平成23年度 卒業論文

論文題目

推論による情報漏えい防止のための ハイパーグラフによる依存関係のモデル化とアルゴリズム

神奈川大学大学院 工学研究科 電気電子情報工学専攻

学籍番号　201070087

鈴木　遼

指導担当者　木下宏揚　教授

目 次

図 目 次

表 目 次

第1章

序論

昨今, インターネット技術の発展に伴い, 人類が創出する情報量は爆発的に増加している．また, mixi や twitter などの SNS やクラウドサービスの台頭によりそれらの情報へのアクセスおよび解析が容易になった．

そうした背景のもと，ライフログと呼ばれる個々人のディジタル活動記録の有効活用に注目が集まっている．一方で，ライフログはプライバシー情報，個人情報などを直接あるいは間接的に含んでおり，ライフログの有効活用だけに注目するのではなく，情報漏えいの問題にも注意を払わなければならない．

このような情報爆発時代においては, 膨大な情報を高速に解析する攻撃に対応できる新しい情報漏えい対策が必要となる．

従来の主な情報漏えい対策の目的は, 個人情報や企業機密のような秘密情報がそのまま丸ごと漏えいしないようにすることであった．例えば, 誰がどの情報にアクセス可能かどうかを規定するアクセス制御リストによって情報漏えいの危険を監視するためのモデルやシステム, アルゴリズムなどが研究されている [8, 10, 12, 13].

しかし, 一つ一つの情報それ自体は秘密情報でなかったとしても，それらが複数集まり何らかの推論を施すことによって, 秘密情報を抽出できてしまうこともある．例えば, twitter は「今から○○へ行ってきます」「到着なう」のように, それ自体は秘密ではない情報を時刻情報と共に発信する場であるが, 過去にその人物が発信した全ての出発・到着情報と時刻を twitterAPI などを用いて自動取得しそれをもとに推論解析することで発信者の住所の詳細な部分的情報を抽出できる可能性がある．また, SNS では知人同士のつながりのようなコミュニティのリンクをたどることが可能なため, ある人物の本名や所属等の個人情報が特定されると, その情報を元に, その知人の

本名や所属までもが推論によって暴かれてしまう可能性がある．

このような推論解析攻撃に対抗するためには膨大な情報群とその間にある推論関係を常に監視し何らかの問題を未然に検知して警告するようなシステムが必須である．しかし, そのためには情報間の推論的依存関係を記述するモデルが必要である．そこで本論文では, ハイパーグラフを用いたモデル化を提案し, そのモデル上で考えるべき課題を提起する．

本論文の構成は次のとおりである．第 2 節では, covert channel と呼ばれる ”隠れた情報経路 ”による情報漏えいをアクセス制御リストを用いて監視するモデルを紹介し, そのモデルだけでは推論による情報漏えいを防げないことを示す．また, 実際にどうやって推論による情報が漏えいが発生するかを示す. そして, 複数のデータ構造からグラフを選択したのかを示す. 第 3 節では, グラフの頂点彩色について紹介したのち, 推論的依存関係の有向グラフによるモデル化を試みる．また, 推論による情報漏えい対策のためにアクセス制御リストの修正が必要な場合があることを示し, 今後の課題とする．第 4 節では, 有向グラフでは表現できない依存関係を有向ハイパーグラフを用いることでモデル化することを提案と推論を考慮したグラフに対して安全なリスト着色をすることが出来ない ACL を判別するアルゴリズムについて提案する.

第2章

基礎知識

2.1　アクセス行列

アクセス行列とは主体 (Subject) と客体 (Object) の関係を表した行列のことで，主体と客体の関係にはR(Read:読み込み可), W(Write:書き込み可能),RW(Read+Write:読み書き可能), φ (Phi:読み書き不可) の 4 種類の権限がある.

	S1	S2
O1	ϕ	W
O2	R	R

図 2.1　アクセス行列

2.1.1 主体（Subject）

主体とはネットワークやデータベース内で管理されている客体にアクセスする行為者であり, ユーザーに相当する.

- 名前……主体の名前
- 競合……管理しているコミュニティの情報
- 階層……コミュニティで指定されたセキュリティレベル
- 役割……客体の権限を決定する役割

2.1.2 客体（Object）

客体はネットワークやデータベース内で管理されている情報であり, ファイルに相当する.

- 名前……客体の名前
- 競合……管理している主体のコミュニティの情報
- 階層……コミュニティで指定されたセキュリティレベル
- 所有……管理している主体の情報
- プライベート……管理している主体の情報

2.1.3 コミュニティ（Community）

コミュニティとは, コミュニティの属性, コミュニティに属する主体, および, コミュニティが管理する客体とその属性の集まりから成る社会システムに相当する. コミュニティ(Community) 同士には利害関係があり, 管理している主体には組織的に階層レベルや役割を割り振られる. コミュニティにも様々な種類があるが, インターネット上のコミュニティを考えてみると主体の役割や利害関係, 或いはプライベートな情報（個人情報）が複雑に絡み合っている. 現在, 求められているのはこのように複雑に絡み合ったコミュニティにおいて実現するセキュリティモデルである. それが実現されたのが Community Based Access Control Model である.

2.2 Covert Channel

2.2.1 間接情報フロー

Covert Channel とはアクセス行列において, 本来客体（Object:データやそれを含む情報）に直接アクセスする権限（Permission:アクセス権）がない主体（Subject:利用者, ユーザー）なのにも関わらず, アクセス権限を持つ第三者の力を借りて間接的にその客体できるようになってしまう. その時に発生する客体に対しての主体へのアクセス権限が矛盾した不正な経路を Covert Channel という. またこれを間接情報フロー [2] と呼ぶ. 以下の流れが例である. 初期状態 S2 は直接 O1 の情報を読み込むことができないが以下の流れで読むことが出来てしまう.

1. S1（Subject）が O1（Object）を読み込む
2. S1 が O1 で読み込んだ情報を O2（Object）に書き込む
3. S2（Subject）が O2 を読み込む
4. 発生した Covert Channel より読めないはずの O1 の情報を S2 が読める

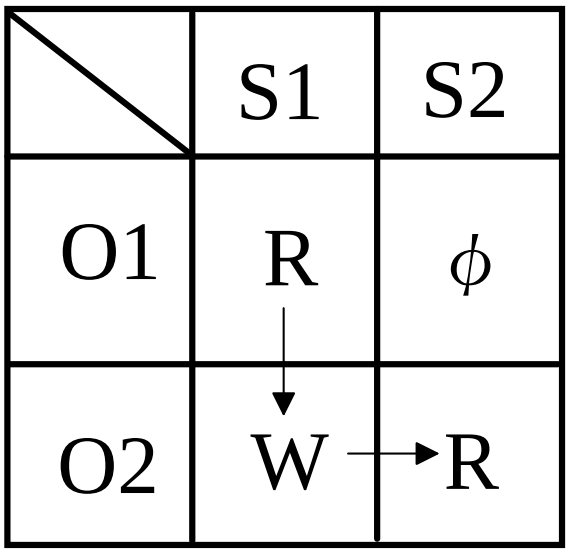

図 2.2 Covert Channel の例

このような流れで不正な情報流出が発生してしまうためアクセス制御を行う推論エンジンとしては出来る限りこれが発生するのを防ぐ検出と訂正を的確に行えるようにするのが情報フィルタに必要とされる機能である.

2.2.2 実際に発生する Covert Channel

不正な情報経路である Covert Channel を全て塞いでしまえば安全なシステムを構築することが出来るように見えるが, 単独では隠れチャンネル (Covert Channel) が存在しないようなコンピュータでもネットワークに接続されたコンピュータ群が協調することによって, 隠れチャンネルを構成できてしまう. つまり, 単独では安全なコンピュータでも, それがネットワークを構成すると安全ではなくなるような状況が簡単に存在し得るのである. このようなネットワーク構成機能の問題点が Covert Channel で利用される. 例えば以下のような例が挙げられる.

- 会社の機密データを社外へ持ち出したり, 社外の人間（社外の PC）でも見れるようにする
- mixi[4] 等の SNS の個人データが掲示板やブログ等不特定多数へ流出
- スパイウェア等, 個人 PC から情報を持ち出すためにこれを用いて通信を行い, 検知を困難とする

WWW 等, 不特定大多数が利用するネットワークでは意図しなくても Covert Channel が発生してしまう恐れがあるのでそういった情報網では比較的安易に情報漏洩が起こりうる. このように Covert Channel は今のネットワーク社会にとって情報を安易に流出させてしまう存在なのである.

2.2.3 Community Based Access Control Model

Community Based Access Control Model [1] は Covert Channel の制御を実現するためのセキュリティモデルの１つである.Access Control Agent System がこのモデルには組み込まれていて, コミュニティを用いた Covert Channel 分析が行なえる. まずユーザ数とファイル数を抑制し, 整理するために共通する属性を持った小規模なユーザの集合を Community と定義する. 各コミュニティではそれぞれ内部で Covert Channel 分析を行い,Covert Channel がおきないように制御する. 外部コミュニティと通信した場合の Covert Channel 分析は自コミュニティ, アクセス要求者, その要求について全て分析する.Covert Channel 分析は,Subject や Object の関係を以下のようなアクセス行列で表現し,Covert Channel の検出をする. 属性はアクセス制御や Covert Channel との関連性から競合, 所有, 階層, 役割, プライベートの５つを使用する. このとき, アクセス行列

から図1 のような2 × 2 行列のパターンを全て取り出し,Covert Channel 分析を行なう. このやり方により,2 × 2 行列全てのパターンだけでなくそれらを組み合わせることで3 × 3 やそれ以上の場合も Covert Channel を全て検出することが出来る. このようにこのモデルは Covert Channel を防ぐのに元々適したモデルであるので, この推論機能をベースとして改良し, 新たな処理のアクセス制御を行なう.

2.2.4 セキュリティモデル

セキュリティモデル [1] は, アクセス制御システムを構築する上で, セキュリティポリシーを具体的な論理的形式で表現したものである. そこには制御したいサービスや組織構造が反映される. 最も単純な型では,permission(read,write, ¬ read, ¬ write) であり,Subject（主体）,Object（客体）を含めた３つでアクセストリプルと呼び, それをシステムで如何扱うかによってアクセス制御が行われる. 従来のセキュリティモデルには様々な種類があり, 使用される状況に応じて使い分けたり, 複数使用する等しているが今回の実験ではアクセストリプルの構造や Covert Channel の検出・訂正を考慮したモデルとして Community Based Access Control Model を改良する. このモデルには幾つかのモデルには無い機能を持ち, 他のモデルと併用することで Covert Channel の検出機能及び今回追加する訂正機能を実行することができる.

2.2.5 情報フィルタ

情報フィルタとはCovert Channel検出時にそのCovert Channelが無くなるように特定の権限を変更することである.情報フィルタには4種類の方法があり,それぞれ一長一短がある.3種類はフロー経路の権限を禁止して遮断するのに対し,Read権限を許可する方法は情報共有の拡大の意味を持つ.以前は読めなかった客体が修正により普通に読めるようになれば,不正経路ではなくなるのでCovert Channel自体は無くすことができる.情報フィルタの具体的な処理を以下にまとめていく.図のようにCovert Channelが発生して検出された場合,以下,図2.3の(a)(b)(c)(d)のいずれかを適用すればCovert Channelが解消される.

1. (S1,O1)のREAD権限を削除
2. (S1,O2)のWRITE権限を削除
3. (S2,O1)にREAD権限を添付
4. (S2,O2)のREAD権限を削除上記のどの情報フィルタを選択するかは各コミュニティのセキュリティポリシーや主体のアクセス履歴,ユーザがどういう方針で処理するか定めるユーザーポリシーを考慮して決定するが,(a)から(d)のどの場合でもCovert Channelは訂正できる.

	S1	S2
O1	ϕ	R
O2	ϕ	RW

(a)

	S1	S2
O1	ϕ	ϕ
O2	R	RW

(b)

	S1	S2
O1	ϕ	R
O2	R	W

(c)

	S1	S2
O1	R	R
O2	R	RW

(d)

図 2.3 情報フィルタ

2.3 推論

ここでは，まず Prolog に関する基礎知識として記号論理について述べる. そして，Prolog について解説し，Community Based Access Control で用いられる推論について述べる.

2.3.1 命題論理

(propositional logic) である. 1 つの命題 (proposition) は真 (true) または偽 (false) のどちらかの値を持つ. 命題とは, 例えば「太郎は父親である」といったような事実を表したものであると考えればよい. 1 つの命題を 1 つの記号, たとえば英文字の p によって代用するものとする. 1 つまたは複数の命題を論理演算子 (logical opertor) で結合してできる式を論理式 (logical formula) という.A,B をそれぞれ論理式とすると,. 以下のような性質を表すことが出来る.

- 否定 (negation) ¬ A (not A, ∼ A)
- 論理積 (conjunction) A ∧ B (A and B)
- 論理和 (disjunction) A ∨ B (A or B)
- 含意 (implication) A ⊃ B (A → B, A ⇒ B)

これらは論理式と論理式の関係であるが，これら自体も論理式である.

2.3.2 述語論理

述語 (predicate) とは, 例えば「鳥は空を飛ぶ」のように何かの関係を表現するものである. 述語は関数のように扱うことができ，引数をとれる. 引数としてとるものを項 (term) と言う．項の個数が n だとすれば，その述語は n 項述語であると言える. 例えば ”Taro drinks water. ” という文は，

- drinks(Taro, water)

という式で表せる．この中で drinks は 2 項述語, Taro と water は項である．変数 (variable) も項である. このような式を原子論理式 (atomic formula) という．また，述語論理においては，例えば，ある特定の固体 x のもつ述語的性質 P(x) が真であるかどうかを聞くだけでなく，集合 D のすべての要素 x について P(x) が成立するかどうかを聞くことが出来る. これを，

- ∀ xP(x)

と書く. これは集合 D の ”すべての x について，その x は性質 P を持つ ”という命題である. ∀ のことを全称記号（universal quantifier）という. この論理式の真偽を決めることは，定義域 D が有限の場合には可能であるが，D が無限の場合には困難な問題となる.「すべての人間には親がある」の論理表現については以下のようになる.

- ∀ x(Man(x) → Parent(x))

となる. そして，”すべての ”という概念の対になる ”少なくとも一つは存在する ”という概念もある．これは，∃ を用いて表現され，存在記号（existential quantifier）と言う.「愛する事が出来る人がいる」というのは，

- ∃ x Love(x)

と表すことができる. ∀，∃ の両者を量記号，または限量子（quantifier）と言う. 上式はある固体 a ∈ D であって，Love(a) が真であるということであり，その場合に ∃ x Love(x) は真となる.

2.3.3 Horn 論理

Prolog については，後で述べるが，述語論理の特殊形である Horn 論理 (Horn logic) について簡単に触れる. Prolog は述語論理の特殊形である Horn 論理に基づいている.Horn 論理では, 扱う論理式をつぎのような形式のものに限定する.

- p1 ∧ p2 ∧ ⋯ ∧ pn ⊃ q

ここで p1 , p2 , ⋯ , pn , q は原子論理式または原子論理式に否定がついたものでありリテラル (literal) と呼ぶ. このような論理式をホーン節 (Horn clause) という. Prolog のプログラムはホーン節の集合からなる. 節のホーン節を Prolog では, 次のように記述する.

- q :- p1 , p2 , ⋯ , pn.

”:- ”は右辺を前提, 左辺を結論とする含意を表す 2 項演算子である. 左辺を頭部 (head) , 右辺を本体 (body) と呼ぶ. 1 つの節は必ずピリオド ”. ”で終る. 頭部あるいは本体を省略した節は特殊な意味をもつ.

- 規則 (rule) q :- p1 , p2 , ⋯ , pn.
- 事実 (fact) q.
- 目標 (goal) :- p1 , p2 , ⋯ , pn.

規則は， p1 かつ p2 かつ... かつ pn ならば q，事実は q は事実であるという事を表し，目標は p1 かつ p2 かつ... かつ pn でないかどうか調べよという事を意味している. 「事実」は頭部のみからなる節であり, このときは ”:- ”を省略する. 一般に Prolog のプログラムは, 複数の規則と複数の事実, そして 1 つの目標から構成され, 実行は, 与えられた規則と事実から目標が導けるかどうかを証明するという形で行なわれる.

2.3.4 Prolog

Prolog とは，論理学を基にして作られたプログラミング言語であり，述語表記されたものを処理させる事が出来る．Prolog は，物事の間の関係（属性）を定義していく事で，物事を抽象化し，問い合わせをする事により，結論を導きだす．この物事の関係を事実と言う．例えば「正弘は太郎の父親である」という関係があり，prolog で表すと以下のようになる.

father.pl 3

```
isFatherOf(masahiro,taro).
```

ここで，以下のような関係を定義する．

parent.pl 3

```
male(masahiro).
male(taro).
female(hanako).
isFatherOf(masahiro,taro).
isMotherOf(hanako,taro).
isParentOf( X, Y ) :- isFatherOf( X, Y ).
isParentOf( X, Y ) :- isMotherOf( X, Y ).
```

このような関係が与えられた時に,「太郎の父親は誰か」という質問をしてみる．すると以下のような結果になる．

output of father.pl

```
1 ?- isFatherOf(X,taro).
X = masahiro ;
No
```

Prolog の変数というのは，何が入っているのかは決まっていない．Prolog は質問に対して答える時に，その質問に一致する事実があるかどうかを調べる．このことをパターンマッチングといい，これにより変数の値が決まる．この場合，”isFatherOf(X,taro). ”が質問になる．この意味は，taro の父親に当てはまる X があればそれを出力するということである．つまり，Prolog はパターンマッチングする X を探す．parent.pl の ”isFatherOf(masahiro,taro). ”で「taro の父親は masahiro」であることが定義されいてるため, ”X=masahiro ; ”が結果として返る．この X を自由変数といい，この自由変数にデータが代入される．次は，二つの変数 X,Y を用いた時の例を示す．

output of father.pl

```
1 ?- isParentOf(X,Y).
X = masahiro Y = taro ;
X = hanako Y = taro ;
No
```

質問は ”isParentOf(X,Y). ”であり，親子関係がある X,Y を出力させるというものである．まず，X について当てはまるのは ”isParentOf(X, Y) :- isFatherOf(X, Y). ” より masahiro であり，X が masahiro の時にパターンマッチングするのは taro である．そして，もう一つの関係である ”isParentOf(X, Y) :- isMotherOf(- X, Y). ”より，X が hanako のときパターンマッチングするのが，taro なので，この二つが結果として返される．

2.3.5 Community Based Access Control に於ける推論

Community Based Access Control で実現される推論機能とは，情報フィルタに組み込まれる推論エンジンによって実現される．推論エンジンは，Subject 及び Object に属性を付与し，演繹推論規則を用いることでアクセスルールと照合し権限を決めるという機能を持つ．つまり，まだ権限の決められていないアクセス行列中のブランクの権限を決める事が出来る．また，異なった属性から推論された権限の矛盾や重複の訂正，すでに権限が与えられている場合の権限の重複や矛盾も検出し訂正する事が出来る．この場合の訂正は以下の 3 つ方法が考えられる．

- 競合・プライベート・階層属性を優先する
- ユーザ側に問い合わせる
- 決められたセキュリティポリシーに適した権限を矛盾した場合に自動推論

今回は，競合・プライベート・階層属性を優先にする事で実現している．これは，Covert Channel を検出を目的としたセキュリティポリシーにしているためである．これにより，Community Based Access Control で，問題にしている Covert Channel の検出も可能とする．

2.4 推論による情報漏えい

一つ一つの情報それ自体は秘密情報でなかったとしても，それらが複数集まり何らかの推論を施すことによって, 秘密情報を抽出できてしまうことがある．そのような攻撃を推論攻撃と呼ぶ．推論攻撃への対策はデータベースのセキュリティ課題として研究されてきた．推論攻撃の一種として, 情報間の統計的な関連に注目した研究がある [8]．データベースに蓄積されているデータは，それぞれが無関係に独立に存在しているわけではなく，統計的あるいは意味的に関連している場合が多い. 一つ一つの情報それ自体は秘密情報でなかったとしても，それらが複数集まり何らかの推論を施すことによって, 秘密情報を抽出できてしまうことがある．そのような攻撃を推論攻撃と呼ぶ．例えば, ある発言者が同じ地名，若しくは駅名などの単語を頻繁に発言している場合，この地名や駅名と発言者と間に何らかの関連があることが統計的に分析できてしまう.

推論は統計的手法以外にも様々なものがありそれら全ての推論解析攻撃に対抗するためには膨大な情報群とその間にある推論関係を常に監視し何らかの問題を未然に検知して警告するようなシステムが必須である．しかし, そのためには情報間の推論的依存関係を記述するモデルが必要である.

どのような推論手法であっても, いくつかの情報からある情報を導くことに変わりは無い．その依存関係をモデル化できれば, 推論手法によらない対策を考えることができるかもしれない．図.2.4 はあるオブジェクト集合におけるオブジェクト間の依存関係を洗い出してリスト化したものである.

推論元オブジェクト	推論	導出オブジェクト
O1,O2,O3	⇒	O4
O4,O6	⇒	O5
O3,O6	⇒	O8
O6,O8	⇒	O7
O4	⇒	O6

図 2.4 object 間の依存関係リスト

リストの 2 行目は, オブジェクト O1,O2,O3 とある推論によってオブジェクト O4 が

導出されてしまうことを意味している．このようなリストを作ることそれ自体も重要で難しい問題であるが, 本研究の目的は, このようなリストが与えられたときに, そのリストを解析し covert channel を検知するために必要なモデルを考察することである．

ACL は直接的にオブジェクトを読み書きできるかどうかだけを表したリストであるから, 仮に ACL 上では covert channel が無かったとしても, 推論によって ACL に反する情報アクセスが可能かもしれない．即ち, ACL と図.2.4 のような依存関係リストを合わせて初めて発覚する covert channel があり得るということである．第 3 章では, ACL と依存関係リストを同時にグラフ表現をする．

2.4.1 ライフログ

ライフログという言葉は近年になって使われだした新しい言葉であり，学術的な定義はまだなされていない．本稿では，ライフログとは「個々人がいつどこで誰と何をしたのか，どのような情報を受け取り，どのような情報を発信したのか，そういった“ 個人の軌跡 ”のうち，ディジタル情報として計算機上に記録されたもの」と定義する．

ライフログを解析し個人の傾向を分析することで個人個人の好みに合わせたサービスを提供したり，ある集団の共通点をデータマイニングの手法で抽出するといったことが可能となる．例えば，過去に購入した書籍からおすすめの書籍を画面上に表示したり，ある病気を患っている人たちの生活習慣から病気の原因となる悪習慣を抽出するなど，様々な活用が考えられる．個人の過去の行動から嗜好を分析して広告を表示する手法は行動ターゲティング広告 (behavior targeting advertisement) と呼ばれ，研究されている [21]．GPS やカメラ，センサを備えたウェアラブルな機器を用いてライフログを自動的に記録するための手法も研究されている [19, 20, 18]．Twitter のフォロワー数などから個人の“ 情報発信者としての信頼度 ”を割り出そうとする研究もある [23, 24].

そうしたライフログの活用可能性に注目が高まる一方で，ライフログの取り扱い方によっては，プライバシーの侵害や，個人情報・企業機密情報の漏えいが生じる危険性が指摘されている [22, 26, 27, 25]．例えば，携帯端末の GPS 機能によって記録された移動ログをマーケティングの参考にしようとする場合には，誰の移動ログなのかを伏せた状態でプライバシーを保護しながらデータマイニングをしなければならない．別の例として，A 社が収集した書籍購入履歴と B 社が収集した書籍購入履歴

を互いに共有する際には，個人情報を除いた履歴のみを共有するべきであるが，仮にそういう配慮をしたとしても，A 社が所持しているC さんの履歴を分析することで得られたC さんの嗜好と，B 社が所持していたとある匿名者の履歴を分析することで得られた嗜好が酷似していれば，その匿名者がC さんである確率が高いことをA 社は知ることができ，C さんがB 社でどのような書籍を買ったのかがA 社に漏洩してしまうだろう．

このようなタイプの情報漏えいを防ぎつつ，データマイニングを行おうとする研究分野として，“ プライバシー保護データマイニング (Privacy Preserving Data Mining) ” がある [15, 16, 17, 14]．文献 [14] によるプライバシー保護データマイニングの目的を以下に引用する．

> プライバシー保護データマイニングのゴールは，そのような消費者・小売業者・納入業者のどちらにとっても有利な状況を可能にすることである．すなわち，データ内の知識は活用のため引き出され，個人のプライバシーは保護され，データの所有者はデータの乱用や漏洩から守られる． [[14], P.4]

同文献にはプライバシー保護データマイニングの手法や未解決課題の考察などがまとめられている．

本研究は情報漏えい防止のみを目的としているため，データマイニングと情報漏えい防止を両立させようとするプライバシー保護データマイニングとは立場が異なるが，後に述べる提案モデル上で定義された安全性を保ちつつデータマイニングを行うといった応用研究への発展が期待できる．

2.4.2 ライフログのアクセス制御

ライフログは，日記のように自己に関する事象や人間関係を含む外部との相互作用を自らが記録する「手動記録型」と，GPSやカメラ，各種センサなどウェアラブルな計測機器や監視カメラ，アクセスログなどにより自動的に記録される「自動記録型」に大別できる．

手動記録型ライフログには，ブログ等に書き込んだテキストや画像，音声，動画，プロフィールなどのコンテンツ，他人が発信したコンテンツへのマーキング（Facebookのいいね！ボタンなど），ネットショッピングサイトでのお気に入り登録などがある[28]．自動記録型ライフログには，GPSなどの位置情報や監視カメラの映像など被記録者の実環境での行動に関する情報，電子商取引の履歴，Web閲覧記録などネットワーク環境での行動に関する記録などがある[29]．また，これらに対するメタデータとして，ライフログのデータ間の同期やコンテキストなどに関する情報もライフログである．

ライフログの特徴として，直接の情報漏えいやプライバシー侵害にはならないようなライフログに関してはセキュリティ面が軽視され，被記録者本人が自身のライフログを安易に公開してしまいやすい点が挙げられる．しかしながら，ライフログはどんな情報であってもすべて個人にまつわる情報であり，１つ１つのログに関しては公開しても問題が無いように思えても，複数のログと推論規則から，重大なプライバシー侵害となりうる情報が導かれてしまう可能性がある．例えばtwitterで「おはよう」とつぶやいただけであっても，その時刻が昼過ぎであることから生活リズムが漏洩し，泥棒にとって進入しやすい時間帯を推論されてしまう危険がある．

このような問題は，暗号理論的な手法では対策が難しい．なぜなら，この種の情報漏えいはある意味で被記録者本人のミスが原因であり，極端に言えば秘密鍵をうっかり公開してしまうのを防ぐ問題と同種だからである．そこで，本研究ではこの問題をアクセス制御の問題として捉えて対策する道を模索する．従来の情報漏えい対策と異なるのは，被記録者の１度のミスで全てが漏れるわけではないという点である．複数のログが漏れて初めて推論によって重大な情報が漏洩するということは，そうなる前の段階で対策をする時間的余裕があるということである．

この問題はライフログに限らず，推論規則が存在するようなあらゆるリソース空間における問題でもある．本研究ではアクセス制御の問題としてより一般化して議論するために，次節以降ではライフログを含むあらゆる情報リソースをオブジェク

トと呼び，抽象的に扱い議論を進める．前述のようにライフログには手動記録型と自動記録型があり，さらにそれぞれ多様な形式で記録されており，データの種類や大きさも様々であるが，アクセス制御においてはそれらのデータを識別さえできればデータそのものを読み込んで処理する必要は無く，オブジェクトとしての識別子を付加しておいて識別子とアクセス権限のリストだけを読み込んで管理すれば済むため，アクセス制御モデルはどのようなライフログに対しても適用可能である．

2.5 データ構造

与えられたデータを, 計算の高速化のために構造をもたせて記憶する手法の総称. 目的に合わせ, あるいくつかの機能を, 高速, あるいは省メモリで行えるよう設計される. 例えば, ある言葉を n 個の言葉の辞書データから検索するとき, 通常の配列では, 最悪で挿入・削除に $O(n)$, 検索に $O(\log n)$ か, 挿入と削除に $O(1)$, 検索に $O(n)$ の時間を要する. 二分探索木というデータ構造は, これらの機能を $O(\log n)$ 時間で行い, 使用メモリは $O(n)$ である.[5] 主に扱われるなデータ構造は以下の通りである.

- 連結リスト
- 配列
- スタック
- キュー
- 木構造
- ハッシュテーブル
- ルックアップテーブル
- グラフ

2.5.1 連結リスト

連結リスト（れんけつリスト, 英: Linked list）は, 最も基本的なデータ構造の一つであり, 他のデータ構造の実装に使われる. リンクリスト, リンクトリストとも表記される.

一連のノードが, 任意のデータフィールド群を持ち,1 つか 2 つの参照（リンク）により次（および前）のノードを指している. 連結リストの主な利点は, リスト上のノードを様々な順番で検索可能な点である. 連結リストは自己参照型のデータ型であり, 同じデータ型の別のノードへのリンク（またはポインタ）を含んでいる. 連結リストは場所が分かっていれば, ノードの挿入や削除を定数時間で行うことができる（場所を探すのにかかる時間はリスト上の順番の条件などにも依存するし, 後述する片方向リ

ストなのか双方向リストなのかにも依存する）. 連結リストにはいくつかの種類があり, 片方向リスト, 双方向リスト, 線形リスト, 循環リストなどがある.

連結リストは多くのプログラミング言語で実装可能である.LISP や Scheme といった言語は組み込みでこのデータ構造を持っていて, 連結リストにアクセスするための操作も組み込まれている. 手続き型やオブジェクト指向型の言語（C 言語,C++,Java）では, 連結リストを作るには mutable（更新可能）な参照を必要とする.

2.5.2 配列

配列（はいれつ,Array）は, プログラミングにおけるデータ構造の一つ. 科学技術計算分野ではベクトルという場合もある.

配列なのはデータの集合であり, 添え字でインデックスされたものを指す. 古典的なプログラミング言語では同じデータ型の集合に限定されるが, 比較的新しい言語や多くの高水準言語では異なった型も格納することができる. 通常, 変数には 1 つの値しか格納できない. しかし, ときには, ある関係を持つ複数の値を格納できる変数があると都合の良いことがある. その場合に用いられるのが配列である. 例えば,6 人の生徒の平均点を計算するプログラムを書くとする. それぞれの生徒の点数を格納する変数は, 愚直に考えれば, 次の C 言語で書かれた例のように個別に宣言することになるだろう.[3]

2.5.3 スタック

スタックは, ノード（何らかのデータを持ち, 別のノードを指し示すことができる構造）のコンテナ（データを集めて格納する抽象データ型の総称）であり,2 つの基本操作プッシュ(push) とポップ (pop) を持つ. Push は指定されたノードをスタックの先頭（トップ）に追加し, 既存のノードはその下にそのまま置いておく. Pop はスタックの現在のトップのノードを外してそれを返す. よく使われる比喩として, 食堂にあるバネが仕込まれた台に皿や盆を積み重ねておく様子がある. そのようなスタックでは利用者は一番上（トップ）の皿だけにアクセスすることができ, それ以外の皿は隠されている. 新たに皿が追加される（Push される）と, その新しい皿がスタックのトップとなり, 下にある皿を隠してしまう. 皿をスタックから取る（Pop する）と, それを使うことができ, 二番目の皿がスタックのトップとなる. 二つの重要な原則がこの比喩で示され

ている. 第一は後入れ先出し (LIFO: Last In First Out) の原則である. 第二はスタックの中身が隠されているという点である. トップの皿だけが見えているため, 三番目の皿がどういうものかを見るには一番目と二番目の皿を取り除かなければならない.[3]

2.5.4 キュー

キュー (queue), あるいは待ち行列はコンピュータの基本的なデータ構造の一つ. データを先入れ先出し (FIFO: First In First Out) のリスト構造で保持するものである. キューからデータを取り出すときには, 先に入れられたデータから順に取り出される. キューにデータを入れることをエンキュー (Enqueue) , 取り出すことをデキュー (Dequeue) という. プリンタへの出力処理や, ウィンドウシステムのメッセージハンドラ, プロセスの管理など, データを入力された順番通りに処理する必要がある処理に用いられる.

キューの変形として, 先頭と末尾の両端から入出力を行えるものを両端キュー (double-ended queue, deque) という.

キューとは逆にLIFO（後入れ先出し）のリスト構造を持つデータバッファをスタックと呼ぶ.[3]

2.5.5 木構造

木構造は, ノード（節点, 頂点）とノード間を結ぶエッジ（枝, 辺）あるいはリンクで表される. ノードには何らかのデータ（値, 条件, 別のデータ構造, 別の木構造）が付属している. 木構造内の各ノードは, 0 個以上の子ノード（child nodes）を持ち, 子ノードは木構造内では下方に存在する（木構造の成長方向は下とするのが一般的である）. 子ノードを持つノードは, 子ノードから見れば親ノード（parentnode）である. 同じ親を持つノード同士を兄弟という. ノードは高々1 つの親ノードを持つ. 最底辺の子ノードから, あるノードまでのエッジ数を, そのノードの「高さ」という. （後述する）根ノードの「高さ」は, 木構造の「高さ」である. 逆に根ノードから最底辺に向かってのエッジ数を「深さ」という.

木構造の頂点にあるノードを根ノード（root node）と呼ぶ. 頂点にあるため, 根ノードは親ノードを持たない. 木構造に対する各種操作は, 一般に根ノードを出発点とす

る（アルゴリズムによっては, 葉ノードから開始して根ノードに到達して完了するものもある）. 他のノードにはエッジあるいはリンクを辿ることで必ず到達できる. 形式的定義では, そのような（根から特定ノードまでの）経路は常に一意である. 図で示す場合, 根ノードが一番上に描かれるのが普通である. ヒープなどの木構造では, 根ノードは特別な属性を持つ. 木構造内の全てのノードは, そのノードを頂点とする部分木の根ノードと見なすことができる.

木構造の最底辺にあるノードを葉ノード（leaf node）と呼ぶ. 最底辺にあるため, 葉ノードは子ノードを持たない.

内部ノード（internal node,inner node）は, 子ノードを持つノード, すなわち葉ノード以外のノードを意味する.[3]

2.5.6 ハッシュテーブル

ハッシュテーブルはキーをもとに生成されたハッシュ値を添え字とした配列である. 通常, 配列の添え字には非負整数しか扱えない. そこで, キーを要約する値であるハッシュ値を添え字として値を管理することで, 検索や追加を要素数によらず定数時間 O(1) で実現する. しかしハッシュ関数の選び方（例えば, 異なるキーから頻繁に同じハッシュ値が生成される場合）によっては, 性能が劣化して最悪の場合 O(n) となってしまう.[3]

2.5.7 ルックアップテーブル

ルックアップテーブル（Lookup table）とはコンピュータにおいて, 効率よく参照や変換をする目的でつくられた配列や連想配列などのデータ構造のことをいう. 例えば大きな負担がかかる処理をコンピュータに行わせる場合, あらかじめ先に計算できるデータは計算しておき, その値を配列（ルックアップテーブル）に保存しておく. コンピュータは配列から目的のデータを取り出すことによって, 計算の負担を軽減し効率よく処理を行うことができる. またあるキーワードを基にあるデータを取り出すとき, その対応を表としてまとめたものもルックアップテーブルといえる.[3]

2.5.8 グラフ

一般に, グラフは$G = (V, E)$と記述される．ここで, Vは頂点集合, EはVの2頂点対の集合である．$u, v \in V$, $(u, v) \in E$のとき, Gが無向グラフならば, $(v, u) \in E$かつ$(u, v) = (v, u)$であるが, Gが有向グラフのときはそうとは限らない．ある頂点$u \in V$から, ある頂点$v \in V$へ辺をたどって到達可能な時, その到達経路を(u, v)歩道といい, どの頂点も高々1回しかたどらない(u, v)歩道を(u, v)道と呼ぶ.

アクセス行列とcovert channelは有向グラフによってモデル化できる．グラフとは頂点集合とそれらを結びつける辺の二つで構成され,「点とそれを結ぶ線」の「つながり方」に着目して問題を考えるためのデータ構造である. つながり方だけではなく, 辺でつながれた2頂点の順序を考慮してその有向性を矢印で表現したものを特に有向グラフと呼ぶ．その場合, 有向性のないグラフは無向グラフと呼ばれ, 有向グラフと区別される．また全ての頂点間に辺が存在するグラフを完全グラフと呼ぶ. 有向グラフ$G = (V, E)$は頂点集合V，有向辺集合としてVの順序対の集合Eを持つ. ここでは自己ループと多重辺は持たないものとする. 有向辺(u, v)に対してuを始点, vを終点と呼ぶ. 有向グラフGの有向辺集合$A(G)$として$\{((u, v), (w, x)) \mid v = w, (u, v), (w, x) \in A(G)\}$を持つ有向グラフとして定義される. 有向グラフ$G$において, $S \subseteq V(G)$が独立集合であるとき，任意Sの2頂点間に有向辺が存在しないときを言う.

図.2.3のアクセス行列をグラフで表現したものを図.2.5に示す.

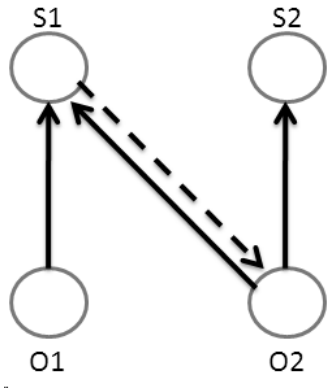

図 2.5 アクセス行列のグラフ表現

このグラフにおいて, 実線はread, 破線はwriteを意味する．グラフG上で, O_iから

S_i への直接の矢印が無い, 即ち, $(O_i, S_i) \notin E$ にも関わらず, O_i から S_i への (O_i, S_i) 歩道が存在するときその歩道は covert channel を意味する．したがって, グラフによってモデル化することで ACL 上の covert channel 検知問題に対してグラフの歩道発見アルゴリズムを適用できるようになる [13]．また, グラフ化することで人間に対して視覚的に危険を訴えることができる．

第3章

有向グラフと頂点着色によるモデル化

3.1　有向グラフと頂点着色によるモデル化

3.1.1　グラフの頂点着色

グラフ $G = (V, E)$ の頂点着色とは, グラフの頂点に色を塗ることである．即ち, ある色集合 C と写像 $c : V \to C$ を与えることである．特に, $(v, w) \in E$ であるようなどの 2 頂点 $v, w \in V$ についても $c(v) \neq c(w)$ となるとき c を頂点彩色と呼ぶ．色数 $|C|$ が制限されている時に頂点彩色可能かどうかを判定する問題や, 可能ならばその彩色を求める問題はグラフ理論において重要な問題として研究されており, さまざまなアルゴリズムが考案されている．一般に, ある条件 P を満たす頂点着色を P 着色と呼ぶことにする．

頂点着色は次のように一般化できる．グラフの頂点にあらかじめ色のリストが与えられているとする．即ち, 写像 $L : V \to 2^C$ が与えられているとする．このとき, G の頂点着色 c で, 全ての頂点 $v \in V$ に対して $c(v) \in L(v)$ を満たすものを G のリスト着色と呼ぶ．任意の頂点 $v \in V$ に対して $L(v) = C$ ならば通常の頂点着色である．c が頂点彩色のときは, 特にリスト彩色と呼ぶ．また, c が P 着色のときは, P リスト着色と呼ぶことにする．各頂点 v における色リストのサイズ $|L(v)|$ が制限されている時にリスト彩色可能かどうかを判定する問題や, 可能ならばその着色を求める問題が通常の頂点彩色と同様に広く研究されており, さまざまなアルゴリズムが考案されている．

3.1.2 推論による頂点着色のグラフ表現

まず, オブジェクト間の依存関係リストを次のようにグラフ化する．頂点集合Vをオブジェクト集合とし, 依存関係リスト上で$O_{i_1}, \cdots, O_{i_k} \in V$から$O_j \in V$が導出可能ならば有向辺$(O_{i_1}, O_j), \cdots, (O_{i_k}, O_j)$を描く．さらにそのグラフに対する色リストを用いてACLを次のように表現する．ただしここでは議論を簡単にするためにACLにおいてread可能か否かのみに着目する．まず, 色集合Cをサブジェクト集合とする．そして, ACL上で, あるサブジェクト$S_i \in C$があるオブジェクト$O_j \in V$をread可能ならば, $S_i \in L(O_j)$としO_jの色リストにS_iを加える．例として, 図.2.4をグラフで表現し, あるACLに従って色リストを与えたものを図.3.1に示す.

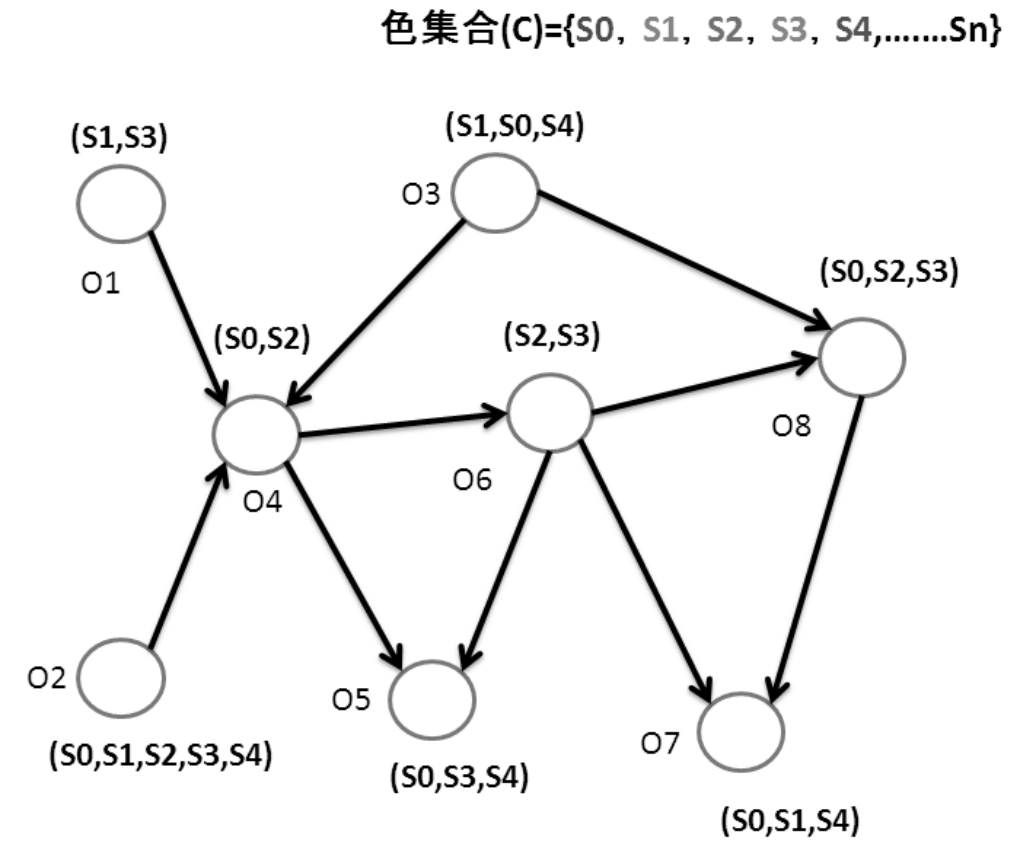

図 3.1 推論的依存関係とACLの有効グラフ表現

サブジェクトS_iがオブジェクトO_jをreadしたとき頂点O_jに色S_iを塗るとしよう. 例えば, 図.3.1においてO_1, O_2, O_3が全てS_1で塗られたとする．このとき, 推論によっ

てO_4が導出可能なことを考えればO_1, O_2, O_3が全てS_1で塗られた時点でO_4もS_1で塗るべきである．しかしその一方で, $S_1 \notin L(O_4)$, 即ち, ACL上ではS_1はO_4をreadできないことになっているので, これは推論による情報漏えいを意味している．このとき, その頂点着色はリスト着色の定義にも反していることに注目すると推論による情報漏えいに関する安全性を次のように定義できる．

定義：あるP着色がリスト着色ならばその着色は推論に対して安全であるという．ここで, $P =$「任意の頂点vに対して, vを終点とする全ての有向辺$(u_1, v), \ldots, (u_k, v)$の始点$u_i$が同一色で塗られているならば, $c(v) = c(u_i)$でなければならない.」とする．

このモデル上で考えるべき課題は状況に応じて様々であると思われるが, 本論文ではACLそのものに含まれる問題を見直すための初歩的な課題を提起する．

3.1.3 ACL修正問題

情報が実際にreadされるタイミングには時差がある．例えば, 前節の例では, S_1がO_1, O_2をreadしたときに, その後O_3が読み込まれたときの危険性を検知しS_1がO_3をreadする前にO_3のACLを修正し, $L(O_3)$からS_1を削除してしまえば良い．そのためには, オブジェクトの読み込み状況を常に監視し未来に起こり得るP着色を随時計算しそれが推論に対して安全かどうかを判定するアルゴリズムが必要である．その際に, どの段階で危険とみなすのかという問題もある．例えばS_1はO_1, O_2をreadした段階で自動的にO_3が読めなくなってしまう．そこで, S_1がO_1をreadした段階で残りはO_2, O_3のどちらかしかreadできないことを検知して警告したり, ACLを早い段階で柔軟に修正できるようになる．

また, そもそも最初からACLに問題があるケースもある．図3.2はO_6がS_2にreadされると, S_2はO_3をreadできなくなる．なぜなら, 推論によってS_2はO_8を導出できてしまうからである．するとO_3はS_1しか読めない．S_1がO_3をreadすると, 同様に, O_2はS_0しか読めなくなる．すると, O_4もS_0しか読めないので, もしS_0がO_2とO_4をreadすると, S_0は推論によってO_5を導出してしまう．よって, このグラフは推論に対して安全なP着色が存在しないので, ACLを見直さなければならない．このようなグラフが与えられたときに, ACLを見直すべきかどうかを判定するためには, 安全なP着色が存在するかどうかを判定するアルゴリズムが必要である．

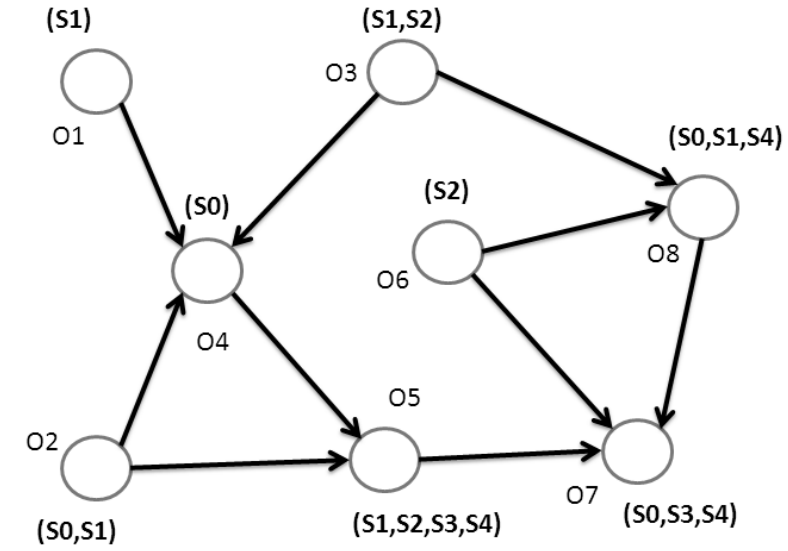

図 3.2 色が塗れない object があるグラフ構造

3.1.4 有向グラフ表現の問題点

有向グラフでは表現不可能な依存関係リストが存在する．例えば図3.3において, O_1, O_2 から, O_4 が導出でき, O_1, O_3 からも O_4 が導出できる．これを有向グラフで表現しようとすると O_1, O_2, O_3 から O_4 へ有向辺を描くことになり, しかしそれでは O_1, O_2, O_3 から O_4 が導出できるという意味になってしまう.

推論元オブジェクト	推論	導出オブジェクト
O1,O2	⇒	O4
O1,O3	⇒	O4
O1,O2,O4	⇒	O5
O1,O3,O5	⇒	O2

図 3.3 有向グラフでは表現不可能なリスト

この問題を回避するために, 次節で有向ハイパーグラフを用いたモデル化を提案する.

第4章

提案モデル

4.1　有向ハイパーグラフによるモデル化

4.1.1　ハイパーグラフの定義

グラフにおいて辺とは 2 頂点対のことであった．これは辺は 2 個の頂点からなることを意味する．この個数制限を自由にすることで一般化したものがハイパーグラフである [6, 9]．ハイパーグラフは $H = (V, E)$ と記述される．ここで, V は頂点集合, $E \subseteq 2^V$ は V の部分集合族である．

グラフとは異なり，ハイパーグラフは紙上に図示するのが困難である．そのため，グラフのような図解をされることは少なく，集合論の用語で表されることが多い．

4.1.2　有向ハイパーグラフの定義

有向ハイパーグラフ $H = (V, E)$ は，頂点集合 V と有向辺の集合 E から構成される．ここで, 有向辺とは，空ではない互いに素な V の 2 つの部分集合 S, T の順序対 (S, T) である．

グラフの頂点着色, 頂点彩色, P 着色, リスト着色, リスト彩色はいずれもハイパーグラフに拡張可能であるが, ハイパーグラフのリスト彩色についてはあまり研究がなされていない [7].

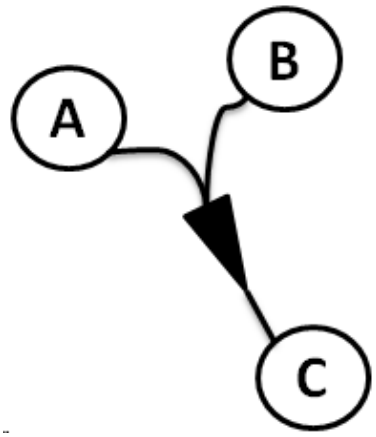

図 4.1 有向ハイパーグラフ

4.1.3 提案モデル

まず, オブジェクト間の依存関係リストを次のようにグラフ化する．頂点集合Vをオブジェクト集合とし, 依存関係リスト上で$O_{i_1}, \cdots, O_{i_k} \in V$から$O_j \in V$が導出可能ならば有向辺$(\{O_{i_1}, \ldots, O_{i_k}\}, \{O_j\})$を描く．さらにそのグラフに対する色リストを用いてACLを次のように表現する．ただしここでは議論を簡単にするためにACLにおいてread可能か否かのみに着目する．まず, 色集合Cをサブジェクト集合とする．そして, ACL上で, あるサブジェクト$S_i \in C$があるオブジェクト$O_j \in V$をread可能ならば, $S_i \in L(O_j)$としO_jの色リストにS_iを加える．ハイパーグラフをFig.4.2に示す.

サブジェクトS_iがオブジェクトO_jをreadしたとき頂点O_jに色S_iを塗るとしよう. 例えば, Fig.4.2においてO_1, O_2が全てS_1で塗られたとする．このとき, 推論によってO_4が導出可能なことを考えればO_1, O_2が全てS_1で塗られた時点でO_4もS_1で塗るべきである．しかしその一方で, $S_1 \notin L(O_4)$, 即ち, ACL上ではS_1はO_4をreadできないことになっているので, これは推論による情報漏えいを意味している．このとき, その頂点着色はリスト着色の定義にも反していることに注目すると推論による情報漏えいに関する安全性を次のように定義できる.

定義：あるP着色がリスト着色ならばその着色は推論に対して安全であるという. ここで, $P =$「任意の頂点vに対して, vを終点とする全ての有向辺S,Tの始点集合Sが同一色で塗られているならば, Tの頂点も同じ色で塗られなければならない.」とする.

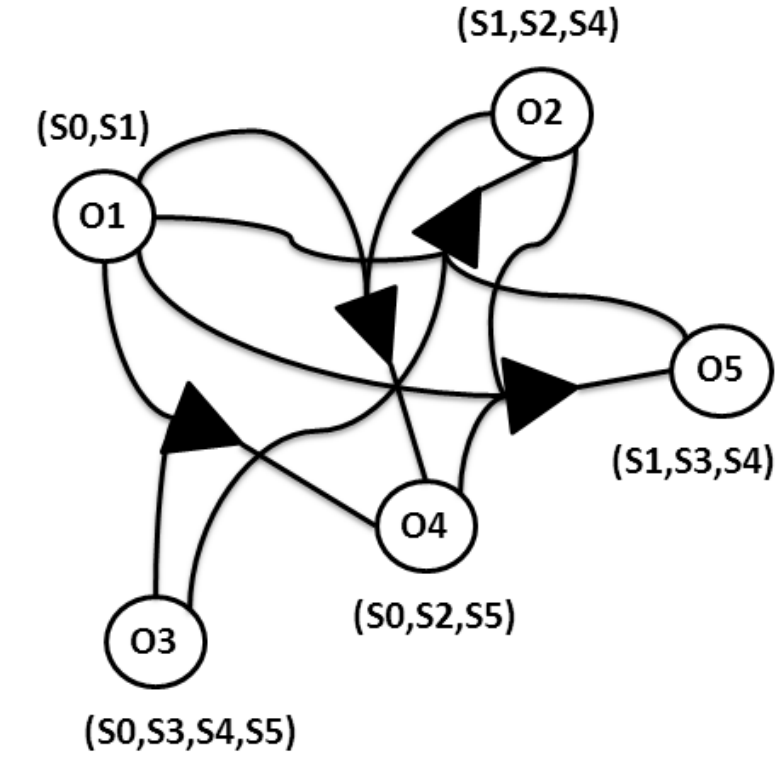

図 4.2 有向ハイパーグラフを用いた提案データ構造

4.2 提案アルゴリズム

推論によるリスト着色が一つでも存在しなければそのACLはreadすることが出来ないオブジェクトが存在することになる. そのようなACLはアルゴリズムの目的は与えられた依存関係とリストで構成されたグラフがリストの組み合わせが一つでも存在していれば, そのACLは推論い対して安全なリストの組み合わせが存在していると言うことを示す.

本稿で読み込み不可能のオブジェクトが存在しないリストの組み合わせを判別するのが目的のため, 効率の考慮はされていない. また, ここで述べるreadの許可は実際に情報のreadをすることを表しているわけではない。まず最初に, 与えられたグラフ全体のノードに含まれているリストを見て$L=1$のノードに含まれているサブジェクトのreadを許可する. $L=1$のノードが複数ある場合, ノードを結ぶ線の数が少ないものから許可を与える. $L=1$のノードにreadの許可を全て与えたときに推論によりサブジェクトを導出できるノードが存在するときそのノードには導出元のサブジェクトを与える. このとき既に推論によりノードのLに存在しないサブジェクトが与えられている場合. そのACLは不自由なACLとなる. その次に$L \geqq 2$のノードをそれぞれLの値が少ないものから順番に許可を与える. 全てのノードに許可を与えたら推論により導出されるサブジェクトが存在するのであればそのノードは着色に沿ってサブジェクトを変更する. このときに図4.3のような$(O_1, O_2) \rightarrow (O_5), (O_3, O_4) \rightarrow (O_5)$だったとき$O_1$と$O_2$が$S_0$で塗られ,$O_3$と$O_4$が$S_3$で塗られていたら$O_5$を$S_0$と$S_3$のどちらで塗るのかという問題がある. $(O_5, O_6), (O_7)$で,O_6にS_3が塗られていて,O_7のLにS_3が無いときもしO_5をS_0で塗ったときO_5とO_6がS_3→推論でO_7もS_3になってしまうケースを見逃す可能性がある. このような場合はO_5にS_0を与える場合とS_3を与える場合の両方を記述することにする.

このようにしてサブジェクトの組み合わせを全て行い, 推論による頂点着色を満たすリストの組み合わせが一つも存在しなければそのACLは推論による情報漏えいを考慮したとき, 読み込めないオブジェクトが存在する不自由なACLとする.

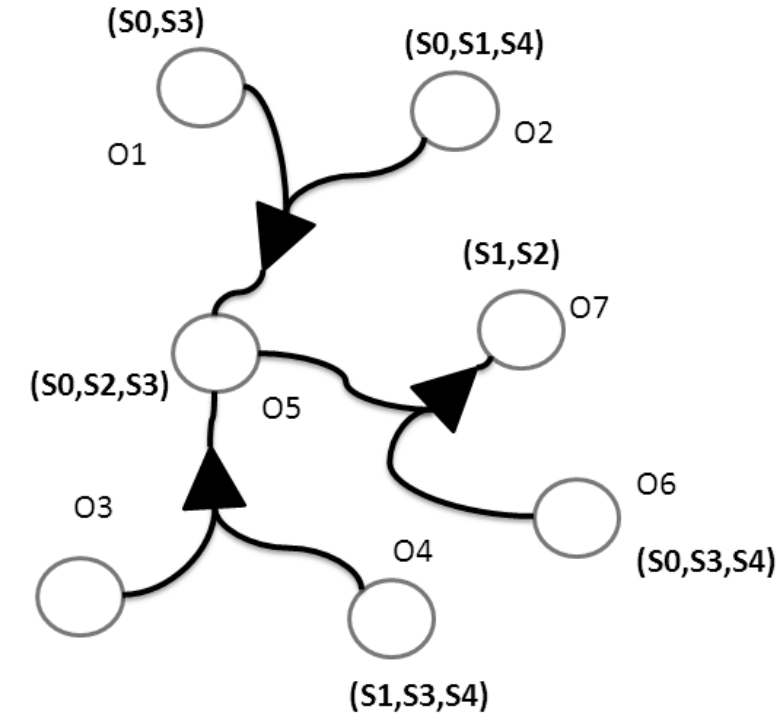

図 4.3 複数の推論ができるグラフ

第5章

結論

本論文では, 推論による情報漏えいを未然に防ぐために必要な推論的依存関係をハイパーグラフによってモデル化することを提案し, 推論攻撃に対する安全性を定義した．アクセス制御リストモデルにおける covert channel 解析だけでは推論による情報漏えいを防げないことを示した．推論的依存関係をハイパーグラフで表現し, アクセス制御リストを色リストで表現することで, ハイパーグラフ上の頂点着色問題に帰結できる可能性を示した．推論による情報漏えい対策のためにアクセス制御リストの修正が必要な場合があることを示した．アクセス制御リストの修正を行う必要がある ACL を判別するためにグラフを用いて安全な頂点着色が存在するかを検出するアルゴリズムについて提案した. アクセス制御リストを効果的に修正するためには, 提案モデル上でどのような問題設定をすれば良いか, また, 本稿で提案した安全性の定義以外にも安全性は考えられるかなどの考察. またアルゴリズムを用いて安全な頂点着色の組み合わせの数などからどのようにして ACL の自由度を評価するかなどの考察が今後の課題である.

謝辞

本研究を行なうにあたり，終始熱心に御指導していただいた木下宏揚教授と鈴木一弘助手に心から感謝致します．また，公私にわたり良き研究生活を送らせていただいた木下研究室の方々に感謝致します．

2012年2月

鈴木　遼

参考文献

[1] ”酒井剛典, 森住哲也, 畔上昭司, 小松充史, 稲積泰宏, 木下宏揚：“ Covert Channel 分析メカニズムと EJB による情報フィルタの構築 ”,2006 年暗号と情報セキュリティシンポジウム, (2006).”

[2] 内野雄策:”SNS における情報漏洩を防止するための情報フィルタの適用 (2009)”

[3] ”ウィキペディア（Wikipedia)”
http://ja.wikipedia.org/wiki/

[4] ”株式会社ミクシィ”
http://mixi.jp/

[5] ”OR 事典” ”http://www.weblio.jp/cat/academic/orjtn”

[6] Giorgio Gallo, Giustino Longo, Sang Nguyen, Stefano Pallottino: ”DIRECTED HYPERGRAPHS AND APPLICATIONS”,(1992), http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.54.829.

[7] Penny Haxell, Jacques Verstracete: ”List coloring hypergraphs”, THE ELECTRONIC JOUNAL OF COMBIN The electronic journal of combinatorics 17, #R129, (2010).

[8] 木下宏揚: ”データベースのアクセス制御に関する考察”, Symposium on Cryptography and Information Security(SCIS1996),SCIS96-10D(1996).

[9] Andre Kundgen, Eric Mendelsohn, Vitaly Voloshin: Colouring planar mixed hypergraphs, The electronic journal of combinatorics 7, #R60 ,(2000).

[10] 森住哲也：“ 直観主義論理の意味論に基づく統合セキュリティモデル” , 博士論文, 博第 3 号, 情報セキュリティ大学院大学,(2008).

[11] Chii-Ren Tsai, Virgil D. Gligor, C. Sekar Chandersekaran, ”A Formal Method for the Identification of Covert Storage Channels in Source Code,” sp, pp.74, 1987 IEEE Symposium on Security and Privacy, 1987

[12] 戸田瑛人, 市瀬浩, 鈴木一弘, 森住哲也, 木下宏揚: ”プッシュ型 Web システムに於ける情報フロー制御の提案”, 信学技報,vol.109,no.4,SITE2009-2,pp.51-56, (2009).

[13] 戸田瑛人, 森住哲也,(鈴木一弘), 木下宏揚: ”MapReduce を用いたクラウドの情報漏洩解析”, Symposium on Cryptography and Information Security(SCIS2010),3E4-2,(2010).

[14] J. ヴァイダヤ, C.W. クリフトン, Y.M. ズー（著）, 嶋田茂, 清水將吾（訳）: プライバシー保護データマイニング, シュプリンガー・ジャパン, ISBN978-4-431-10233-5, (2010).

[15] Agrawal, Rakesh；Srikant, Ramakrishnan: Privacy-preserving data mining, ACM SIGMOD Record, Vol. 29, No. 2, pp. 439–450 (2000).

[16] Blum, Avrim; Ligett, Katrina; Roth, Aaron: A learning theory approach to non-interactive database privacy, In Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC), pp. 609-618 (2008); Full Version, arXiv:1109.2229 (2011).

[17] 佐久間淳, 小林重信: プライバシ保護データマイニング, 人工知能学会誌, 24 巻 2 号, (2009).

[18] Minamikawa, Atsunori; Kotsuka, Nobuhide; Honjo, Masaru; Morikawa, Daisuke; Nishiyama, Satoshi; Ohashi, Masayoshi: RFID Supplement for Mobile-Based Life Log System, Applications and the Internet Workshops, 2007. SAINT Workshops 2007. International Symposium on, (2007).

[19] Aizawa, Kiyoharu: Digitizing Personal Experiences: Capture and Retrieval of Life Log, Multimedia Modelling Conference, 2005. MMM 2005. Proceedings of the 11th International, pp.10–15 (2005).

[20] Abe, Masanobu; Morinishi, Yuji; Maeda, Atsuhiko; Aoki, Masakatsu; Inagaki, Hirohito: A life log collector integrated with a remote-controller for enabling user centric

services, Consumer Electronics, IEEE Transactions on, Vol. 55, pp.295–302 (2009).

[21] Chen, Ye; Pavlov, Dmitry; Canny, John F.: Large-scale behavioral targeting, KDD '09 Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining,

[22] ライフログ活用サービスWG, ライフログ活用サービスWGからの中間報告, 総務省「利用者視点を踏まえたＩＣＴサービスに係る諸問題に関する研究会」, 第４回会合 (2009).

[23] Yamasaki, Shigeichiro: A Dynamic Trust Estimation Method for 'Persona' from the Human Relationship of Social Web, Applications and the Internet (SAINT), 2010 10th IEEE/IPSJ International Symposium on, pp.300–303 (2010).

[24] Yamasaki, Shigeichiro: A Trust Rating Method for Information Providers over the Social Web Service, Applications and the Internet (SAINT), 2011 IEEE/IPSJ 11th International Symposium on, pp.578–582 (2011).

[25] 新保史生: ライフログの定義と法的責任, 情報管理, 53(6), pp.295–310 (2010).

[26] O'Hara, Kieron; Tuffield, Mischa M.; Shadbolt, Nigel: Lifelogging: Privacy and empowerment with memories for life, Identity in the Information Society, Vol. 1, No. 1, pp.155–172 (2009).

[27] Rawassizadeh, Reza; Tjoa, A Min: Securing Shareable Life-logs, Social Computing (SocialCom), 2010 IEEE Second International Conference on, pp.1105–1110 (2010).

[28] 相澤清晴: ライフログの実践的活用:食事ログからの展望, 情報処理, Vol. 50(7), pp.592–597 (2009).

[29] Eagle, Nathan; Pentland, Alex (Sandy): Reality Mining: Sensing Complex Social Systems, Personal and Ubiquitous Computing, Vol. 10(4), pp.255–268 (2006).

発表文献

1. 鈴木 遼, 鈴木一弘, 森住哲也, 木下宏揚, “ 推論による情報漏えい防止のためのハイパーグラフによる依存関係のモデル化 ”電子情報通信学会 情報セキュリティ研究専門委員会（ISEC 研）, 2011 年暗号と情報セキュリティシンポジウム（SCIS2011）,2F3-1,2011 年 1 月

2. 鈴木一弘, 鈴木 遼, 森住哲也, 木下宏揚, “ 推論による情報漏えい防止のためのハイパーグラフモデル ”信学論 D(載録決定)

3. 鈴木 遼, 鈴木一弘, 森住哲也, 木下宏揚, “ 推論による情報漏えい防止のためのハイパーグラフによる依存関係のモデル化の改良 ”技術と社会・倫理研究会 (SITE),2012 年 3 月 (予定)