耐タンパハードウェアを用いたモバイルエージェント保護手法

春木洋美 † 河口信夫 ‡∗ 稲垣康善 †
† 名古屋大学大学院工学研究科
‡ 名古屋大学大型計算機センター
∗ 名古屋大学統合音響情報研究拠点
haruki@inagaki.nuie.nagoya-u.ac.jp

近年，計算機の小型・高性能化とネットワークの発達に伴い，電子情報の手軽な交換・共有が可能になった．しかし，個人情報等のデータを扱う場合，従来の技術ではデータの読み書きや送信に対する許可・拒否などのデータ所有者の意図を反映できないという問題がある．そこで，コードと実行状態を持って移動するモバイルエージェント技術を利用する．この技術により，コードに書かれた手続きでデータを処理できるため，データ所有者の意図に沿ったデータ保護が可能となる．しかし，従来のシステムでは，移動先のホストからモバイルエージェントを保護できない．本稿では，各ホストに信頼できる箇所，耐タンパハードウェアの存在を仮定したモバイルエージェント保護手法を提案する．本手法では，耐タンパハードウェアによる公開鍵暗号方式を用いることによりホストから秘密のデータを隠匿する．また，耐タンパハードウェアの利用により生じる問題点を挙げ，その解決策を提案する．本手法により，モバイルエージェントが持つデータを保護できるため，モバイルエージェントを用いた柔軟なデータ管理が可能となる．

Protecting Mobile Agents using Tamper Resistant Hardware

Hiroyoshi Haruki† Nobuo Kawaguchi‡∗ Yasuyoshi Inagaki†
†Graduate School of Engineering, Nagoya University
‡Computation Center, Nagoya University
∗Center for Integrated Acoustic Information Research(CIAIR), Nagoya University

Recently, it is getting possible to make communication with computers by advances in computer and wireless communication technology. However, under such environment, once a personal data has been sent to other hosts, it is impossible to control the redistribution of the data. In this paper, we propose a data protecting method using mobile agents. Because mobile agent can migrate with code and state, owner can control mobile agents to perform a flexable data protection. However, mobile agent can be tampered by malicious host. We propose a mobile agent protection system with tamper resistant hardware. The tamper resistant hardware has public key encryption system. We solve the problem producted by using the tamper resistant hardware.

1 はじめに

近年，計算機の小型・高性能化に伴い，簡単に大量の情報を扱うことが可能になった．また，ネットワークの急速な発達に伴い，電子情報の手軽な交換・共有が可能になった．このような環境では，名刺やスケジュールといった個人的なデータを交換・共有する場合がある．しかし，電子情報は自由に修正や再配布ができてしまうため，名刺等の個人情報も，一度配布してしまえば，どのように扱われるかの制限はできない．これでは，紙の名刺よりも，電子情報の方が扱いにくくなってしまう．

データの読み書きや再配布の許可，回数制限，時間制限といったデータ所有者の意図が，データを配布した後にも反映できれば，個人情報などの電子的な交換・共有をより手軽に安心して行うことが可能になる．PDF[1]では，文書情報へのフラグの付加によりデータ保護を実現している．しかし，事前に送信先に閲覧するためのアプリケーションが必要である．また，PDF で決められた範囲内でのデータ保護しかできないため，データ所有者が保護する範囲を自由に決めることができない．

我々は，データ所有者の意図をデータ配布後にも反映させるために，コードと実行状態を持って移動するモバイルエージェントを用いたデータ保護手法を提案する．これにより，所有者の意図を反映するコードを付加してデータが配布可能となる．また，事前に移動先にデータを扱うためのアプリケーションをインストールしておく必要がない．さらに，モバイルエージェント技術では，所有者が自由にコードを作成できるため，従来の PDF などで行われてきたデータ管理よりも柔軟なデータ管理を行うことができる．

しかし，モバイルエージェントには，移動先のホストから攻撃を受けるというセキュリティ上の問題がある．モバイルエージェントは，移動先にモバイルエージェントを実行するために必要なコードと実行状態を持って移動する．しかし，移動先ホストのモバイルエージェントシステムが信用できない場合，すべてのコードと実行状態を渡すため，モバイルエージェントは実行状態の中に秘密のデータを持つことはできなくなる．また，秘密のデータを暗号化して移動したとしても，モバイルエージェントを実行するために復号機能を移動先に渡す必要があるため，ホストは暗号化されたデータを解読できる．

本研究では，アドホックネットワークでも扱えるように，信頼できるサーバや所有者自身のホストの存在を仮定しない．本稿では，各ホストに信頼できる箇所，耐タンパハードウェアの存在を仮定したモバイルエージェ

ント保護手法を提案する．本手法では，耐タンパハードウェアにデータの暗号化・復号化を扱わせることによりホストから秘密データを隠す．本手法の要点を以下に示す．

1. 耐タンパハードウェアを用いたデータの暗号化・復号化による秘密データの漏洩防止
2. 耐タンパハードウェア専用の CA(Certified Authority) の仮定による耐タンパハードウェア公開鍵偽装の防止
3. 復号要求元のモバイルエージェントの認証による耐タンパハードウェアへの不正アクセスの防止
4. モバイルエージェントマネージャのデータベース保持による Java におけるバイトコード獲得問題の解決

ここで，アドホックネットワークとは，必要に応じて一時的に構築されるネットワークであり，耐タンパハードウェアとは，変更を行うと正常に動作しないハードウェアである．

以下，本稿では，まず，第 2 章において，モバイルエージェントとセキュリティについて説明し，第 3 章において，耐タンパハードウェアを用いたモバイルエージェント保護手法と耐タンパハードウェアを用いる場合に生じる問題点を挙げ，その解決策について説明する．また，第 4 章において，本手法を Java を用いて実装する場合に生じる問題点を挙げ，その解決策について説明する．また，本手法のプロトタイプについて報告する．さらに，第 5 章において，関連研究との比較について述べる．

2 モバイルエージェントとセキュリティ

モバイルエージェントとは，コードと実行状態を持って自律的に移動するプログラムである．モバイルエージェントを用いることにより，遠隔実行におけるプログラムの非同期実行やネットワーク負荷の軽減を図ることが可能となる．また，負荷分散やプログラムの自動インストール，電子商取引などモバイルエージェントを用いた応用が数々考えられる．ここで，モバイルエージェントは，モバイルエージェントシステムと呼ばれるランタイム上で実行される．

モバイルエージェントを用いる上で重要になることはセキュリティである．セキュリティとして問題となる場面は大きく分けて，以下の 3 点が挙げられる．

1. モバイルエージェントからホストへの攻撃
   - 悪意を持ったモバイルエージェントによるホストにある重要なデータの盗聴，改竄
2. 中継ホストからモバイルエージェントへの攻撃
   - 中継ホストによるモバイルエージェントが持つコードの盗聴，改竄
   - 中継ホストによるモバイルエージェントが持つ実行状態の盗聴，改竄
3. 移動先ホストからモバイルエージェントへの攻撃
   - 移動先ホストによるモバイルエージェントが持つコードの盗聴，改竄
   - 移動先ホストによるモバイルエージェントが持つ実行状態の盗聴，改竄

1 番目の問題については，署名技術を用いることにより出所を調べ，出所によるアクセス制御を行うことによりモバイルエージェントからの攻撃を防ぐことができる．2 番目の問題については，暗号化技術を用いることにより，移動途中は暗号化されたモバイルエージェントが移動するため，中継ホストからの攻撃を防ぐことができる [2, 3]．3 番目の問題については，いくつかの研究がされており，詳しくは 5 章で述べる．これらの手法は，1 章で述べたデータ管理を行うモバイルエージェントに必要とされるセキュリティシステムとしては不十分である．

3 耐タンパハードウェアを用いたモバイルエージェント保護手法

3.1 耐タンパハードウェアとモバイルエージェント

本手法では，保護するデータとして個人情報やスケジュールなどのプライバシーに関するデータを対象とする．そこで，本手法ではハードウェア，OS の変更による攻撃はないが，モバイルエージェントシステムは信用できるとは限らないと仮定する．

モバイルエージェントシステムが信用できない場合，モバイルエージェントは移動先のホストによって盗聴・改竄される可能性がある．先述のように，モバイルエージェントはコードと実行状態を持って移動し，モバイルエージェントシステムがモバイルエージェントのコードの登録・ロードとモバイルエージェントの登録を実行する．そのため，移動先のモバイルエージェントシステム，すなわち，ホストは，モバイルエージェントが持つコードと実行状態を獲得できる．そのとき，ホストは獲得したコードに実行状態の中にある秘密のデータを漏らすコードを挿入することにより，データを獲得できる．したがって，モバイルエージェントが秘密のデータを持って移動することができない．

モバイルエージェントの保護のために，ホストがモバイルエージェントが持つコードと実行状態を獲得し，コードを不正に書き換えても，モバイルエージェントが持つ秘密のデータの内容を知られない方法が必要となる．そこで，耐タンパハードウェア (TRH: Tamper Resistant Hardware) を用いたモバイルエージェント保護手法を提案する．ただし，我々は耐タンパハードウェアに入れるべき機能をできる限り小さくすることにより，実用的な手法を提案する．耐タンパハードウェアとは，不正な使用ができず，無理な変更を行うと正常に動作しないハードウェアである．耐タンパハードウェアの一例として，メモリスティック[1] や SD メモリカード[2] があげられる．これらのハードウェアは，データの記憶領域だけでなく，暗号・復号機能，認証機能を持っている [4]．この認証

[1] http://www.memorystick.org/
[2] http://www.panasonic.co.jp/avc/home/sd/

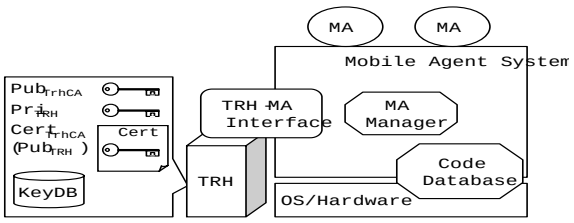

図 1: 本手法で想定するモバイルエージェントシステム

機能により，耐タンパハードウェアが持つ秘匿データは特定のアプリケーションからしか見られない．

本手法では，耐タンパハードウェアを用いた公開鍵暗号方式によるデータのやり取りを実施する．モバイルエージェントが持つ秘密のデータを移動先のホストの耐タンパハードウェアの公開鍵を用いて暗号化することにより，移動先のホストは耐タンパハードウェアを用いない限り，モバイルエージェントが持つ秘密のデータを解読できなくなる．

図 1 に，今回想定するモバイルエージェントシステムの構成を示す．ただし，TRH-MA Interface は耐タンパハードウェアとモバイルエージェント間のインターフェースであり，MA Manager はモバイルエージェントを管理し，Code Database はコードを保管しているコンポーネントである．

3.2 耐タンパハードウェアの公開鍵の信頼性

今回想定する耐タンパハードウェアを用いた保護手法では，公開鍵暗号方式を用いるため，あらかじめ耐タンパハードウェア固有の公開鍵を互いに交換する必要がある．しかし，悪意のあるホスト B は，自分の公開鍵 Pub_B を耐タンパハードウェアの公開鍵と偽って攻撃対象のホスト A と交換する可能性が考えられる．

ここで問題となっていることは耐タンパハードウェアの公開鍵の信頼性問題である．耐タンパハードウェアが他ホストから獲得した公開鍵が『本当に』他ホストの耐タンパハードウェアの公開鍵であるかを知る手段が必要である．そこで，耐タンパハードウェアの公開鍵専用の Certified Authority(TrhCA) の存在を仮定する．この TrhCA は耐タンパハードウェアに対してだけ証明書を与える．また，耐タンパハードウェアは，固有の公開鍵 Pub_{TRH} を持つのではなく，TRH 固有の証明書 $Cert_{TrhCA}(Pub_{TRH})$ を持つ．この証明書には，TRH の公開鍵と TRH の情報が含まれている．悪意のあるホストは，自分の公開鍵に対する TrhCA の証明書を獲得できないため，自分の公開鍵を耐タンパハードウェアの公開鍵として偽装することは困難となる．表 1 に，本手法想定の耐タンパハードウェアが保持するデータを示す．

3.3 復号要求元の認証

悪意のあるホスト，及び，悪意のあるモバイルエージェントシステムは，モバイルエージェントのコードと暗号化データを含む実行状態を獲得できる．図 2 のように，悪意のあるホストが耐タンパハードウェアに対して，獲得したコードと暗号化データを含む実行状態を用いて復号化要求を行うことにより秘密のデータを獲得するという攻撃が可能である．

表 1: 本手法の TRH が保持するデータ

変数名	詳細
$Cert_{TrhCA}(Pub_{TRH})$	耐タンパハードウェア公開鍵を含む証明書
Pub_{TrhCA}	TrhCA の公開鍵
Pri_{TRH}	耐タンパハードウェア秘密鍵
$KeyDB_{TRH}$	他の耐タンパハードウェア公開鍵保管用データベース

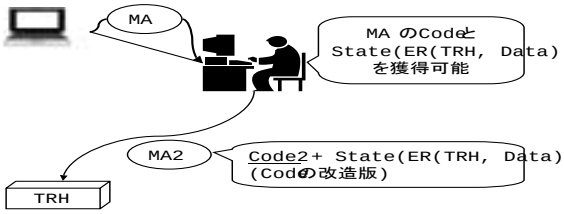

図 2: TRH に対する不正な復号化要求の例

この問題は，あるデータに対して耐タンパハードウェアを用いて復号要求できるモバイルエージェントは，データの持ち主であるモバイルエージェントだけとしていないことにある．そこで，復号要求元の認証が不可欠となる．

ここで，モバイルエージェントのコードは移動しても変わらない．また，悪意のあるホストは，モバイルエージェントのコードを書き換えてモバイルエージェントのデータを獲得しようとする可能性がある．したがって，要求元のモバイルエージェントのコードがデータを暗号化したときのコードと同じであるかどうかを確認すること，つまり，コードの完全性の検証によりデータの復号を許可すればよい．

3.4 インターフェイスの認証

耐タンパハードウェアとモバイルエージェントを仲介するインターフェース (TMI: TRH-MA Interface) は，暗号化データ，復号化データを扱う．そのため，このインターフェイスが改竄されたとき，ホストに秘密のデータが漏れる可能性がある．そこで，耐タンパハードウェアはインターフェイスを認証する必要がある．

3.5 安全なモバイルエージェントコード

モバイルエージェントは，復号化データを扱うため，不用意な復号化データを伴った移動，永続化，他からの不正アクセスに注意しなければならない．この問題はプログラマが注意すべき問題として残る．

3.6 本手法の手続き

本手法は，(1) 耐タンパハードウェアの公開鍵交換，(2) 移動元でのデータの暗号化，(3) 暗号化モバイルエージェントの移動，(4) 移動先でのデータの復号化の 4 つの手続きから成る．ただし，$ED(Key, Data)$ は，共通鍵 Key を用いて $Data$ を暗号化したデータを示し，$ER(Key, Data)$ は，公開鍵 Key を用いて $Data$ を暗号化したデータを示す．また，表 2，表 3 は，TMI，及び，TRH が保持している機能を示す．ここで，モバイルエージェントは暗

表 2: 本手法の TMI が保持する機能

関数名	関数の詳細
enc(MA, data, NodeID)	MA に対応するコードを獲得し，TRH の enc 関数を実行
dec(MA, eData)	MA に対応するコードを獲得し，TRH の dec 関数を実行
setCert(Certificate)	TRH の setCert 関数を実行
getCert()	TRH の getCert 関数を実行

表 3: 本手法の TRH が保持する機能

関数名	関数の詳細
enc(code, data, NodeID)	NodeID の TRH 公開鍵と code により data を暗号化
dec(code, eData)	TRH 自身の秘密鍵と code により eData を復号化
setCert(Certificate)	耐タンパハードウェアの公開鍵を登録
getCert()	TRH 証明書を渡す

号化データと復号化データを受け取るために，setEncData(eData, NodeID) と setDecData(data) を持つ．

(1) 耐タンパハードウェアの公開鍵交換　データの暗号化を行うために，事前にホスト間で TRH の公開鍵の交換が必要になる．ホスト B からホスト A に TRH の公開鍵を移動させる場合について示す．

1. TRH 証明書交換エージェント (TRHCertTransAgent) は，TMI_B の getCert() の実行により，TRH_B から $Cert_{TrhCA}(Pub_{TRH_B})$ を獲得（3.2 節）
2. TRHCertTransAgent は，ホスト B からホスト A へ証明書 $Cert_{TrhCA}(Pub_{TRH_B})$ を持って移動
3. TRHCertTransAgent は，TMI_A の setCert ($Cert_{TrhCA}(Pub_{TRH_B})$) を実行して，TRH の公開鍵 Pub_{TRH_B} を登録．
   (a) TRH_A は，入力された証明書が TrhCA によって作成されたかを Pub_{CA} を用いて認証
   (b) 認証成功時に，DB_{TRH_A} に移動元の公開鍵 Pub_{TRH_B} を登録

図 3 は，以下の手続きを UML のシーケンス図を用いて表現したものである．

(2) 移動元でのデータの暗号化

1. モバイルエージェント MA は，TMI_A の enc(MA, $Data$, Host B) を実行
   (a) TMI_A は MA に対応するコードを獲得
      i. Code Database の認証（認証フェーズ (1)）
      ii. MA に対応するコード $code$ を獲得
   (b) TRH_A との間で相互認証（3.4 節, 認証フェーズ (2)）
   (c) TRH_A の enc($code$, $Data$, Host B) を実行し，データを暗号化
      i. TRH_A は，$code$ から共通鍵 $sKey$ を生成
      ii. 共通鍵 $sKey$ を用いて $ED(sKey, Data)$ を作成（3.3 節）
      iii. 移動先の TRH 公開鍵 Pub_{TRH_B} を用いて $ER(Pub_{TRH_B}, ED(sKey, Data))$ $(= eData)$ を作成（3.1 節）
      iv. TMI_A に暗号化データ $eData$ を返す
   (d) MA の setEncData($eData$, Host B) を実行
2. MA は，setEncData が実行されることにより，暗号化データ $eData$ を獲得

(3) 暗号化モバイルエージェントの移動

1. ホスト A は MA が保持する $Code$ と $eData$ を含む $State$ を移動先の公開鍵 Pub_B で暗号化
2. 暗号化 State に対して署名を生成
3. 配送エージェントは，暗号化 $Code$，暗号化 $State$，署名の 3 つを持って移動
4. ホスト B は，署名を確認し，確認できれば暗号化 $Code$ と暗号化 $State$ を秘密鍵 Pri_B で復号化
5. ホスト B は，$Code$ と $State$ を用いて MA をモバイルエージェントとして実行させる

(4) 移動先でのデータの復号化　モバイルエージェントは，$Code$ と $eData$ を含む $State$ を持っている．

1. MA は，移動後，$Data$ を利用するため，TMI_B の dec(MA, $eData$) を実行
   (a) TMI_B は MA に対応するコードを獲得（上記の (2)-1-(a) と同じ）
   (b) TRH_B との間で相互認証（3.4 節, 認証フェーズ (3)）
   (c) TRH_B の dec(MA, $code$, $eData$) を実行し，データを復号化する
      i. TRH_B は自分の TRH 公開鍵 Pri_{TRH_B} を用いてデータ $ED(sKey, Data)$ を復元（3.1 節）
      ii. $code$ から共通鍵 $sKey$ を作成
      iii. 共通鍵 $sKey$ を用いてデータ $Data$ を復元（3.3 節）
      iv. TMI_B に復号化データ $Data$ を返す
   (d) MA の setDecData($Data$) を実行
2. MA は，setDecData が実行されることにより，復号化データ $Data$ を獲得

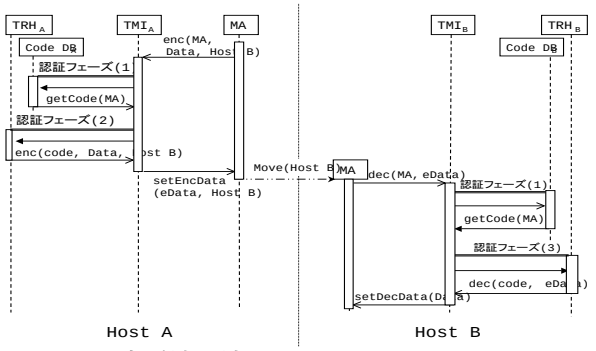

図 3: 本手法の流れの UML シーケンス図

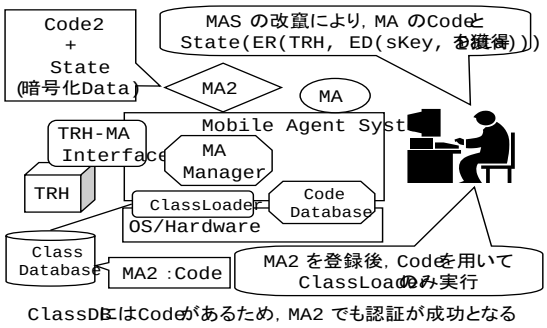

図 4: Java バイトコード獲得問題の一例

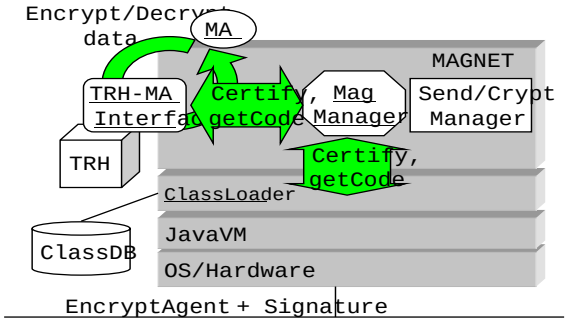

図 5: プロトタイプシステムの概略図

4 実装

今回，本システムの実装において，Java 言語 (JDK1.3, JCSI2.0 [3]) を用いた．ここで，本システムではハードウェア，OS，JavaVM の変更による攻撃はできないが，モバイルエージェントシステムは信用できるとは限らないとする．また，耐タンパハードウェア専用の CA が存在すると仮定する．

4.1 Java バイトコード獲得問題

3.3 において，復号要求元の認証にコードを鍵とした共通鍵暗号方式を用いている．Java で実装する上で，要求元のモバイルエージェントのバイトコードをどのように獲得するかという問題がある．Java には，実行中のモバイルエージェントからコードを獲得するメソッドは存在しない．そのため，ClassLoader がクラスを作成するときに，クラス名とバイトコードを対としたクラスデータベースを保持する必要がある．しかし，クラスデータベースにあるバイトコードは実行中のバイトコードと等しいとは限らない．例えば，図 4 のように，モバイルエージェント MA からバイトコード $Code$ と暗号化データ $ER(Pub_{TRH}, ED(sKey, (Data))$ を含む実行状態 $State$ を獲得できる．ただし，$sKey$ は，$Code$ から生成された共通鍵を示す．それから，$Code$ の改造版 $Code2$ と暗号化データ $ER(Pub_{TRH}, ED(sKey, (Data))$ を含む実行状態 $State$ を持つモバイルエージェント $MA2$ を作成する．$MA2$ をモバイルエージェントとして登録後，クラスデータベースにあるコードを $Code$ に書き換える．そのとき，耐タンパハードウェアはクラスデータベースにあるバイトコードを元に復号処理を行うので，$MA2$ に復号データを渡してしまう．これは，Java でコードからオブジェクトを生成する作業とオブジェクトをモバイルエージェントとして登録する作業が別に存在するためである．

この解決策として，オブジェクトをモバイルエージェントとして登録する作業をする MA Manager がオブジェクトとバイトコードを対とする MA テーブルを保持する．MA Manager はオブジェクトをモバイルエージェントとして登録するときに，対応するコードを ClassLoader が持つクラスデータベースから獲得し，これを MA テーブルに入れる．ただし，このテーブルは MA Manager が終了すると共に消滅してしまうものとする．

[3] http://security.dstc.edu.au/projects/java/jcsi.html

最後に，テーブル，及び，データベースの正当性を保つために，ClassLoader と MA Manager が信頼できるかどうかを認証する必要がある．ここで，MA Manager は ClassLoader の証明書を保持し，TMI は TrhCA の公開鍵 Pub_{CA} と MA Manager の署名を保持する．ClassLoader の認証は，TrhCA による ClassLoader の証明書を獲得し，Pub_{CA} による検証によって行う．MA Manager の認証は ClassLoader からクラスコードを獲得し，それを用いた署名の検証によって行う．

4.2 プロトタイプの作成

本手法のプロトタイプをモバイルエージェントシステム MAGNET(Mobile AGent NETwork)[5] に実装した．図 5 に，プロトタイプの概略図を示す．

プロトタイプでは，モバイルエージェント（コードと実行状態）は移動時に移動先の公開鍵を用いて暗号化され，実行状態には移動元の電子署名が添付されている．移動先では電子署名の確認をして移動先ホストの秘密鍵で復号される．その役割を果たすコンポーネントが Send/Crypt Manager である．また，プロトタイプでは仮想の耐タンパハードウェアを想定し，Java で実現している．また，下線は，認証を必要とする，つまり，証明書，あるいは，署名を必要とするコンポーネントを示し，矢印はコンポーネント間のデータのやりとりを示す．

4.3 本手法における攻撃に対する防御

表 4 に，様々な攻撃に対する本手法を利用した場合の防御の可否を示した．ただし，表中の○は防御できることを、△は一部防御できることを、×は防御できないことを示す．

表 4: 本手法における攻撃に対する防御

攻撃者	攻撃方法	防御可否
モバイル	システムの盗聴	○
エージェント	システムの改竄	○
中継先ホスト	コードの盗聴	○
	コードの改竄	○
	データの盗聴	○
	データの改竄	○
移動先ホスト	コードの盗聴	×
	コードの改竄	△
	データの盗聴	○
	データの改竄	○
	TRH のなりすまし	○

5 関連研究

セキュリティを考慮したモバイルエージェントシステムとして Aglets[6]，D'Agent[7] があげられる．

Aglets は，IBM により開発された Java モバイルエージェントシステムである．Aglets ではセキュリティドメインと呼ばれる同質のセキュリティが期待される領域内でのみ安全な通信を認めている．各セキュリティドメインには 1 つのマネージャが存在し，ドメインマネージャがセキュリティドメインに属するサーバに対して，ドメインの証である共通鍵を配布する．しかし，この手法では，ドメイン内での通信しか安全な通信ができないことやドメインを形成する際にはマネージャが存在する必要があるため，出会ったその場での安全な通信を行う場合には適さない．

D'Agent は，Tcl，Java，Scheme といった複数言語で書くことができるモバイルエージェントシステムである．D'Agent では，モバイルエージェントをコンポーネント化し，秘密データを持つコンポーネント以外のコンポーネントを他の端末に移動させ，秘密のデータが必要なときに移動先からコンポーネントが秘密のデータを持つコンポーネントを獲得して連れて行く方式をとっている．しかし，この手法では，信頼できるサーバ，あるいは，作成者自身のホストが実行時に同一ネットワーク上に存在する必要がある．

耐タンパハードウェアを用いたモバイルエージェント保護手法として，G.Wilhelm[8] の手法がある．Wilhelm は，耐タンパハードウェア内にモバイルエージェントシステム全体が保存され，モバイルエージェントは耐タンパハードウェア内で実行される手法を提案した．しかし，システム全体の挿入は，高性能で大容量のメモリを持つ耐タンパハードウェアが必要となるため，実用的とはいえない．

本研究では，アドホックネットワーク上でも扱えるモバイルエージェント保護システムであり，かつ，実用的な容量の耐タンパハードウェアで扱うことができる．

6 まとめ

本稿では，耐タンパハードウェアを用いたモバイルエージェントを保護する手法について提案した．本手法では，各ホストに耐タンパハードウェアの存在を仮定し，公開鍵暗号方式を用いたモバイルエージェントのデータのやりとりを行っている．そして，耐タンパハードウェアにアクセスできるモバイルエージェントを制限している．これにより，モバイルエージェントが持つコードと実行状態を獲得されても実行状態の中にある秘密のデータの盗聴・改竄を防ぐことができた．

また，Java 上で実装する上で問題となるバイトコード獲得問題についても MA マネージャがモバイルエージェントとバイトコードを対に持つテーブルを保持することにより解決した．

今後は，本手法における保護手法の安全性と必要実行時間を調査する必要がある．さらに，今回は特定のホスト間のみについて考えたが，グループにおけるモバイルエージェント保護手法についても考える必要がある．

参考文献

[1] Portable Document Format Reference manual: http://partners.adobe.com/asn/developer/pdfs/tn/PDFSPECS.pdf

[2] DiCelie, Joseph. Shigenori Kino , "Concordia and Its Security Features", in Proc. Japan Society of Software Technology Workshop, pp. 133-140, August 13-15(1998)

[3] 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, 第 1 回インターネットテクノロジワークショップ (WIT98) 論文集, 日本ソフトウェア科学会 インターネットテクノロジ研究会 (1998)

[4] 上野正巳，庵祥子，三宅延久，武井英明：不正コピー防止を考慮したコンテンツ販売システム，情報処理学会研究報告，IM Vol.2000 Num.36，pp.17-24 (2000)

[5] Nobuo Kawaguchi, Katsuhiko Toyama, and Yasuyoshi Inagaki: MAGNET:Ad Hoc Network System based on Mobile Agents, Computer Communication,Vol.23, No.8, pp761-768(2000)

[6] Kouichi Ono：A Security Model for Mobile Agent Framework Aglets，コンピュータソフトウェア, Vol.17 No.4，pp.2-14(2000)

[7] Robert S. Gray et al: D'agents: Security in a Multiple-Language, Mobile Agent System, in Vigana G. (ed): Mobile Agents and Security, Lecture Notes in Computer Science, Vol.1419(1998)

[8] U. G. Wilhelm. Increasing privacy in mobile communication systems using cryptographically protected objects. In Verlasliche IT-Systeme 1997, pp.319-334, Freiburg (Brsg.), Germany, November (1997).