修 士 論 文

耐タンパ・モバイルエージェントを実現するセキュリティ機構の提案と実装

北陸先端科学技術大学院大学
情報科学研究科情報システム学専攻

長谷川 信

2002年3月

修 士 論 文

耐タンパ・モバイルエージェントを実現するセキュリティ機構の提案と実装

指導教官　二木厚吉 教授

審査委員主査　二木厚吉 教授
審査委員　片山卓也 教授
審査委員　権藤克彦 助教授

北陸先端科学技術大学院大学
情報科学研究科情報システム学専攻

010087 長谷川 信

提出年月: 2002 年 2 月

概 要

本研究の目的は，不正な内部解析や盗み見に対する防御機構を持つモバイルエージェント（耐タンパ・モバイルエージェント）のアーキテクチャを提案することである．

モバイルエージェントはネットワークを通じて移動し，様々なホスト上でタスクを実行するプログラムである．このようなモバイルエージェントは通信回線の接続状況に影響されにくいといった特長を持つ．実際に携帯電話を用いた情報検索システムにモバイルエージェントを利用したものがあり，携帯電話を用いた電子商取引システムにもモバイルエージェントの利用が期待されている．しかし，その実用化にあたりモバイルエージェントのセキュリティを十分考慮する必要がある．

モバイルエージェントのセキュリティ問題は，1）不正なエージェントによるホストへの攻撃，2）不正なホストによるエージェントへの攻撃に大別される．既存の研究の多くは上記 1 の問題に対してある程度有効な解を与えているが，上記 2 の問題に関しては，概念的・理論的なアプローチに終始しているものが多く，その実用性と有効性に疑問がもたれる．しかし，モバイルエージェントのセキュリティを考慮するうえで，上記 2 の問題を避けて通るわけにはいかない．

特に，モバイルエージェントを用いて電子商取引システムなどを構築する場合，モバイルエージェントに持たせた個人情報をどのように保護するかが非常に重要な問題となる．例えば，モバイルエージェントのデータがエージェントの移動中に盗聴される可能性がある．これには通進路を暗号化することで対処できるが，モバイルエージェントが移動した後，移動先のホスト上で悪意のあるモバイルエージェントなどにデータを不正に取得・改竄される危険性もある．この問題は通進路の暗号化だけでは十分ではなく，アプリケーションレベルにおけるデータの保護機構が必要であることを示唆している．

上記の背景をふまえて本研究では，1）モバイルエージェントごとに保護機構を導入，2）モバイルエージェントをメタレベル・アーキテクチャとして構成，3）モバイルエージェントごとにセキュリティポリシを導入などのアプローチより，不正な内部解析や盗み見に対するモバイルエージェントのアーキテクチャを提案する．また，本研究で提案したセキュリティ機構を Java ベースのアプリケーションフレームワーク（ART）として実装し，実験を行った．

目 次

図 目 次

表 目 次

第1章　はじめに

本章では，本研究の背景，目的を述べ，最後に本論文の構成について述べる．

1.1　本研究の背景

モバイルエージェントとは，ネットワーク上の複数のホスト間を移動し，移動先のホスト上でタスクを実行するプログラムである．その移動においては，プログラムコードだけでなく，プログラムの内部状態や実行状態を保持したままホスト間を移動する．この結果，モバイルエージェントは移動前に処理していたタスクを移動先のホスト上でも継続して処理することができる．また，エージェントの移動後は，通信回線を切断するすることができる．このため，モバイルエージェントは，携帯端末が仮定しているような無線通信を行う環境に利用が期待されている．

モバイルエージェントの適用分野の一つとして期待されている携帯端末との連携による電子商取引では，エージェントが複数の仮想店舗を巡回し，ユーザの代わりに情報収集，電子決済などの処理を行うことが可能になる．しかし，その実用化にあたりセキュリティ問題が最大の障害となる．

このようなモバイルエージェントのセキュリティ問題は，不正なエージェントによるホストへの攻撃と，不正なホストによるエージェントへの攻撃に分けられる．前者に対しては，Java アプレットにみられる sandbox モデルやアクセスコントロールリスト (ACL) のようにリソースアクセスを制御する手法や，エージェントを認証する手法がある．後者については，盗み見，内部解析，改竄などの攻撃に対して，1）暗号化したエージェントを移動先のホストでそのまま実行する，2）エージェントを難読化する，3）実行トレースにより攻撃を検出する，などのアプローチが提案されている．しかしこれらの実装例は一部であり，実用的なアプリケーションに適用可能な手法は確立されていない．

電子商取引をモバイルエージェントで実現するためには不正なホストによるエージェントへの攻撃に対処することが重要である．これは，エージェントの持つ秘密情報が移動先のホストによって盗み見，内部解析され，悪用される危険性があるためである．本論文では，移動先のホストの盗み見，内部解析からモバイルエージェントの秘密情報を保護するセキュリティ機構について述べる．

1.2 本研究の目的

本研究の目的は，不正なホストの攻撃に対処するモバイルエージェント（耐タンパ・モバイルエージェント）のセキュリティ機構を提案することである．

前説で述べたように，携帯電話を用いた電子商取引システムにモバイルエージェントの利用が期待されている．しかし，その実用化にあたりモバイルエージェントのセキュリティ問題への対処が必要である．

そこで，本研究では以下を前提として，

- 携帯端末を使った電子商取引をモバイルエージェントで実現する．
- モバイルエージェントが持つデータが，不正なホストにより盗み見，内部解析される問題を取り扱う．
- 本研究は新しい暗号化・難読化技術の提案ではない．

以下のアプローチから，既存の暗号化・難読化技術との融合により，耐タンパ・モバイルエージェントを柔軟に構成するためのアーキテクチャを提案する．

- 各モバイルエージェントに保護機構を導入
- 保護機構のモジュール化
- 各モバイルエージェントにキュリティポリシを導入
- 保護機構の難読化

また，本研究で提案するセキュリティ機構をアプリケーションフレームワークとして実装することでセキュアなモバイルエージェントプログラムの作成を支援する．

1.3 本論文の構成

本論文の構成は以下となる．第 2 章では，モバイルエージェントの基本的概念，構成について説明し，既存のモバイルエージェントシステムを幾つか紹介する．第 3 章では，具体例を用いて本研究で対象とするモバイルエージェントのセキュリティ問題について述べる．第 4 章では，本研究で提案するセキュリティ機構について述べる．第 5 章では，提案したセキュリティ機構をアプリケーションフレームワークとして実現する．第 6 章では，アプリケーションフレームワークを用いて実際に作成した例題：電子商取引エージェントについて説明し，実装したシステムを考察する．第 7 章では，関連研究としてモバイルエージェントのセキュリティ問題に対する既存の対処手法と耐タンパ・ソフトウェアについて説明する．第 8 章では，本論文をまとめ，今後の課題について述べる．

第2章　モバイルエージェントシステム

本章では，はじめに移動性や自律性といったモバイルエージェントの基本的な概念について説明する．次に既存のモバイルエージェントシステムを幾つか紹介する．最後に本研究で使用する AgentSpace を例に，モバイルエージェントの構造を説明する．

2.1　モバイルエージェントとは

モバイルエージェントの標準化仕様であるＯＭＧのＭＡＳＩＦでは,「人や組織のために自律的に活動するコンピュータ・プログラム」を「エージェント」の定義とし,「実行を開始したシステムには拘束されないエージェント」を「モバイルエージェント」の定義としている．しかし，この定義はエージェントの研究者すべてが合意しているものではなく，モバイルエージェントという用語の明確な定義は定まっていない．このため，各々の論文で定義は異なるが，一般的に以下の性質を持ったプログラムであるとされる．

- 移動性
  ネットワーク上のホスト間を移動し，複数のホストにまたがるタスクを処理できる．
- 自律性
  移動先で外部からのアクションやイベントが無くても処理を維持継続できる．また，外部から情報を取り入れ，自らの振る舞いを自分で決定できる．
- 協調性
  エージェント同士が互いに通信し合い，協調的に振る舞うことができる．
- 局所性
  システム全体の情報を持つ必要はなく，局所的な情報のみで行動できる．
- 適応性
  移動先のホストの環境に対応して，適切に行動できる．

従来のクライアント・サーバシステムと比較した特徴として，モバイルエージェントは通信回線の障害に対する強さがある．従来のクライアント・サーバシステムでは（図 2.1参照），クライアントとサーバ間に，大量の通信が発生する．クライアントとサーバ間の回線が高速なら問題はないが，回線が頻繁にディスコネクトするような携帯端末が用いた環

境では，送受信以外は通信回線の維持が必要ないモバイルエージェントの利用が有効である．

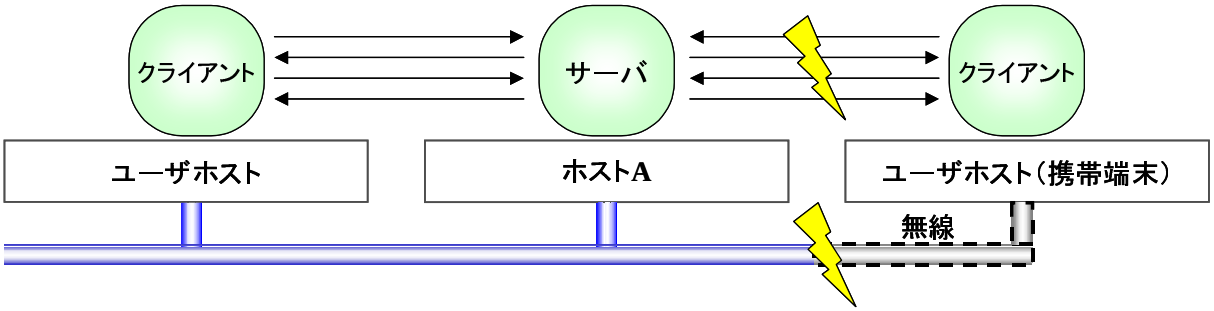

図 2.1: クライアント・サーバシステムの問題

2.2 既存のモバイルエージェントシステム

既存のモバイルエージェントシステム 1994 年に米 General Magic 社（アップル・コンピュータ社が設立）が Telescript と呼ぶエージェント技術の仕様を公開してから，さまざまなモバイルエージェントシステムが企業や研究機関などから提案されてきた．本節では，既存の代表的なモバイルエージェントの概要を述べる．

2.2.1 Telescript

Telescript は，米 General Magic 社により開発された世界初の商用モバイルエージェントシステムである [2][3]．商業的には成功しなかったが，その後モバイルエージェント技術に大きな影響を与えた．

Telescript の根本となっているアイデアは，コンピュータネットワークを「データ」だけではなく「動いているプログラム」つまりプロセスが行き交う場にしようということである．モバイルエージェントという用語は，通信技術に関するものであり，人工知能やインタフェースエージェントのような擬人性は無いとされている．

Telescript のエージェントは独自のオブジェクト指向言語である Telescript 言語で記述され，仮想機械を通じて実行される．エージェントのデータ化は深い継続実行（Strong migration, Strong mobility）[4] が可能であり，プログラムコードと共に内部変数値や実行時内部データ（プログラムカウンタ，内部スタック）などの実行環境を転送する．また，Telescript では，プレースと呼ばれる非移動プロセスを初めて導入した．このプレースはネットワーク上の各ホストに 1 つ以上用意され，モバイルエージェントにリソースやサービスを提供する．エージェントはプログラム中で go() 命令を実行することによりプレー

表 2.1: Aglets のコールバックメソッド

メソッド名	呼び出されるタイミング
onCreation	エージェントのライフサイクルの中で 1 度だけ， エージェントが生成されたとき
onDisposing	エージェントが終了するとき
onCloning	エージェントが複製される前
onClone	エージェントが複製された時に，複製により生成された エージェントの onClone メソッドが呼び出される
onCloned	エージェントが複製された時に，オリジナルのエージェ ントの onCloned メソッドが呼び出される．
onDispatching	移動の直前に呼び出される
onReverting	エージェントがリモートホストから呼び戻された時
onArrival	移動先のホストに到着したとき
onDeactivating	エージェントが非活性化される直前
onActivation	エージェントが活性化された時

ス間を移動しながら，移動先のプレースからサービスを受けたり，同一プレース上のエージェントと通信しながら処理を進める．

Telescript 言語の実行環境である Telescript Engine の他に Tabriz と呼ばれる開発環境があり，Web から Telescript を操作するための機能や Telescript プロセスから HTTP リクエストを行うための機能がライブラリとして提供されている．

セキュリティ対策としては，不正なエージェントから移動先のホストを守る機構として，エージェント認証，セーフインタプリタなどの機能を備えている．

2.2.2 Aglets

Aglets は，IBM の東京基礎研究所が開発している Java ベースのモバイルエージェントシステムである [5]．オープンソースであり，IBM Public Licence のもとに公開されている．エージェントは，1）生成直後，2）移動直前・直後，3）終了直前，などの所定のタイミングに呼び出されるコールバックメソッド群からなる Java 言語オブジェクトである．Aglets は Java のシリアライゼーションを利用しているため，エージェントのデータ化は浅い継続実行（weak migration, weak mobility）[4] であり，内部変数値は転送対象となるが，メソッド内のローカル変数やプログラムカウンタは対象外となる．

セキュリティ対策としては，不正なエージェントから移動先のホストを守る機構として，エージェント認証，セーフティインタプリタなどの機能を備えている．また，不正なホストからエージェントを保護する機構として，セキュリティドメイン認証が取り入れら

れている [6]．表 2.1に Aglets のコールバックメソッドと，各メソッドが呼び出されるタイミングを示す．

2.2.3 Voyager

Voyager は，ObjectSpace 社により開発された Java ベースのモバイルエージェントシステムであり [7]，最も広く普及しているといわれる．Voyager は，独自の ORB[1] を基礎とし，リモートホスト上に Java オブジェクトを生成したり移動したりでき，その特別な場合としてモバイルエージェントを実現している．エージェントのデータ化は Aglets と同様に浅い継続実行であるが，移動先で呼び出すメソッドを明示的に指定することができる．

セキュリティ対策としては，リモートホスト間のエージェント通信を SSL で保護している他，不正なエージェントから移動先のホストを守る機構として，エージェント認証，セーフインタプリンタなどの機能を備えている．

2.2.4 Plangent

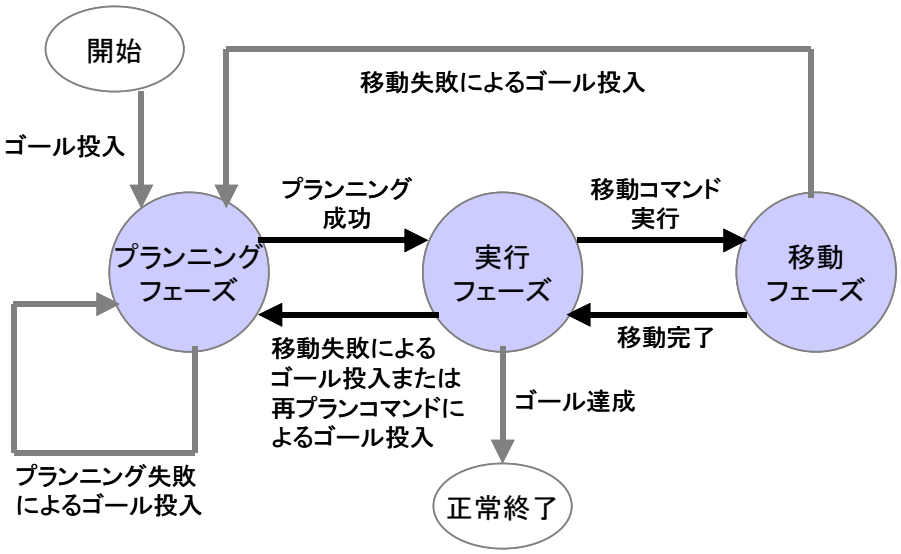

図 2.2: Plangent の動き

Plangent は，東芝 S & S 研究所により開発されている Java ベースのモバイルエージェントシステムである [8]．Plangent の特徴は，エージェントがプランニング機能，プラン実行機能を持つ点である．

[1] ORB(Object Request Broker) は，分散オブジェクト間のメソッド呼び出しを仲介し，各メソッド呼び出しにおいて，実際に呼ばれるオブジェクトやメソッドを選択・実行する機構である．

- プランニング
  ユーザによって与えられた要求を達成するために自らの行動計画を立案する機能.

- プラン実行
  プランニングによって決定された行動計画を実行する機能.

- 再プランニング機能
  移動先のホスト（Plangent ではノードと呼ぶ）でエージェントの実行が失敗した場合には，失敗の根拠となった情報の更新を行い行動計画を生成し直す機能.

これらの機能により，1)「状況」として現在のネットワーク環境や各所で提供されるサービス状況，2)「目的」としてユーザがネットワークから得たい情報や利用したいサービスの内容，などをエージェントに与えると，図 2.2のように，エージェント自身がプランニング，プラン実行を行う自律性が実現される．

Plangent は Java により作成されたシステムであるが，エージェントはプラン記述を考慮した独自のスクリプト言語（PlangentScript）によって記述され，エージェントはこのスクリプト定義を解釈／展開しながら実行するインタプリタを装備する．エージェントのデータ化は Telescript と同様に深い継続実行であり，プログラムカウンタや内部スタックといった実行時内部データも転送の対象となる．

セキュリティ対策としては，不正なエージェントから移動先のホストを守る機構として，エージェント認証の他，プランニング機能に制約処理機能を追加する手法がとられている [9]．また，不正なホストからエージェントを保護する機構としては，ホスト認証が取り入れられている．

2.2.5 AgentSpace

図 2.3: MobileSpaces のモバイルエージェント

AgentSpace は，国立科学研究所の佐藤一郎氏によりモバイルエージェントの開発研究用フレームワークを目的として開発された，Java ベースのモバイルエージェントシステムである [10]．AgentSpace の特徴は，エージェントの移動が高速であることが挙げられる．これは，エージェントの実行に必要なクラスのうち，移動先に存在しないクラスファイルをエージェントが全て持って移動するためである．これに対し，既存のモバイルエージェントシステムの多くは，エージェントが移動先のホストで必要になった時点でクラスをオンデマンドに転送する．このため AgentSpace は，1 度の通信で必要とする資源を一括転送できることからディスコネクトといったネットワーク障害に強いが，エージェントのサイズが大きくなり通信コストが大きくなる．エージェントのデータ化は Aglets,Voyager と同様に浅い継続実行である．

セキュリティ対策としては，転送するエージェントデータの暗号化が可能である．

佐藤一郎氏はこの他に，同じく Java ベースのモバイルエージェントシステムである MobileSpaces も公開している [11]．MobileSpaces は，Mobile Ambients の概念を取り入れた高階モバイルエージェントシステムであり，図 2.3のように，エージェントの中にエージェントを包含したまま移動することができる．

第3章　事例

本章では，具体例をもとにして，本研究が対象とするモバイルエージェントの問題を明確にする．

3.1　携帯端末を用いたモバイルエージェント

モバイルエージェントは，前述したように通信回線の接続状況に影響されにくいという特徴がある．そのためディスコネクトが頻繁に怒るような無線を用いた環境に利用が期待されている．実際，携帯電話を用いた情報検索サービスや携帯端末用のミドルウェアなどにモバイルエージェントを利用したものもあり，携帯電話を用いた電子商取引システムの実現にもモバイルエージェントの利用が期待されている．しかし，その実用化にあたりモバイルエージェントのセキュリティを十分考慮する必要がある．

3.2　インターネットを巡回する電子商取引エージェント

電子商取引システムをモバイルエージェントを利用して実現するには，そのモバイルエージェントのセキュリティを十分考慮する必要があること前説で述べたが，モバイルエージェントのセキュリティ問題は，モバイルエージェントのセキュリティ問題は，1）不正なエージェントによるホストへの攻撃，2）不正なホストによるエージェントへの攻撃に大別される．上記 1 の問題に対しては実用的な対処手法が提案されているが，上記 2 の問題に対しては，研究段階のものが多く有効な対処手法は確立されていない．しかし，モバイルエージェントのセキュリティを考慮するうえで，上記 2 の問題を避けて通るわけにはいかない．

特に，モバイルエージェントを用いて電子商取引システムなどを構築する場合，モバイルエージェントに持たせた個人情報をどのように保護するかが非常に重要な問題となる．例えば，モバイルエージェントのデータがエージェントの移動中に盗聴される可能性がある（図 3.1参照）．これには通進路のデータを暗号化することで対処できるが，モバイルエージェントが移動した後，移動先のホスト上で他の悪意のあるモバイルエージェントなどにより，データを不正に取得・改竄される危険性もある（図 3.2参照）．この問題は通進路のデータの暗号化だけでは十分ではなく，アプリケーションレベルにおけるデータの保護機構が必要であることを示唆している．

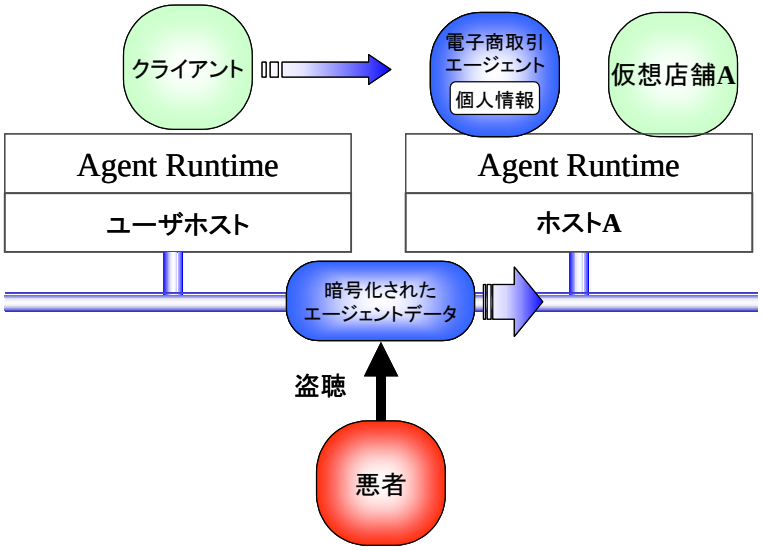

図 3.1: 移動中のモバイルエージェントが盗聴される問題

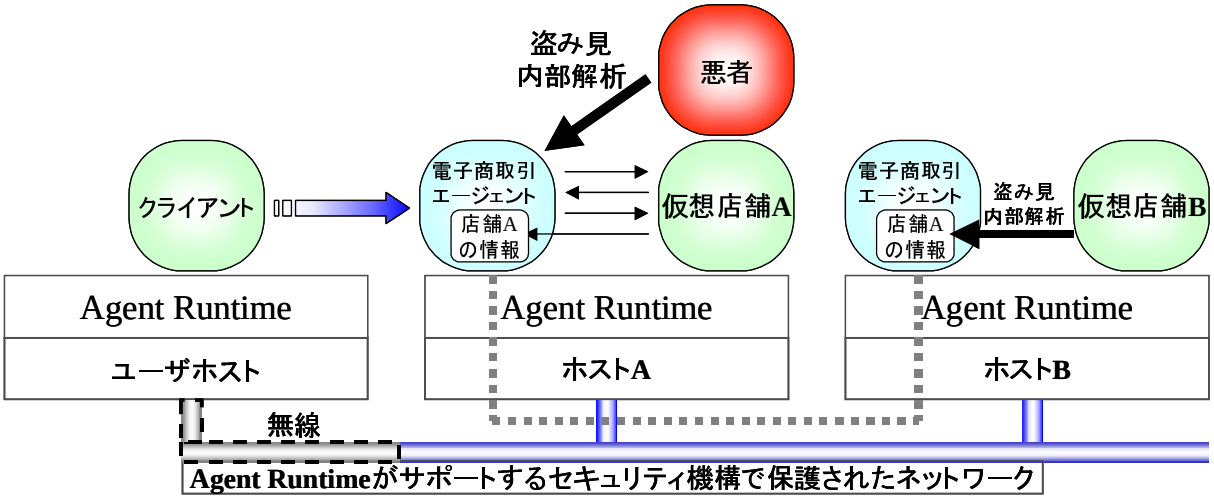

図 3.2: 移動先のホストでモバイルエージェントが内部解析・改竄される問題

3.3 考察

このような事例から，本研究では不正なホストによる攻撃に対して，通進路のデータの暗号化に加えて，アプリケーションレベルにおけるデータの保護機構が必要であると考え，不正なホストによる攻撃，特に不正な内部解析や盗み見に対する防御機構を持つ耐タンパ・モバイルエージェントを柔軟に構成するためのアーキテクチャを提案する．本研究の目的は新しい暗号化・難読化技術の提案ではない．既存のそれらとうまく融合できるようなモバイルエージェントのアーキテクチャの設計が最終目標であり，本研究は暗号化・難読化技術の研究と競合するものではなく，それらと相補的な関係に位置づけられる．

第4章　本研究のセキュリティ機構

本章では，本研究で提案するセキュリティ機構について述べる．はじめに前提とアプローチを設計方針として述べる．次に電子商取引を例題とし，セキュリティ機構を実現するためのモバイルエージェントのアーキテクチャと秘密情報の保護手法について説明する．最後に秘密情報へのアクセスを制限するセキュリティポリシについて説明する．

4.1　設計方針

第3章の考察に基づき本研究では，不正なホストによる攻撃，特に不正な内部解析や盗み見に対する防御機構を持つ耐タンパ・モバイルエージェントを柔軟に構成するためのアーキテクチャを提案する．

対象にとするモバイルエージェントの問題や本研究の位置づけなどを，本研究のセキュリティ機構における前提として以下に述べる．

本研究の目的は新しい暗号化・難読化技術の提案ではない．既存のそれらとうまく融合できるようなモバイルエージェントのアーキテクチャの設計が最終目標であり，本研究は暗号化・難読化技術の研究と競合するものではなく，それらと相補的な関係に位置づけられる．

本研究における前提

- 携帯端末を使った電子商取引をモバイルエージェントで実現する．
  本研究で提案するセキュリティ機構は，このような環境を想定して提案する．

- モバイルエージェントが持つデータが，不正なホストにより盗み見，内部解析される問題を取り扱う．
  不正なホストによる攻撃にも様々なものがある．全ての攻撃に対処することは非常に困難である．本研究では対象とする攻撃を盗み見や内部解析に絞り，対処手法を提案する．不正なホストの攻撃には不正なエージェントによるエージェントへの攻撃も含まれるとする．

- 本研究は暗号化・難読化技術の研究と競合するものではなく，それらと相補的な関係に位置づける．

本研究の目的は新しい暗号化・難読化技術の提案ではなく，既存のそれらとうまく融合できるようなモバイルエージェントのアーキテクチャの設計が最終目標である．

このような前提下のもと，本研究では以下のアプローチによりモバイルエージェントの持つ秘密情報を保護する．

本研究のアプローチ

- モバイルエージェントごとにデータの保護機構（暗号・復号化）を導入
  本研究で考えるアプリケーションレベルにおける保護機構として，モバイルエージェント自身に暗号・復号下機構を内蔵させる．これにより，あらかじめ所持するデータはもとより移動先で新しく得たデータも自分で保護することが可能になる．

- 保護機構のモジュール化
  暗号･復号化機構などの保護機構は，アプリケーションロジックから分離し，モジュール化する．

- モバイルエージェントをメタレベル・アーキテクチャとして構成
  モバイルエージェントは保護機構のモジュール群などから構成されるメタレベルと，アプリケーションロジックやモバイルエージェントの基本機構などから構成されるベースレベルの二層構造で構成する．このようなモバイルエージェントのアーキテクチャは，メタレベルの変更や再利用により，既存の暗号化・難読化技術を柔軟に取り入れることが可能であり，本研究の最終目標である既存の技術との融合を実現する．

- 保護機構の難読化
  モジュール化した保護機構を難読化することにより，内部処理の処理内容や実行順序の解析を困難にする．

- モバイルエージェントごとにセキュリティポリシを導入
  秘密情報に適用するセキュリティ用件をセキュリティポリシとして定義し，エージェントに持たせることにより，復号化する情報を柔軟に変更することが可能となる．

暗号化と難読化の定義

本研究では，暗号化・難読化技術を用いてモバイルエージェントのデータを保護する．そこで，本研究における暗号化と難読化の定義は以下とする．

- 暗号化
  情報の表現を組み替えて，第三者が利用できないようにする技術とする．暗号化の対象は文字列とプログラムコードとし，暗号化された状態では実行が不可である．

- 難読化
  プログラムのアルゴリズム解析を困難にし，解析に手間や時間を要するように仕向ける技術とする．難読化の対象はプログラムコードとし，難読化したプログラムは，難読化前のプログラムと同じ結果を生じる．

本研究では，以上の前提とアプローチを設計方針とし，不正な内部解析や盗み見に対する防御機構を持つ耐タンパ・モバイルエージェントを柔軟に構成するためのアーキテクチャを提案する．

4.2 耐タンパ・モバイルエージェントの構成

本節では，本研究で提案する耐タンパ・モバイルエージェントの構成について述べる．

本研究が最終目標とする，既存の暗号化・難読化技術との融合が可能な耐タンパ・モバイルエージェントのアーキテクチャを実現するために，以下のような特徴をもつモバイルエージェントの構成を提案する．

耐タンパ・モバイルエージェントのアーキテクチャの特徴

- モバイルエージェントに導入する保護機構のモジュール化
- モバイルエージェントのアプリケーションロジックと保護機構を分離
- ベースレベルとメタレベルからなる二層構造でモバイルエージェントを構成

モバイルエージェントのデータの保護は，暗号化機構，復号化機構，セキュリティマネージャなどのセキュリティ機構により実現する．これらのセキュリティ機構は各機能別にモジュール化する．モジュール化することにより，各機構の柔軟な変更や再利用が可能になる．

既存の耐タンパ・ソフトウェア技術はソフトウェアの保護機構とアプリケーションロジックを入り組ませて耐タンパ性を実現している．しかし，このような構造は，ソフトウェアの変更を非常に困難なものにする．さらに，一度でも保護機構が解析されてしまった場合，ソフトウェアを全て作り直さなくてはいけないようなアドホックな実装となっている．それに対し本研究のように保護機構とアプリケーションロジックを分離することにより，例え一度解析されてしまっても，解析されてしまった保護機構だけを柔軟に変更することで対処でき，それ以外の部分の再利用も可能である．

本研究では，図 4.1のように，ベースレベルとメタレベルからなる二層構造でモバイルエージェントを構成する．すなわちエージェントの本来の目的となるアプリケーションロジックとセキュリティに関わる機能を分離したモバイルエージェントのアーキテクチャを提案する．このような二層構造でモバイルエージェントを実現することにより，メタレベルの柔軟な変更や再利用が可能となる．

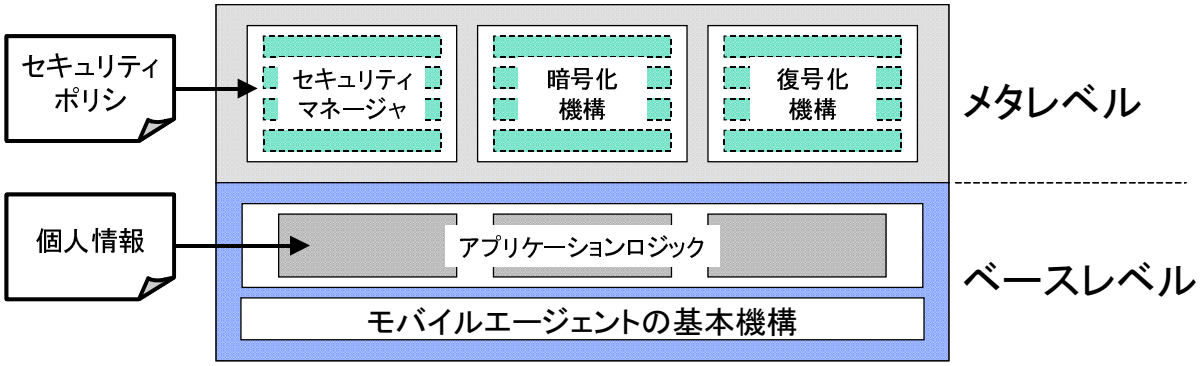

図 4.1: 耐タンパ・モバイルエージェントの構成

ベースレベルとメタレベルの本研究における意味は，メタレベルの保護機構を用いてベースレベルのデータを暗号化して保護することから，操作を施す側と操作が反映される側としての意味を持つ．

4.2.1 ベースレベル

本研究では二層構造のモバイルエージェントのアーキテクチャを提案する．そのうちの一層であるベースレベルについて説明する．ベースレベルの特徴は以下となる．

ベースレベルの特徴

- モバイルエージェントの基本機構，アプリケーションロジックなどから構成される．
- ベースレベル単体でもモバイルエージェントとして機能する．
- メタレベルによって暗号・復号化される．

モバイルエージェントの基本機構とは，モバイルエージェントとしての基本的な機能を提供する機構である．

例えば前述した既存のモバイルエージェントシステムである AgentSpace のモバイルエージェントの基本機構とは以下のような機能を提供する機構である．

ベースレベルのモバイルエージェントの基本機構

- エージェントの永続化
  エージェントを永続化する機能．

- エージェントの起動
  永続化されたエージェントの情報を読み取り，活性化する機能.
- エージェントの移動
  エージェントを任意のエージェントランタイム（ホスト）に移動する機能.
- エージェントの停止
  エージェントを停止する機能.
- エージェント間通信
  同じエージェントランタイム上のエージェントと通信する機能.

これらの機能はモバイルエージェントシステムのエージェントランタイムによって行われる機能であり，エージェント自身が提供している機能ではないため，厳密にはエージェントが内蔵する機構という表現は正当ではない．しかし，モバイルエージェントはこれらの機能をコールバックメソッドを用いて実行する．したがって本研究ではこのようなコールバックメソッド群をエージェントの基本機構と呼ぶ.

アプリケーションロジックとは，アプリケーション特有の機能のことであり，エージェントの実行目的となる部分である．電子商取引を例とすると以下のような機能である．

ベースレベルのアプリケーションロジック

- 巡回パターン
  エージェント移動機構を利用して複数のパターンで巡回する機能
- 秘密情報
  電子商取引で用いるユーザの個人情報
- 情報収集
  移動先の仮想店舗の情報を取得するための機能
- 電子決済
  仮想店舗と電子決済を行うための機能

電子商取引を実現する全ての機能は電子商取引のアプリケーションロジックあり，電子商取引で必要になる秘密情報なども，アプリケーション特有の情報なのでアプリケーションロジックである.

このようにベースレベルはモバイルエージェントの基本機構と電子商取引のアプリケーションロジックより構成され，電子商取引エージェントとして機能することが可能である.すなわち，ベースレベルとは，セキュリティを考慮していない従来のモバイルエージェントを構成するものである．このようなことから以下の利点が考えられる.

ベースレベルの利点

- 保護機構から分離されたアプリケーションロジックだけの変更が可能
- 既存のモバイルエージェントに容易に保護機構を付加することが可能

また，ベースレベルのアプリケーションロジックは秘密情報として扱われ，メタレベルによって保護される．

4.2.2 メタレベル

二層構造におけるベースレベルの上位に位置するメタレベルについて説明する．メタレベルの特徴は以下になる．

メタレベルの特徴

- セキュリティ関連のモジュール群より構成
  柔軟なセキュリティ機構を実現するために，ベースレベルのアプリケーションロジックから分離された耐タンパを実現するためのセキュリティ機能モジュール群によりメタレベルは構成される．
- 難読化による保護
  ベースレベルは，メタレベルにより不正な内部解析や盗み見といった攻撃から保護されるが，メタレベル自身を守るセキュリティ機能は備えていない．そこで既存の難読化技術を用いて，ベースレベルのモジュール群を難読化する．

本研究のアプローチで述べたように暗号・復号化機構などのデータの保護機構はアプリケーションロジックから分離しモジュールとして実現する．このようなモバイルエージェントのアーキテクチャはメタレベルの変更や再利用により，既存の暗号化技術を柔軟に取り入れ可能とする．メタレベルは以下のセキュリティ関連モジュールにより構成される．

メタレベルの保護機構モジュール

- 暗号化機構
  秘密情報を暗号化するための機構．ベースレベルの秘密情報を暗号化するためにセキュリティマネージャにより管理される．
- 復号化機構
  暗号化した秘密情報を復号化するための機構．暗号化機構と同様にベースレベルの秘密情報を暗号化するためにセキュリティマネージャにより管理される．

- セキュリティマネージャ
  読み込んだセキュリティポリシに基づき，秘密情報の暗号・復号化を管理するための機構.

メタレベルの問題

これらのモジュールによりエージェントのデータの保護を実現する．しかしメタレベルは自身に対して行われる内部解析を防御する機構を備えていない．耐タンパ・モバイルエージェントを実現するにはメタレベルの保護も必要である．メタレベルの攻撃の対処として，既存の難読化技術を取り入れる．難読化とは前述したようにプログラムのアルゴリズム解析を困難にし，解析に手間や時間を要するように仕向ける技術である．具体的な難読化手法は後述する．

メタレベルの利点

- ベースレベルのアプリケーションロジック全体の保護が可能
- メタレベルの変更や再利用により，既存の暗号化技術を柔軟に取り入れ可能
- 既存のモバイルエージェントにメタレベルを付加することで，容易に保護機構を導入することが可能

メタレベルはエージェントのデータを保護するための機能をすべて備える．また，アプリケーションロジックから分離されていることにより，メタレベルの柔軟な変更や再利用が可能である．このような柔軟な機構は，暗号化可能な秘密情報を扱う全てのモバイルエージェントに適用可能であると考えられる．したがってアプリケーションフレームワークとして実装することで，耐タンパ・モバイルエージェントの容易な実装を支援する．

メタレベルの欠点

しかし，このような二層構造は本来モバイルエージェント全体に対して行う不正な内部解析をメタレベル一層に限定される可能性がある．この問題に対しては二層構造を用いる利点である柔軟性により，頻繁に保護機構を変更することや，複雑な難読化を施して解析を困難にすることで対処する．

4.3 セキュリティポリシ

秘密情報に適用するセキュリティ用件は，セキュリティポリシとして外部のポリシファイルに定義する．本研究で用いるセキュリティポリシは，2つの要素の組からなる単純なものである．セキュリティポリシには，エージェントの持つ秘密情報の重要度を定義する．

4.3.1 秘密情報の重要度

エージェントの持つ秘密情報には，適用するセキュリティ用件として，その情報の重要度を定義する．それらはセキュリティポリシとして外部のポリシファイルに定義し，エージェントの実行時にポリシファイルを読み込ませる．セキュリティポリシファイルには，キーと値の 2 つ要素の組により記述する．キーには秘密情報の識別子を記述し，値にはその秘密情報の重要度を記述する．

＜ 秘密情報の識別子 ： その重要度 ＞

秘密情報の重要度は，1 つの情報に 1 つ定義する．したがって，秘密情報が複数ある場合には，各情報ごと個別に重要度を定義する必要がある．電子商取引を例とすると図 4.2のようになる．

氏名	:	重要度 1
住所	:	重要度 4
電話番号	:	重要度 3
メールアドレス	:	重要度 2
購入希望商品名	:	重要度 1
購入希望価格	:	重要度 3

図 4.2: 電子商取引のセキュリティポリシ

セキュリティポリシは外部ファイルとして定義することにより，柔軟な変更が可能となる．

このように秘密情報に重要度を定義することにより，移動先のホストに公開する秘密情報を決定するときの基準とする．公開する情報の決定は，秘密情報の重要度と公開対象となるホストの信頼度に基づいて行われる．

4.3.2 ホストの信頼度

本研究ではモバイルエージェントがセキュリティポリシに定義した情報の信頼度とホスト（エージェント）の信頼度により，公開する情報を柔軟に変更する．このホストの信頼度について以下に説明する．本研究はホスト上の不正なエージェントによる攻撃を対象とするため，ここでいうホストとは仮想店舗などのエージェントであるとする．

事例：インターネットオークション業者の身元確認

近年利用者が急増しているインターネットオークションなどで，盗品が出品され売買されることを防ぐ目的の法規制として，インターネットオークション業者やホームページで古物を売買している業者は身元確認のために，公安委員会に届け出が義務付けられた．

今後ホストの信頼性の問題は重要になると考えられる．

本研究ではモバイルエージェントの移動先となるホストに信頼度を定め，ホストはモバイルエージェントにその信頼度の公開が義務付けられているとした．

前提

- ホストとはエージェントが情報公開する対象であるモバイルエージェントである．
- ホストの信頼度は，信頼のおける第三者により定められている．
- ホストはエージェントに対しホストの信頼度を提示する．

信頼度およびその認定者の明確な定義は，本研究の対象外であるので詳細を避けるが，具体例として，デジタル証明書に採用されている認証局（CA）のような組織を想定する．信頼度の基準については，対象となるホストで過去に起きたセキュリティ問題やホストの運用期間などの運用実績を基準として定められているとする．

このように定められたホストの信頼度と秘密情報の重要度に基づき，セキュリティマネージャは移動先のホストに公開する情報を決定する．

エージェントの信頼度

本研究ではモバイルエージェントを利用したアプリケーションのセキュリティ問題を対象としている．このようなアプリケーションでは1つのホスト上に複数のエージェントが存在する場合がある．例えば本研究で例題とする電子商取引においても，あるホスト上にショッピングモールのようなプレースが存在し，そこに仮想店舗であるモバイルエージェントが集まり，それら仮想店舗で取引を行うような場合も考えられる．このとき，既存のホスト認証によりホストが認証できてもそのホスト上の仮想店舗モバイルエージェントに悪意がある場合，安全は保証できない．このような状況ではモバイルエージェントの認証が非常に重要になってくる．本研究で定めるホスト（モバイルエージェント）の信頼度とは以上のようなことを想定して提案するものである．

4.4 秘密情報の保護機構

本節では，秘密情報の保護機構である暗号・復号化機構とセキュリティマネージャについて述べる．

4.4.1 暗号・復号化機構

本研究では，モバイルエージェントに暗号・復号化機構を内蔵させる．このように，モバイルエージェント自身が暗号・復号化機構を持つことにより以下の利点が得られる．

エージェントが保護機構を内蔵する利点

- 移動先のホストに依存しない暗号技術が利用できる．
  例えば移動先のホストにより用いる暗号アルゴリズムが異なる場合や移動先のホストが暗号化機構を備えていない場合でも，エージェントは秘密情報を暗号・復号化できる．また，移動先のホストの環境を気にすること無く暗号技術の変更が可能である．

- モバイルエージェントが自律的に暗号・復号化を行える．
  モバイルエージェント自身が暗号・復号化機構を持つため，エージェントが望む状況で秘密情報の暗号・復号化を行うことが可能である．

本研究では既存の暗号化技術用いることを前提とし，以下の暗号化技術などの利用が期待できる．

利用が期待できる暗号化技術

暗号化技術とは対象となる情報の表現を組み替えて，第三者が利用できないようにする技術である．基本的には鍵と呼ばれるパラメタを用いてその鍵にもとづき情報の変換（暗号化）を行うものである（図 4.3参照）．その情報を復号化するときにも暗号化に使われた鍵を用いることにより復号化可能となる．以下に本研究で利用が期待できる暗号化技術について説明する．

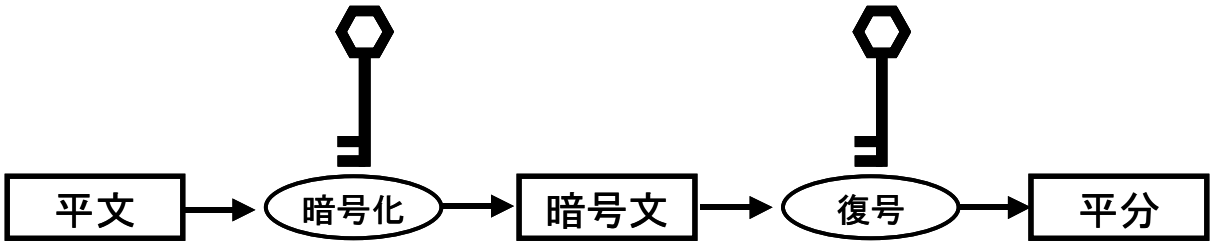

図 4.3: 鍵を使う暗号化と復号化

- シーザー暗号（図 4.4参照）
  古代に暗号を利用していた有名な人物であるユリウス・カエサル（Julius Caesar）

が用いた暗号技術である．この暗号は非常に単純なもので，アルファベットを文字を一定の文字数だけ循環させるものである．

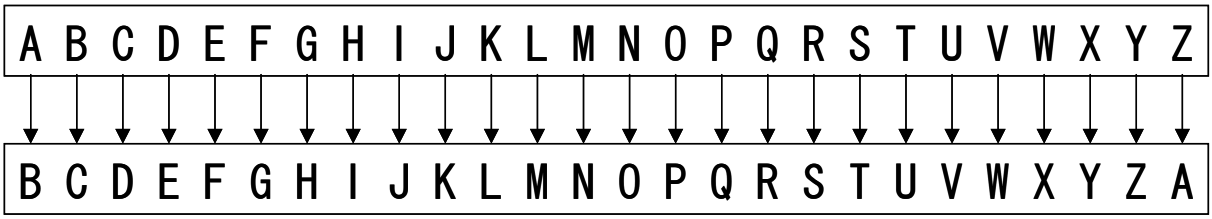

図 4.4: シーザー暗号

- BASE64 エンコード暗号化技術というよりは変換技術である．BASE64 エンコードとは，対象となるデータの 8bit2 文字分を 6bit3 文字に分割し，大文字と小文字のアルファベットや数字など，64 種類の文字を使ってデータを表現し直す変換技術である．

- DES（Data Encryption Standard）（図 4.5参照）
  NIST により米国内の機密扱いではない情報を保護する標準暗号化アルゴリズムとして採用された暗号化技術である．DES は，一般の解読攻撃に対して非常に有効だったが，1990 年代半ばには差分解読と線形解読にもとづく攻撃が発見され，1999 年には 24 時間以内に解読された．しかし，保護されている情報よりも DES を攻撃するコストのほうが高くつく場合は，ある程度のセキュリティが期待でき，一般に利用できるコンピュータリソースと解読技術を使って解読されることを防ぐ．

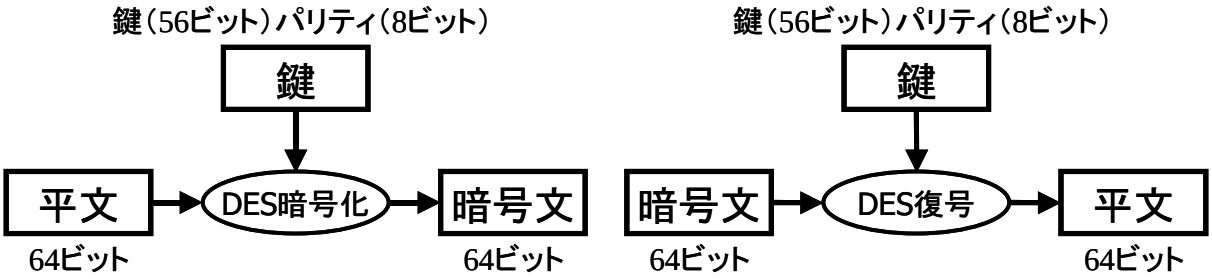

図 4.5: DES（Data Encryption Standard）

- RSA（Rivest, Shamir, Adleman）最も有名な公開鍵アルゴリズムで，大きな数を因数分解することの難しさを利用してセキュリティを確保する暗号化技術である．ディジタル署名にも利用されている．しかし，非常に処理速度が遅いといった欠点があり，上記 DES の実装と比較して 100〜1000 倍の時間がかかってしまう．だが，まだ破られていない暗号化技術である．

4.4.2 セキュリティマネージャ

セキュリティマネージャは，セキュリティポリシにより定義された秘密情報の重要度と移動先のホストの信頼度に基づき，秘密情報を移動先のホストに公開する．セキュリティマネージャの主な役割は，公開する情報の決定と内蔵する暗号・復号化機構を用いてその情報を復号化することである．

ホストに移動してから，次のホストに移動するまでの基本的なセキュリティマネージャの動作は以下になる（図 4.6参照）．

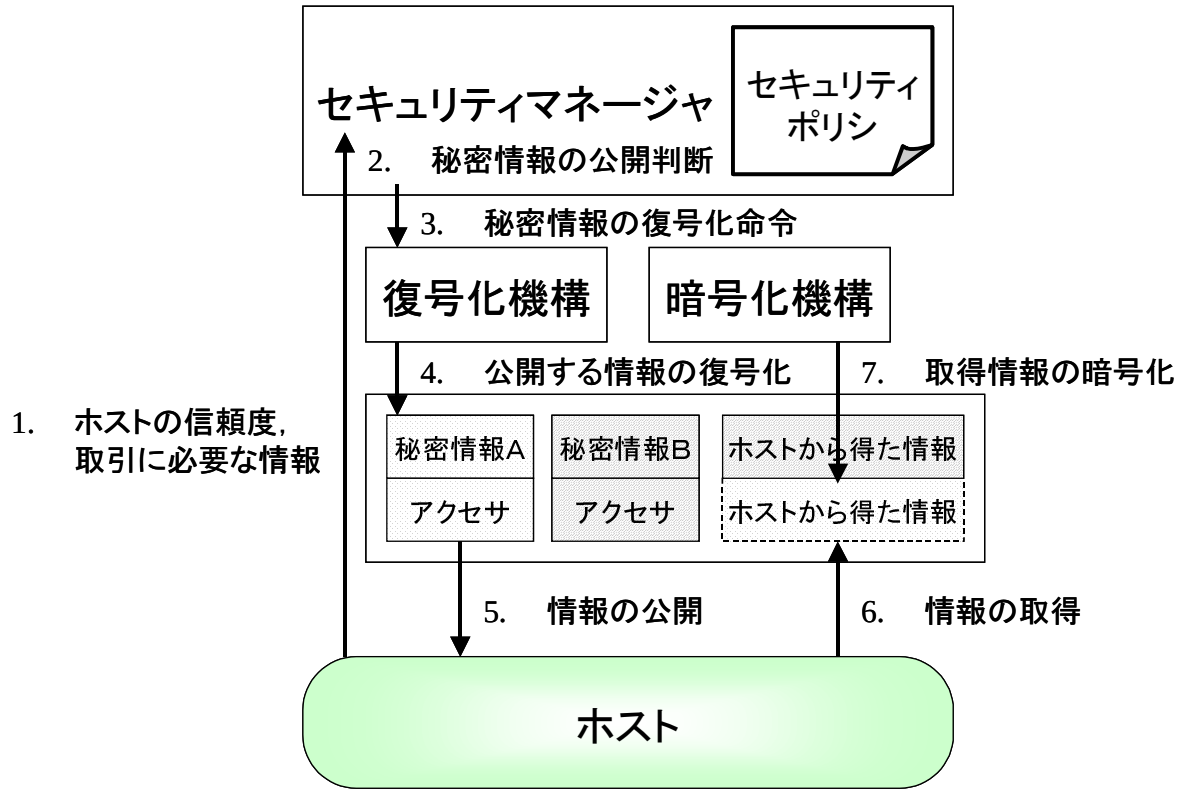

図 4.6: セキュリティマネージャの動作

セキュリティマネージャの動作

1. 対象となるホストの信頼度を確認

2. ホストの要求を確認

3. 要求された情報で公開する情報を判断

4. 公開する情報を復号化

5. ホストから取得した情報を暗号化

公開する情報の判断方法を以下に説明する．
例えば，取引先のホストの信頼度が 3 である場合，情報の重要度が 3 以下の情報は公開対象とし，4 以上は非公開対象とするといった単純な方法である．

4.5 保護機構の難読化

秘密情報は，暗号方式を知られてしまうと簡単に復号化されてしまう．そこで，メタレベルの暗号・復号化機構は，移動先のホストに解析されないように難読化する必要がある．同様に，メタレベルの機構であるセキュリティマネージャ，ホスト認証機構なども難読化する．
本研究は，以下のような既存の難読化手法を用いる（第 7 章 関連研究 耐タンパ・ソフトウェア参照）．

本研究で用いる既存の難読化手法

- メタレベルの機構はセルと呼ぶ実行可能な最小単位のモジュールに分割（図 4.7参照）
  プログラムを難読化するために準備

- 時間軸と空間軸の両方にセルを分散
  セルを連続アドレスに存在させてしまうと，主記憶の単純なスキャニングにより取得できてしまう．そこで時間軸と空間軸の両方にセルを分散する．

- セルのインタりーブおよびマルチスレッド化
  演算やループ処理の正しさは保ったまま，セルはインタリーブして実行する．また，セルをプラットホームの提供するマルチ・スレッド機能を利用し，並列実行することにより，実行順序がプログラミングだけでは決まらなくなるため，実行順序の追跡を困難にする．

- ダミーセルの挿入
  セルのりーブおよびマルチスレッド化の際，実行結果に副作用を及ぼさないダミーセルを挿入することにより，実行内容の解析をより困難にする．

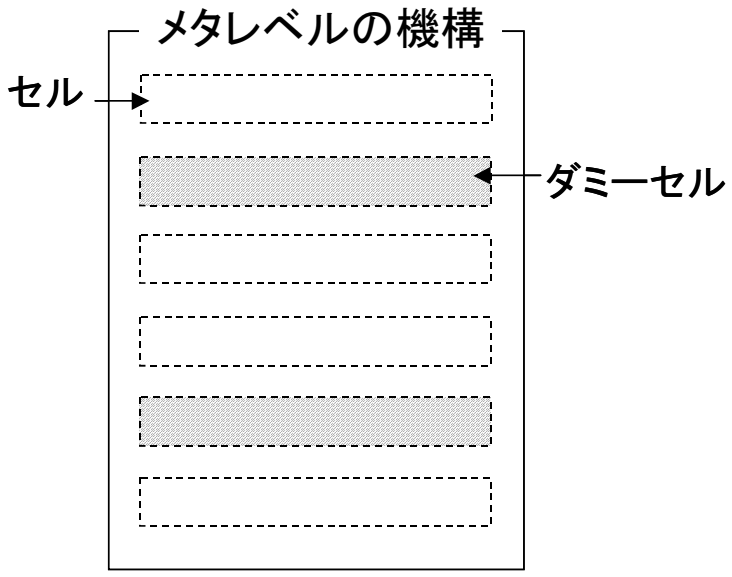

図 4.7: 実行可能なセルとダミーセルに分割

第5章　実装

本章では，本研究で実装したアプリケーションフレームワークについて説明する．アプリケーションフレームワークは，第4章で提案したセキュリティ機構をJavaで実装したものであり，ここではARTと呼ぶ．

5.1　実装環境

- OS
  Windows 2000 Professional
- Java
  Java(TM)2 SDK Standard, Edition Version 1.3.1(JDK 1.3.1_02)
- 開発ツール
  JBuilder 5 Enterprise
- モバイルエージェントシステム
  AgentSpace1(2000年10月5日版)

5.2　AgentSpaceの概要と仕組み

本節では，実装のベースとなるモバイルエージェントシステムであるAgentSpaceについて，その概要と仕組みを述べる．

AgentSpaceは，Java言語を利用した分散計算用ミドルウェアであり，モバイルエージェントの作成を可能にするフレームワークとそのエージェントの実行や移動などを処理するエージェントランタイムシステムの二つの部分から構成されている．

5.2.1　エージェントランタイムシステム

AgentSpaceのエージェントランタイムシステムは，Javaの仮想機械（VM）上で動作し，エージェントの起動，永続化，停止を管理する．また，他のエージェントランタイムシステムとのエージェントの送受信を行うことができ，エージェントを受信するとその

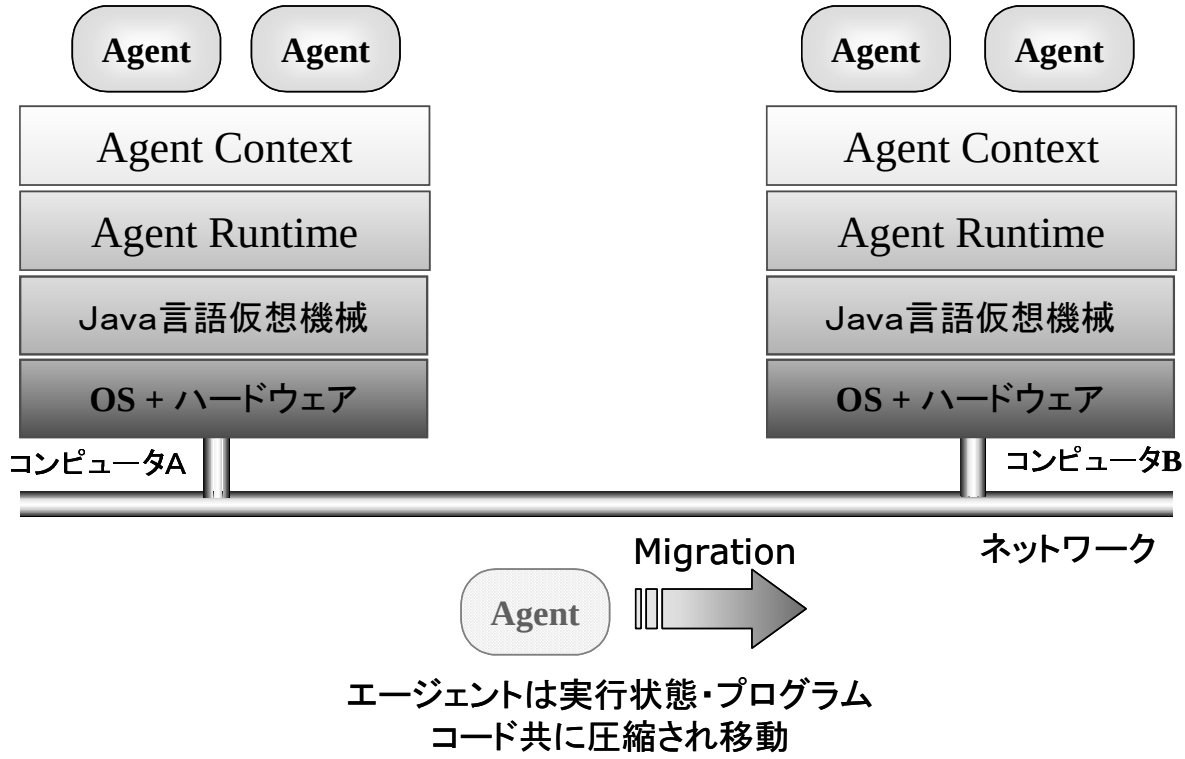

図 5.1: AgentSpace システム構成図

エージェントを実行可能にし，転送要求のあったエージェントを他のエージェントランタイムシステムに転送する．

エージェントランタイムシステムは，各エージェントのインタフェースとなるエージェントコンテキストと，エージェントの移動や永続化を管理するエージェントランタイムから構成されている（図 5.1参照）．

エージェントランタイムシステムの機能には以下のものがある．

- エージェントの永続化
  Java のシリアライゼーション機能を利用してエージェントを永続化する機能．永続化された情報には，エージェントのプログラムコードとインスタンス変数などの実行状態が含まれる．
- エージェントの起動
  永続化されたエージェントの情報を読み込み，エージェントを生成する機能．
- エージェントの移動
  エージェントを指定されたエージェントランタイムシステムに転送する．また，他のエージェントランタイムシステムから受信したエージェントを実行可能な状態にする．
- エージェントの停止
  エージェントプログラムの実行を停止する機能．
- エージェント間通信
  同じエージェントランタイムシステム上のエージェント間通信の実現．エージェント間通信には，非同期メッセージ通信，同期メソッド呼び出し，非同期メソッド呼び出し（フューチャ通信）がある．

5.2.2 コールバックメソッド

AgentSpace におけるモバイルエージェントプログラムは，一つまたは複数の Java クラスから構成され，そのうち一つ以上のクラスは，クラス Agent のサブクラスとして定義される．Agent クラスは，表 5.1に示すコールバックメソッドから構成され，各メソッドはエージェントの状態変化の前後に呼び出される．クラス Agent のサブクラスでこれらのメソッドをオーバーライドすることで状態変化に対応した動作の記述が可能となる．

5.2.3 エージェント間通信

AgentSpace では，セキュリティを配慮してエージェントが別のエージェントを直接参照することができない．そこで，エージェント間で通信を必要とする場合には，AgentSpace

表 5.1: AgentSpace のコールバックメソッド

メソッド名	呼び出されるタイミング
void init()	クラスファイルからエージェントを合成するときに一度だけ呼び出される.
void create()	エージェントを生成した直後に呼び出される．初期化などはここに記述する.
void destroy()	エージェントが停止する直前に呼び出される.
void dispatch(URL url)	エージェントが他のコンピュータに移動する直前に呼び出される．引数 url には，移動先の URL アドレスが入る
void arrive()	エージェントが移動先のコンピュータに到着した直後に呼び出される.
void suspend()	エージェントが永続化される直前に呼び出される.
void resume()	エージェントが活性化された直後に呼び出される.
void duplicate()	エージェントの複製が生成される直前に呼び出される.
void parent(AgentIdentifire aid)	エージェントの複製が生成された直後に，複製元のエージェントで呼び出される．引数 aid には，複製によって生成されたエージェントの識別子が入る.
void child(AgentIdentifire aid)	エージェントの複製が生成された直後に，複製によって新たに生成されたエージェントで呼び出される．引数 aid には，複製元となったエージェントの識別子が入る.

が提供する AgentContext クラスの API を用いて通信を行う．本節では，AgentSpace が提供するエージェント間通信について解説する．なお，以下で述べるエージェント間通信 API は，同一ホスト上のエージェントとしか通信ができない．これは，他のホストにいるエージェントとは移動後にローカル通信を行うべきであるという AgentSpace の設計方針に基づくものである．

エージェント間通信により送信するメッセージは，Message クラスによって指定する．Message クラスのコンストラクタで，受信側エージェントが呼び出すメソッド名を指定し，Message クラスのメソッド setArg(Object arg) で呼び出すメソッドの引数を指定する．

例えば，

```
Message msg = new Message("append");
msg.setArg("foo");
msg.setArg("bar");
```

のように作成したメッセージ msg を送信した場合，受信側エージェントではメソッド append(”foo”,”bar”) が呼び出される．

- 非同期メッセージ通信

  `void send(AgentIdentifier aid, Message msg)`

  識別子 aid をもつエージェントに Message クラスのインスタンス msg で与えられたメソッドを呼び出す（図 5.2参照）．送信側エージェントは，メソッドの返値を受け取ることはできないが，受信側エージェントの状態に関わらず処理を継続することができる．ただし，送信した時点で，受信側エージェント自体および対応するメソッドが存在しない場合，そして引数の数や型が一致しない場合でもエラーを返すことはない．

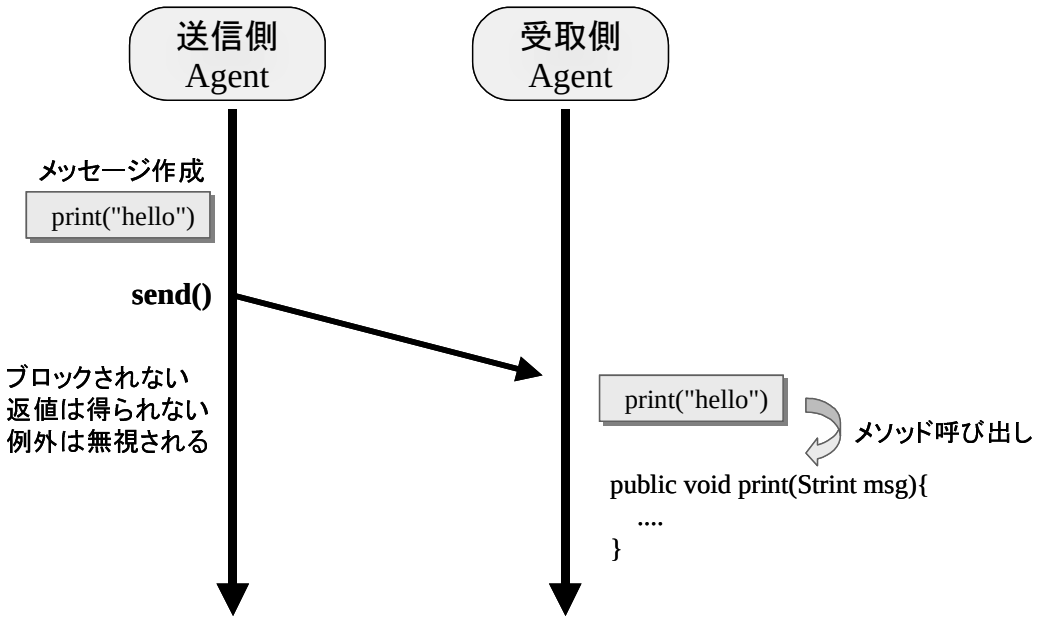

図 5.2: 非同期メッセージ通信

- 同期メソッド呼び出し

  `Object Call(AgentIdentifier aid, Message msg)`

  識別子 aid をもつエージェントに Message クラスのインスタンス msg で与えられたメソッドを呼び出す．返値は呼び出したメソッドの返値となる（図 5.3参照）．このとき，送信側エージェントは受信側エージェントが結果を返すまでブロックされる．ただし，送信した時点で，受信側エージェント自体および対応するメソッドが存在しない場合，そして引数の数や型が一致しない場合はエラーを返す．

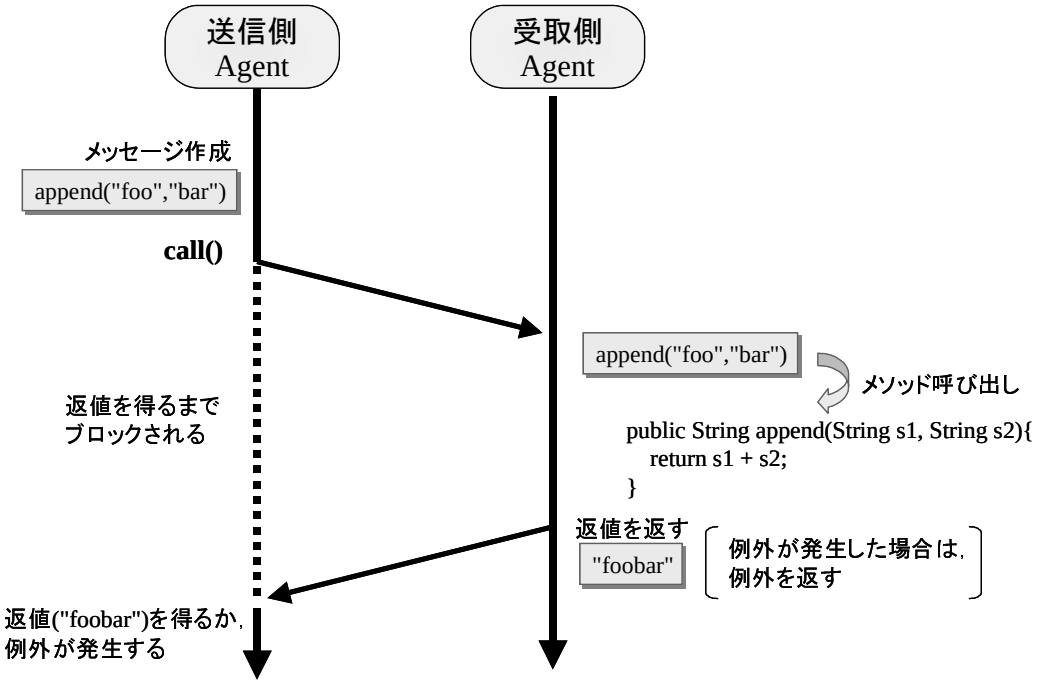

図 5.3: 同期メソッド呼び出し

- 非同期メソッド呼び出し（フューチャ通信）

  `void future(AgentIdentifier aid, Message msg)`

  識別子 aid をもつエージェントに Message クラスのインスタンス msg で与えられたメソッドを呼び出す．送信側エージェントは，メソッドの返値を受け取ることはできないが，受信側エージェントの状態に関わらず処理を継続することができる．送信側エージェントは，Future クラスの getReply() メソッドを通じて返値を得ることができる（図 5.4参照）．ただし，送信側エージェントが getReply() メソッドを呼び出した時点で，受信側エージェントが返値を返していない場合，返値が返されるまで送信側エージェントはブロックされる．なお，送信した時点で，受信側エージェント自体および対応するメソッドが存在しない場合，そして引数の数や型が一致しない場合でもエラーを返すことはない．

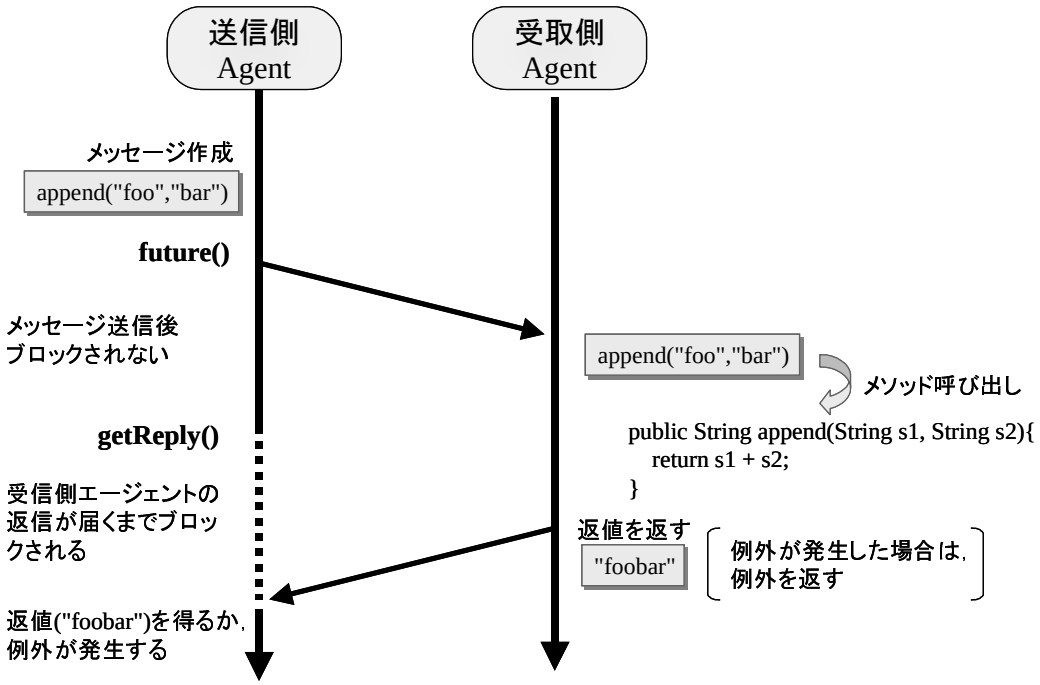

図 5.4: 非同期メソッド呼び出し（フューチャ通信）

5.3　アプリケーションフレームワークの構成

本研究で提案したセキュリティ機構をアプリケーションフレームワークとして実装した．このアプリケーションフレームワークを ART と呼ぶ．ART は以下の特徴をもつ．

アプリケーションフレームワーク ART の特徴

- メタレベル（暗号・復号化モジュール）の柔軟な変更が可能
- 暗号方式として単純なシーザー暗号，BASE64 エンコード方式にもとづく暗号アルゴリズムなどを提供
- 難読化機能としてマルチスレッド化したダミー処理を実現
- セキュリティ・ポリシを外部ファイルとして取り込み可能
- メタレベルとセキュリティポリシを管理するセキュリティマネージャを提供
- 複数の巡回パターンを提供

アプリケーションフレームワークは，耐タンパ・モバイルエージェントのベース（クラスライブラリ），ベースレベル（クラスライブラリ），メタレベル（クラスライブラリ）などから構成される．これらについて以下で説明する．また，図 5.5にアプリケーションフレームワークの構成を示す．

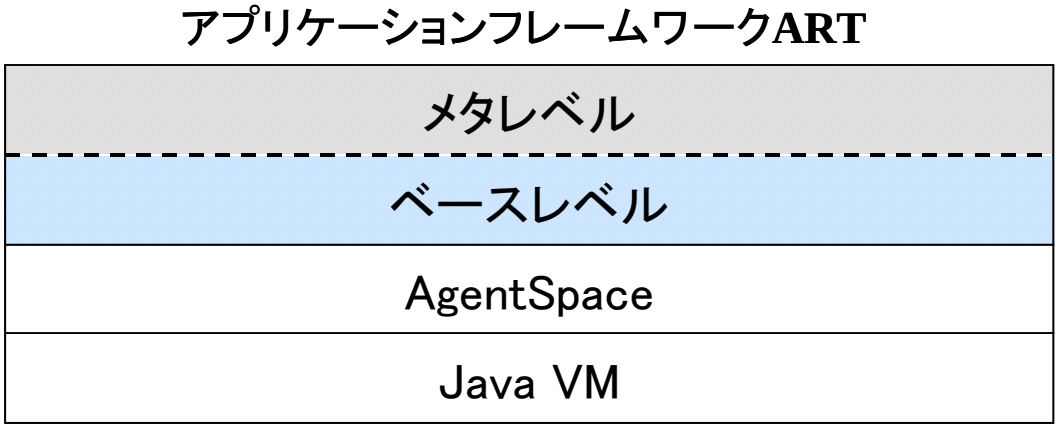

図 5.5: アプリケーションフレームワーク ART の構成

- 耐タンパ・モバイルエージェントのベース（クラスライブラリ）
  AgentSpace が提供する Agent クラスを継承することにより，モバイルエージェント

を実現するクラスであり，クラスライブラリであるベースレベルとメタレベルを用いて耐タンパ・モバイルエージェントを実現する．このクラスライブラリは，電子商取引エージェントの GUI，ファイル管理，エージェント間通信を用いた電子商取引プロトコル，自動巡回などの機能を提供する．

- ベースレベル（クラスライブラリ）
  ベースレベルの機能を持つクラス群から構成されるパッケージである．電子商取引を行うための基本的な機能を実装したクラス群より構成される．ユーザのホストで読み込んだ個人情報や仮想店舗から得た情報など，電子商取引に必要となる秘密情報は，情報の種類別に分類し，対応する各クラスにより管理する．秘密情報を管理する各クラスは，管理する秘密情報とその情報へのアクセサにより構成される

- メタレベル（クラスライブラリ）
  メタレベルの機能を持つクラス群から構成されるパッケージである．暗号化，復号化，セキュリティマネージャなどのセキュリティ機能を提供するクラス群から構成される．セキュリティマネージャはユーザのホストでセキュリティポリシを読み込み，セキュリティポリシに基づきベースレベルの秘密情報を管理する各クラスをシリアライズ後暗号化する．

5.3.1 パッケージ構成

ART は Java のクラスファイル群から構成される．図 5.6 に，これらクラスファイルのパッケージ構成を示す．

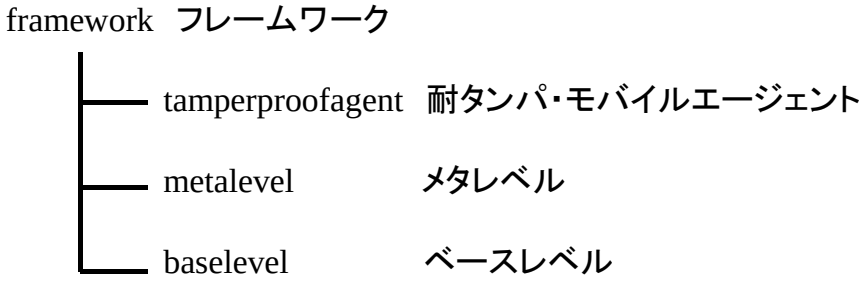

図 5.6: ART のパッケージ構成

パッケージの概要は以下になる．

- tamperproofagent
  耐タンパ・モバイルエージェントを構成するための基本的な機能をもつクラス群（表 5.2参照）から構成されるパッケージである．

表 5.2: tamperproofagent に含まれるクラスの概要

クラス名	概要
TamperProofAgent	耐タンパ・モバイルエージェントの主となるクラス
TamperProofAgent_Frame	ファイル読み込み機能を提供
TamperProofAgent_Utilities	ユーティリティを提供

表 5.3: metalevel に含まれるクラスの概要

クラス名	概要
AgentSecurityManager	暗号・復号化機構を管理するクラス
CaesarCipher	シーザー暗号を実現するクラス
Base64Encode	Base64Encode を実現するクラス

- metalevel
  本研究で提案したメタレベルを実装したパッケージである．セキュリティ関連の機能をもつクラス群（5.3）から構成される．

- baselevel
  本研究で提案したベースレベルを実装したパッケージである．電子商取引を実現するためのクラス群 5.4より構成される．

5.3.2 クラス構成

ART を構成する主なクラスの構成を図 5.7に示す．
クラスの概要は以下になる．

- Agent
  AgentSpace で提供されるエージェントの雛形クラス．このクラスを継承することにより AgentSpace 上のモバイルエージェントとして機能する．主な機能をしてはコールバックメソッド群やエージェント間通信などがある．

- TamperProofAgent
  耐タンパ・モバイルエージェントの主となるクラスであり，Agent クラスを継承する．このクラスはメタレベルパッケージおよびベースレベルパッケージをインポートすることで，耐タンパ・モバイルエージェントとして機能する．また，Agent が提供するエージェント間通信やエージェントコンテキストを利用した機能などを拡張した機能を実装する．

表 5.4: baselevel に含まれるクラスの概要

クラス名	概要
ECApplicationLogic	電子商取引を実現するクラス
ECA_RoundManager	巡回パターンにより移動先を決定するクラス
ECA_Serializable	ベースレベルのシリアライズを管理するクラス
ECA_PrivateData	個人情報とそのアクセサを保持するクラス
ECA_CatalogueList	価格表リストとそのアクセサを保持するクラス
ECA_PriceList	価格表とそのアクセサを保持するクラス
ECA_ReliabilityList	信頼度リストとそのアクセサを保持するクラス
ECA_VirtualShopList	仮想店舗アドレスリストとそのアクセサを保持するクラス

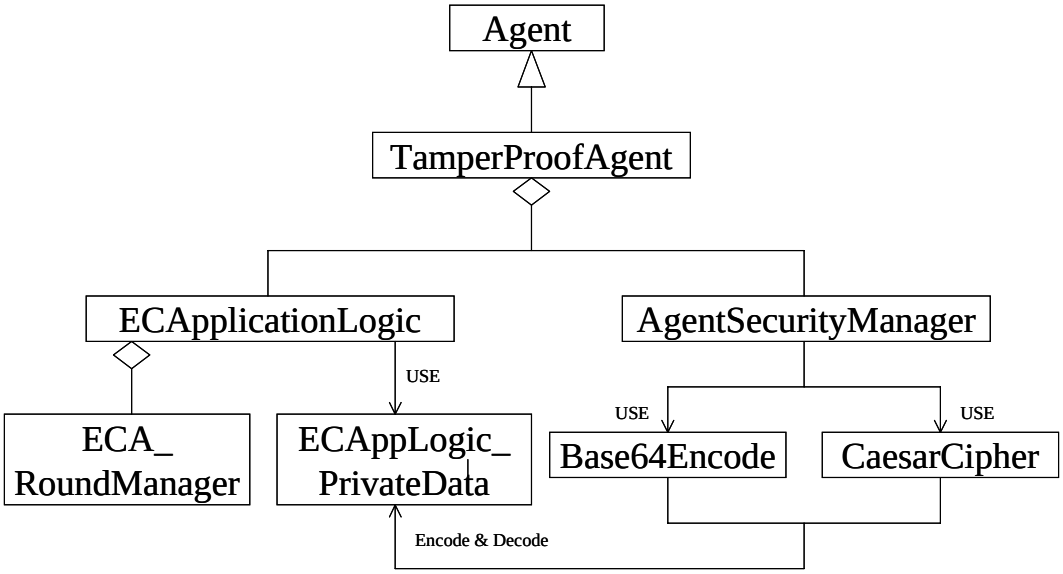

図 5.7: アプリケーションフレームワークのクラス構成

表 5.5: アプリケーションフレームワークの主なクラスの機能

クラス名	概要
Agent	AgentSpace で提供されるエージェントの雛形クラス
TamperProofAgent	耐タンパ・モバイルエージェントの主となるクラス
AgentSecurityManager	暗号・復号化機構を管理するクラス
ECApplicationLogic	電子商取引を実現するクラス
ECA_RoundManager	巡回パターンにより移動先を決定するクラス
ECA_PrivateData	個人情報とそのアクセサを保持するクラス
CaesarCipher	シーザー暗号を実現するクラス
Base64Encode	Base64Encode を実現するクラス

- AgentSecurityManager
  メタレベルパッケージのセキュリティマネージャであり，メタレベルの暗号・復号化モジュールを管理する．暗号・復号化モジュールを新たに追加する場合は，このクラスを変更することにより利用可能となる．

- ECApplicationLogic
  ベースレベルパッケージの電子商取引を実現するクラス．巡回パターンに応じた巡回先を決定する機能などを備える．

- ECA_RoundManager
  ベースレベルの巡回マネージャである．電子商取引エージェントは仮想店舗を出発するごとに巡回マネージャを呼び出し，次の移動先仮想店舗を決定する．

- ECAppLogic_PrivateData
  電子商取引で必要となる個人情報を保持するクラス．個人情報へのアクセサ関数を持つ．このクラスは個人情報ごとシリアライズし暗号化される．

- Base64Encode
  メタレベルの提供する暗号アルゴリズム．このクラスは Base64Encode を実現する．String 型および byte 配列型に対応している．

- CaesarCipher
  メタレベルの提供する暗号アルゴリズム．このクラスはシーザー暗号を提供する．String 型および byte 配列型に対応している．

第6章　実験・考察

本章では，例題を用いて本研究で提案したセキュリティ機構を実験・考察する．

6.1　例題：電子商取引エージェントシステム

本研究で提案したセキュリティ機構の利用が期待できるものとして，個人情報を必要とするインターネットサービスなどが考えられる．具体的には，書籍や電化製品などをインターネットを利用して購入する電子商取引や航空機チケットの予約，ホテルの宿泊予約，懸賞応募サイト，インターネットオークションなどがある．

本研究で提案したセキュリティ機構は，これら全てのサービスに適用可能だが，ここではその１つである携帯端末などを利用する電子商取引エージェントを例題として実験する．電子商取引エージェントは携帯端末などのユーザのホストを出発した後，仮想店舗のあるホストを巡回し，ユーザの代わりに情報収集，電子決済などを行う．

本研究では，携帯端末の利用を前提としているが，携帯端末を使った電子商取引にモバイルエージェントを利用する利点は以下になる．

携帯端末を用いた電子商取引にエージェント利用する利点

- ネットワーク上の仮想店舗に対して，情報収集や電子決済，決済時の個人情報の入力などの処理を，モバイルエージェントに代行させることで，ユーザの手間が軽減.
- モバイルエージェントの送受信以外は通信路を維持する必要がないため，ディスコネクトが頻繁に起るような携帯端末の環境に有効.

携帯端末では，入力端末の機能が制限されている場合があり，入力する情報量が多い場合はユーザ負荷が大きくなる可能性がある．例えば，インターネットブラウザを用いて個人情報を入力する電子商取引の場合，ユーザは取引対象となる各仮想店舗毎に個人情報を入力する必要がある．しかし，モバイルエージェントを用いた場合，一度だけ個人情報と希望する処理を入力してしまえば，後はモバイルエージェントがユーザの代理として複数の仮想店舗を巡回し，取引を行ってくれる．

前述するように，携帯端末などのディスコネクトが頻繁に起るような環境においてモバイルエージェントの利用は有効である．しかし，既存のモバイルエージェントシステムには，エージェントの実装したクラスが移動先で必要になったときに，オンデマンドにク

ラスを転送する方式を採用するシステムもある．しかし，本研究ではベースとなるモバイルエージェントシステムに AgentSpace を採用することで，電子商取引エージェントに全ての機能をもたせることができた．したがって，エージェントの送受信を成功させれば，完全に通進路を切断することが可能である．これはバッテリーの制限がある携帯端末で，消費電力の減少にも有効である．

電子商取引エージェントを例題とした実験では，このような利点を活かしたシステムを構築した．

6.1.1 システムの概要

電子商取引エージェントシステムの概要として，

1. **電子商取引の巡回パターン**
2. **保護対象とする秘密情報**
3. **仮想店舗の特徴**

について述べる．

本章で実験する電子商取引エージェントシステムは，電子商取引エージェントが，ネットワーク上の複数の仮想店舗を巡回し，ユーザの代わりに情報収集，電子決済などを行う．

本システムにおいて，電子商取引エージェントには複数の巡回パターンが用意されている．巡回パターンを用いることにより以下の利点が得られる．

巡回パターンを用いる利点

- 電子商取引の自動化を実現する．
- 巡回パターンを複数用意することでユーザの希望により近い取引方法を提供する．

電子商取引エージェントの巡回パターン

- 情報収集
  エージェントは複数の仮想店舗を巡回しながら，移動先の仮想店舗から商品の価格表を取得し，ユーザのホストに持ち帰る．
- 条件付き決済
  エージェントは複数の仮想店舗を巡回しながら，1）希望商品の有無，2）希望価格，3）仮想店舗の信頼度，などの決済条件を満たす仮想店舗が見つかれば，その時点で決済を行う．
- 価格優先決済
  エージェントは複数の仮想店舗を巡回後，希望商品の価格が最も安い仮想店舗に移

動し，決済を行う．なお，最安値の仮想店舗が複数存在する場合は，その中で最も信頼度が高い仮想店舗で決済を行う．

- 信頼度優先決済
  エージェントは複数の仮想店舗を巡回後，希望商品が有る信頼度が最も高い仮想店舗に移動し，決済を行う．なお，信頼度が最も高い仮想店舗が複数存在する場合は，その中で希望商品の価格が最も安い仮想店舗で決済を行う．

電子商取引エージェントシステムにおいて，エージェントが持つ秘密情報には，電子商取引で必要とするユーザの個人情報，セキュリティポリシ情報，仮想店舗で得られる情報などがある．また，これらの情報を使用するためのアクセサも秘密情報として扱う．以下に暗号化の対象となるこれらの秘密情報について述べる．

電子商取引エージェントで用いる秘密情報

- 電子商取引で必要な個人情報
  電子商取引エージェントはユーザの代理として，電子商取引に必要な個人情報を持つ．個人情報には，図6.1で示すように，ユーザの氏名，年齢，住所，電話番号，Eメールアドレス，購入希望商品名，購入希望価格，取引対象の条件[1]などがある．また，これらの情報は，情報の識別子と情報の内容の組により保持される．

```
NAME        :  Makoto HASEGAWA
AGE         :  26
ADDRESS     :  4-28 SHIJIMA KANAZAWA-SHI ISHIKAWA ‥
TELEPHONE   :  076-298-6212
E-MAIL      :  makoto-h@jaist.ac.jp
GOODS       :  ThinkPad
PRICE       :  200000
RELIABILITY :  4
```

図 6.1: 電子商取引エージェントの持つ個人情報

- セキュリティポリシ情報
  セキュリティポリシは，秘密情報に適用するセキュリティ用件である．セキュリティポリシ情報には，図6.2で示すように電子商取引エージェントの持つ各秘密情報の重要度が定義されている．

  セキュリティポリシ情報に記述されている情報の識別子は，上記の電子商取引で必要な個人情報で記述されている情報の識別子と対応している．したがって，図6.3の

[1]取引対象の条件とは，仮想店舗の信頼度を指し，ここで指定した信頼度より高い信頼度の仮想店舗だけを取引対象とする．

ように,「NAME」という識別子で対応しているセキュリティポリシ情報と個人情報から，情報の内容である「Makoto HASEGAWA」が重要度「1」の秘密情報であるといえる．

```
NAME        :   1
AGE         :   2
ADDRESS     :   4
TELEPHONE   :   3
E-MAIL      :   2
GOODS       :   1
PRICE       :   3
RELIABILITY :   3
```

図 6.2: 電子商取引エージェントの持つセキュリティポリシ情報

図 6.3: 情報の識別子による対応

- 移動先の仮想店舗で得られる情報
  電子商取引エージェントは，移動先の仮想店舗で，仮想店舗の名前，仮想店舗の信頼度，仮想店舗のアドレス，購入希望商品の価格，全商品の価格表，決済確認などの情報を取得する．これらの情報は他の仮想店舗に知られたくないので，取得後に暗号化し,エージェントが保持する．

- 秘密情報へのアクセサ
  電子商取引エージェントプログラムには，エージェントが保持する秘密情報を取り扱うための処理（アクセサ）が記述されている．この秘密情報へのアクセサは，秘密情報として扱い，暗号化による保護の対象とする．このように，秘密情報へのアクセサを暗号化することにより，秘密情報が必要な場合以外は秘密情報を扱えないようにする．

電子商取引エージェントは，移動先の仮想店舗と商取引を行う．仮想店舗の特徴は以下になる．

仮想店舗の特徴

- 取り扱う全商品の価格表を，取引相手の電子商取引エージェントに公開する
- 仮想店舗自身の信頼度を，取引相手の電子商取引エージェントに公開する
- 決済を行う際，必ず取引相手の個人情報を必要とする．
- 全商品には在庫が定められていて，購入された商品の在庫は減っていく．

仮想店舗は，電子商取引エージェントに取り扱う商品とその価格を提示する．同様に仮想店舗は自身の信頼度も提示する．このとき，もし仮想店舗がその信頼度の提示を拒否したり，明らかに不正な信頼度を提示するような場合は，電子商取引エージェントはこのようなホストを不正なホストと判断し取引を中断する．

6.1.2 電子商取引エージェントの動作と記述

電子商取引エージェントは，指定された巡回パターンで複数のホスト上の仮想店舗を巡回する．ことのき，電子商取引エージェントは，セキュリティポリシに合致した情報だけを復号化し，移動先の仮想店舗に公開する．また，移動先の仮想店舗から得た情報は暗号化し，次の仮想店舗に移動する．電子商取引エージェントは巡回パターンで定められた目的の処理が終了したらユーザのホストに帰還する．

本システムでは，複数の巡回パターンに応じて電子商取引エージェントが自律的に行動する．巡回パターンを複数用意することで，よりユーザの希望に近い取引の自動化を実現する．

以下に本システムにおける電子商取引エージェントの動きとして，全体像と各部の詳細について，実装コードの記述をふまえて述べる．

まず始めに，電子商取引エージェントシステムの準備としてユーザが行わなくてはならない処理について述べる．但しここで述べるのは基本的な処理のみとし，本システムを動作させるための実行環境，インストール方法，実行方法については付録 A にて説明する．

電子商取引エージェントの実行方法

1. 電子商取引エージェントを起動（図 6.4参照）
   AgentSpace 上から電子商取引エージェントをロードする．ロードボタンを押すことによりファイルダイアログが表示されるので,「TamperProofAgent.jar」を選択し，開くボタンを押すことにより電子商取引エージェントが起動する．
2. 巡回パターンを指定（図 6.5参照）
   電子商取引エージェントのポップアップメニューから希望する巡回パターンを選択す

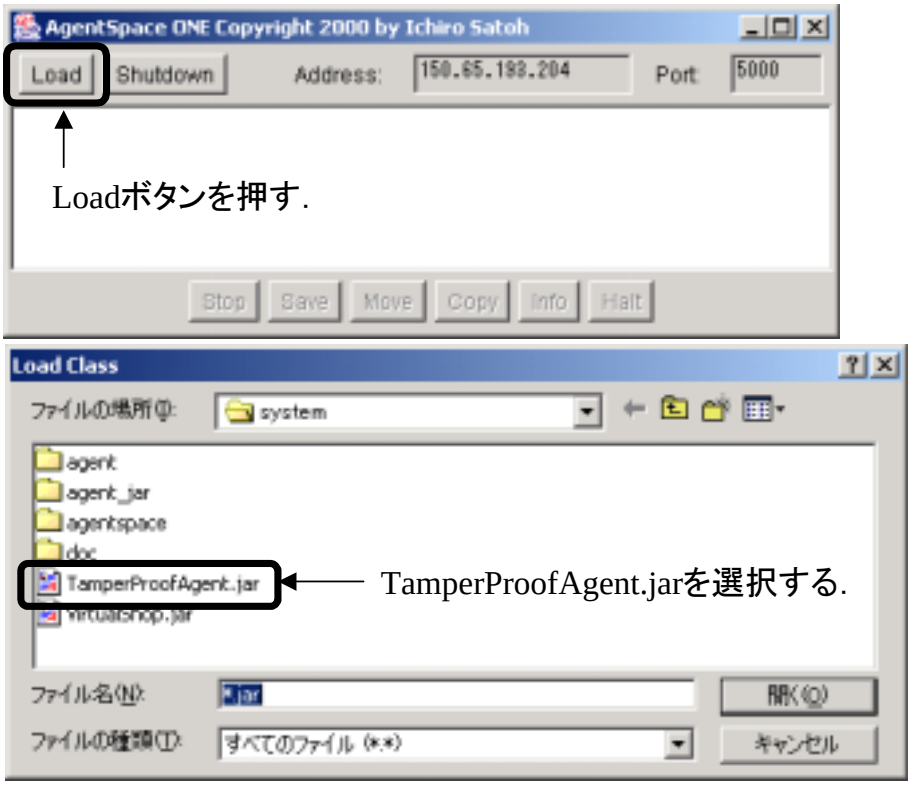

図 6.4: 電子商取引エージェントの起動

る．メニュー項目，Catalogue，Direct，Low priced，High Reliability はそれぞれ，情報収集，条件付き決済，価格優先決済，信頼度優先決済に対応する．

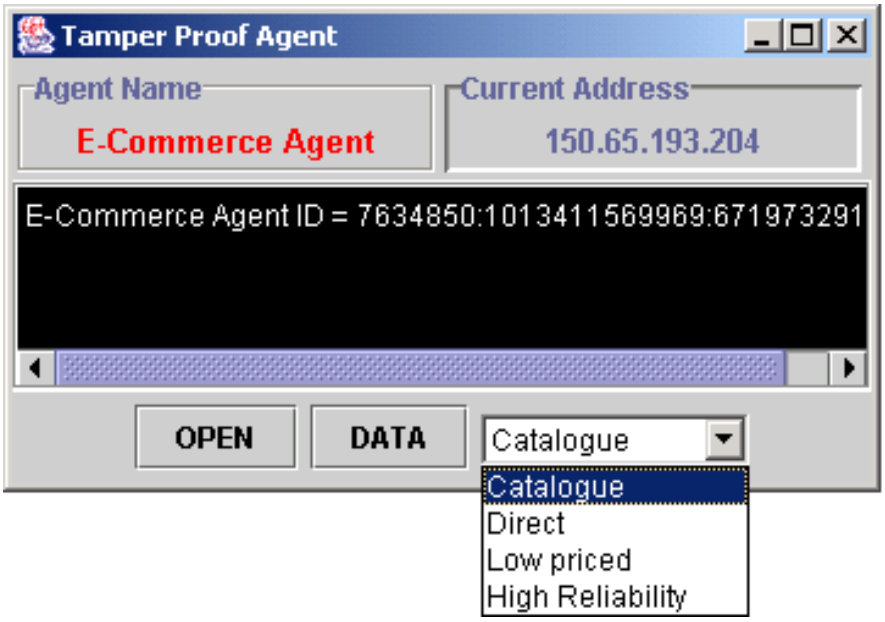

図 6.5: 巡回パターンの指定

3. セキュリティポリシファイルを入力（図 6.6参照）
   電子商取引エージェントの OPEN を押すことによりファイルダイアログが表示されるので,「security.policy」を選択し，開くボタンを押すことによりセキュリティポリシ情報が入力される．
4. 個人情報ファイルを入力（図 6.7参照）
   セキュリティポリシファイル入力後に再度ファイルダイアログが表示されるので,「private.data」を選択し，開くボタンを押すことにより個人情報が入力される．
5. 巡回対象となる仮想店舗のアドレスリストファイルを入力（図 6.8 参照）
   個人情報ファイル入力後も再度ファイルダイアログが表示されるので,「url.list」を選択し，開くボタンを押すことにより仮想店舗のアドレスリストが入力される．

以上の処理をユーザが行うだけで，電子商取引エージェントは自動的に仮想店舗を巡回し，巡回パターンに基づく電子商取引を行う．
電子商取引エージェントの動作の流れは以下となる．

電子商取引エージェントの動作の流れ

1. ユーザのホストで巡回パターンの選択およびセキュリティポリシ，個人情報，巡回対象仮想店舗アドレスリストを入力

図 6.6: セキュリティポリシファイルの入力

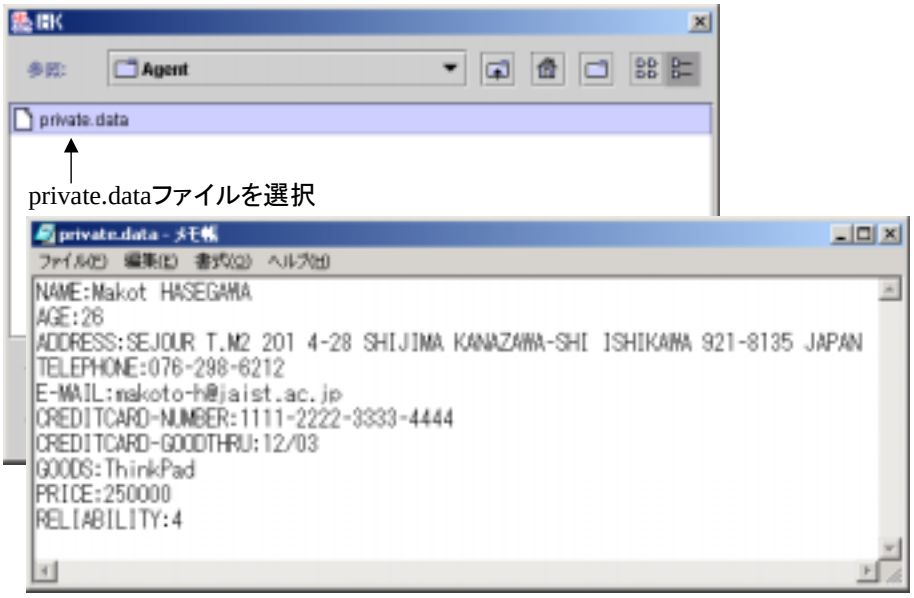

図 6.7: 個人情報ファイルの入力

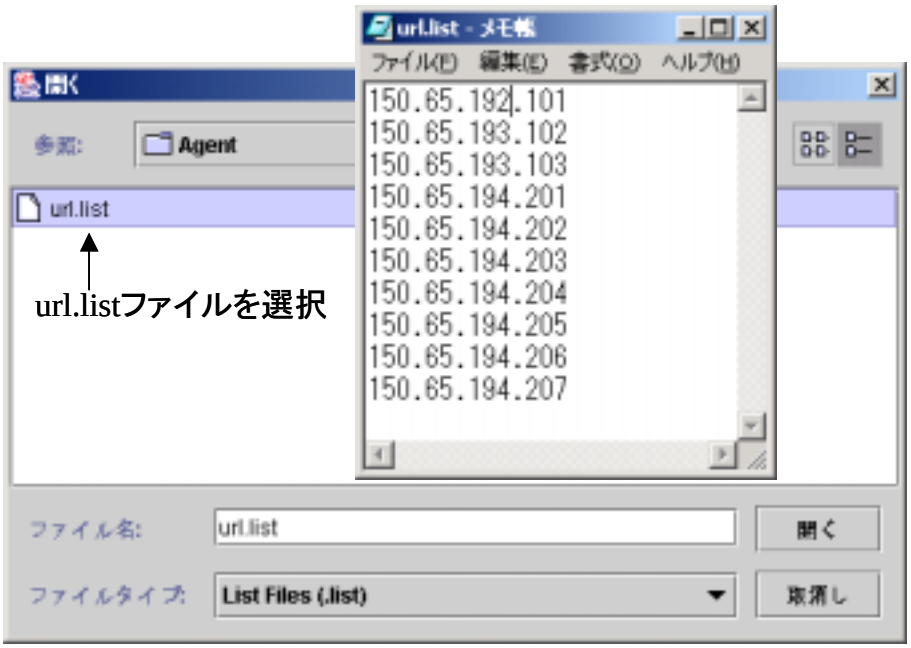

図 6.8: 仮想店舗のアドレスリストファイルの入力

2. 巡回マネージャにより移動先の決定

3. 移動先に到着後，その移動先がユーザのホストであるか，目的の仮想店舗か，それ以外の仮想店舗（巡回対象リストの 1 つ）であるか確認

   - ユーザのホストなら全ての情報を復号化し，得た情報を全て公開する．
   - 目的の仮想店舗なら公開情報選択プロトコル，決済プロトコルの実行
   - 巡回対象リストの 1 つであるなら信頼度の確認を行う

4. 信頼度が不正なら巡回マネージャにより移動先を決定し，信頼度が正当なら一般的なホストの動作を行う．

以下に巡回マネージャ，電子商取引プロトコル，一般的なホストの動作を説明する．

電子商取引エージェントの移動先は巡回マネージャによって決定される．巡回マネージャは以下の動作になる．

巡回マネージャ

巡回対象仮想店舗アドレスリストに移動先があるならそこに移動してリストからそのアドレスを削除する．リストが空ならターゲットホストセレクタにより巡回パターンにも

とづく移動先を決定する.

巡回マネージャの疑似コード

```
public void roundManager() {
  String url = null;
  AgentContext ac = getAgentContext();
  TamperProofAgent_Utilities utilities =
                              new TamperProofAgent_Utilities();
  if(iterator.hasNext()){              //リストにアドレスがある場合
    //そのアドレスを移動先と指定する
    url = (String)mapentry.getKey();
    //リストからそのアドレスを削除
    tourList.remove(url);
    //移動する
    ac.dispatch(url);
  }
  else {                               //リストが空の場合
    ECA_RoundManager rp =
      new ECA_RoundManager(birthURL,tourMode,
                           reliabilityList,priceList,virtualShopList);
    //ターゲットホストセレクタで移動先を決定
    targetHostURL = rp.targetHostSelector();
    //ターゲットホストセレクタで決定された仮想店舗へ移動
    ac.dispatch(utilities.urlParser(targetHostURL));
  }
}
```

ターゲットホストセレクタ

巡回パターンにもとづき移動先を決定する．情報収集，条件付き決済ならば，もう移動先が存在しないためユーザのホストに戻る．価格優先決済なら今まで巡回した仮想店舗で最低価格の仮想店舗を価格表リストより検索し，移動先に指定する．同様に信頼度優先決済なら信頼度が一番高い仮想店舗を移動先に指定する.

ターゲットホストセレクタの疑似コード

```
public URL targetHostSelector() {
```

```
    //巡回パターンが価格優先決済の場合
    if(tourMode.equals("Low priced")){
      targetHostURL = lowPrice(priceList,reliabilityList,
                               virtualShopList);
    }
    //信頼度優先決済の場合
    else if(tourMode.equals("High Reliability")){
      targetHostURL = highestReliability(priceList,
                                         reliabilityList,
                                         virtualShopList);
    }
    //情報収集，条件付き決済の場合はユーザホストアドレスを指定
    else{
      targetHostURL = birthURL;
    }
    return targetHostURL;
  }
```

電子商取引エージェントと仮想店舗間において，商取引開始から終了までは，基本的に以下の 3 つの取引プロトコルにより構成される．

電子商取引プロトコル

1. 価格表取得プロトコル
   電子商取引エージェントは，全ての巡回パターンで，移動先の仮想店舗から全商品の価格表を取得する．よって，電子商取引エージェントが仮想店舗の価格表を取得する処理は，この価格表プロトコルにより提供される．仮想店舗から取得した価格表は暗号化し，次のホストに移動する．

価格表取得プロトコルの疑似コード

```
    //価格表取得プロトコル
    public void getCatalogueTransaction() {
      //アクセサごと暗号化された全ての価格表リストを復号化
      catalogueListDeSerializable();
      AgentSecurityManager asm = new AgentSecurityManager(privateData);
      //価格表を取得
      catalogue = sendMessage(hostID,start,"catalogue");
      //価格表を暗号化
```

```
    catalogue = asm.hashmapEncoder(catalogue);
    //価格表をホスト名をキーとして価格表リストに追加
    catalogueList.put(getName(hostID),catalogue);
    //アクセサごと暗号化
    catalogueListSerializable();
  }
```

2. 公開情報選択プロトコル
   仮想店舗と取引を行うためには，ユーザの個人情報を公開する必要がある．しかし，移動先の仮想店舗に，電子商取引エージェントの持つ全ての情報を公開するとは限らない．そこで，この公開情報選択プロトコルでは，仮想店舗が取引に必要とする個人情報の識別子から，セキュリティマネージャがセキュリティポリシに合致する個人情報の識別子だけを選択する．

公開情報選択プロトコルの疑似コード

```
  //公開情報選択プロトコル
  public void baseTransaction() {
    AgentSecurityManager asm = new AgentSecurityManager(privateData);
    TamperProofAgent_Utilities utilities =
                                new TamperProofAgent_Utilities();
    //仮想店舗が要求する情報を取得
    request = sendMessage(hostID,start,"request");
    //仮想店舗が要求する個人情報のの Key だけを暗号化
    request = asm.hashmapKeyEncoder(request);
    //セキュリティポリシの value だけ復号化（Key だけが暗号化）
    securityPolicy = asm.hashmapValueDecoder(securityPolicy);
    //仮想店舗に公開できる情報を選択
    matchPolicy = asm.policyMatching(securityPolicy,request);
    //仮想店舗に公開できる情報だけで取引ができるか問い合わせる
    permission = sendMessage(hostID,
                             asm.hashmapKeyDecoder(matchPolicy),
                             "permission");
  }
```

3. 決済プロトコル
   この決済プロトコルでは，仮想店舗との決済処理を行う．電子商取引エージェントは，仮想店舗との取引に必要な個人情報から，セキュリティポリシに合致した情報だけを

復号化し，仮想店舗に公開し決済を行う．決済を終えた電子商取引エージェントは，決済情報を暗号化し，ユーザのホストへ移動する．

決済プロトコルの疑似コード

```
//決済プロトコル
public void buyTransaction() {
  //アクセサごと暗号化された個人情報を復号化
  privateDataDeSerializable();
  AgentSecurityManager asm = new AgentSecurityManager(privateData);
  TamperProofAgent_Utilities utilities =
                              new TamperProofAgent_Utilities();
  //仮想店舗に公開出来る情報を選択
  matchData = asm.securityManager(securityPolicy,
                                    privateData,request);
  //選択した情報を公開して商品購入
  buy = sendMessage(hostID,asm.hashmapDecoder(matchData),"buy");
  //決済情報を暗号化
  buy = asm.hashmapEncoder(buy);
  //復号化していたセキュリティポリシの値を再び暗号化
  securityPolicy = asm.hashmapValueEncoder(securityPolicy);
  //アクセサごと個人情報を暗号化
  privateDataSerializable();
  AgentContext ac = getAgentContext();
  //購入したら必ずユーザホストに帰る
  ac.dispatch(utilities.urlParser(birthURL));
}
```

一般的なホストの動作

始めに価格表取得プロトコル，次に公開情報選択プロトコルを実行する．仮想店舗が取引を拒否した場合は巡回マネージャにより移動先の決定，取引可能なら仮想店舗に購入希望商品の在庫が在るか確認する．在庫が無いなら巡回マネージャにより移動先の決定，在庫が在るなら巡回パターンにより以下の動作に別れる．

- 条件付き決済の場合
  条件を満たすなら決済プロトコルにより商品購入，満たさないなら巡回マネージャにより移動先の決定．
- 情報収集，価格優先決済，信頼度優先決済の場合

その仮想店舗の購入希望商品の価格，仮想店舗の信頼度，仮想店舗のアドレスを取得し，巡回マネージャにより移動先の決定.

一般的なホストの動作の疑似コード

```
//一般的な巡回対象ホストでの動作
public void regularHost() {
  AgentSecurityManager asm = new AgentSecurityManager(privateData);
  ECApplicationLogic eca = new ECApplicationLogic();
  getCatalogueTransaction();         //価格表取得プロトコル
  baseTransaction();                 //公開情報選択プロトコル
  //ホストが取引を許可する場合
  if(permission.get("ans").equals("true")){
    privateDataDeSerializable();
    //ホストに希望する商品が在る場合
    if(eca.stock(catalogue,privateData,asm.encode("GOODS"))){
      //巡回パターンに対応する動作
      regularHostTransaction();
    }
    //在庫が無い場合は次のホストに移動
    else{
      roundManager();
    }
  }
  //ホストが取引を許可しなかったので次のホストに移動
  else{
    roundManager();
  }
}
```

巡回パターンに対応する動作の疑似コード

```
public void regularHostTransaction(){
  AgentSecurityManager asm = new AgentSecurityManager(privateData);
  ECApplicationLogic eca = new ECApplicationLogic();
  //条件付き決済なら条件を満たすか判断
  if(tourMode.equals("Direct")){
    //条件が全て満たされるなら決済プロトコルにより商品購入
```

```
    if(eca.condition(asm.decode(clprice),asm.decode(pvprice),
                     hostrbty,asm.decode(pvrbty))){
      buyTransaction();
    }
    //条件が満たされないなら巡回マネージャにより移動先を決定
    else{
      securityPolicy = asm.hashmapValueEncoder(securityPolicy);
      roundManager();
    }
  }
```

以下が各巡回パターンのエージェントの動作である.

各巡回パターンにおける電子商取引エージェントの動作

情報収集におけるエージェントの動作

- 情報収集
  情報収集巡回パターンは以下のステップとなる.
  1. エージェントはユーザの個人情報を持たずユーザのホストを出発する.
  2. 巡回対象のホストを全て巡回したらステップ 4，巡回対象が残っているならステップ 3.
  3. 価格表取得プロトコルにより価格表取得し暗号化する．ステップ 2
  4. 巡回対象の全仮想店舗を巡回したらユーザのホストへ戻り，価格表を復号化する.

条件付き決済におけるエージェントの動作

- 条件付き決済
  条件付き決済巡回パターンは以下のステップとなる.
  1. エージェントはユーザの個人情報，セキュリティポリシ情報，購入条件情報を保持しユーザのホストを出発する.

2. 巡回対象のホストを全て巡回したらステップ 8，巡回対象が残っているならステップ 3.

3. 価格表取得プロトコルにより価格表を取得し暗号化する．

4. 公開情報選択プロトコルにより公開情報を選択する．

5. ホストに希望商品の在庫があるかチェックする．在るならステップ 6，無いならステップ 2

6. エージェントは移動先の仮想店舗でユーザの指定した条件で購入可能か判断する．可能ならステップ 7，不可能ならステップ 2

7. 決済プロトコルにより商品を購入し，決済情報を暗号化する．

8. ユーザのホストに戻り，取引で得た情報を復号化する．

価格優先決済におけるエージェントの動作

- 価格優先決済
  価格優先決済巡回パターンは以下のステップとなる．
  1. エージェントはユーザの個人情報，セキュリティポリシ情報，購入希望商品情報を保持しユーザのホストを出発する．

  2. 巡回対象のホストを全て巡回したらステップ 7，巡回対象が残っているならステップ 3.

  3. 価格表取得プロトコルにより価格表を取得し暗号化する．

  4. 公開情報選択プロトコルにより公開情報を選択する．

  5. ホストに希望商品の在庫があるかチェックする．在るならステップ 6，無いならステップ 2

  6. 情報収集として仮想店舗の価格，信頼度，アドレスを取得し，それらを暗号化する．ステップ 2

7. これまで取得した情報をもとに，一番価格が安い仮想店舗に移動する

8. 情報公開プロトコル，決済プロトコルにより商品を購入し，決済情報を暗号化する．

9. ユーザのホストに戻り，取引で得た情報を復号化する．

信頼度優先決済におけるエージェントの動作

- 信頼度優先決済
  信頼度優先決済巡回パターンは以下のステップとなる．
  1. エージェントはユーザの個人情報，セキュリティポリシ情報，購入希望商品情報を保持しユーザのホストを出発する．

  2. 巡回対象のホストを全て巡回したらステップ 7，巡回対象が残っているならステップ 3.

  3. 価格表取得プロトコルにより価格表を取得し暗号化する．

  4. 公開情報選択プロトコルにより公開情報を選択する．

  5. ホストに希望商品の在庫があるかチェックする．在るならステップ 6，無いならステップ 2

  6. 情報収集として仮想店舗の価格，信頼度，アドレスを取得し，それらを暗号化する．ステップ 2

  7. これまで取得した情報をもとに，信頼度が一番高い仮想店舗に移動する

  8. 情報公開プロトコル，決済プロトコルにより商品を購入し，決済情報を暗号化する．

  9. ユーザのホストに戻り，取引で得た情報を復号化する．

6.2 考察

本節では，例題である電子商取引エージェントによる実験について考察する．

6.2.1 耐タンパ・モバイルエージェントの動作

実験した電子商取引エージェントの行動は大きく以下の 2 つに分けられる．

- セキュリティに関する動作

- 電子商取引に関する動作

セキュリティに関する動作とは，電子商取引における秘密情報の保護手法である．本研究の電子商取引エージェントはセキュリティポリシにより，柔軟な秘密情報の暗号・復号化を実現した．このような動作は全てセキュリティマネージャで管理し，実現した．セキュリティマネージャにより実現した動作は以下になる．

セキュリティマネージャで実現した動作

- 仮想店舗の信頼度とセキュリティポリシに合致する情報の選択
  仮想店舗の信頼度と取引に必要な情報をもとに，セキュリティポリシに合致し，取引に最低限必要な情報だけを選択した．

- 合致した情報の暗号・復号化
  選択した情報のもつ識別子をもとに，実際の秘密情報を暗号・復号化機構を用いて暗号・復号化した．

このような動作によりユーザはセキュリティポリシの変更だけで，電子商取引エージェントで秘密情報を取り扱うことができた．しかし，本研究で用いたセキュリティポリシは非常に単純なものであるため，秘密情報の管理は非常に制限されたものであった．今後セキュリティポリシの記述形式をより考慮することにより柔軟な秘密情報の管理の実現が期待できる．

一方，電子商取引に関する動作とは，電子商取引エージェントが行う情報収集や電子決済のことである．電子商取引エージェントはユーザの個人情報から決済条件を読み取り巡回パターンを定められることで，ユーザの代理としてユーザの希望を反映した電子商取引を行った．ユーザの代理として行った動作は以下になる．

ユーザの代理として電子商取引エージェントが行った動作

- 仮想店舗の情報収集巡回対象となる仮想店舗の全ての商品と価格情報を収集
- ユーザの希望（商品，価格，仮想店舗）による購入希望購入商品と希望価格を満たす店舗での購入，購入希望商品が一番安い店舗での購入，購入希望商品を取り扱う一番信頼度の高い店舗での購入などをサポートする．

このように，ユーザは巡回パターンと決済条件が含まれる個人情報を入力するだけで，電子商取引を行うことが可能である．携帯端末などは入力インターフェースが制限されている場合が多い．このような環境においては一度だけ個人情報を入力しそれを端末内に保存し，取引ごとに変わると思われる希望商品などの情報だけを変更するといった使い方が可能な本システムは，ユーザの入力などの手間を軽減するのに非常に役立つと思われる．巡回パターンを増やすことにより，よりユーザの希望にあった取引の実現が期待できる．

6.2.2 セキュリティ機構の柔軟性

電子商取引エージェントの実験においては複数の暗号方式と難読化を用いてそれらを変更しながら，実際にエージェントの秘密情報の保護機構として情報を暗号・復号化しながら電子商取引を行った．

実験で用いた暗号方式は以下になる．

実験で使用した暗号方式

- シーザー暗号
- BASE64 エンコード

実験により，これらの暗号方式はモジュール化することにより容易な変更や再利用が可能であることが確認できた．

同様に実験で用いた難読化技術は以下になる．

実験で用いた難読化技術

- 関数名に意味のない名前使用
- セルに分割およびダミーセルの挿入
- 処理マルチスレッド化

これらは実装が容易であり，難読化としては非常に単純なものであるため，複雑な手順をふむような難読化手法の実装は今後の課題とする．実験では上記 3 つの難読化手法をいろいろな組合わせで適用したメタレベルを複数用意して，複数の暗号方式と共に変更しながらしかしながら実験を行うことにより，本研究で提案するセキュリティ機構の柔軟性が確認できた．さらにアプリケーションフレームワーク ART を用いることで，既存のモバイルエージェントにセキュリティ機構を容易に付加することが可能となる．

6.2.3 セキュリティ機構の問題点

本研究で提案するセキュリティ機構の問題点を以下に挙げる．

メタレベルの内部解析

本研究ではメタレベルとベースレベルの二層構造によりモバイルエージェントを構成することにより，柔軟性を得ることができたが，その結果内部解析の対象を限定される問題が生じる．既存の耐タンパ・ソフトウェアでは，アプリケーションロジックの中に保護機構も入り組ませることにより，保護機構をプログラム全体に隠蔽していた．このため保護機構を解析するにはプログラム全体を解析する必要があった．しかし，本研究では保護機構がプログラムの半分に片寄っているため，全体を解析しなくてもメタレベルだけを解析すればよい．この問題に対しては難読化を用いて保護機構の解析を困難なものにすることで対処する．また，この構造で得られた柔軟性により保護機構だけを頻繁に変更することで，保護手法の特定を困難にする．

第7章　関連研究

本章では，はじめに不正なエージェント，不正なホストのそれぞれのセキュリティ問題に対する既存の対処手法を紹介する．次に耐タンパ・ソフトウェアについて紹介し，その次に電子商取引で考慮すべきセキュリティ問題を説明し，最後にこれら関連研究を議論する．

7.1　不正なエージェントからホストを守る手法

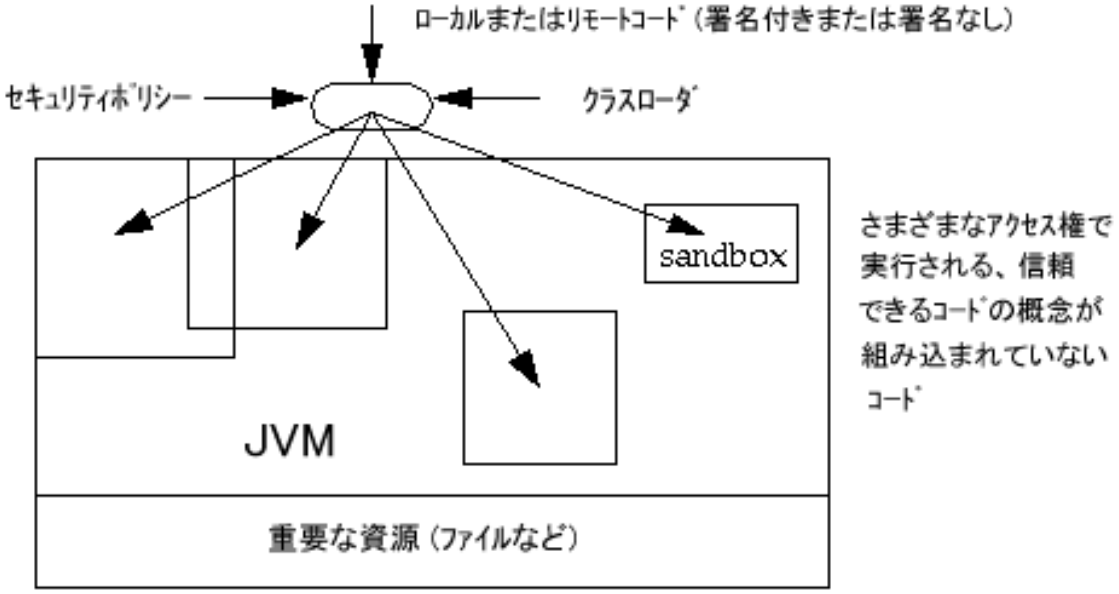

図 7.1: Java セキュリティモデル

モバイルエージェントは，ネットワーク上のホストに移動し，プログラムを実行する．そのため，モバイルエージェントの中にトロイの木馬のような悪意のあるプログラムが仕込まれていた場合，移動先のホストは，リソースを不正に使用される危険性がある．また，エージェントの作成者に悪意がなくても，エージェントプログラムのバグによる作成者の意図しないエージェントの動作により，移動先のホストのリソースを不正に使用される危険性もある．

このようなセキュリティ問題を引き起こす不正なエージェント[1]から移動先のホストを守る手法として以下が提案されている．また，これらの手法は既存のモバイルエージェントシステムにも取り入れられている．

- セーフインタプリタ
  エージェントの実行をインタプリタ上に制限し，インタプリタがエージェントのリソースアクセスを制御する手法である．インタプリタは，エージェントのリソースアクセス権を記録したACL（Access Control List）を読み込み，リソースアクセスを制御する．正確にはモバイルエージェントではないが具体的なのもとして，Javaのアプレットにみられるsandboxモデルがこの技術に当てはまると考えられる（図7.1）．

  セーフインタプリタは，悪意を持ったエージェント，エージェントプログラムのバグによる不正な動作の両方に対応できる．

- エージェント認証
  エージェントプログラムの作成者およびエージェントの所有者（エージェントオブジェクトの生成者）を認証し，信頼できるエージェントだけを実行する手法である．エージェントの作成者および所有者は，エージェントに電子署名を付加する．移動先のホストはこの電子署名でエージェントを認証する．

  エージェント認証は，悪意を持ったエージェントには対応できるが，エージェントプログラムのバグには対応できず，認証を通り抜けたエージェントがリソースを不正使用しても対処できない．

- 証明運搬コード（Proof-Carrying Code）
  証明運搬コード（PCC）は，答えをチェックすることは，答えを作り出すことよりしばしば簡単であるという観察に基づいている．エージェントが正しいというキーとなる理由を知っているのは，エージェントの作成者であって，移動先のホストではない．よってエージェントの作成者は，エージェントが安全に動作することを証明する証明書をエージェントに付加し，移動先のホストでこれをチェックする．証明書では，型安全やリソース使用の安全性が証明される．しかし，PCCは，1）形式的検証技術が必要である，2）CPU（DEC Alpha）インストラクションのレベルでしか実証されていない，などの困難さがあるものの，インタプリタやランタイムチェックなしにエージェントを安全に実行できる．

[1]本論文では，悪意のある動作によるリソースの不正利用や，バグにより作成者の意図しない動作によるリソースの不正利用などを行うエージェントを不正なエージェントとする．

7.2 不正なホストからエージェントを守る手法

モバイルエージェントは，移動先のホストに対して自身の持つ情報を秘密にするのは非常に困難だとされている．これは，移動先のホストがエージェントを実行するには，エージェントのプログラムコード，エージェントの実行状態などを知らなければならないためである．このため，移動先のホストの管理者が悪意を持っていたり，あるいは移動先のホストが悪意のある者の制御下にあった場合，エージェントの情報は，盗み見，内部解析，改竄などの攻撃により不正使用される危険性がある．

このようなセキュリティ問題を引き起こす不正なホスト[2]からエージェントを守る手法として以下が提案されている．

7.2.1 ホスト認証

モバイルエージェントが移動先のホストを認証し，危険だと思われるホストには近づかないことによりエージェントを守る手法である．同様の手法として，移動先のホストが所属するセキュリティドメインを認証するセキュリティドメイン認証 [6] がある．一般にセキュリティドメインとは，同質のセキュリティが保持されていることが期待される領域のことで，会社の一部門の組織などが想定される．これらの手法では，エージェントは不正なホストに移動しないため攻撃を受けないが，信頼できるホストおよびドメインの決定基準が問題となる．

7.2.2 Mobile Cryptgraphy

暗号化されたまま実行可能なプログラムの概念を定式化し，それをモバイルエージェントの保護に応用することにより，盗み見，特定の目的を持った改竄などの攻撃からエージェントを守るための手法である [12]．具体的には，CEF（暗号化された関数による計算：Computing with Encrypted Function）という概念で，エージェントプログラムを暗号化したまま移動先のホスト上で実行することにより，特定の目的を持った改竄を防ぐことができる．しかし，Mobile Cryptgraphy は論理的には実現可能であるが，モバイルエージェントに適用するには，1）指数的な規模になる万能ブーリアン回路の使用，2）CEF プロトコルにおける膨大な量のコミュニケーション，3）任意のプログラムを CEF で保護できる暗号方式が存在しない，などの問題がある．以下に CEF プロトコルの例を示す．

- 要求
  Alice は関数 f を計算するアルゴリズムを持っている．Bob は入力 x を持っていて Alice のために f(x) を計算してあげたい，しかし，Alice は Bob が f に関する本質的

[2]本論文では，エージェントに対して，盗み見，内部解析，改竄などの攻撃を行うホストを不正なホストとする．

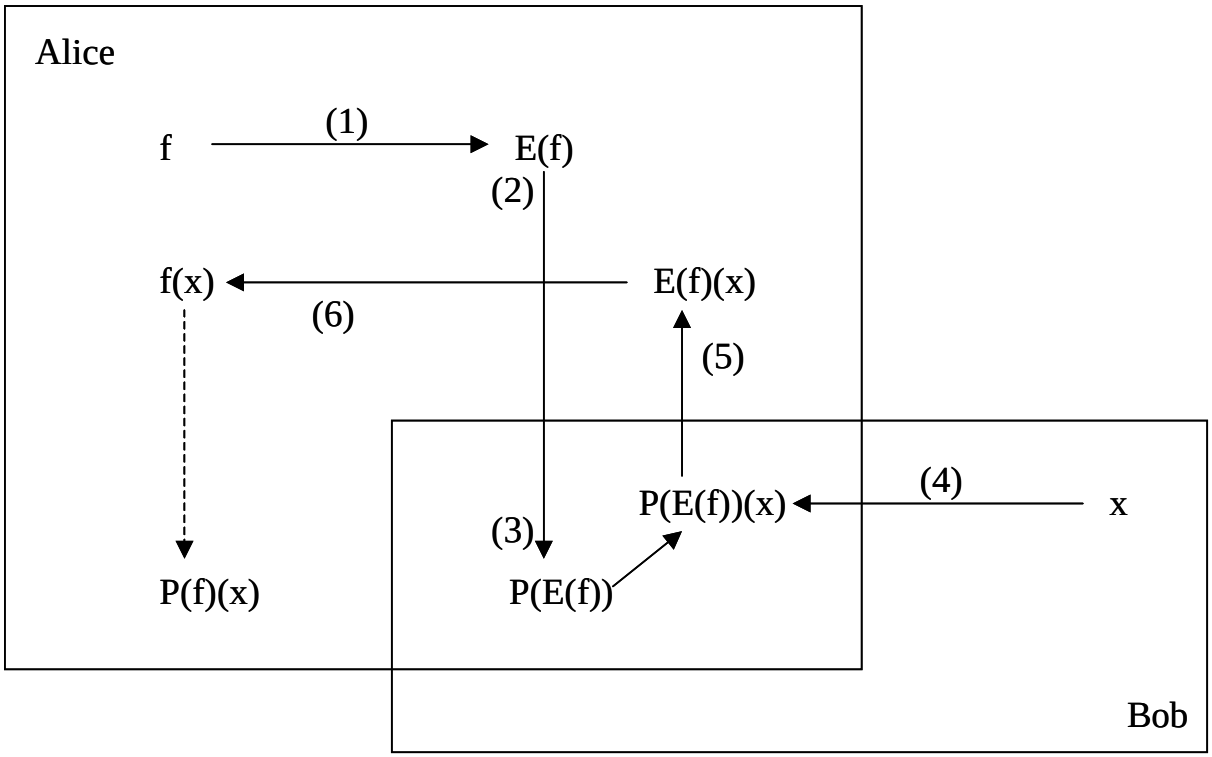

図 7.2: CEF プロトコルの例

なことを一切知ることを望まない．さらに，Bob と Alice は f(x) の計算中に対話を必要としない．

- CDF プロトコルの手順（図 7.2参照）

(1) Alice は f を暗号化する．
(2) Alice は $E(f)$ を実装するプログラム $P(E(f))$ を作成する．
(3) Alice は $P(E(f))$ を Bob に送る．
(4) Bob は $P(E(f))$ を x に関して計算する．
(5) Bob は $P(E(f))(x)$ を Alice に送る．
(6) Alice は $P(E(f))(x)$ を解読して $f(x)$ を得る．

7.2.3 エージェントの難読化

プログラム難読化には明確な定義がある訳ではないが，プログラムアルゴリズム解析を困難にし，解析に手間や時間を要するように仕向ける技術である．エージェントプログラムを難読化することにより，不正なホストの盗み見，特定の目的を持った改竄を困難にする．しかし，現在難読化には絶対的なアルゴリズムが存在しないので，モバイルエージェントのセキュリティ機構としては，難読化したエージェントプログラムの解析時間をエージェントのライフタイムとして制限する Time Limited Blackbox Security 手法 [15] が提案されている．

プログラムの難読化の簡単な例を以下に示す．

- 変数名，手続き名に意味のない名前を使用した難読化
  Java 関連の開発ツールとして，クラス・データを入力すると変数名やクラス名，メソッド名などを，数字を組み合わせた，意味のないシンボルに置き換えたクラス・データを出力するものが開発されている（図 7.3）．しかし，プログラムで使用されるクラス・データのシンボルを別のものに置き換えただけでは，プログラムの制御構造自体は保存されたままである．このため，デバッガを用いた逆アセンブルにより容易に解析できる．

- 偽装コードの埋め込み
  偽装コードを使用した難読化として，1）レジスタ加算を行ったら必ず減算を行うなど，コード列として実行時に意味のある働きを持たないコードを挿入する手法を新規生成型偽装コード，2）オリジナル・コードの一部を流用して，実行時にオリジナル・コードと同じ働きをするタイプの複製生成型偽装コード，などが提案されている [13][14].

- 並列処理の利用
  ある実行スレッドが並列に動作しているスレッドとの間で密に連携を取り合ったり，

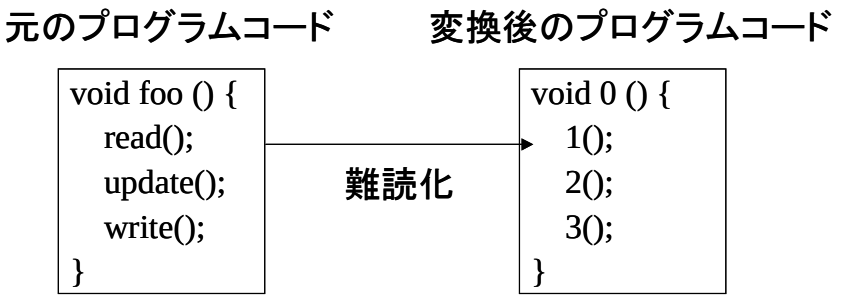

図 7.3: 意味のない名前を使用した難読化の例

変数を共有したりすることによって，解析しようとしているコード部分以外の要素で，変数の内容や動作に変動が生じる．そのような変動のために，プログラムを解析するための手がかりをつかみにくくする．

- データ駆動型プログラムによる難読化
  図 7.4は，順次的な制御の流れを，制御変数 c と $swich$ 文，ループを用いて書き換えた例である．Obfoo() は制御変数 c の初期値に外部から 1 を与えられることで foo() と同じ処理を行う．ただし，c の初期値が 1 であることを知らない限り，Obfoo() が foo() と同じ処理を行うことは分からない．このため，c という変数に制御の流れを隠していると考えることができる．

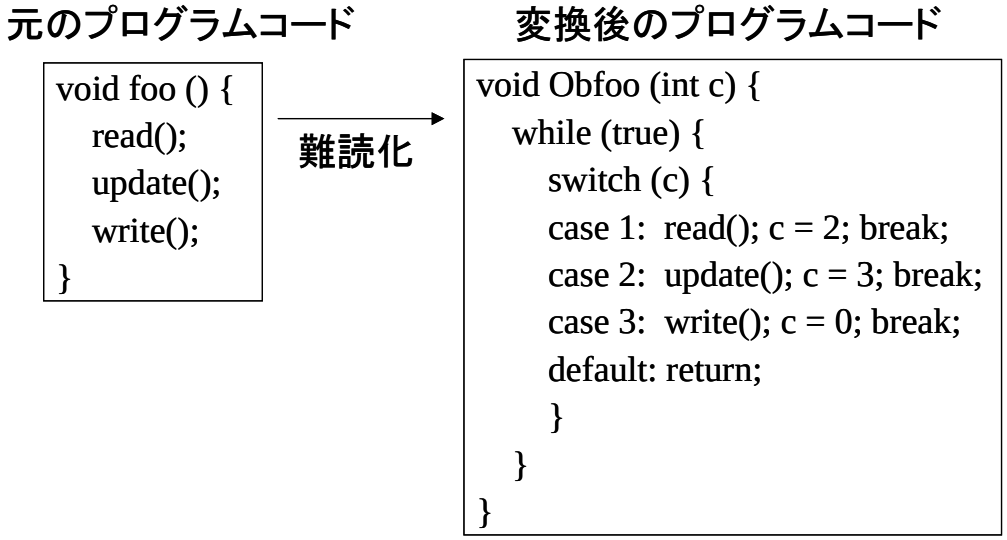

図 7.4: データ駆動型プログラムによる難読化の例

- メタ・プログラムによる難読化
  実行モジュール自体には，メタ・レベルのインタプリタのみを与え，問題となるア

ルゴリズムを，そのインタプリタによって処理されるデータ内に隠蔽する．例えば図 7.5のような文字列をデータとして LISP 風のインタプリタに評価させる．

Java ベースのエージェントの場合には，アプリケーション専用の仮想マシン M を用意し，エージェントのデータ領域にその M によって記述されたコードを保持すると同時に，エージェント自体に，仮想マシン M のインタプリタとしての機能を持たせる．このように Java 仮想マシン上でエージェントの一部である M 仮想マシンを動作させ，M で記述されているコードを実行することにより，解析をより困難なものとする．

```
(define Obfoo
        (
                lambda()(
                        (prog1)
                        (prog2)
                        (prog3)
                )
        )
)
```

図 7.5: メタ・プログラムによる難読化の例

7.2.4 Nバージョン難読化エージェント

モバイルエージェントのセキュリティ技術を，1）情報隠蔽理論，2）フォールト・トレラント・ソフトウェア理論，3）ソフトウェア検証理論，などの三者が合流する地点に形成されていくという視点に立脚して提案された手法である．具体的には図 7.6に示すように，別々の難読化を施したエージェントプログラムを N 本用意し，実行時に同一のマシンもしくは異なるマシンでその N 本のエージェントを実行させ，結果の有効性を多数決で決定する．

このような手法は元々ソフトウェアの信頼性向上を目的としているが，N 本のエージェントプログラムは，それぞれ難読化されていて内容の解読を困難にしているため，不正なホストの盗み見や内部解析に対処できる．また，結果の多数決を取ることにより，不正なホストによるエージェントの不正利用や改竄の有無を検出できる．

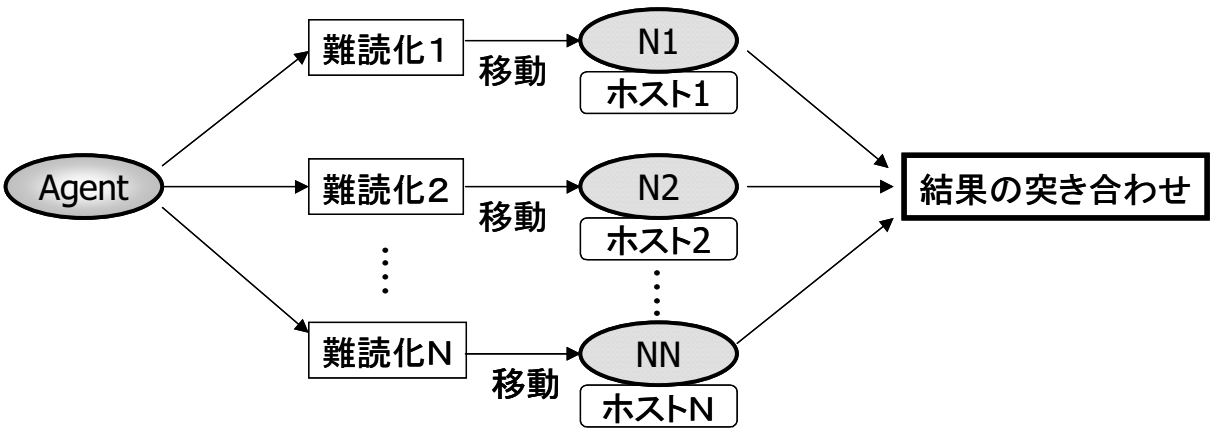

図 7.6: Nバージョン難読化エージェント

7.2.5 巡回・秘密情報管理エージェント

秘密情報をエージェント本体から分離してユーザのホスト上で管理し，秘密情報へのアクセスを制限する手法である．

7.2.6 実行トレース

エージェントの各ホスト上での実行トレースを検査することで，エージェントが改竄，不正使用されたことを検出するための手法である [16][17]．実行トレースとは，例えば図7.7のような，外部からエージェントへ入力された値の履歴である．実行トレースを用いた手法としては，エージェントがネットワーク上の複数のホストを巡回し，出発元のホストに戻ってきてから，全ての実行トレースを検査する手法や，次のホストに移動する度に，前のホストの実行トレースを検査する手法が提案されている．これらの手法では，実行トレースは正しいという前提のもと，エージェントが不正使用や改竄されたことは検出できるが，攻撃そのものを防ぐことはできない．

10 read(x) 11 y=x+z 12 m=y+1 13 k=cryptInput 14 m=m+k	10 x=5 11 12 13 k=2 14
Code fragment	**Trace of code fragment**

図 7.7: 実行トレースの例

7.3 耐タンパ・ソフトウェア

本節では，David Ausmith, Gray Graunke[18] による「耐タンパ・ソフトウェア（Tamper-resistant Software）[3]」の概念について説明する．

耐タンパ・ソフトウェアとは，ソフトウェアの不正な内部解析，改変を防止する仕組みを持つソフトウェアのことである．ソフトウェアの知的財産を保護するため，金融関係の処理や電子認証などを実装するソフトウェアで要求される高度な機密性を実現するために，そのようなソフトウェアを開発する技術が必要となる．従来から，ソフトウェアの違法コピーの防止，不正なユーザによる使用を防止するために，パーソナル・コンピュータ用のソフトウェアを中心に，プログラムの難読化やコピープロテクトなどが，試みられてきた．しかし，それらはアド・ホック的なもので，一般的に利用可能な技術としては定着していなかった．Aucsmith らの耐タンパ・ソフトウェアは，そのような状況の中で，コンセプト，アーキテクチャ概要，開発方法の概要を公開した研究成果である．

7.3.1 設計コンセプト

この耐タンパ・ソフトウェアは，主にパーソナルコンピュータをターゲットとするソフトウェアの保護を目的とする．そこで，このようなソフトウェアに対する攻撃のカテゴリを以下の 3 種類に分類する．

カテゴリ 1 通信手段を介したソフトウェアの攻撃.

カテゴリ 2 パソコン上などで動作する攻撃用プログラムを利用した攻撃.

カテゴリ 3 デバッガ，インサーキット・エミュレータ（ICE），ロジック・アナライザなどを使い，ターゲットコンピュータを掌握した状態にて行う攻撃.

このとき，カテゴリ（3）の攻撃に対しては，部分的な解決にとどめてる．その決定は，全ての攻撃に対処するにはコストがかかりすぎること，パーソナル・コンピュータ向けのソフトウェアの攻撃によって得られる利益では，悪者にとって，高価なインサーキット・エミュレータなどを入手するコストに見合わないこと，などの配慮に基づく．さらに，耐タンパ・ソフトウェアを設計は以下の 4 つの基本原則を適用する．

原則 1 時間軸，空間軸の両方に秘密の要素を分散させる.
秘密の要素を，主記憶装置の単純なスキャニングによって取得できるような連続アドレスに存在させない.

[3]不正な内部解析（逆解析）や改変に対して防護機能を備えるソフトウェア.「tamper」とは，不正に中身をいじるといった意である．従来の試みとしては，逆アセンブラなどで簡単に解析できないようにする難読化技術や，デバガで解析しようとすると暴走してしまうような技術などが市販パッケージ・ソフトウェアに組み込まれた実績がある．

原則 2 処理をばらばらに分割し，難読化する．
処理を細かく分割し，それぞれをインタリーブして実行することにより，難読化を実現する．

原則 3 プログラムごとに異なるコードを組み込む．
プログラムの定型部分を狙った攻撃を阻止するために，プログラムごとに異なる暗号化鍵データやコード・シーケンスを使用する．

原則 4 相互チェックにより実行状態を検証する．
協調動作している全てのプログラムが正しく動作しているかを相互チェックにより正しさの検証を行う．

7.3.2 アーキテクチャ

Aucsmith らの耐タンパ・ソフトウェアは，3.3.1 で述べた 4 つの基本原則にしたがい，IVK（Integrity Verification Kernel）と呼ぶ小さなプログラムモジュールを開発し，保護したいプログラムに IVK を組み込むことにより耐タンパ・ソフトウェアを実現する．IVK には，1）コード・セグメントまたはプログラムの正しさの検証，2）他の IVK との通信，などの機能を備える．IVK は単体でもターゲットプログラム[4]が正確に実行されているかの監視，検証が可能であるが，IVK を複数用意し相互チェックを行わせることによってより信頼性，安全性を高めることができる．

このとき，IVK には，高度な機密性が要求される．そのため，IVK には以下のような保護対策がなされている．

1. タスクのインタリーブ
   IVK は必要に応じて，さまざまな別のタスクの実行を管理できる．例えば，3 つタスク A，B，C があり，分割したソフトウェア部品をそれぞれ a_i, b_i, c_i とする．この場合 IVK は「$a_1a_2a_3a_1b_2b_3b_1c_2c_3$」と順次に実行させるのではなく，「$a_1b_1c_1a_2b_2c_2a_3b_3c_3$」という風に実行し，プログラムを解析しようとする側に，理解し難しいように実行させることができる．
2. 秘密の要素の分散格納
   IVK のバイパスを防ぐために，IVK には少なくても 1 つの秘密が埋め込まれている．通常，この秘密は署名を行うための，秘密鍵である．この秘密鍵は小さく分割し，IVK 全体に散りばめ隠す．
3. 難読化されたコード
   IVK のプログラム自体が暗号化され，そのプログラムは実行時に自分自身を書き換えながら実行する．復号化した部分は，処理が終わり次第，再び暗号化する．さらに，同じメモリ番地が異なるオペレーション・コードで再利用される．

[4]IVK を組み込むことにより保護の対象とするプログラム

4. 組み込むごとにコードを変える
   IVK はターゲット・プログラムごとに違うものが生成され組み込まれる．つまり，組み込み先のコードの性質や暗号化鍵データに応じて異なる IVK をその都度生成する．これにより，悪者が複数種のターゲット・プログラムを入手して IVK を比較しても，共通部分を抽出しにくくする．
5. 決定的な挙動を示さない
   可能なときには，プラットホームが提供するマルチ・スレッドの機能を利用することにより，スレッドの実行順序がプログラミングだけでは決まらないようにし，実行順序の追跡を難しくする．

7.3.3 IVK の作成方法

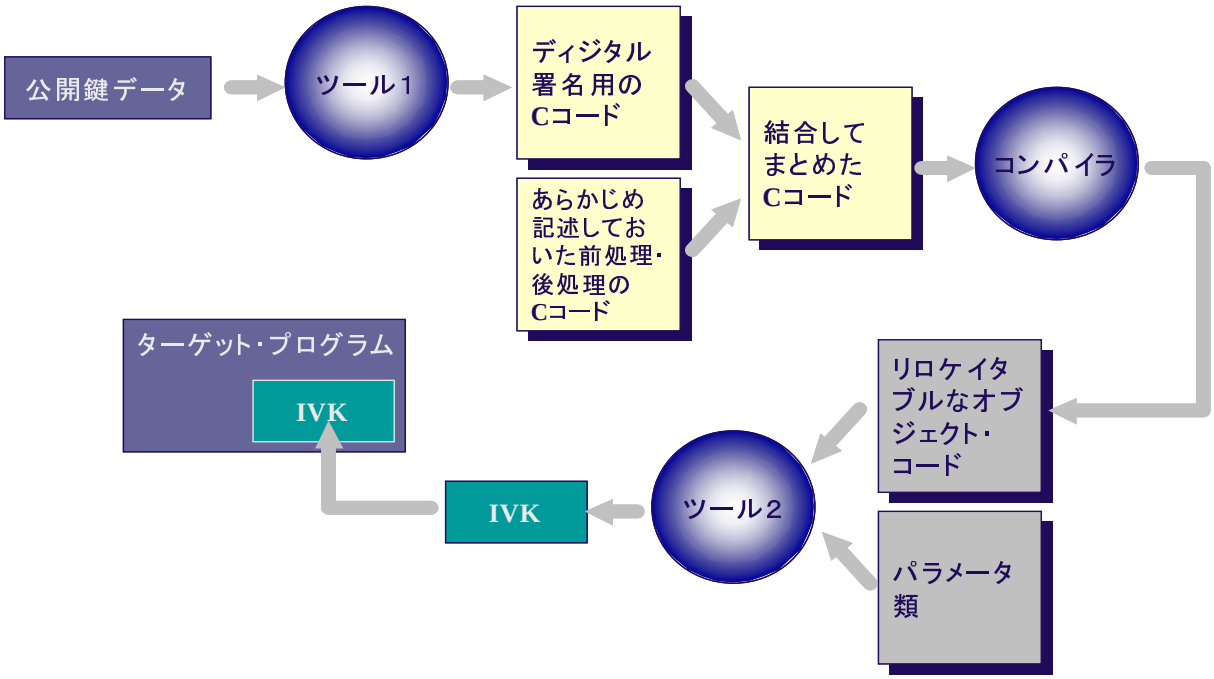

図 7.8: IVK の作成の流れ

IVK は基本的には，C 言語によるプログラムを想定している．IVK の作成には，1）固有のソース・コード生成ツール，2）オブジェクト・コード・レベルの変換ツール，などの 2 種類のツールが用いられる．図 7.8で示すように，IVK の作成からターゲット・プログラムへ組み込むまでの流れは以下の 5 ステップからなる．

ステップ 1 ツール 1 に公開鍵データを入力する．
入力した鍵データを使い，あとで IVK の正しさを調べる．ツール 1 は，公開鍵デー

タ用の予備計算を実行し，デジタル署名処理のCコード（C言語で記述したソース・コード）を生成する．

ステップ2 生成したCコードをほかのCコードとインタリーブしながら組み合わせる．
ほかのCコードとは，IVKのコードを実行するのに必要な前処理，後処理などをあらかじめ記述したものとする．

ステップ3 IVK用のオブジェクト・コード生成．
インタリーブを終えたソース・コードをコンパイラにかけてIVK用のオブジェクト・コードを生成する

ステップ4 ツール2にオブジェクト・コードを入力する．
ツール2は，オブジェクト・コードを暗号化されたバイト列にする．このバイト列がIVKになる．

ステップ5 IVKの組み込み．
ターゲット・プログラムはあらかじめIVKを埋め込む場所を開けておく必要がある．そこにIVKを組み込み，作業は終了する．

7.4 電子商取引システムのセキュリティ機構

電子商取引という言葉は非常に広い意味で使用される．本研究ではBtoCと呼ばれる企業対消費者間のインターネット販売を対象とする．インターネット上には，電子マネーやネットバンキングなどを利用した様々な電子商取引システムが存在するが，クレジットカードを用いて電子決済を行う場合に消費者が被害にあうセキュリティ問題としては，主に以下が考えられる．

1. 取引相手の仮想店舗が存在しない．

2. 仮想店舗との通信データが途中のサーバで盗聴され，悪用される．

3. クレジットカードの情報が仮想店舗により悪用される．

4. 消費者の個人情報が仮想店舗により悪用される．

1,2の問題に対処する手法として，電子決済にSSL（Secure Socket Layer）プロトコルが用いられる．SSLでは通信相手の認証および通信データの暗号化が行われるため，実在する取引相手と安全に通信を行うことができる．しかし，SSLを使用したクレジットカード決済の場合，消費者から仮想店舗には安全にカード情報が送信されるものの，3の

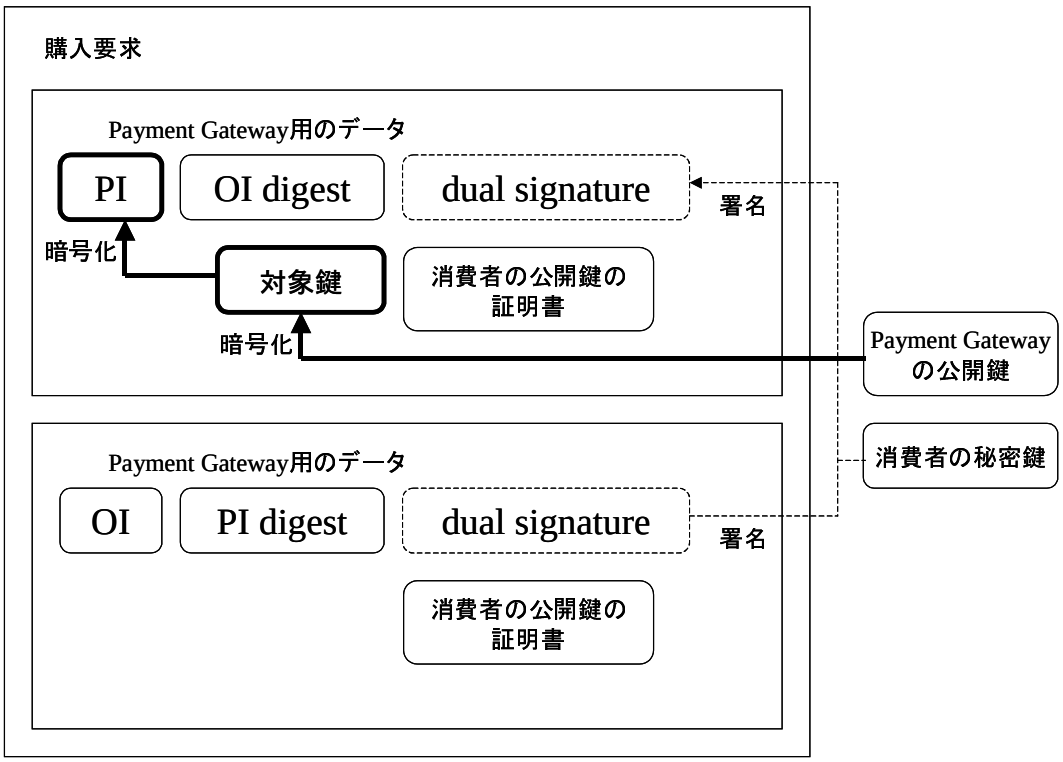

図 7.9: SET の購入要求データの内容

問題である仮想店舗がカード情報を悪用する危険性が残る．この問題に対応する決済用のプロトコルとして，Visa や MasterCard などにより開発された SET（Secure Electronic Transaction）がある．SET では，カード情報がクレジットカード認証機関である Payment Gateway の公開鍵で暗号化されて仮想店舗に送信されるため，仮想店舗がカード情報を盗み見することができない．

以下に，インターネット上でカード決済を行う場合の一般的な購入手順を示す．

(1) 消費者が商品を見る（Web, CD-ROM）．

(2) 消費者が商品を選ぶ．

(3) 仮想店舗がオーダ・フォーム（注文票）を提供する．

(4) 消費者が仮想店舗にオーダを送る．

(5) 消費者がクレジットカードを選択する．

(6) **消費者が仮想店舗に購入要求を送る．**

(7) **仮想店舗が Payment Gateway にカードの認証依頼を送る．**

(8) **仮想店舗が消費者にオーダ確認を送る．**

(9) 仮想店舗が消費者に商品を発送する．

(10) **仮想店舗が Payment Gateway に支払いを要求する．**

このうち SET で規定されているのは（6），(7)，(8)，(10）の手順である．手順（6）で消費者から仮想店舗に送られる購入要求には，オーダ（OI:Order Information）と支払い指示（PI:Payment Instruction）が含まれる．クレジットカードの情報は PI に含まれる．OI は Payment Gateway に見せる必要はなく，PI は仮想店舗に見せる必要はない．SET では，必要のないデータを見せないようにすることでデータの悪用を防いでいる．これを実現するために購入要求のデータは図 7.9に示す内容となっている．図中の dual signature とは，OI のメッセージダイジェストと PI のメッセージダイジェスト連結したもののメッセージダイジェストである．

$$dualsignature = MD(MD(OI) + MD(PI))$$

表 7.1: 盗み見や内部解析への対処

既存の対処手法	盗み見や内部解析への対処
ホスト認証	事前に不正なホストがわかっていなければならない.
Mobile Cryptography	理論的には対処できるが，実行できるシステムがない.
エージェントの難読化	時間制限付きでしか対処できない.
Nバージョン難読化エージェント	時間制限付きでしか対処できない.
実行トレース	対処できない.

7.5 議論

本章では，不正なエージェントの攻撃に対処する手法から，本研究が対象とするモバイルエージェントのセキュリティ問題である不正なホストによる攻撃の対処手法，また非常に関連が深い耐タンパ・ソフトウェアや電子商取引のセキュリティ機構について説明した．不正なホストの攻撃に対処する既存の手法を表7.1にまとめたが，表で示すように既存の手法では不正なホストの攻撃に十分に対処できない．これは移動先のホストがエージェントプログラムを解釈し，実行しなければならないためである．このため，モバイルエージェントを移動先のホストの盗み見や内部解析から保護することは非常に難しい問題である．

本節で既存の関連研究について成果や課題，本研究への適用性について議論する．

ホスト認証はエージェントの移動先のホストを認証し攻撃される可能性があるホストには近づかないようにするため，認証結果が確実なものであるという前提下ではほとんどの攻撃を避けることができる．課題としては，信頼できるホストの決定基準が問題となり，本研究で用いたホストの信頼度についても同様な問題であるといえる．本研究ではホスト認証を保護機構の 1 つとして取り入れた．

Mobile Cryptgraphy は論理的なアプローチであり実装できるシステムが存在しない．

エージェントの難読化は，モバイルエージェントプログラムの解析を困難にする技術である．しかし，絶対的なアルゴリズムは存在しないためなんらかの制約が必要である．Time Limited Blackbox Security 手法はプログラムの解析時間がエージェントのライフタイムとして制限する手法であり，難読化手法の 1 つとして参考にしたい．

N バージョン難読化エージェントは，ソフトウェアの信頼性を向上する技術と難読化を取り入れることでセキュリティ技術として応用しているものである．既存の技術を取り入れるといった点では，本研究と同じ方向性にあるのかもしれない．しかし，実用性は考慮されていなく，モバイルエージェントの柔軟性が損なわれている手法となっている．

実行トレースはエージェントの各ホスト上での実行トレースを検査することで，エージェントが改竄や不正使用されたことを検出する手法である．実行トレースは正しいとい

う前提のもとではあるが，本研究への適用は容易であると考えられる．

耐タンパ・ソフトウェアは，ソフトウェアに耐タンパ機構を備えることで，不正な内部解析や改竄に対処する手法である．本研究はモバイルエージェントに耐タンパ機構を備える目的なので，この耐タンパ・ソフトウェアとは近い関係がある．耐タンパ・ソフトウェアは，保護対象となるアプリケーション全体に，小さく分割された保護機構を分散し実行することでソフトウェアの正当性を証明する技術である．このように保護機構はソフトウェア全体に分散されているので，保護方式を解読されてしまったからといって，保護機構だけを変更することができず，ソフトフェア全体の再構築が必要になる．また，アプリケーションの変更時にも同様のことがいえる．ほとんどの耐タンパ・ソフトウェアがこのようなアドホックな実装となっている．

本研究の耐タンパ・モバイルエージェントは保護機構とアプリケーションロジックを分離した構造をもつ．このためアプリケーションロジックの変更と保護機構の変更を独立して行うことが可能である．しかし，このような構造は保護機構が全体構造の半分に片寄っているため，内部解析の対象を保護機構に限定される可能性がある．これは，暗号化された情報を解読するよりも，なにも保護されていない保護機構を解析するこで情報を手にいれた方が容易なためである．この問題に対しては難読化を用いて保護機構の解析を困難なものにすることで対処する．また，この構造で得られた柔軟性により保護機構だけを頻繁に変更することで，保護手法の特定を困難にする．

第8章　おわりに

本章では，本研究全体をまとめ，本研究で得られた結論と今後の課題を述べる．

8.1　まとめ

本研究では，耐タンパ・モバイルエージェントを実現するセキュリティ機構を提案した．

本研究で提案したセキュリティ機構は，不正なホストによるモバイルエージェントへの攻撃，特に不正な内部解析や盗み見に対する防御機構を持つモバイルエージェント（耐タンパ・モバイルエージェント）を柔軟に構成するためのアーキテクチャである．

モジュール化した暗号・復号化機構を各モバイルエージェントに導入することにより，アプリケーションレベルでのデータの保護と，暗号技術の柔軟な変更や再利用を実現した．各モバイルエージェントは秘密情報に適用するセキュリティ用件を定義したセキュリティポリシ保持し，これを変更することで柔軟なセキュリティ対策が可能となる．

また，提案したセキュリティ機構を Java ベースのアプリケーションフレームワークである ART（Architecture for realizing Tamper-proof-agent）として実装した．

ART を用いることにより，暗号技術の柔軟な変更が可能な耐タンパ機構を持つモバイルエージェントを容易に作成することができる．

最後に，ART を用いて電子商取引エージェントを作成し，提案したセキュリティ機構の柔軟性を確認した．

8.2　今後の課題

今後の課題としては以下が挙げられる．

- セキュリティポリシの記述方式
  本研究では非常に単純な記述形式のセキュリティポリシを用いたが，セキュリティポリシはその記述方式だけで 1 つの研究分野とされるほど，セキュリティにとって重要なものである．より柔軟なセキュリティ機構を構築するには，セキュリティポリシの記述方式を改善する必要がある．

- 保護機構の難読化技術
  本研究の保護機構には，難読化技術が必要不可欠であるが，ART 用いた難読化技術

は，実装が容易で比較的単純なものである．しかし，本研究のセキュリティ機構をより実用的なものにするには，複雑な難読化技術を導入する必要がある．

謝辞

指導教官の二木厚吉教授には，本研究を行うにあたり，充実した研究環境をはじめ終始御指導と御助言をして頂きました．ここに感謝の意を表し，心より御礼申し上げます．天野憲樹助手には，本研究についての多大な技術的御指導と有益な御助言をして頂きました．心より感謝致します．緒方先生をはじめ言語設計学講座の皆様には，セミナーの際に色々と貴重な意見を頂きました．感謝の意を表します．佐藤一郎助教授には，快く AgentSpace を利用することを許可して頂きました．感謝の意を表します．最後に，日頃より共に研究活動を行ってきた二木研究室修士課程学友諸氏，他分野の立場から助言して頂いた三井実氏，川野邊誠氏，須藤隆氏，長谷川勝巳氏，森田綾子氏に感謝致します．

参考文献

[1] David M. Chess, Security Issue in Mobile Code System, Lecture Notes in Computer Science, Vol. 1419, pp1–14, Springer(1998).

[2] White J. E, Telescrypt Technology, The foundation for the electronic marketplace, White paper. Geveral Magic,Inc. 1994.

[3] 山崎重一朗, 津田宏, Telescript 言語入門, アスキー出版局, 1996

[4] 佐藤一朗, モバイルエージェントの動向, 人工知能学会, Vol. 14, No. 4, pp598–605, 1999.

[5] IBM Aglets Software Development Kit, http://www.trl.ibm.co.jp/aglets/

[6] 小野康一, 大島満, 移動エージェント Aglets のセキュリティ・モデル, コンピュータソフトウェア, Vol. 17, No. 4, pp2–14, 2000.

[7] http://www.objectspace.com/products/voyager/

[8] http://www.toshiba.co.jp/plangent/

[9] 田原康之, 長谷川哲夫, 大須賀昭彦, 本位田真一, 知的ネットワークエージェント Plangent のセキュリティ・セーフティ, インターネットテクノロジーワークショップ, 1998.

[10] http://research.nii.ac.jp/ ichiro/agent/

[11] 佐藤一郎, 高橋美奈子, 棚橋杏子, 吉野裕子, モバイルエージェントの階層的な構成と移動, 日本ソフトウェア科学会全国大会, 1998.

[12] Tomas Sander, Christain F. Tschudin, Protecting Mobile Agents Against Malicious Hosts, Lecture Notes in Computer Science, Vol. 1419, pp44–60, Springer(1998).

[13] 村山隆徳, 満保雅浩, 岡本栄司, 植松友彦, ソフトウェアの難読化について, 電子情報通信学会技術研究技報 情報セキュリティ, Vol.95, No. 353, ISEC95-25, pp9–14, 1995.

[14] 村山隆徳, 満保雅浩, 岡本栄司, 植松友彦, プログラムコードの難読化について, SCIS 暗号と情報セキュリティシンポジウム, SCIS96-8D, 1996.

[15] Fritz Hohl, Time Limited Blackbox Security:Protecting Mobile Agents from Malicious Hosts, Lecture Notes in Computer Science, Vol. 1419, pp92–113, Springer(1998).

[16] Fritz Hohl, A Protocol to Detect Malicious Hosts Attacks by Using Reference States, Technical Report Nr, 1999.

[17] Giovanni Vigna, Cryptographic Trace for Mobile Agents, Lecture Notes in Computer Science, Vol. 1419, pp137–153, Springer(1998).

[18] David Aucsmith, Gray Granuke, 逆解析や改変からソフトを守る タンパ・レジスタント・ソフトウェア技術の詳細, 日経エレクトロニクス, 1998.1.5(No.706).

付 録A

ここでは，本アプリケーションフレームワークおよび作成した例題アプリケーションを動作させるための，実行環境，インストール方法，実行方法について説明する．

A.1　実行環境

本システムは最小限の構築環境としてネットワークで通じた 3 つのホストを必要とする．その構成内容は 1 つのユーザのホストと 2 つ以上の仮想店舗のホストとなる．これら各ホストは本システムを動作させるために以下の実行環境を整える必要がある．

各ホストの実行環境は以下になる．

- OS
  Windows 98/2000
- Java
  JDK 1.3
- モバイルエージェントシステム
  AgentSpace1(2000 年 10 月 5 日版)

A.2　インストール方法

電子商取引エージェントを実行するすべてのホストでは以下をインストールしておく必要がある．

1. JDK
   http://java.sun.com/j2se/
   より JDK をダウンロードする．インストールの詳細については JDK 付属のマニュアルにて記載されているので，ここでは省略する．
2. AgentSpace
   http://research.nii.ac.jp/ ichiro/agent/
   より AgentSpace1(2000 年 10 月 5 日版) をダウンロードする．アーカイブファイルは zip 形式で圧縮されているので解凍する．解凍後のファイル構成例は以下になる．

```
(agentspace)-|-README
             |-COPYRIGHT
             |-(doc)
             |-(system)-|-agentspace-|-*.class システムのクラス
                        |            |-*.java  システムのソース
                        |-*.agent    エージェントプログラム
                        |-*.java     各エージェントのソースプログラム
                        |-*.class    各エージェントの class ファイル
                        |-*.mk       各エージェントの Makefile
                        |-Makefile   AgentSpace システムの Makefile
```

環境変数 CLASSPATH に上記の system ディレクトリが含まれている必要がある．もし，環境変数 CLASSPATH に「.」（カレントディレクトリ）が含まれているならば，上記の system ディレクトリをカレントディレクトリにすることで実行可能となる．

A.3 起動方法

本システムでは全てのホストで AgentSpace を起動し，その AgentSpace 上においてシステムを実行する．AgentSpace の起動方法は以下になる．

1. コマンドプロンプトから AgentSpace の system ディレクトリで以下のように入力し，AgentSpace を起動する．

   ```
   java␣agentspace.Agentserver
   ```

2. 図 A.1のように AgentSpace が起動したら，各ホストでエージェントを実行する．

ユーザのホストにおける電子商取引エージェントの実行方法と，仮想店舗のホストにおける仮想店舗の実行方法を以下に説明する．

- ユーザのホスト
  ユーザのホストでは，電子商取引エージェントを起動する．起動方法は以下の手順となる（図 A.2参照）．
  1. AgentSpace の Load ボタンを押す．
  2. ファイル選択画面が表示されるので，
     TamperProofAgent.jar を選択し，開くボタンを押す．
  3. 電子商取引エージェントが起動する（図 A.3参照）．

- 仮想店舗のホスト
  仮想店舗のホストでは仮想店舗を起動する．起動方法は以下の手順となる（図 A.4参照）．

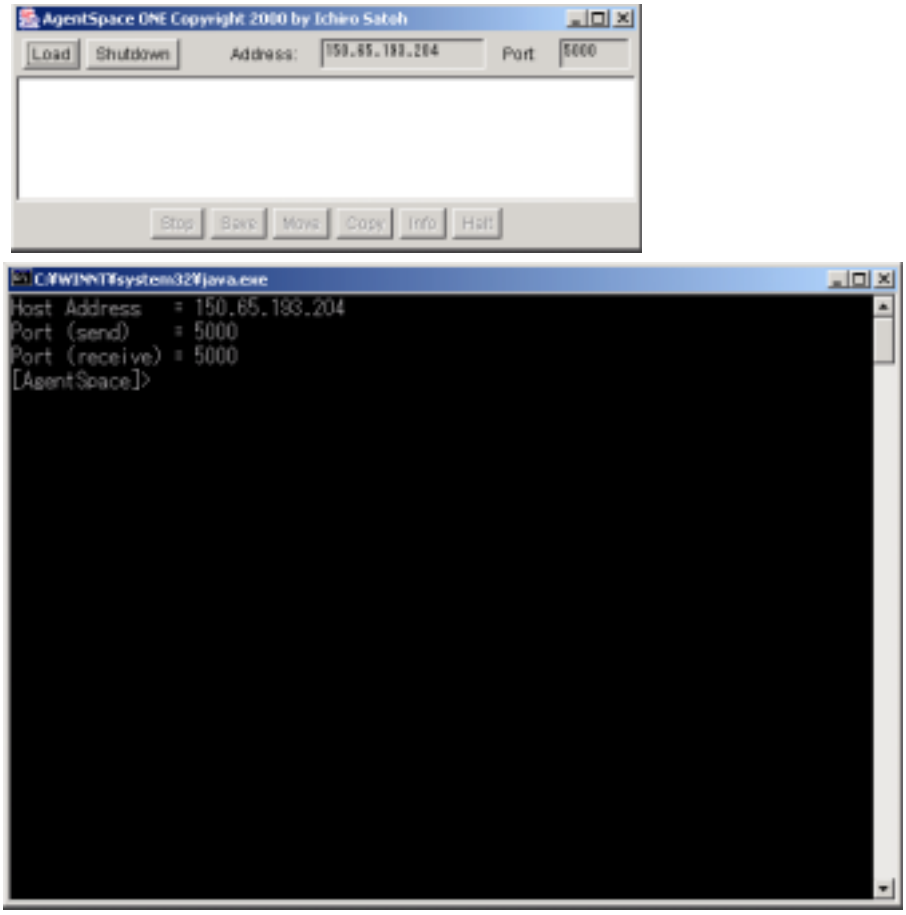

図 A.1: AgentSpace の起動

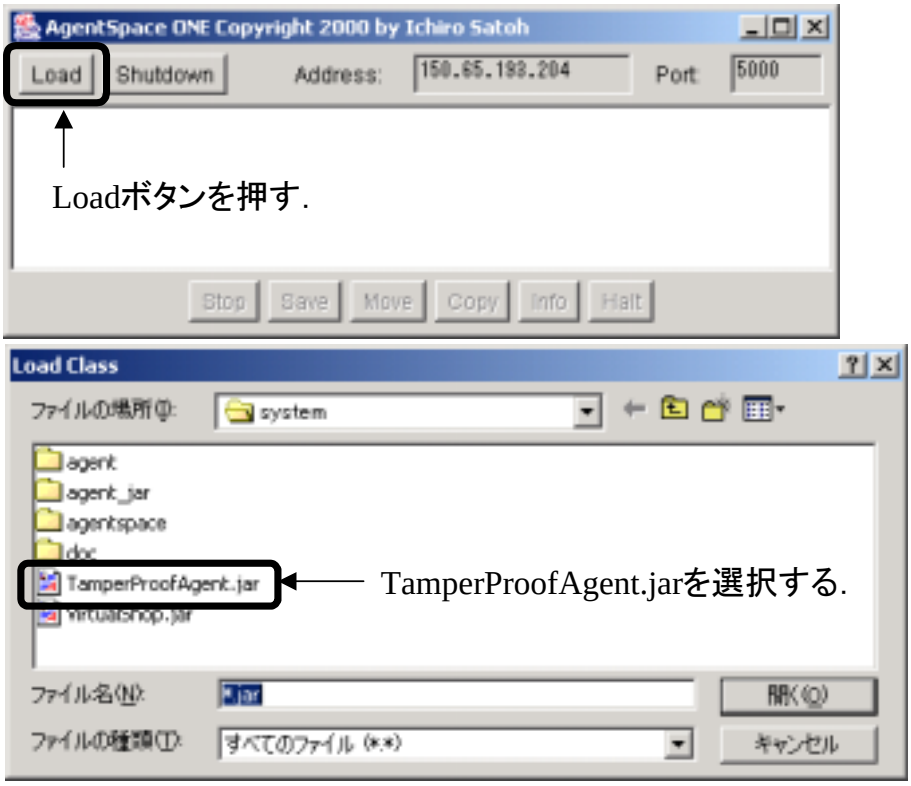

図 A.2: 電子商取引エージェントの起動

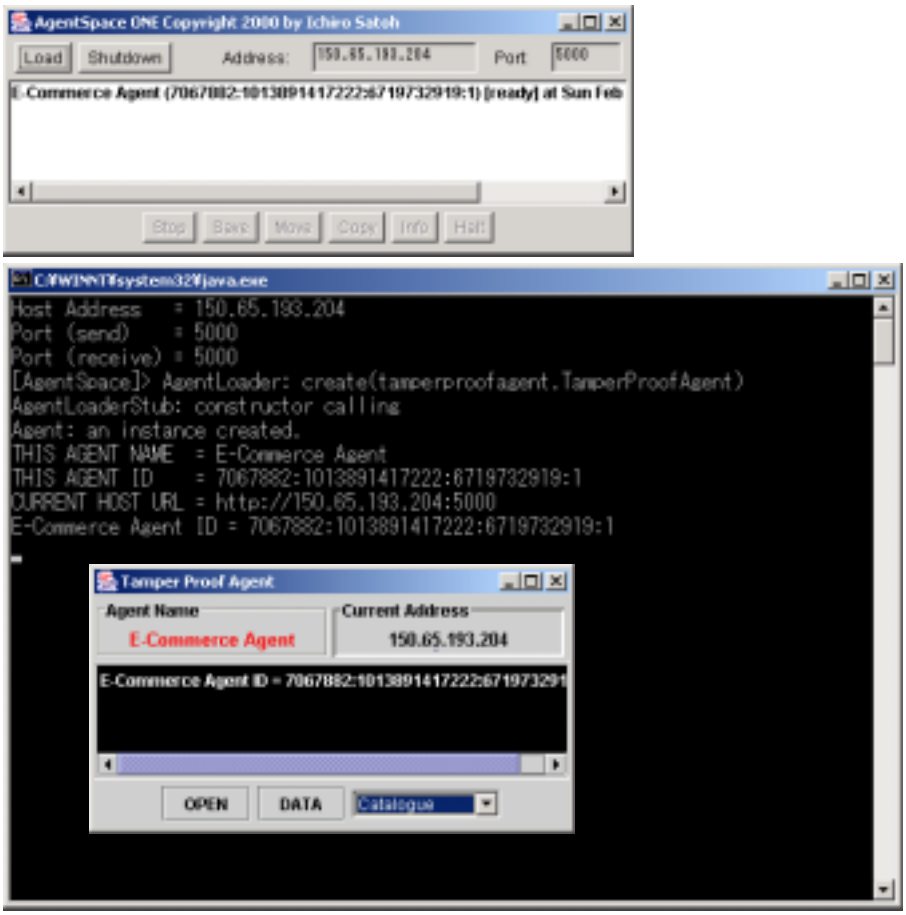

図 A.3: 電子商取引エージェント

1. AgentSpace の Load ボタンを押す.
2. ファイル選択画面が表示されるので,
   VirtualShop.jar を選択し，開くボタンを押す.
3. 仮想店舗（エージェント）が起動する（図 A.5参照）.

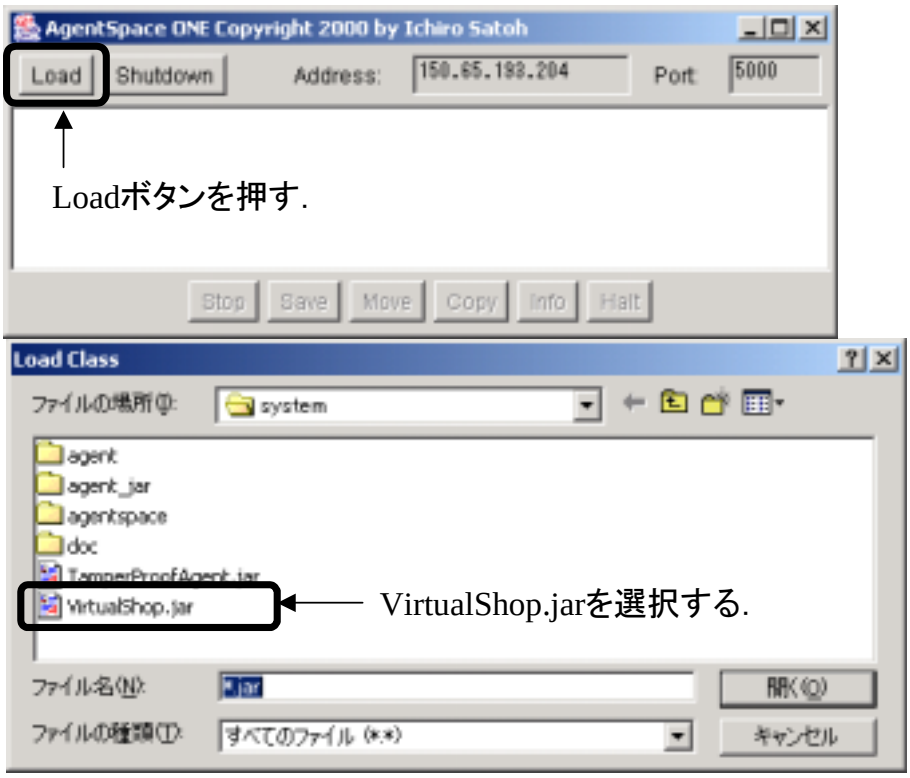

図 A.4: 仮想店舗（エージェント）の起動

A.4 設定方法

本システムを実行するための各エージェントの設定方法について以下に説明する.

- 電子商取引エージェント
  電子商取引エージェントは実行時に以下の 3 つのファイルを読み込む.
  - セキュリティポリシ定義ファイル
    ファイル名：security.policy
    取り扱う個人情報のセキュリティ用件を定義したファイルである．記述形式は以下になる.

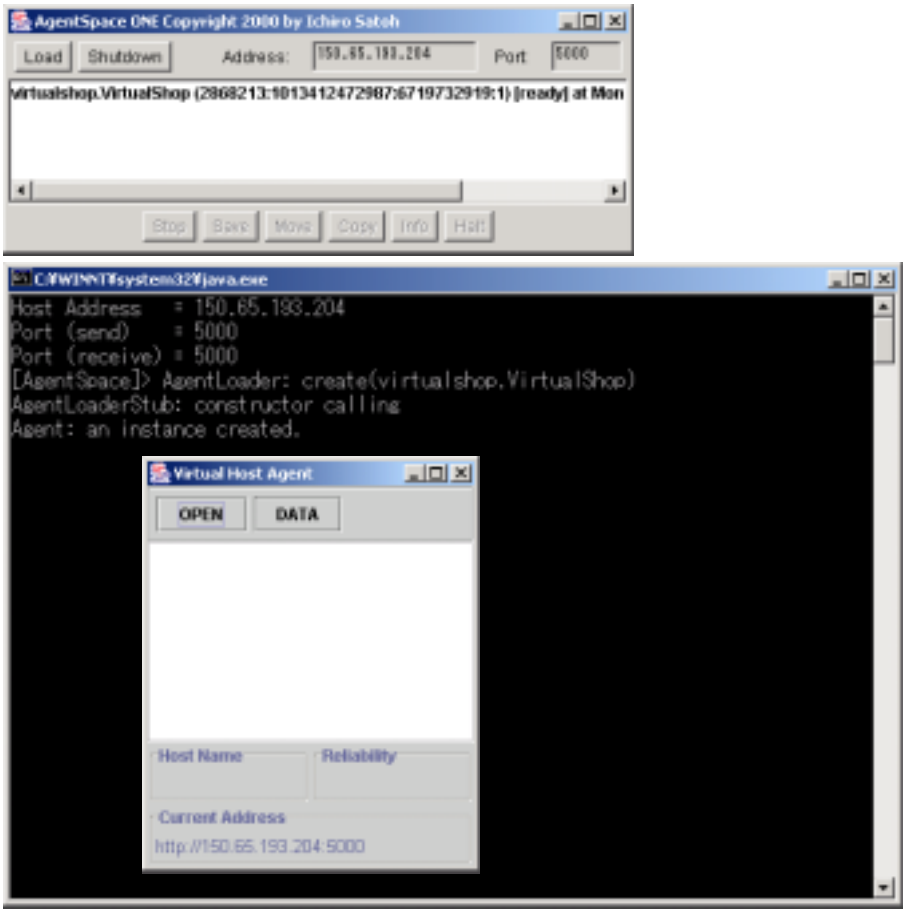

図 A.5: 仮想店舗（エージェント）

表 A.1: 本システムで取り扱うファイルの概要

ファイル名	概要
TamperProofAgent.jar	電子商取引エージェントファイル
security.policy	セキュリティポリシ定義ファイル
private.data	個人情報記述ファイル
url.list	巡回対象仮想店舗アドレスリストファイル
VirtualShop	仮想店舗エージェントファイル
shop.inf	仮想店舗設定ファイル

情報の識別子（String 型） ： 情報の重要度（int 型）
（例） NAME:1

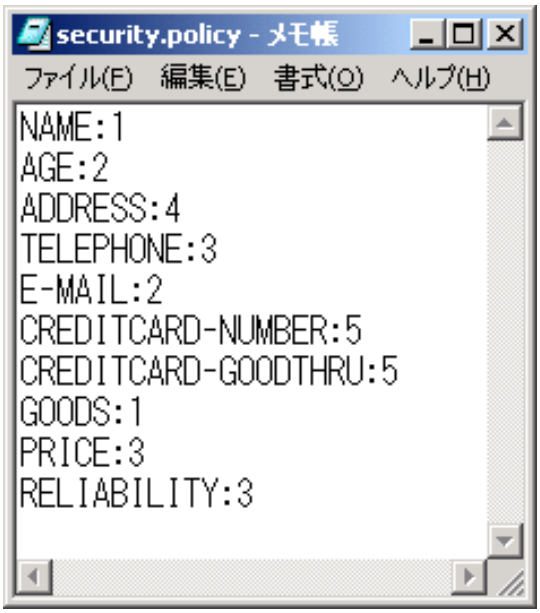

図 A.6: セキュリティポリシ定義ファイル

– 個人情報記述ファイル
ファイル名：private.data
取り扱う個人情報の情報内容を記述したファイルである．記述形式は以下になる．

情報の識別子（String 型） ： 情報の内容（String 型）
（例） NAME:Makoto HASEGAWA

– 巡回対象仮想店舗アドレスリストファイル
巡回対象となる仮想店舗の IP アドレスを記述したファイルである．記述形式は以下となる．

IP アドレス（String 型）
（例） 150.65.192.101

- 仮想店舗
仮想店舗は実行時に設定ファイルを読み込む．設定ファイルには以下の 4 種類の情報を記述する．

– 仮想店舗情報
ファイル名：virtualshop.txt

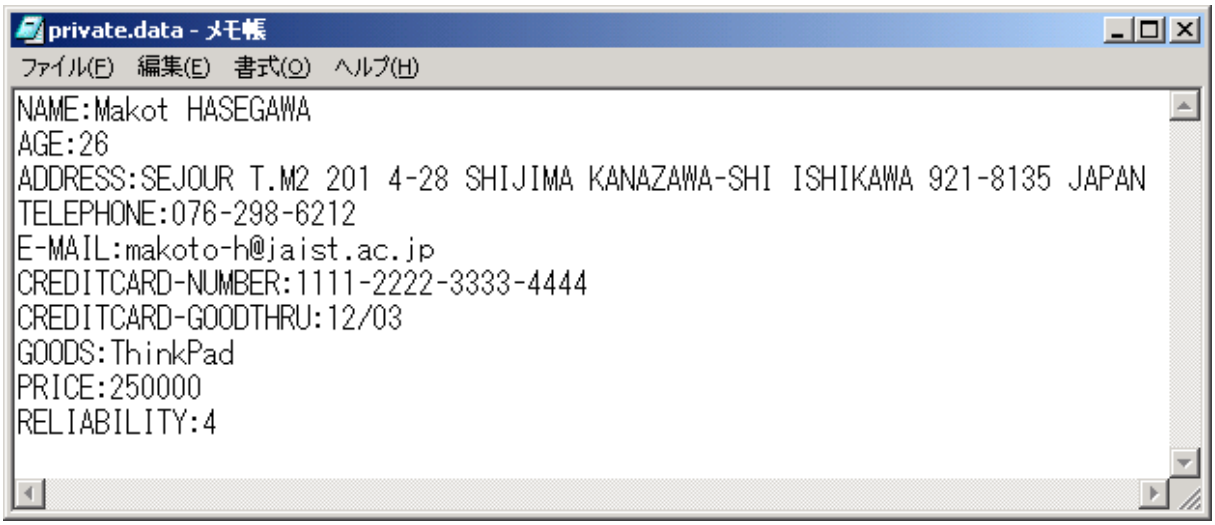

図 A.7: 個人情報記述ファイル

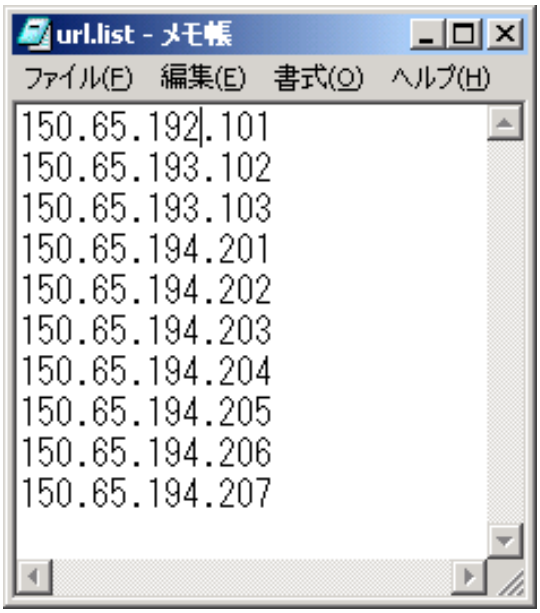

図 A.8: 巡回対象となる仮想店舗のアドレスリストファイル

仮想店舗情報として仮想店舗名，仮想店舗の信頼度を記述する．記述形式は以下になる．

設定情報識別子（String 型），仮想店舗名（String 型），信頼度（int 型）（例）
SHOPNAME,VirtualShop1,5

– 取引許可情報
仮想店舗が要求する個人情報を電子商取引エージェントが全て公開しなかったときの返答を記述する．記述形式は以下になる．

設定情報識別子（String 型），返答識別子（String 型），返答（boolearn 型）
（例） PERMISSION,ans,true

– 要求情報
電子商取引エージェントに公開を要求する情報の識別子を記述する．記述形式は以下になる．

設定情報識別子（String 型），公開情報識別子（String 型）（例）
REQUEST,NAME

– 取り扱い商品情報
仮想店舗が取り扱う商品名，商品価格，在庫数を記述する．記述形式は以下になる．

商品名（String 型），商品価格（int 型），在庫数（int 型）（例）
ThinkPad,200000,3

A.5 実行方法

本システムを実行するための手順について説明する．

- 電子商取引エージェント

セキュリティポリシ定義，個人情報，巡回対象アドレスなどの設定をファイルを記述した後，電子商取引エージェントにそれらを読み込ませる．電子商取引エージェントは各種設定ファイルを読み込んだ後，自動的に巡回を開始してしまうため，その前に巡回パターンを設定する必要がある．以下に巡回開始までの手順を示す．

1. 巡回パターンの設定（図 A.10参照）

電子商取引エージェントのポップアップメニューから希望する巡回パターンを選択する．メニュー項目，Catalogue，Direct，Low priced，High Reliability はそれぞれ，情報収集，条件付き決済，価格優先決済，信頼度優先決済に対応する．

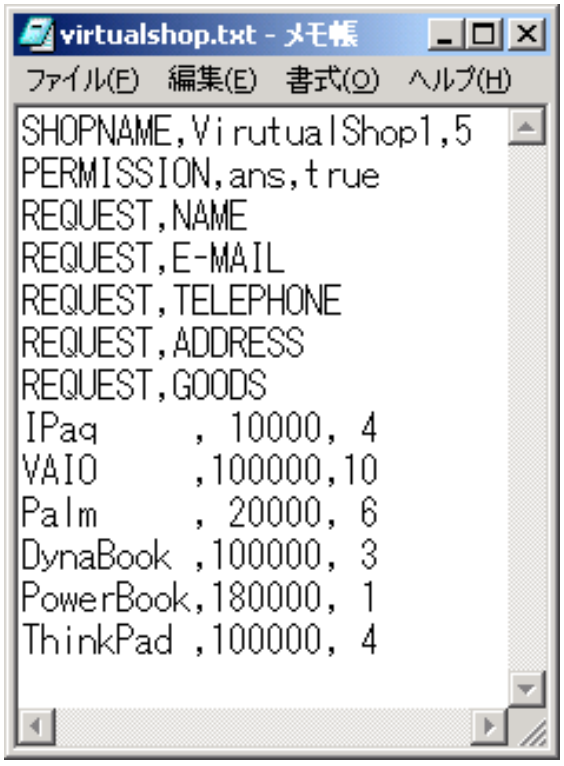

図 A.9: 仮想店舗設定ファイル

– 情報収集
  巡回対象の仮想店舗を全て巡回し，全てのホストの価格表を取得してユーザのホストに戻る．
– 条件付き決済
  個人情報で記述した「PRICE」,「GOODS」,「RELIABILITY」を全て満たす仮想店舗に到着し次第そこで決済する．
– 価格優先決済
  巡回対象の仮想店舗を全て巡回し，希望商品「GOODS」が最低価格の仮想店舗で決済する．
– 信頼度優先決済
  巡回対象の仮想店舗を全て巡回し，希望商品「GOODS」を販売する信頼度が一番高い仮想店舗で決済する．

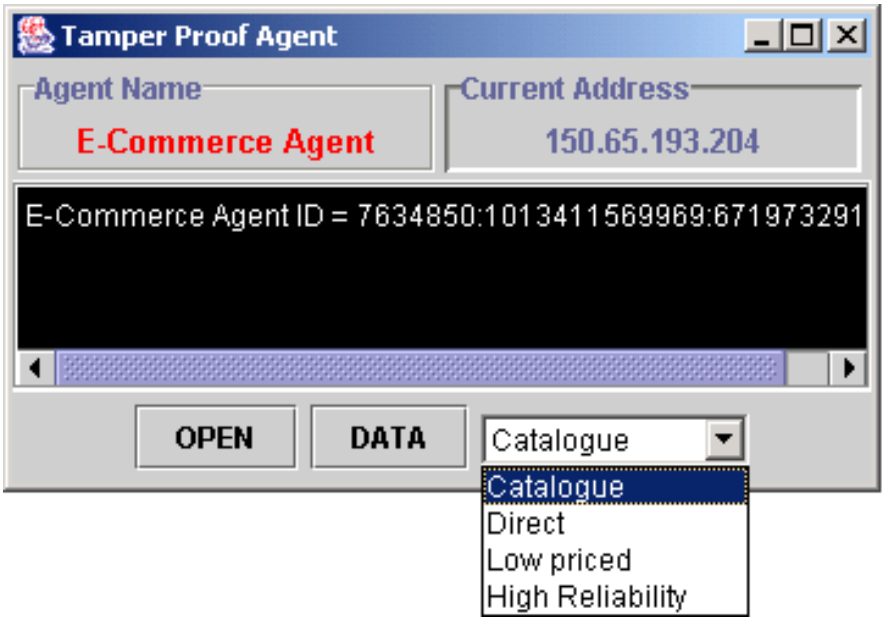

図 A.10: 巡回パターンの指定

2. 各種設定ファイルの読み込み

各種設定ファイルは，電子商取引エージェントの OPEN ボタンを押すことにより入力する（図 A.11参照）．

(a) セキュリティポリシ定義ファイルの読み込み（図 A.12参照）．

電子商取引エージェントの OPEN を押すことによりファイルダイアログが表示されるので,「security.policy」を選択し，開くボタンを押すことによりセキュリティポリシ情報が入力される．

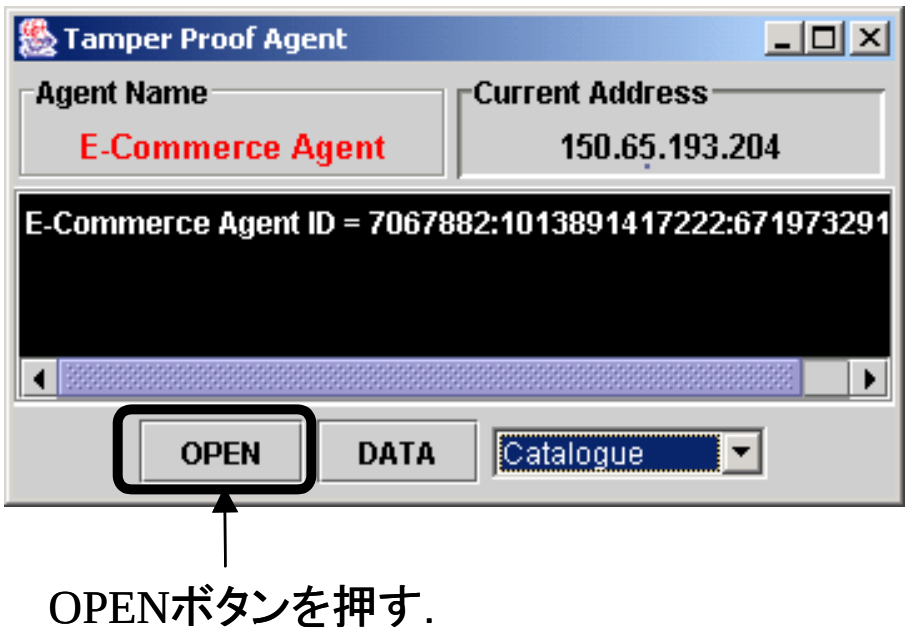

図 A.11: 設定ファイルの読み込み

図 A.12: セキュリティポリシファイルの読み込み

(b) 個人情報記述ファイルの読み込み（図 A.13参照）.

セキュリティポリシファイル入力後に再度ファイルダイアログが表示されるので,「private.data」を選択し，開くボタンを押すことにより個人情報が入力される.

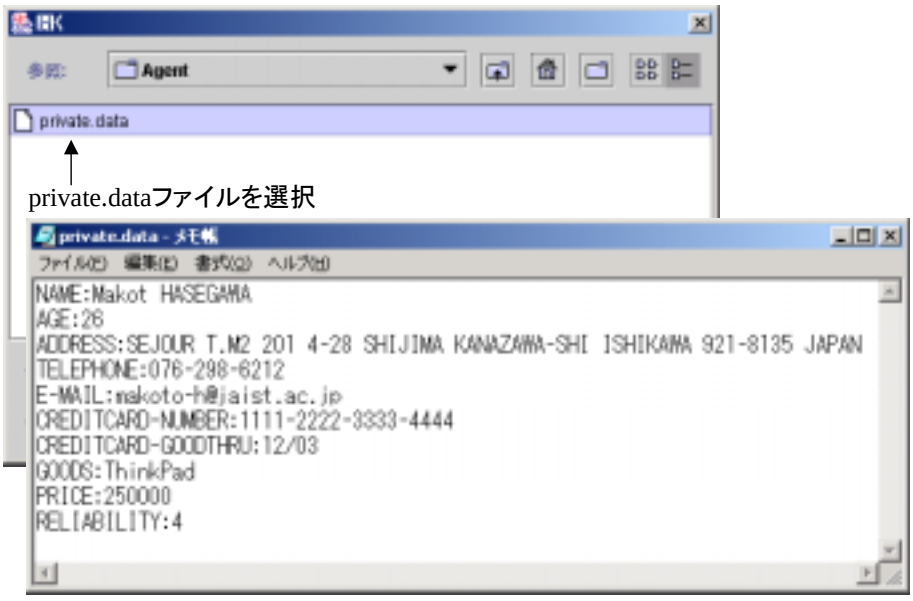

図 A.13: 個人情報記述ファイルの読み込み

(c) 巡回対象仮想店舗アドレスリストファイルの読み込み（図 A.14 参照）.

個人情報ファイル入力後も再度ファイルダイアログが表示されるので,「url.list」を選択し，開くボタンを押すことにより仮想店舗のアドレスリストが入力される.

- 仮想店舗

仮想店舗設定ファイルを記述した後，仮想店舗にそれを読み込ませる．仮想店舗は設定ファイルを読み込んだ直後から実行開始となり，電子商取引エージェントの待ち状態に入る．以下に手順を説明する.

1. 設定ファイル入力（図 A.15参照）

   仮想店舗設定ファイルは，電子商取引エージェントの OPEN ボタンを押すことにより入力する.
2. 設定ファイル選択（図 A.16参照）

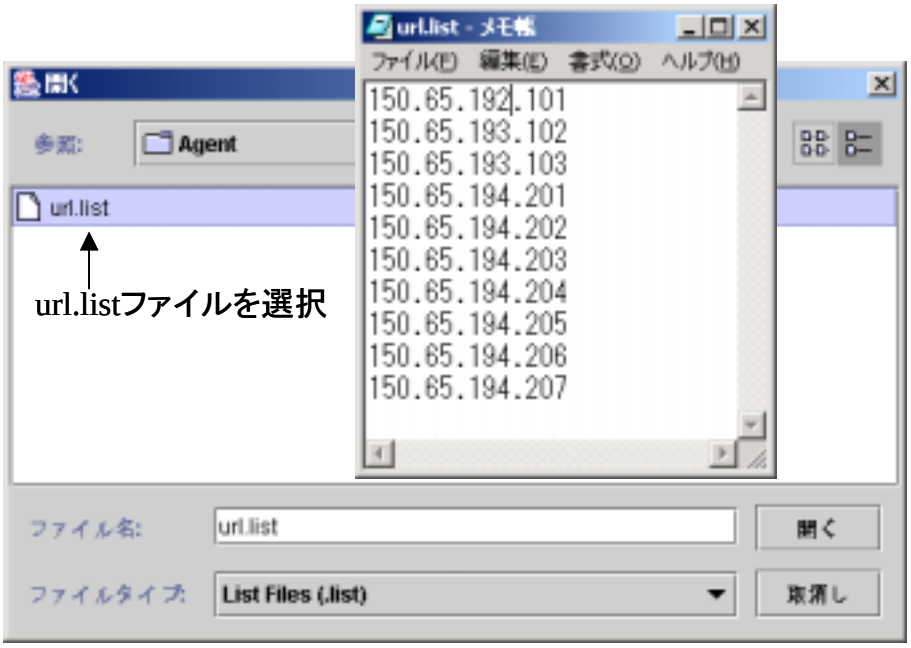

図 A.14: 巡回対象仮想店舗アドレスリストファイルの読み込み

図 A.15: 設定ファイルの読み込み

「virtualshop.txt」を選択し，開くボタンを押すことにより仮想店舗設定ファイルが入力される．

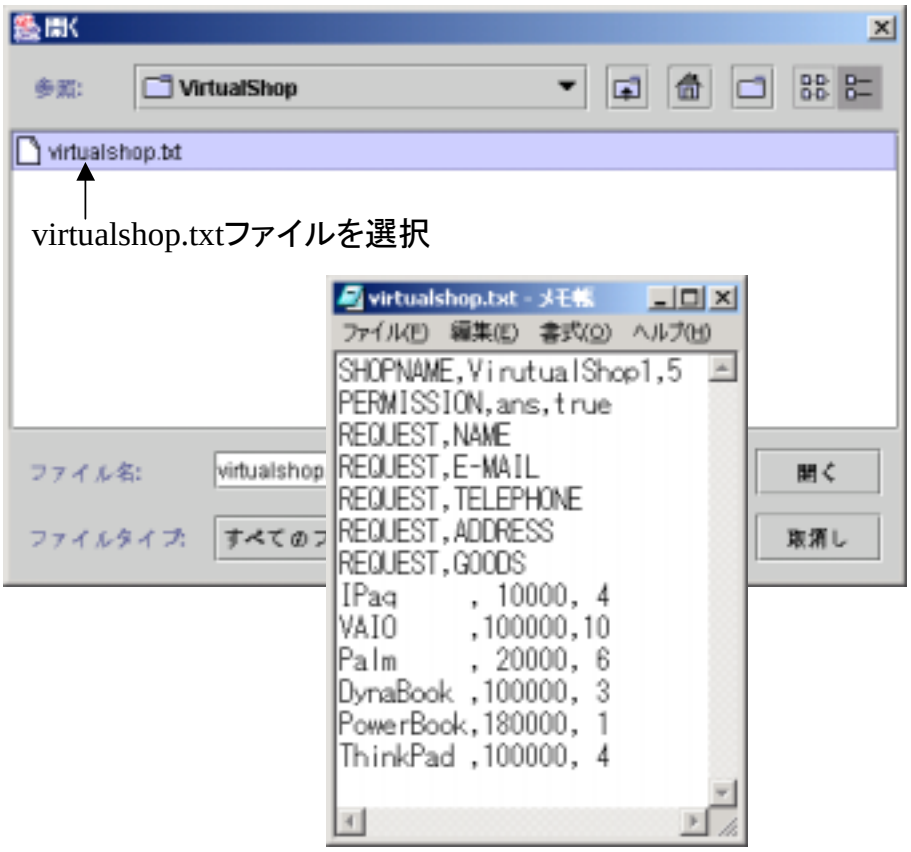

図 A.16: 仮想店舗設定ファイルの読み込み

3. 電子商取引エージェント待ち状態（図 A.17参照）

以上の設定を終えることで，仮想店舗は電子商取引エージェント待ち状態に入る．

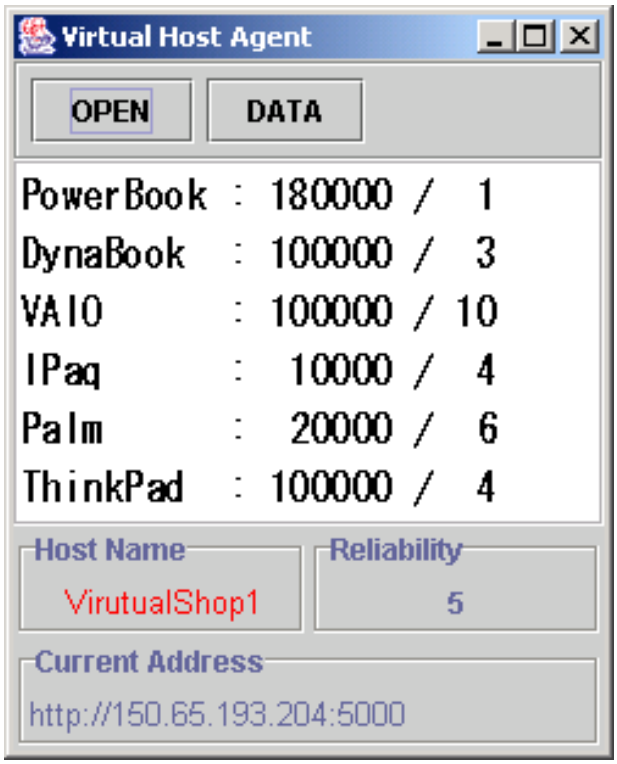

図 A.17: 仮想店舗の電子取引エージェント待ち状態