必要がないケースに着目する．たとえば，業務によってはある時間帯のみの要求だけでよい場合や，他の処理の後でないと実施不可能な作業等が存在する．これらの業務から生じる要求は，必ずしも恒常的なアクセス権を必要としない．要求を満たす十分な時間を要求に割り付けることができれば，生じた余剰時間を他の要求に割り当てることができる．

時刻を用いた論理を論じる場合，時制論理があげられる．時制論理は，命題論理等の通常の論理体系に「つねに」や「いつか」といった時間の概念を取り入れて構成された論理体系である．時制論理を取り上げた研究としては，時間的性質を組み込んだ部分仕様を組み合わせて，条件に満足するシステム全体としてのLOTOS 記述を導出する研究[3]や，アクセスの順序を時制論理と代数的仕様で評価する形式的仕様記述支援ツールの作成の研究[4]等がある．本稿では，上記のような仕様を組み合わせたり評価したりする研究とは異なり，ユーザの要求条件と提案するシステムの仕様という限定された条件の中で，矛盾している要求条件を時間分割を用いて解決するアルゴリズムを論ずる．

本稿はこの点に着目し，矛盾する要求がユーザから寄せられた状況において，SLA 時間分割アルゴリズムを用いて要求の時間分割と要求実行順序の決定を試みる．システム管理者はこのアルゴリズムによって，矛盾する要求が構成された状況においても，機械的な解決を行える可能性を得る．2 章では階層的アクセス制御を導入したシステムにおいて矛盾を含む要求の構成例を述べ，本稿がこの構成に対して用いるアプローチを示す．3 章では SLA 時間分割アルゴリズムを例に沿って述べる．

2. 時間分割の概念

階層的アクセス制御を利用するにあたり，その階層レベル設定を決定する手法として，SLA アルゴリズムが利用される．本章では，SLA アルゴリズムの利用前提となる階層モデルとそこで生じる問題点について述べ，問題を解決するために用いる時間分割の概念と具体的な手法について説明する．

2.1 Bell and LaPadula（BLP）モデル

階層的アクセス制御を導入したセキュリティモデルに BLP（Bell and LaPadula）モデル[5]がある．BLP モデルでは，階層的なレベルと非階層的なカテゴリの 2 つの要素からなるラベル（サブジェクトのラベルは clearance，オブジェクトのラベルは classification と呼ばれる）が定義され，それらの間における半順序関係（あるいは束）に基づいて情報フロー制御が実施される．本稿ではこのうち階層的なレベル，すなわち全順序関係にのみ着目する．一般的な半順序関係はレベルとカテゴリの付与法も含めて抜本的な検討が必要であり，これらは今後の課題としたい．

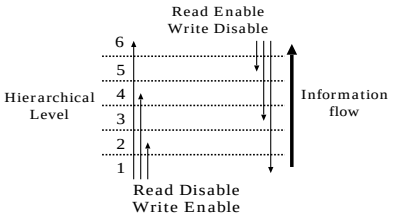


図 1 階層的アクセス制御の原則
Fig. 1 The principle of hierarchical access control.

全順序に限定した場合の BLP モデルは，階層的なレベルが存在する中で下位階層レベルから上位階層レベルに対してのみ書き込み許可があり，階層レベルの上下関係で情報フローの有無が明確に判定できる利点がある．この BLP モデルの情報フローにおける原則を，**図 1** に示す．ここで図中の矢印は，情報フローの向きを示す．

この原則により，現在サーバシステムを運営するにあたって広く用いられている UNIX オペレーティングシステムに見られるような，意図しない情報のフローが BLP モデルでは発生しない．すなわち UNIX が用いているアクセス制御手法は一般的に ACL（Access Control List）方式と呼ばれ，個々のファイルに対して細かなアクセス制御が可能である反面，与える権限を十分に検討しないと，予期しないファイルから情報がフローすることがある．

これに対して，BLP モデルにおいては情報フローの一方向性が存在するため，下位階層レベルに上位階層レベルの情報がフローすることはない．機密性設計の評価に関して，階層的セキュリティレベルを使った強制アクセス制御は，ACL 等の任意アクセス制御よりも高いランク付けがされている[6]．

2.2 矛盾を含む要求

システム管理者が適切な階層レベルを設計するにあたり，まずユーザや部署等が望むアクセス権限の要求が収集される．これらはデータを流したいという処理要求と，データを流したくないという機密要求で表される．なお，コンピュータのアクセスにかかわるユーザやファイル・プログラム等をエンティティと総称する．

SLA は機密要求に対し，終点のエンティティが始点のエンティティよりも，必ず高い階層レベルを割り

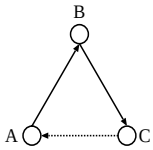


図 2 矛盾する要求例
Fig. 2 An example of demand including inconsistency.

付けるように動作する．これは BLP モデルの原則に基づき，上位階層から下位階層への情報フローを防止するためである．この SLA のレベル割付けによって，機密要求が求めている終点のエンティティから始点のエンティティに対する情報フローは防止される．しかし要求には，BLP では表現しきれない要求の組合せが含まれることがある．このような要求の構成例を，**図 2** に示す．ここで，図 2 のように処理要求と機密要求をグラフとして図示したものを統合要求グラフと呼び，エンティティを丸印で表す．処理要求は実線で表し，始点から終点に向かう情報フローを示す．また，機密要求は破線で表し，終点から始点への情報フローを禁止する．

図 2 において，エンティティ C から A に対する機密要求が存在する．したがって，エンティティ A にはエンティティ C よりも，高い階層レベルを割り付けなくてはならない．エンティティ C と A の階層レベルをそれぞれ，1 と 2 と仮定する．

次に，エンティティ A から B への処理要求とエンティティ B から C への処理要求に着目する．これらのエンティティ間において，少なくとも終点のエンティティにおける階層レベルは，始点のエンティティ以上の階層レベルでなければならない．エンティティ B の階層レベルは少なくとも 2 以上，エンティティ C も同様に 2 以上にする必要がある．すると，先ほどエンティティ C に割り付けた階層レベルでは，上位階層から下位階層への情報フローを実施できず処理要求を満たすことが不可能となる．強制的にエンティティ C に階層レベル 2 を割り付けると，エンティティ A との機密要求が成り立たなくなる．

あるエンティティからの要求をたどっていったとき，経路によって元のエンティティに戻る場合，本稿ではその経路をループしているという．SLA アルゴリズムは，機密要求を含むループを検出した場合，この要求が矛盾しており実現不可能であることをシステム管理者に通知してアルゴリズムを停止した．システム管理者はこの要求を申請したユーザに連絡をとり，再度の要求検討を求める必要がある．

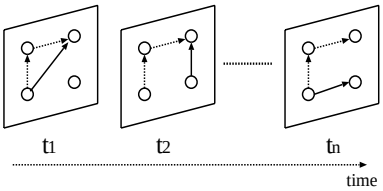


図 3 時間ごとに異なる統合要求グラフ
Fig. 3 Time-dependent integrated demand graph.

2.3 矛盾を含む要求の分割

本稿では前節の問題に対し，各処理要求を必ずしも恒常的に確保しておく必要がないというケースに着目する．SLA では要求の時間変化を考慮しておらず，割り付けられた階層レベルは恒常的にユーザの要求を満たすように設計される．しかし実際には，特定の時間帯には一部の要求が必要でなかったり，ある処理の前かもしくは後だけに要求を満たすことができればよかったりする等の，限定された要求の場合がある．SLA 時間分割アルゴリズムでは，恒常的な要求として扱っていたこれらの要求に時間的制限を設ける．時間ごとに異なる統合要求グラフが存在するイメージを，**図 3** に示す．

なお，High-Water-Mark 方式に見られる階層レベルの変更をともなう実行制御については，階層レベルの変更を実施したエンティティとつながる要求が満たせなくなる場合があるため，本稿では取り上げていない．本稿の手法は，一度決定された階層レベルを用い続けるという前提での利用とし，階層レベルの変更をともなう解決法は別稿としたい．

SLA 時間分割アルゴリズムでは，矛盾する要求を構成するループを最も少ないエンティティをたどるよう（これを最小の構成と呼ぶ）に取り出し，統合要求グラフを分割する．統合要求グラフの分割は，次の原則で実施される．

- 統合要求グラフを，ループを構成する処理要求の数で分割する．
- 各統合要求グラフには 1 つの処理要求が存在し，他の統合要求グラフの処理要求と重複しない．
- 機密要求は分割後の各統合要求グラフすべてに存在する．

機密要求は時間不変であり，ある時間帯において突然消失することはない．したがって，機密要求は分割後のどの統合要求グラフにも記入され，すべての時間帯において情報のフローに制限を与える．これらの原則を基に，図 2 のグラフを分割した状態を**図 4** に示す．分割した各グラフは識別のため，順に (a)，(b) とする．

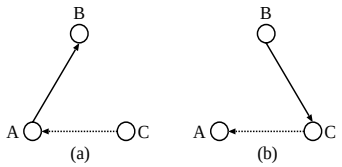


図 4 要求の時間分割
Fig. 4 Time division of a demand.

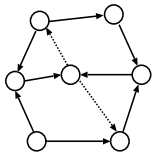


図 5 矛盾を含む統合要求グラフ
Fig. 5 The example of integrated demand graph including the contradictory demand.

表 1 実行順序の組合せによる情報フロー
Table 1 Information flow by combination of execution.

実行順序	Entity A	Entity B	Entity C	判定
a → b	B,C	A,C	A,B	不適
b → a	B,C	A,C	B	適

2.4 要求の実行順序決定

統合要求グラフの分割後，これらの統合要求グラフを用いる適切な順序を決定する．たとえば，図 4 のグラフは (a)(b) の順に用いる場合，時間とともにエンティティ A の情報がエンティティ C にフローし，エンティティ A と C の間に設定された機密要求を満たすことができない．(a) のグラフにはエンティティ A と B の間に処理要求が存在するため，(a) を用いるとエンティティ B に対してエンティティ A の情報がフローする可能性がある．ここで続けて (b) を用いると，先ほどエンティティ B にフローしたエンティティ A の情報が，エンティティ C にフローする可能性がある．

しかし，グラフを用いる順序を (b)(a) とすると，エンティティ A の情報はエンティティ C にフローしない．図 4 における統合要求グラフを用いる順序と情報フローの関係を，**表 1** に示す．

表 1 は，左欄に示す順序で統合要求グラフを用いた際に，各エンティティにどのエンティティの情報がフローする可能性があるかを示す．たとえば，「a → b」の順序における「Entity A」の欄には，「B,C」と示されている．これは，この順序を用いた際に，最終的にエンティティ A には，エンティティ B と C の情報が存在する可能性があることを意味する．

図 2 の統合要求グラフでは，エンティティ A の情報がエンティティ C にフローしないことが求められているため，表 1 の「Entity C」の欄にエンティティ A が含まれている順序の「a → b」は適切でない．システム管理者はこの表から適切と判断された「b → a」を選択するとよい．

2.5 本稿の技術の位置付け

近年，システムの設計等においてこれを構築するためには，単純に要求を満たす解を求めるだけではなく，セキュリティを意識した設計が強く求められている．本稿の技術はこの意識を反映しており，セキュリティを含めた解の提示に意義があると考えている．

たとえば，ワークフロー的業務等では一定の作業の流れがあり，各々の作業においてデータの入出力要求が存在する状況で，これを満たす設計が想定される．そうしたとき本稿の技術は，要求を集めてアルゴリズムを実行することにより，その要求が妥当であるかどうか，またどのような順序でデータフローと作業を分割して実施すると要求を満たしつつセキュリティを維持することができるかどうか，提示することが可能となる．この結果，恒常的にアクセス権限を与える従来の SLA アルゴリズムではセキュリティを満たせないような問題に対しても，SLA 時間分割アルゴリズムでは解の提示が可能となる場合がある．

3. 時間分割アルゴリズム

本章では，SLA 時間分割アルゴリズムを実施するための具体的な手法を，**図 5** の統合要求グラフを用いて示す．アルゴリズムは大別して 2 つの区分から構成される．まず，統合要求グラフから矛盾を含む要求を構成する部分を，最小構成で検出する．その後，検出された部分を要求単位に分割し，要求を矛盾なく実行可能な順序の組合せを求める．

3.1 問題定義

本稿が取り上げる問題設定を，下記に示す．

- 処理要求と機密要求の収集を実施し統合要求グラフを作成したとき，階層レベルの割付けが不可能である要求の構成が生じることがある．この構造を生じさせる要因は要求のループであり，この要求を階層構造を用いて表現すると，上位階層から下位階層への矢印となって表される．この構造は，階層アクセス制御の原則に反する．

本稿が目標とする設定変更を，下記に示す．

- ループが存在する統合要求グラフで処理要求の実行順序を変更することによって，ループが実現可能となる場合があるため，この順序を求める．

設定変更を行ううえで考慮される制限事項を，下記

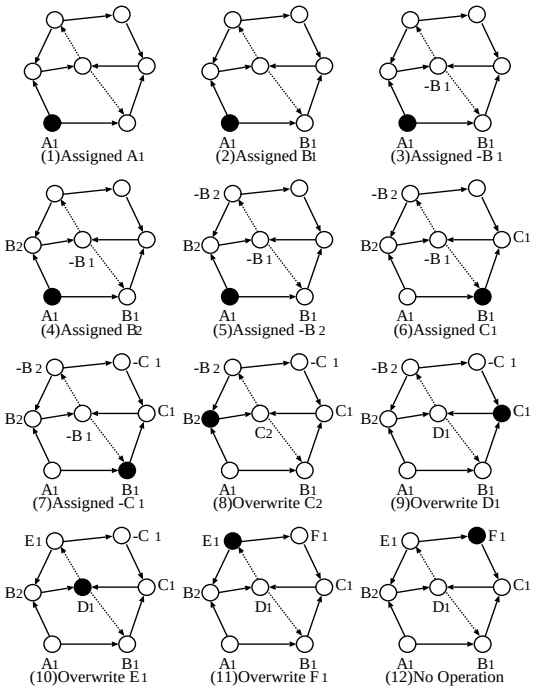


図 6 ラベル割付け過程
Fig. 6 Labeling process.

に示す.

- 各要求辺の先に存在するエンティティには，要求辺の根であるエンティティの情報がフローする.
- 機密要求辺の先に存在するエンティティの情報は，その機密要求辺の根であるエンティティへフローしてはならない.

3.2 統合要求グラフへのラベル付け

最初に，エンティティを一意に識別するためのラベルを各エンティティに割り付ける．また，この処理によって矛盾を含む要求を構成するエンティティのペアを検出する．**図 6** に，本節で述べる (S1) から (S6) までの手順によってエンティティにラベルが割り付けられる過程を示し，アルゴリズムを述べる.

(S1) 処理要求，機密要求の入力辺がないエンティティ（入力される辺がなく 0 であることを，入次数 0 と表現する）を見つけ，A_1 というラベルを振る．入次数 0 のエンティティが複数ある場合には，順に A_2，A_3，$\cdots$ と振る．ただし，入次数 0 のエンティティが存在しない場合，階層的に最も低いレベルとなるエンティティを決定するために，システム管理者が A_1 とすべきエンティティを手動で作成し，統合要求グラフにおいて最も低い階層レベルとするエンティティに A_1 からの要求を入力する．なお，手動で A_1 を作成したときのみ，要求を入力するエンティティの選定によってはラベル付けが行われないエンティティを生じる可能性がある．(S6) までの一連のアルゴリズムを終了してもラベル付けが行われなかったエンティティがある場合は，A_1 からの要求を入力するエンティティを変更して，(S1) から再度実行するものとする.

(S2) ラベルの付け終わっているエンティティの中で，最もラベルの値の低いエンティティを選択する．ラベル値の大小は英字をまず比較し，A を最小としたアルファベット順で比較する．英字ラベルが同一である場合には，英字に振られている数値で比較する．ここで選択され処理の対象となったエンティティを，現エンティティと呼ぶ（図中，黒丸によって示す）．選択できるエンティティがなくなった場合は，アルゴリズムはここで終了する．例では，(S1) によってラベル付けされた A_1 のみが存在するため，A_1 が現エンティティとなる．なお，再び (S2) が処理される際には，後述する (S6) によって現エンティティが選択されないように除外されているため，「ラベルの付け終わっているエンティティの中で，最もラベルの値の低いエンティティ」という条件に合致するエンティティは，B_1，B_2，$\cdots$ と変化していく.

(S3) 現エンティティから出る矢印を列挙し，順に現エンティティの記号を 1 つ進めた記号を用いて（たとえば B_1，B_2，$\cdots$，のように）矢印の先の各エンティティ（次エンティティとする）にラベル付けする．すでにラベルが付けられていたエンティティに対しては，新しいラベルで上書きする．ただし上書き時には，後述するループチェック用の記憶領域に保存されたエンティティの組合せを参照し，Entity from で示されるエンティティが現エンティティと同じラベルかつ，Entity to と上書き対象であるエンティティに振られているラベルが同じである場合には，上書きを行わない.

(S4) 次エンティティ（複数ある場合はそれぞれ）に入っている矢印を列挙し，次の条件に合致するエンティティからの入力を検出する.

- ラベルがまだ付けられていないエンティティ
- ラベルにマイナス記号が振られているエンティティ

このような入力が存在しなかった場合は，(S6) へ.

(S5) (S4) で検出されたエンティティを Entity from，次エンティティを Entity to とし，Entity from に次エンティティをマイナスとしたラベルを振って，ループチェック用の記憶領域へ対で記憶する（Entity from が複数あるときは，$-B_1a$，$-B_1b$，$\cdots$ と振る）．次エンティティが複数ある場合は，それぞれの次エンティティについて実施する．この情報は一意に識別するために，ループナンバ（これを LNo とする）を検出さ

表 2 ループチェック用記憶領域の構成
Table 2 Composition of storage area for loop check.

LNo	Entity from	Entity to
1	$-B_1$	B_1

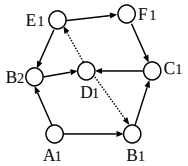


図 7 ラベル割付け後の統合要求グラフ
Fig. 7 Integrated demand graph after label assignment.

表 3 ラベル割付け後のループチェック用記憶領域
Table 3 Storage area for loop check where label is stored.

LNo	Entity from	Entity to
1	D_1	B_1
2	E_1	B_2
3	F_1	C_1

れた順に 1，2，3，$\cdots$ と振る．これらの情報が格納された記憶領域の例を，**表 2** に示す．

ここで，以後の処理でこれらのエンティティにラベルが再度割り付けられた場合は，同時にループチェック用記憶領域のラベルも更新する．たとえば，B_2 が現エンティティである際，(S3) アルゴリズムにおいて $-B_1$ エンティティは C_2 エンティティで上書きされる（図 6 の 8）．このとき，ループチェック用記憶領域にあるすべての $-B_1$ は，C_2 へ機械的に置換される．(S6) 現エンティティを (S2) において選択されたエンティティから除外して，(S2) からの手順を繰り返す．

ラベルの割付けが完了した状態を**図 7** に示す．ループチェック用の記憶領域に保存されたラベルもアルゴリズムの進行とともに書き換わり，最終的に Entity from の値が**表 3** で示されるラベルに変化する．

ラベル付けアルゴリズムは幅優先探索をもとにしている．ループチェックを行っているので，同一ループを繰り返しラベル付けすることはない．したがって辺の数を e としたとき，探索の計算量（辺をたどる回数）は通常の幅優先探索と同様に $O(e)$ である．

また，入次数 0 のエンティティが複数ある場合，これらのエンティティへのラベル付けは A_1，A_2，A_3，$\cdots$ と任意に行うことができる．このことを説明するために，統合要求グラフに新たにエンティティ ϕ を追加する．エンティティ ϕ から入次数 0 の各エンティティ（この集合を S とする）へ処理要求辺（実線の矢印）を追加した統合要求グラフを考えると，3.2 節に述べたアルゴリズムにより，S に属するエンティティは ϕ から見て次エンティティとなり，そのラベル付けは A_1，A_2，A_3，$\cdots$ と任意である．このようにしてすべてのラベル付けが終了した段階でエンティティ ϕ を除去すれば，元の統合要求グラフのラベル付けが完了する．

3.3 ラベル付け動作の評価

本稿の基礎となるラベル付けおよびループ検出のアルゴリズムについて，次のように評価できる．まず，本アルゴリズムの動作を阻害する要因として，次の 3 点を示す．

- ラベルの付け忘れ．
  - アルゴリズムが終了しても，ラベルが振られていないエンティティがある．
- アルゴリズムが停止しない．
  - ループを検出せず，永久にラベルを上書きし続ける．
- 意図しないラベル付け．
  - 要求を満たしていないラベルが振られる．

①ラベルの付け忘れ

このアルゴリズムは，幅優先探索法によってラベル付けを実施している．ラベルの付け忘れが発生すると仮定した場合，幅優先探索法がすべてのノードを回りきらない場合があることになる．しかし幅優先探索法は連結グラフであれば，すべてのノードを巡回することができるので，ラベルの付け忘れは発生しない．

②アルゴリズムが停止しない

アルゴリズムが停止しないと仮定する．まず，ループを含まない統合要求グラフに対してこのアルゴリズムを用いる場合，幅優先探索法によってエンティティが巡回されエンティティ数も有限であるため，アルゴリズムの停止に支障がないことが示される．そこで，ループを含む統合要求グラフにおいてアルゴリズムが停止しないためには，ループの検出に失敗する必要がある．

しかしこのアルゴリズムは，次エンティティに入力される要求のうち，下記 2 点の条件に合致するエンティティからの要求を，ループを構成する可能性があるとして検出する．

- ラベルがまだ付けられていないエンティティ
- ラベルにマイナス記号が振られているエンティティ

これらをループと判断する根拠は，探索方法に由来する．幅優先探索では，未探索のエンティティは探索済みのエンティティよりも先に位置する．このため，これらの条件に合致するエンティティから入力される要求のみが，ループを構成する可能性を持つ．そのた

表 4 時間分割対象ループテーブルの構成
Table 4 Composition of target loop table for time division.

LNo	Pt	Pv	終了状態

め，ループの可能性がわずかでもある要求は必ず列挙されるため，ループの検出漏れが発生しない．このことから，アルゴリズムが停止しないために必要な条件を満たすことができず，仮定が矛盾すると説明できる．

③意図しないラベル付け

意図しないラベルが付けられると仮定する．この仮定の成立には，次の 2 点のどちらかが可能でなければならない．

- 大きすぎるラベル付け
  - 例：D_1 とすべき場所を E_1 とする

現エンティティは現エンティティに割り付けられたラベルよりも，1 つだけ大きい英字を，次エンティティに割り付ける．この割り付けるラベルは必ず 1 つだけ大きな英字であり，必要以上の大きなラベル付けが生じない．

- 小さすぎるラベル付け
  - 例：D_1 とすべき場所を C_1 とする

探索されたときに値を満たさないラベルが付けられていた場合，そのエンティティは値の大きいラベルによって上書きされる．値の大きいエンティティへ値の小さいエンティティが上書きするという処理はなく，次エンティティには現エンティティよりも 1 つ以上大きな値のラベルが割り付けられ，かつ，次エンティティに入っている要求のうち 1 つに，次エンティティよりも 1 つだけラベルの小さいエンティティからの入力が必ず存在する構成となる（ループとして検出されたエンティティの組合せは除く）．したがって，小さすぎるラベル付けを生じない．

したがって上記 2 点のどちらも成立することはなく，意図しないラベル付けのために必要な条件を満たすことができず，仮定が矛盾すると説明できる．

3.4 矛盾を含む構成の検出

次に，(S1) から (S6) までの処理で検出されたエンティティを元に，ループの全体構成を検出する．ここで利用するテーブルを時間分割対象ループテーブルと呼び，**表 4** に示す．

Pt はループパターンを示す．ループの全体構成を探索する際に複数の探索経路が見つかることがあるため，その経路をループパターンとし，Pt の値によって区別する．Pt は最小値を 1 として 2，3，$\cdots$ と増加させる．Pv にはループチェック用記憶領域の Entity from で示されるエンティティから要求を逆にたどったときのエンティティが，(S7) から (S11) までのアルゴリズムによってカンマ記号で区切って順に記録される．

(S7) (S1) から (S6) までの処理によって，ループチェック用の記憶領域には，ループを構成する一部となるエンティティが対で記憶されている．ここより，LNo の最も小さいものを検査対象として選択する．これを，現ループテーブルとする．この例ではまず，D_1 から B_1 に対する機密要求の組合せが現ループテーブルとして選択される．

(S8) 現ループテーブルの LNo を時間分割対象ループテーブルの LNo 欄に記入し，Pt の欄には 1 を，あわせて Pv 欄には現ループテーブルの Entity from で示されているエンティティを記入する．例ではまず Pv に D_1 が記入される．

(S9) Pv の最後尾に記入されているエンティティに入っている矢印をすべて列挙し，その矢印の根であるエンティティを，Pv の最後尾にカンマ記号で区切って記入する．矢印の根となるエンティティが複数ある場合は，時間分割対象ループテーブルに Pt を 1 増加させた行を新規に作成して Pv の内容を複製し，矢印の根となるエンティティを Pv の最後尾にカンマ記号で区切って記入する．この例では，エンティティ D_1 からたどることができるエンティティは B_2 と C_1 があるため，Pv に B_2 を記入するとともに，Pt を 2 とする行を新規に作成して C_1 を記入する．

(S10) (S9) において，現ループテーブルの Entity to で示されるエンティティを記入した場合（現ループテーブルのエンティティを含むループ発見を意味し，終了状態①とする），同じエンティティを 2 度記入した場合（現ループテーブルとは別のループ発見を意味し，終了状態②とする），入次数 0 のエンティティを記入した場合（ループとは無関係の経路をたどったことを意味し，終了状態③とする），これらの場合は探索をここで終了し (S11) へ進む．この例では，Pv に記入したエンティティは終了状態①〜③のいずれにも該当しないため，探索は終了しない．(S9) へ戻る．

(S11) ループチェック用の記憶領域から現ループテーブルを削除し，(S7) から繰り返す．ループチェック用の記憶領域が空になっていたら，このアルゴリズムは終了する．

例を用いた際にできあがる時間分割対象ループテーブルを，**表 5** に示す．また，抽出されたループ構成を **図 8** に示す．これらのループ構成は一意に識別するためにラベルを振る．ラベルの命名規則は，LNo(x).Pt(y)

表 5 時間分割対象ループテーブル
Table 5 Target loop table for time division.

LNo	Pt	Pv	終了状態
1	1	D_1 , B_2 , A_1	③
1	2	D_1 , C_1 , B_1	①
1	3	D_1 , B_2 , E_1 , D_1	②
1	4	D_1 , C_1 , F_1 , E_1 , D_1	②
2	1	E_1 , D_1 , B_2	①
2	2	E_1 , D_1 , C_1 , B_1 , D_1	②
2	3	E_1 , D_1 , C_1 , F_1 , E_1	②
2	4	E_1 , D_1 , C_1 , B_1 , A_1	③
3	1	F_1 , E_1 , D_1 , B_2 , A_1	③
3	2	F_1 , E_1 , D_1 , C_1	①
3	3	F_1 , E_1 , D_1 , B_2 , E_1	②

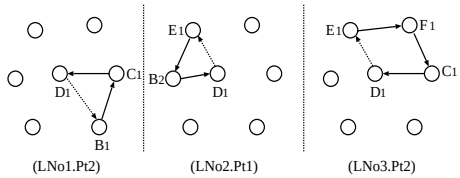


図 8 最小ループ構成の検出
Fig. 8 Detection of minimum loop composition.

の形とし，(x) には LNo を，(y) には Pt の値を入れるものとする．

終了状態の①で示されるものが，得られたループ構成である．Entity from で示されるエンティティから要求を逆にたどるこの探索は，Entity from で示されるエンティティを根とする幅優先探索に基づくものである．したがって Entity to で示されるエンティティまで最も早くたどりついた経路が最短であり，ループの構成するうえでこれ以上の短い経路は存在しない．このアルゴリズムによって求められた最短経路のループを，最小ループ構成と呼ぶ．また，ループの構成とは無関係のエンティティをたどった行については，この時点で削除を行う．

ここで，ループ構成が検出されなかった LNo があった場合，そのエンティティの組合せはループを構成する要因ではないと判断できる．このような LNo が 1 つでも存在した場合は，このエンティティの組合せをループチェック用記憶領域に登録しないようにし，(S1) からの手順を再度繰り返す．例では，各 LNo において必ず 1 つのループ構成が発見されているため，このまま (S7) へ進む．

3.5 最小ループ構成の要求毎分割

ここでは，時間分割対象ループテーブルに記録された最小ループ構成を分割する．分割された最小ループ構成は，後述する実行順序テーブルの作成のために用いられる．

(S12) ここまでの処理によって，時間分割対象ループテーブルには，ループを構成するエンティティが最小構成で記憶されている．ここより，LNo の最も小さいものを検査対象として選択する．LNo が同じ値であれば，Pt の小さいものを選択する．これを，現グラフとする．例では最初に LNo1.Pt2 に示される統合要求グラフが，現グラフとして選択される．

(S13) 現グラフに含まれる処理要求と同数の，現グラフの複製を用意する．用意した複製には，現グラフのラベルである LNo(x).Pt(y) に続けて，ピリオドで区切って a，b，c，$\cdots$ と振る．

(S14) 複製の a から順に，時間分割対象ループテーブルの Pv に示される処理要求の根となるエンティティ順に用い，各エンティティを根とする矢印以外の処理要求を，グラフから消去する．例では LNo1.Pt2 について，エンティティを D_1，C_1，B_1 の順に用いることになるが，D_1 は機密要求の根となるエンティティなので除外され，C_1，B_1 の順でエンティティが用いられる．LNo1.Pt2.a のグラフからは，エンティティ C_1 を根とする矢印以外が削除され，同様に，LNo1.Pt2.b からはエンティティ B_1 を根とする矢印以外が削除される．

(S15) (S12) で選択対象となる最小ループ構成から，現グラフを取り除き，(S12) から繰り返す．すべての最小ループ構成について処理を終えたら，終了する．

(S12) から (S15) までが完了し，各最小ループ構成が要求ごとに分割された状態を**図 9** に示す．

分割を終えた最小ループ構成は分割最小ループ構成と呼び，図 9 の LNo1.Pt2 においては，LNo1.Pt2.a から LNo1.Pt2.b のグラフを指す．また，ループに関連しない処理要求は任意の分割最小ループ構成に含めることができる．例では $A_1 \rightarrow B_1$ および $A_1 \rightarrow B_2$ の処理要求がどのループとも関連せず，任意の分割最小ループ構成に含めても矛盾を引き起こさない．

3.6 実行順序テーブルの作成

ここでは，実行順序テーブルを作成する（**表 6**）．分割を実施した最小ループ構成は，その実行順序によっては，要求に反する情報フローを生じることがあるため，ここで分割後の実行順序を適切なものに絞る．

(S16) 図 9 より，LNo の最も小さい分割最小ループ構成を検査対象として選択する．LNo が同じ値であれば，Pt の小さいものを選択する．これを，現グラフとする．例では最初に LNo1.Pt2.a から LNo1.Pt2.b に示される分割最小ループ構成が，現グラフとして選択される．

(S17) 実行順序を決定する．実行順序は各グラフに割

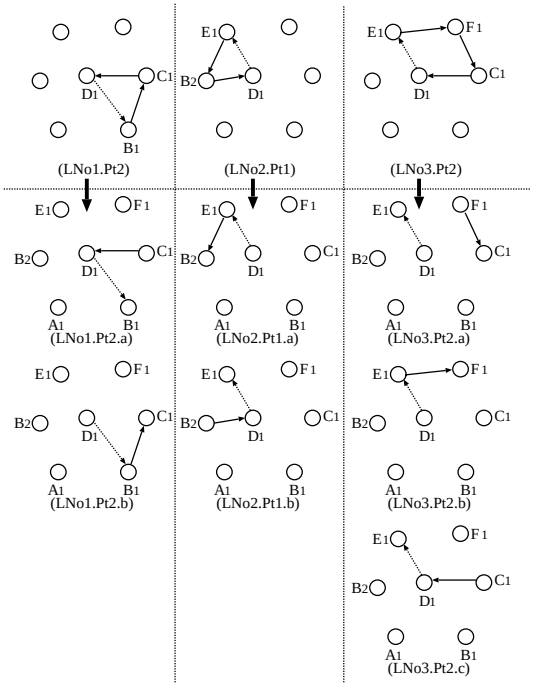


図 9 最小ループ構成の時間毎要求分割
Fig. 9 Sequential demand decomposition of the detected minimum loop.

表 6 実行順序テーブル
Table 6 The execution order table.

分割最小ループ構成	実行順序
LNo1.Pt2.a〜b	a → b
LNo2.Pt1.a〜b	b → a
LNo3.Pt2.a〜c	a → b → c

り付けられたラベルを順に並べ替えた昇順で行うものとする．つまり，a → b の次は b → a である．n 個の分割最小ループ構成がある場合，最良の場合では 1 回で求まるが，最悪の場合は n!回の組合せが生ずる．ただし，組合せ数はどのように処理したいかというユーザの要求（必ず先に処理したい要求や，この順序での処理を必須とする等の要求）によって減少するほか，要求に反する情報フローが発生する組合せ自体が少数であるため，実際の組合せ数は後述する条件によって限定される．例では，LNo1.Pt2 において a → b の実行順序を用いると，エンティティ D_1 へエンティティ C_1 の情報フローのみが発生し，エンティティ B_1 からの情報フローを阻止できるため，適切である．適切である実行順序が求まったら (S18) へ進むが，適切でない場合は (S17) で実行順序の組合せ変更を繰り返す．たとえば，LNo2.Pt1 において a → b の実行順序は，エンティティ E_1 の情報がエンティティ D_1 へフローする．したがって，機密要求の根となるエンティティに，機密要求の先のエンティティの情報がフローしない実行順序が求まるまで，組合せを変更する．

(S18) (S16) で選択対象となる分割最小ループ構成から現グラフを取り除き，(S16) から繰り返す．すべての分割最小ループ構成について処理を終えたら，終了する．

(S17) において分割最小ループ構成の組合せが限定される要因は，要求に反する情報フローを発生させる組合せが主に下記の条件の下で成り立ち，この条件に合致する組合せが少ないためである．

- ①機密要求辺の先のエンティティに処理要求が存在する分割最小ループ構成を，実行順序の最初としている．
- ②次に実行する分割最小ループ構成に，①の処理要求につながる処理要求を含む分割最小ループ構成を選択し，この構成が機密要求辺の根となるエンティティまで続いている．

たとえば，①の条件には LNo3.Pt2.b が該当する．このとき，②を満たす実行順序は，LNo3.Pt2.a→LNo3.Pt2.c だけである．この実行順序のうちどれか 1 つが変わっただけでも，要求に反する情報フローは発生しない．このため，実際には n!回の組合せを行うことなく，実行順序を求めることが可能である．

3.7 全実行順序の決定

分割最小ループ構成内における実行順序が決定した後，各最小ループ構成をどの順序で実施するか決定する．この処理は，各 LNo における最小ループ構成において，他の最小ループ構成と処理要求が共有されている場合に，想定外の順序で要求が実施されてしまうことを避けるものである．例においては図 8 に示されるように，計 3 つの最小ループ構成が存在するため，LNo1 → LNo2 → LNo.3 ··· LNo3 → LNo2 → LNo1 までの 6 通りが考えられる．n 個の最小ループ構成がある場合，最良の場合では 1 回で求まるが，最悪の場合は n!回の組合せが生ずる．ただし，(S17) と同様の理由によって組合せ回数の減少が生じるため，実際の組合せ数は限定される．以下に手順を述べる．

(S19) 各最小ループ構成において，他の最小ループ構成とどの処理要求が共有されているかを調べる．例では，エンティティ C_1 から D_1 への処理要求が，複数の最小ループ構成に含まれている．図 9 より，これらの要求を含む最小ループ構成を選び出し，要求の共有テーブルを作成する．**表 7** に例を用いた場合を示す．

この表では，ある最小ループ構成を実行したことが，他のどの分割最小ループ構成を実行したことと

表 7 要求の共有テーブル
Table 7 The shared-demand table.

最小ループ構成	共有されている分割最小ループ構成
LNo1.Pt2	LNo3.Pt2.c
LNo3.Pt2	LNo1.Pt2.a

表 8 共有された要求を含む実行順序の決定
Table 8 Decision of the execution order including shared demand.

最小ループ構成	実行順序
LNo1.Pt2	a → b
LNo2.Pt1	b → a
LNo3.Pt2	(c) → a → b → c

同じになるかを表している．たとえば，最小ループ構成 LNo1.Pt2 を実行すると分割最小ループ構成 LNo3.Pt2.c を実行したことと同じとなり，最小ループ構成 LNo1 から LNo3 まで順に実行した場合は，LNo3 の実行前にすでに，分割最小ループ構成 LNo3.Pt2.c が実行済みであることになる．

そのため，最小ループ構成の処理要求が他の最小ループ構成と共有されている場合には，先に実行した最小ループ構成によって，後に実行する最小ループ構成が影響を受けないようにしなければならない．**表 8** は，最小ループ構成 LNo1 および LNo2 の実行によって LNo3 の分割最小ループ構成である LNo3.Pt2.c が実行されたことと同じ状態であるため，LNo3 は表 8 のようになる．括弧で示された最小ループ構成は，先に実行された最小ループ構成によって，実施済みとなっている分割最小ループ構成を示す．

例においては LNo3 の実行前にこれらの最小ループ構成が実行されていても，問題は生じない．しかし支障がある場合には，最小ループ構成の実行順序を変更し，問題のない組合せになるまで組合せを変更する．

4. むすび

処理要求と機密要求が存在する状況において，それらの要求を矛盾なく実現する階層レベルを割り付ける方法として，SLA アルゴリズムが知られている．また，SLA アルゴリズムをベースとする拡張 SLA アルゴリズムは自由度と呼ばれる階層レベルの変更可能範囲を求めることにより，階層レベルの割付けに柔軟性を持たせ，より現実の業務に適用しやすいレベル設定を行うことができる．

これに対して本稿では，SLA アルゴリズムで処理の不可能であった矛盾する要求の解決方法を提案している．SLA アルゴリズムにおいて認められていなかった矛盾を含む要求は，矛盾部分の時間分割と実行順序を考慮することによって，その要求を実現できる可能性を生じる．SLA 時間分割アルゴリズムは矛盾を含む要求部分の抽出を行い，これを分割した状況を元に，矛盾を解決したうえで業務に支障の生じにくい階層レベル設定の実行順序を導出することを可能とした．

以上の SLA 時間分割アルゴリズムの実施により，統合要求グラフにおいて矛盾を含む要求が与えられた場合においても，これに適合する権限実行順序が導出可能であることを示した．

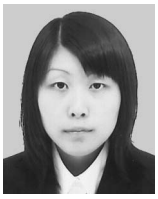
参考文献

1) 永瀬 宏，井上清一，四七秀貴：階層的セキュリティレベルの自由度を用いたアクセス権設定法，情報処理学会論文誌，Vol.41, No.8, pp.2255–2263 (2000).
2) Araki, T., Morizumi, T., Nagase, H., Takenaka, T. and Yamashita, K.: Security Level Assignment By Graph Analysis, *IEICE Trans. Issues on Cryptography and Information Security*, Vol.74, No.8, pp.2166–2175 (1991).
3) 安藤敏彦，加藤 靖，高橋 薫，野口正一：時制論理に基づくプロトコルの LOTOS 仕様の合成，情報処理学会論文誌，Vol.34, No.6-007 (1993).
4) 本位田真一，大須賀昭彦，内平直志：代数的仕様と時制論理によるリアルタイム SA とオブジェクト指向設計の融合手法，情報処理学会論文誌，Vol.33, No.2-008 (1992).
5) Bell, D.E. and LaPadula, L.J.: Secure Computer System: Unified Exposition and Multics Interpretation, Technical Report, MTR-2997 (available as NTIS AD-A023588), MITRECorp. (1976).
6) Denning D.E.R.: *Cryptography and Data Security*, Addison-Wesley, Reading, Massachusetts (1982). 上園忠弘，小嶋 格，奥島晶子（訳）：暗号とデータセキュリティ，培風館 (1988).

(平成 14 年 11 月 28 日受付)
(平成 15 年 6 月 3 日採録)

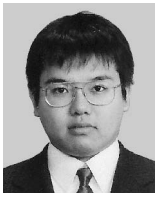
井上 清一（正会員）

平成 10 年金沢工業大学工学部情報工学科卒業．平成 12 年同大学大学院工学研究科修士課程（情報工学）修了．平成 15 年同大学院博士課程単位取得退学．現在，セキュリティファイルシステムの研究に従事．



中盛 友紀

金沢工業大学工学部情報工学科在学中．現在，セキュリティファイルシステムの研究に従事．



下川 徳之（正会員）

平成 13 年金沢工業大学工学部情報工学科卒業．平成 15 年同大学大学院工学研究科修士課程（情報工学）修了．

永瀬 宏（正会員）

昭和 49 年慶應義塾大学理工学部計測工学科卒業．昭和 54 年同大学大学院工学研究科博士課程（電気工学）修了．同年，日本電信電話公社に入社，ATR 通信システム研究所主任研究員を経て，平成 5 年金沢工業大学工学部情報工学科教授就任．以来情報セキュリティ，計算機アーキテクチャの研究に従事．