マルチメディア通信と分散処理　111－29
コンピュータセキュリティ　20－29
（２００３．２．２８）

# 信頼されたホストを仮定したエージェントの実行結果の検証メカニズムの提案

小手川 祐樹 *　　櫻井 幸一 †

概要　モバイルエージェント技術を用いることによって，エージェント管理者は自らの代わりにモバイルエージェントにネットワークを用いた煩雑なタスクの処理を任せることができる．しかし，モバイルエージェントの実行環境であるホストがモバイルエージェントのデータを改竄するといった可能性がある．そのため，モバイルエージェントがエージェント管理者の下へ帰還したときに，その実行結果が本当に正しいものであるかといったことが問題となる．本論文では，信頼されたホストを仮定することによってモバイルエージェントの実行結果が改竄されていないかどうかを検証するメカニズムを提案する．

キーワード　モバイルエージェント，セキュリティ，Trace，改竄検出

# A Mechanism to Detect Tampering on Mobile Agents with Trusted Hosts

Yuki KOTEGAWA *　　Kouichi SAKURAI †

**Abstract**— Mobile agent technologies are powerful mechanism. An agent owner can leave processing of the task which used the network to a mobile agent instead of oneself by using mobile agent technology. However, major problem of mobile agent paradigm is protecting mobile agents from malicious hosts. In this paper, we propose a mechanism to detect tampering on mobile agents with trusted hosts.

## 1　はじめに

モバイルエージェントの定義は様々であるが，本論文における定義は「実行を開始したシステムに拘束されず，利用者の代わりに自律的に活動を行うプログラム」とする [1]．ここでシステムとはモバイルエージェントが稼動するためのランタイムシステムを指す．以後、このシステムを導入している計算機をホストと呼ぶこととする．モバイルエージェントはプログラムコードと共にプログラム内の変数なども同時に移動することにより，移動先ホストで移動前のホストでの実行状態から処理を継続することができる．この特徴によって，頻繁に通信が切断されるような信頼性の低い通信路でも他ホストとの処理を行うことができる．また，通信量を削減することも可能となる．

しかし，モバイルエージェント技術の核であるエージェントの移動に伴いセキュリティの問題が発生する．それらを大きく分けると，エージェントからの攻撃とエージェントへの攻撃とに分けることができる．ホストからエージェントを見た場合，それは外部から来るプログラムであり，エージェントに悪意を持ったコードが含まれていればホスト上のファイルやリソースを不正に使用・盗聴・改竄される可能性へと繋がる．一方，悪意を持ったホストからエージェントの情報を盗聴・改竄される可能性もある．ホス

* 九州大学工学部電気情報工学科 〒812-8581 福岡市東区箱崎 6-10-1, Department of Electrical Engineering and Computer Science, Kyushu University, 6-10-1 Hakozaki, Higashi-ku, Fukuoka City, Japan: kotegawa@tcslab.csce.kyushu-u.ac.jp
† 九州大学大学院システム情報科学研究院 〒812-8581 福岡市東区箱崎 6-10-1, Faculty of Information Science and Electrical Engineering, Kyushu University, 6-10-1 Hakozaki, Higashi-ku, Fukuoka City, Japan: sakurai@csce.kyushu-u.ac.jp

トの保護に関してはアクセス制御を行うことで解決可能であるが，エージェントの保護に関してはエージェントがホスト上で実行されるプログラムであるという本質から困難である．

そのような中で，エージェントを保護する技術として，耐タンパ装置 [2]，プログラムの難読化 [3]，Mobile Cryptography [4] といった技術が提案されている．しかしながら，悪意を持っているホストは耐タンパ装置を設置しないという問題や難読化はその効果を定量的に評価する技法が確立していないという問題，Mobile Cryptography は限られた関数にしか適用できないという問題がある．現時点ではエージェントを完全に保護することは非常に困難であると言える．

エージェントを完全に保護することができないという点から，エージェントが攻撃されたという事実を検出するための技術が注目される．その検出技術の一つとして Trace [5] というアイデアを用いたものがある．Trace はホスト上でのエージェントの実行ログであり，検証者はその Trace を参照してエージェントの実行過程のシミュレーションを行うことで，エージェントの実行結果が正しいかどうか検証する．このアプローチの大きな利点として，ホスト上でエージェントがどのように振舞ったかを把握できることが挙げられる．一方，欠点として，本来の実行に加えて検証のためのシミュレーションとしてエージェントを再実行するため，より多くの実行コストを必要とすることが挙げられる．また，Trace のサイズの肥大化を防ぐためにログをとる対象についても考慮する必要がある．この Trace を用いた既存の改竄検出方式として，Vigna [6]，Hohl [7] がある．

本論文では，エージェントがエージェント管理者の下へ帰還したときに，その実行結果が改竄されてないことを示すためのメカニズムを提案する．本提案方式は二つのエージェントを並列実行することで，既存の方式と比べて実行結果の検証が終了するまでの時間を短縮できる．また，改竄の事実を検出した場合，それ以降の無駄となるであろうタスクの処理を中止する．

本論文では，第 2 章で既存の改竄検出方式について述べ，それらの問題点を述べる．第 3 章で我々の提案方式について述べ，第 4 章で提案方式について考察を行う．最後に，第 5 章で本論文のまとめと今後の課題について述べる．

## 2 既存の改竄検出方式

### 2.1 Vigna の提案方式

Vigna が [6] で提案している方式は，タスクの処理部分とその実行結果の検証処理部分から構成される．エージェントがタスクの処理を終えてエージェント管理者の下へ帰還した後，エージェント管理者がエージェントの実行結果について検証を行う．エージェントは改竄が行われた後もタスクの処理を続けるため，改竄後からエージェント管理者の下へ帰還するまでの間の処理が無駄となる可能性がある．また，検証処理がエージェント管理者の負担にもなる．

### 2.2 Hohl の提案方式

Hohl が [7] で提案している方式では i 番目のホスト $Host_i$ での実行結果の検証を次のホスト $Host_{i+1}$ が行うので，エージェント管理者に対する実行結果の検証の負担はない．途中で改竄が検出された場合は，それ以降のタスクの処理は中止される．しかし，実行結果の検証を行うホスト $Host_{i+1}$ が直前のホスト $Host_i$ と結託した場合，検証処理を行わないことで改竄の事実を隠蔽することができてしまう．また，$Host_i$ で行った実行内容が次ホスト $Host_{i+1}$ に知られてしまうといった問題もある．

## 3 提案方式

### 3.1 要求事項

既存の方式の問題点から，本提案方式の要求事項を以下のものとする．

(1) エージェント管理者の負担の軽減
エージェント管理者はタスクの処理を命じたエージェントを最初のホストへ送り出すだけでよい．

(2) エージェントの実行結果の信頼性
単独の悪意をもったホストによる改竄攻撃や，複数の悪意を持ったホストが結託したエージェントへの改竄攻撃を検出できる．

(3) 改竄検出後の無駄なタスク処理の中止
改竄の事実を検出した場合，それ以降のタスクの処理を行わない．

(4) エージェントの実行内容の秘匿性
エージェントの実行を行ったホストとエージェント管理者以外はその実行内容を知ることができない．

### 3.2 モデル

#### 3.2.1 前提

提案方式を述べるにあたって以下のものを前提とする.

(1) 各ホストは公開鍵暗号を用いて自らが責任をもつデータに対して署名を行える.

(2) タスクエージェントと検証エージェントはホストによって安全な通信路を与えられる.

(3) 信頼されたホストはエージェントを実行するだけである.

(4) 信頼されたホスト以外ではエージェントの保持するデータは盗聴・改竄の可能性がある.

(5) ホスト間の通信路は安全であり, エージェントは移動中に第三者から攻撃を受けることはない.

#### 3.2.2 構成要素

本提案方式は次のエンティティから構成される.

(1) **エージェント管理者**
エージェントに対して責任を負う利用者.

(2) **ホスト**
エージェントのデータに対して盗聴・改竄を行う可能性があるエージェントの実行環境.

(3) **信頼されたホスト**
エージェント管理者が管理するエージェントの実行環境, または耐タンパ装置を備えたエージェントの実行環境 [8].

(4) **タスクエージェント**
エージェント管理者の命じたタスク処理を任意のホストで行うモバイルエージェント.

(5) **検証エージェント**
タスクエージェントの実行結果を検証するエージェント.

#### 3.2.3 記号の定義

$\boldsymbol{AO}$: エージェント管理者
$\boldsymbol{Host_i}$: タスクエージェントの $i$ 番目の実行環境
$\boldsymbol{TrustedHost}$: 信頼された実行環境
$\boldsymbol{TA}$: 任意の実行環境で実行されるタスクエージェント
$\boldsymbol{VA}$: 信頼された実行環境で実行される検証エージェント
$\boldsymbol{C_T}$: TA のプログラムコード
$\boldsymbol{C_V}$: VA のプログラムコード
$\boldsymbol{S_{Ti}}$: $Host_i$ で実行された後の TA の実行状態 (実行結果)
$\boldsymbol{S_{Vi}}$: $S_{Ti}$ の検証を行った後の VA の実行状態 (実行結果)
$\boldsymbol{Sign_i(M)}$: $Host_i$ によるメッセージ $M$ への署名
$\boldsymbol{id_T}$: TA の識別子
$\boldsymbol{id_V}$: VA の識別子 (VA が実行される信頼された実行環境の位置情報を含む)
$\boldsymbol{Trace_i}$: $Host_i$ から TA への入力
$\boldsymbol{PK}$: Trace の暗号化に使用する公開鍵
$\boldsymbol{SK}$: Trace の復号化に使用すると秘密鍵
$\boldsymbol{PK(M)}$: PK を用いて暗号化されたメッセージ M

### 3.3 エージェントの実行と検証手順

#### 3.3.1 エージェント管理者 AO の処理

(1) TA を含むエージェントメッセージ $AM_0$ を作成し, 最初の実行環境 $Host_1$ へ送信する.

$$AM_0 = \{C_T,\ S_{T0},\ Sign_{AO}(S_{T0}),\ PK,\ id_T,\ id_V,\ Sign_{AO}(C_T,\ PK,\ id_T,\ id_V)\}$$

(2) VA を含むエージェントメッセージ $AM_0'$ を作成し, $TrustedHost$ へ送信する.

$$AM_0' = \{C_V,\ S_{V0},\ Sign_{AO}(S_{V0}),\ SK,\ id_T,\ id_V,\ Sign_{AO}(C_V,\ SK,\ id_T,\ id_V)\}$$

#### 3.3.2 実行環境 $\mathbf{Host_1}$ の処理

(1) AO からメッセージ $AM_0$ を受信する.

$$AM_0 = \{C_T,\ S_{T0},\ Sign_{AO}(S_{T0}),\ PK,\ id_T,\ id_V,\ Sign_{AO}(C_T,\ PK,\ id_T,\ id_V)\}$$

(2) 受信したメッセージ $AM_0$ に含まれる署名 $Sign_{AO}(S_{T0})$, $Sign_{AO}(C_T, PK, id_T, id_V)$ を検証する.

(3) TA の実行を開始する.

(4) TA から移動要求を受けたら, TA の実行を停止する.

(5) エージェントメッセージ $AM_1$ を作成し, $Host_2$ へ送信する.

$$AM_1 = \{C_T,\ S_{T1},\ PK,\ PK(Trace_1),\ id_T,\ id_V,\ Sign_{AO}(C_T,\ PK,\ id_T,\ id_V),\ Sign_1(S_{T1},\ PK(Trace_1))\}$$

#### 3.3.3 実行環境 $\mathbf{Host_i}$ の処理

(1) $Host_{i-1}$ からメッセージ $AM_{i-1}$ を受信する.

$AM_{i-1} = \{C_T, S_{Ti-1}, PK, PK(Trace_{i-1}), id_T, id_V, Sign_{AO}(C_T, PK, id_T, id_V), Sign_{i-1}((S_{Ti-1}), PK(Trace_{i-1}))\}$

(2) 受信したメッセージ $AM_{i-1}$ に含まれる署名 $Sign_{AO}(C_T, PK, id_T, id_V)$, $Sign_{i-1}(S_{Ti-1}, PK(Trace_{i-1}))$ を検証する.

(3) 検証メッセージ $VM_{i-1}$ を作成し, $VA$ へ送信する.

$VM_{i-1} = \{S_{Ti-1}, PK(Trace_{i-1}), id_T, id_V, Sign_{i-1}(S_{Ti-1}, PK(Trace_{i-1}))\}$

(4) TA の実行を開始する.

(5) TA から移動要求を受ければ TA の実行を停止する.

(6) エージェントメッセージ $AM_i$ を作成し, $Host_{i+1}$ へ送信する.

$AM_i = \{C_T, S_{Ti}, PK, id_T, id_V, PK(Trace_i), Sign_{AO}(C_T, PK, id_T, id_V), Sign_i(S_{Ti}, PK(Trace_i))\}$

#### 3.3.4 検証エージェント VA の処理

(1) 検証メッセージ $VM_{j-1}$ を $Host_j$ から受信する.

$VM_{j-1} = \{S_{Tj-1}, PK(Trace_{j-1}), id_T, id_V, Sign_{j-1}(S_{Tj-1}, PK(Trace_{j-1}))\}$

もし, このとき不正フラグが立っていれば, $Host_j$ で実行されている TA に対してタスク処理の中止命令を送信する.

(2) $VM_{j-1}$ に含まれる $id_T, id_V$ と自らが保持している $id_T, id_V$ を比較する.

(3) $VM_{j-1}$ に含まれる署名 $Sign_{j-1}(S_{Tj-1}, PK(Trace_{j-1}))$ を検証する.

(4) 秘密鍵 SK を用いて $PK(Trace_{j-1})$ から $Trace_{j-1}$ を復号化する.

(5) 前回の検証結果である $S_{Tj-2}$ と復号化した $Trace_{j-1}$ を用いて $S'_{Tj-1}$ を計算する.

(6) $S'_{Tj-1}$ と VM に含まれる $S_{Tj-1}$ を比較する. これらが一致すれば $Host_{j-1}$ は不正をしていないことが認められる. もし, これらが一致しなかった場合は, $Host_{j-1}$ が不正を行ったとみなし, 不正フラグを立てる.

(7) 検証結果もとにレポートを作成する.

#### 3.3.5 タスクエージェント TA の処理

(1) エージェント管理者に与えられたタスク処理を実行する.

(2) VA からタスク処理の中止命令を受信した場合, VA の滞在している信頼された実行環境への移動要求を現在滞在している実行環境へ行う.

(3) VA から中止命令を受信することなく, 現在の実行環境でのタスク処理を終えれば, 次の実行環境への移動要求を現在滞在している実行環境へ行う. もし, タスク処理がすべて終わっていれば, 次の実行環境は VA が実行されている信頼された実行環境となる.

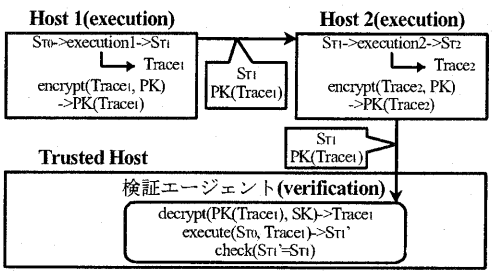


図 1: エージェントの実行と検証

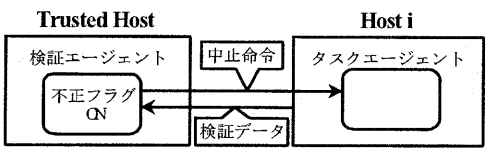


図 2: 不正フラグ ON 時に検証データを受信した場合

## 4 考察

### 4.1 要求事項について

(1) エージェント管理者の負担の軽減

本提案方式では検証処理は検証エージェントが

行うため，エージェント管理者はタスクの処理を命じたエージェントを最初のホストへ送り出すだけでよい．

(2) エージェントの実行結果の信頼性

□単独のホストによる攻撃

単独のホストによる攻撃について考える．悪意をもったホスト Hosti は VA に偽造した検証データを送り，前ホスト Hosti-1 を悪者にしようとするかもしれない．しかし，STi-1; PK(Tracei-1) にはそれぞれ Hosti-1 により署名が施されているので，検証データを偽造することは不可能である．また，悪意をもったホスト Hosti は VA に検証データを送らずに，エージェントが訪れたことを隠そうとするかもしれない．しかし，Hosti は次ホストへエージェントを送信するときに，実行ログ Tracei を暗号化したもの PK(Tracei) とその実行結果 STi に署名をしなければならない．そして，それらは Hosti+1 によって VA に送信されるので，VA に検証データを送らなかったことが明らかになる．

□複数の連続したホストによる攻撃

悪意を持った連続した実行環境 $Host_i$ と $Host_{i+1}$ が結託する場合を考える．まず，$Host_i$ がタスクエージェントの本来の正しい実行状態 $S_{Ti}$ を $S''_{Ti}$ へと改竄したとする．このとき，$Host_i$ は $Host_{i+1}$ へ以下の2つのメッセージを送信する．

$$AM''_i = \{C_T,\ S''_{Ti},\ PK,\ PK(Trace_i),\ id_T,\ id_V,\ Sign_{AO}(C_T,\ PK,\ id_T,\ id_V),\ Sign_i(S''_{Ti},\ PK(Trace_i))\}$$

$$M = \{S_{Ti},\ Sign_i(S_{Ti},\ PK(Trace_i))\}$$

$AM''_i$ は改竄された実行状態をもつエージェントメッセージであり，M は検証をすり抜けるための本来の正しい実行状態とそれに対する署名からなる．これらを受信した $Host_{i+1}$ は VA に以下の検証メッセージ $VM_i$ を送信する．

$$VM_i = \{S_{Ti},\ id_T,\ id_V,\ PK(Trace_i),\ Sign_i(S_{Ti},\ PK(Trace_i))\}$$

VA は VM から $S_{Ti}$ を検証するが，$S_{Ti}$ と $PK(Trace_i)$ との対応が正しいので改竄を検出できない．

次に $Host_{i+1}$ は，TA の初期状態を $S''_{Ti}$ として TA の実行を開始し，$Host_{i+1}$ での TA への入力 $Trace_{i+1}$ を用いて $S_{Ti+1}$ を計算する．そして，次の実行環境 $Host_{i+2}$ へ次のエージェントメッセージ $AM_{i+1}$ を送信する．

$$AM_{i+1} = \{C_T,\ S_{Ti+1},\ PK,\ PK(Trace_{i+1}),\ id_T,\ id_V,\ Sign_{AO}(C_T,\ PK,\ id_T,\ id_V),\ Sign_{i+1}(S_{Ti+1},\ PK(Trace_{i+1}))\}$$

そして，$Host_{i+2}$ は以下の検証メッセージ $VM_{i+1}$ を VA に送信する．

$$VM_{i+1} = \{S_{Ti+1},\ PK(Trace_{i+1}),\ id_T,\ id_V,\ Sign_{i+1}(S_{Ti+1},\ PK(Trace_{i+1}))\}$$

ここで，VA は $Host_{i+2}$ から受信した $VM_{i+1}$ から $Host_{i+1}$ での実行状態 $S_{Ti+1}$ を検証する．しかし，$S_{Ti+1}$ は改竄された実行状態 $S''_{Ti}$ に入力 $Trace_{i+1}$ を用いて計算した結果なので，本来の正しい実行状態 $S_{Ti}$ に $Trace_{i+1}$ を用いて計算したものとは当然異なる．このようにして，$Host_{i+1}$ が不正を行ったことが発覚し，結託した攻撃を検出可能である．

(3) 改竄検出後の無駄なタスク処理の中止

VA から TA への実行中止命令が悪意を持ったホストによって妨害される場合を考える．例えば，$Host_i$ で実行されている TA へ実行中止命令を送信した場合に、$Host_i$ が実行中止命令を TA へ伝えずに破棄したとする．その場合 $Host_i$ では TA の実行を中止することはできないが，VA の不正フラグは ON になったままであるので，不正を行わないホストまで TA が移動した時に中止命令が VA に届き，実行を中止することができる．前述した結託攻撃の場合と同様にすべてのホストが結託しない限り TA の実行を中止することが可能である．

(4) エージェントの実行内容の秘匿性

ホスト $Host_i$ での実行ログである $Trace_i$ は TA の公開鍵 PK によって暗号化されてから，次のホスト $Host_{i+1}$ へ送信される．公開鍵 PK に対応する秘密鍵 SK は信頼されたホスト上で実行される VA が保持しているため，ホスト $Host_{i+1}$ は復号化することができない．暗号化方式が破られない限りエージェントの実行内容の秘匿性は満たされる．

表 1: 提案方式と既存の方式の比較

| | 検証を行う エンティティ | 管理者の 負担軽減 | リアルタイムな 改竄検出 | 実行結果 の信頼性 | 実行内容 の秘匿性 | 実行環境の 制限 |
|---|---|---|---|---|---|---|
| Vigna の方式 | エージェント管理者 | × | × | ○ | ○ | なし |
| Hohl の方式 | 複数の実行環境 | ○ | ○ | △ | × | なし |
| 本提案方式 | 検証エージェント | ○ | ○ | ○ | ○ | あり |

### 4.2 検証のタイミングについて

本提案方式では，TA と VA の二つエージェントを利用することによって，タスクの実行と実行結果の検証を並列して行う．これより，TA は VA の検証結果を待つことなく与えられたタスク処理を続行することができる．例えば，$Host_i$ での TA の実行結果の検証を VA が行っている間でも，TA は $Host_{i+1}$ でタスクの処理が可能である．検証処理が行われるタイミングに関して，提案方式とそれぞれの方式を比較したイメージを図 3 に表す．Vigna や Hohl の方式と比較すると，エージェントがタスクの処理を終え，さらに実行結果の検証が終了するまでの時間が短縮されることが分かる．このことは，エージェントのタスク全体の処理時間が長ければ長いほど他方式にくらべて有効である．しかし，検証処理中に実行されているタスク処理については無駄になる可能性がある．

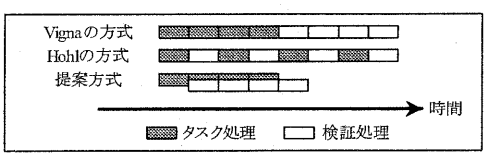


図 3: 各方式の検証タイミングの比較

## 5 まとめ

本論文では，既存の改竄検出方式の問題点を改善した改竄検出メカニズムの提案を行った．我々の提案方式では，エージェントが帰還したとき保持している結果が改竄されていないことが保証される．また，二つのエージェントと並列実行することで，検証処理が終了するまでの時間が短縮できる．しかし，本提案方式には仮定である信頼されたホストを確保しなければならないという問題がある．今後の課題は，Java による本提案方式の実装を行い，実際の時間コストについての調査，考察を行いたい．

## 参考文献

[1] OMG, "Mobile Agent Facility Specification", 2000

[2] Uwe G. Wilhelm, "Cryptographically Protected Objects", EcolePolytechnique Fèdèrale de Lausanne, Switzerland, 1997

[3] Fritz Hohl, "Time Limited Blackbox Security: Protecting Mobile Agents From Malicious Hosts",Springer LNCS 1419, pp. 92-113, 1998

[4] Tomas Sander, Christian F. Tschudin, "Protecting Mobile Agents Against Malicious Hosts", Springer LNCS 1419, pp 44-60, 1998

[5] Bennet Yee, "A Sanctuary for Mobile Agents", DARPA Workshop on Foundations for Secure Mobile Code Workshop, 1997

[6] Giovanni Vigna,"Cryptographic Traces for Mobile Agents", Springer LNCS 1419, pp 137-158, 1998

[7] Fritz Hohl,"A Protocol to Detect Malicious Hosts Attacks by Using Reference States", Technical Report Nr. 09/99, Faculty of Informatics, University of Stuttgart, Germany, 1999

[8] Uwe G. Wilhelm, Sebastian Staamann, Levente Buttyan, "Introducing Trusted Third Parties to the Mobile Agent Pradigm", Springer LNCS1603, pp 469-489, 1999