社団法人　情報処理学会　研究報告
IPSJ SIG Technical Report

2005－DPS－122（22）
2005－CSEC－28（22）
2005／3／22

モバイルエージェント・セキュリティに関する一考察

森　正行†　双紙　正和†　宮地　充子†

† 北陸先端科学技術大学院大学情報科学研究科
〒 923–1293 石川県能美市旭台 1–1

あらまし　モバイルエージェント・セキュリテイの 1 つの手法として, 暗号回路と紛失通信を使った secure function evaluation がある. しかし, secure function evaluation を用いた既存研究では, モバイルエージェントの持つ暗号回路とそれを実行処理するホスト (実行ホスト) の 2 者間における通信手段が明確でなかった. また, 仮にモバイルエージェントの持つ暗号回路への任意の入力がなんらかの方法で可能になった場合, 実行ホストがそれを利用して, 不正な計算などを行えるため, エージェントの秘密情報が漏れる問題があった. 本研究では, モバイルエージェントと実行ホストとの 2 者間の通信を明確にするため, エージェントと実行ホストとの間に信頼できるホストというものを導入し, 信頼できるホストと実行ホストとの通信を行う. ここで, この通信を安全に行うために紛失通信を用いるが, 原始的な 1-out-of-2 紛失通信では効率が悪い. そこで、k-out-of-n 紛失通信と呼ばれるプロトコルをモバイル・エージェントに適応できるように改良する. さらに本稿では, 従来の紛失通信と比較して, 提案方式の方が計算効率がよいことを示す.
キーワード　モバイルエージェント, セキュリティ, 紛失通信, 暗号回路, secure function evaluation

Consideration for mobile agent security

Masayuki MORI†, Masakazu SOSHI†, and Atsuko MIYAJI†

† School of Information Science,
Japan Advanced Institute of Science and Technology(JAIST)
1–1, Asahidai, Nomi, Ishikawa 923–1292 Japan

Abstract　We can utilize secure function evaluation with encrypted circuits and oblivious transfer in order to ensure security of mobile agents. However, most of previous works using secure function evaluation for mobile agent security do not explicitly specify the details of communication between mobile agents and their underlying execution hosts. Futhermore, if an adversary can put arbitrary data into the encrypted circuit of a mobile agent, then the adversary could obtain some secret information stord in the agent. Therefore, in our work first we introduce a trusted host between mobile agents and their execution hosts. Moreover, we use oblivious transfer in communication via the trusted host. Unfortunately, existing 1-out-of-2 oblivious transfer protocols are inefficient for that perpose, so we propose a new oblivious transfer protocol based on a k-out-of-n scheme too realize secure communication for mobile agents. Finally, we show that our oblivious transfer protocol is more efficient than previous ones.
Key words　mobile agent, security, oblivious transfer, encrypted circuit, secure function evaluation

1. はじめに

ネットワーク上のコンピュータを移動し, ユーザの代理として実行処理をするプログラム (ソフトウェア) をモバイルエージェントという. モバイルエージェントは, 自身が他のコンピュータ上で能動的に実行, 移動等を選択する自律的なプログラムであり, 現在, モバイルエージェントの技術は分散オブジェクトに続く次世代のコンピューティング基盤技術として注目を集めている. モバイルエージェントの利点として, エージェントを移動させることによるコンピュータ間の通信遅延の低減, エージェントを複数生成することによる計算処理の並列化等があげられる. しかし, モバイルエージェントを実現するにあたり脅威となる攻撃が大きく分けて 2 つある. 1 つは, 悪意あるモバイエージェントがウイルス等によってホストに被害を及ぼす攻撃, もう一方は, モバイルエージェントが悪意あるホスト上で実行処理される際の秘密データの盗聴や改ざんといった攻撃である. 前者の攻撃においてのセキュリテイ技術は, ウイルス駆除ソフト, JAVA セキュリティ技術等確立したものが存在するが, 後者

における攻撃のセキュリテイ技術は, プログラムが平文として実行されなければならないという制約を持つため, 解決が困難とされてきた. そこで, モバイルエージェントの本格的な実用化に向けて後者の攻撃に対するセキュリテイ技術の必要性が重要視され始めており, 中でも有力な解決策が 2 つ提案されている. 1 つは T. Sander ら [1] による環準同型暗号というものと, もう 1 つは C. Cachin ら [4] による secure function evaluation を使った方法である.

環準同型暗号と secure function evaluation が注目される理由として, プログラムを暗号化したまま実行処理できる, つまり, 暗号化された情報は攻撃者の改ざんといった攻撃が容易でなくなる点があげられる. 環準同型暗号は, 数学的性質の準同型性を利用して, 暗号文の演算を可能にするものであるが, 準同型の乗算に関して安全性に問題 [1] があることが知られている. 一方, secure function evaluation は, 暗号化した回路等を使うことによって, 秘密情報を漏らさず安全に計算するものである. この結果, 悪意あるホスト上でも, プログラムは暗号化されているのでホストからの攻撃を受けずに実行処理ができる.

しかしながら, secure function evaluation を用いた方法でモバイルエージェントのセキュリティを考えたとき, エージェントと実行ホスト間で紛失通信と呼ばれる通信プロトコルを用いられるが, この通信では 1-bit ごとに入力を行わなければならないため, 並列処理をしても計算効率が悪い. そこで本稿では, secure funciton evaluation を用いたモバイルエージェントの保護手法を研究するにあたり, エージェントと実行ホストに間で, より計算効率がよいプロトコルの提案し, 既存の方式との性能比較を示す.

2. 準　　備

2.1 紛失通信

紛失通信 (oblivious transfer) とは, マルチパーティプロトコルに使われるツールである [10]. 最も一般的な紛失通信は 1-out-of-2 紛失通信とよばれ, 送信者が 2 つの情報 m_0, m_1 を持ち, そのうちの 1 つのみが受信者に伝わるが, 受信者がどちらの情報を受け取ったかを送信者は知ることができず, また受信者は選択しなかった情報に関して何 1 つ知ることができないといったプロトコルである.

2.2 k-out-of-n 紛失通信

k-out-of-n 紛失通信ははじめ M. Naor [2] らによって考案されたが, 計算量等, より効率的な提案が C. Chu [8] らによって考案された. このプロトコルは, 送信者が m_1, m_2, ..., m_n 個の情報を持ち, そのうちの k 個の情報が受信者に伝わるが, 受信者がどの情報を受け取ったかを送信者は知ることができず, また受信者は選択しなかった情報に関して何１つ知ることができないといったものである.

受信者は添え字番号 i_1, i_2, ..., i_k を選ぶ. また, 送信者のメッセージは l-bit とする. ここで, このプロトコルは楕円曲線を利用して示す. 公開情報は (P, H_1, H_2, G)

Step1　受信者はランダムに $a_j \in \mathbb{Z}_q^*$ を選び, $Q_{i_j} = H_1(i_j)$ と $A_j = Q_{i_j} + a_j P$ を計算する. ただし $j = 1, 2, ..., k$

Step2　受信者は A_1, A_2, ..., A_k を送信者に送る.

Step3　送信者はランダムに $s \in \mathbb{Z}_q^*$ を選び, $P_0 = sP$, $D_j = sA_j$, $c_i = m_i \oplus H_2(sQ_i, i)$ を計算する. ただし $i = 1, 2, ..., n$, $j = 1, 2, ..., k$

Step4　送信者は P_0, D_1, D_2, ..., D_k, c_1, c_2, ..., c_n を受信者に送る.

Step5　受信者は $K_j = D_j - a_j P_0$ を計算してメッセージ $m_{i_j} = c_{i_j} \oplus H_2(K_j, i_j)$ を入手する. ただし $j = 1, 2, ..., k$

ここで k-out-of-n 紛失通信の送信者, 受信者の安全性は CT-CDH 仮定 [9] の下に保証されている. 記号は 4.6 を参照

3. 既 存 研 究

C.Cachin らは P.Rogaway [7] の暗号回路を用いた secure function evaluation を用いてモバイルエージェントセキュリティを確保する手法を提案した. しかし, [4] では, 悪意あるホストが暗号回路を不正に使用して様々な計算を行えるため, モバイル・エージェントの秘密に関する情報が漏れてしまう可能性もでてきてしまう. そこで C.Cachin らは [5] で信頼できるホストというものを置き, 悪意あるホストの不正な計算を防ぐといった提案をした. 詳細は以下に述べる.

3.1 攻撃者モデル

まず始めに, 攻撃者モデルについて述べる. secure function evaluation における攻撃者モデルは以下のように分類できる.

- honest-but-curious adversary

プロトコルに従うが, 不正に情報を入手しようとする攻撃者

- malicious adversary

プロトコルに従わず, 不正に情報を入手する攻撃者

3.2 暗号回路構築

この節では, secure function evaluation の 1 ツール、暗号回路 [7] の構築を示す.

x, y, z のバイナリ展開した表記を (x_1, x_2, ..., x_{n_x}), (y_1, y_2, ..., y_{n_y}), (z_1, z_2, ..., z_{n_z}) とし, $g(\cdot, \cdot)$ を計算する多項式サイズ回路を C とする. また, 暗号回路を作るアルゴリズムを $construct$, 2 者間での通信プロトコルアルゴリズムを $transfer$, 計算結果を入手するアルゴリズムを $evaluate$ とする. アリス, ボブの 2 者間における secure function evaluation のプロトコ

ルを以下に示す. 仮定として, 攻撃モデルは honest-but-curious でアリスは多項式時間計算能力を持ち, ボブは非多項式時間の計算能力を持つものとする.

Step1　アリスはアルゴリズム A_1 に自身の秘密情報 x を代入する. アルゴリズム A_1 は以下の操作を行う.

- ここで, 各 $x = (x_1, x_2, ..., x_{n_x})$ から $s = (\alpha^{(1)}, ..., \alpha^{(n_x)})$ を出力する.
- 次に $\alpha^{(i)}$ から $(\delta^{(i)}, \beta_0^{(i)}, \beta_1^{(i)})$ を計算する. ただし, $\delta^{(i)} \in_R G$, $\beta_c^{(i)} = g^{\alpha^{(i)}}$ と $\beta_{c\oplus 1}^{(i)} = \delta / \beta_c^{(i)}$, $(i = 1, 2, ..., n_x)$
- メッセージ $m_1 = ((\delta^{(1)}, \beta_0^{(1)}, \beta_1^{(1)}), ..., (\delta^{(n_x)}, \beta_0^{(n_x)}, \beta_1^{(n_x)}))$ としてボブに送る.

Step2　ボブは, アルゴリズム B に, アリスから受け取った m_1 と自身の秘密入力 y を代入する。アルゴリズム B は以下の操作を行う

- アルゴリズム $construct(C, y)$ から暗号回路, 暗号回路入力, 暗号回路出力 $\mathcal{C}$, $(K_{1,0}, K_{1,1})$, ..., $(K_{n_x,0}, K_{n_x,1})$, $(U_{1,0}, U_{1,1})$, ..., $(U_{n_z,0}, U_{n_z,1})$ を出力する.
- $r_0^{(i)}, r_1^{(i)} \in_R \mathbb{Z}_q$ を選び $(e_0^{(i)}, f_0^{(i)}) = (g^{r_0}, K_{i,0}\beta_0^{r_0})$, $(e_1^{(i)}, f_1^{(i)}) = (g^{r_1}, K_{i,1}\beta_1^{r_1})(i = 1, 2...., n_x)$ を用いて $m_{2,i} = (e_0^{(i)}, f_1^{(i)}, e_0^{(i)}, f_1^{(i)})$ を計算する.
- $m_2 = (\mathcal{C}, m_{2,1}, ..., m_{2,n_x}, (U_{1,0}, U_{1,1}), ..., (U_{n_z,0}, U_{n_z,1}))$ メッセージとしてアリスに送る.

Step3　アリスは, アルゴリズム A_2 にボブから受け取った m_2 と Step1 で生成した s を代入する. アルゴリズム A_2 は以下の操作を行う.

- ボブから受け取った $(e_{x_i}^{(i)}, f_{x_i}^{(i)})$ から $f_c^{(i)} / e_c^{(i)\alpha^{(i)}}$ を計算し, $(K_{1,x_1}, ..., K_{n_x,x_{n_x}})$ を入手する. $(i = 1, 2, ..., n_x)$
- $evaluate(\mathcal{C}, K_{1,x_1}, ..., K_{n_x,x_{n_x}})$ を計算して出力結果 $(U_{1,z_1}, ..., U_{n_z,z_{n_z}})$ から z を復元する.

アリス, ボブの安全性は DDH 仮定の下に保証されている.

3.3　2 者間における, 信頼できるホストを使った secure function evaluation

3.3.1　信頼できるホスト

まず, 信頼できるホストを以下に定義する.

- 耐タンパハードウェアを備えたホストである.
- モバイルエージェントに関する秘密情報を入手しない.
- 他のホストと結託をしない.

3.3.2　記号の定義

O　：　モバイルエージェントを生成するホスト;
H　：　モバイルエージェントを実行するホスト;
T　：　耐タンパハードウェアを備えた, 信頼できるホスト;
E_T　：　信頼できるホスト T の公開鍵で暗号化すること;
D_T　：　信頼できるホスト T の秘密鍵で復号すること;
ξ　：　全ての実行ホストに計算を行わせた, 計算結果 $g_l(\cdots(g_1(x, y_1), y_2)\cdots, y_l)$;
$\mathcal{L}$　：　秘密情報 x に相当する暗号回路入力, $(L_{1,0}, L_{1,0})$, ..., $(L_{n_x,0}, L_{n_x,1})$ を示す;
$\mathcal{K}$　：　秘密情報 y に相当する暗号回路入力, $(K_{1,0}, K_{1,0})$, ..., $(K_{n_y,0}, K_{n_y,1})$ を示す;
$\mathcal{U}$　：　計算結果 z に相当する暗号回路出力, $(U_{1,0}, U_{1,0})$, ..., $(U_{n_z,0}, U_{n_z,1})$ を示す;

プロトコルを以下に示す.

Step1　ホスト O は名前等の情報が記入された id を選び, $construct(C)$ を計算する.

- $construct(C)$ より $(\mathcal{C}, \mathcal{L}, \mathcal{K}, \mathcal{U})$ を出力する.
- 信頼できるホストの公開鍵で $\bar{K} = E_T(id||1||(K_{1,0}, K_{1,1})||2||..., ||n_y||(K_{n_y,0}, K_{n_y,1}))$ のように暗号化する.
- 自身の秘密入力 $(x = x_1, x_2, ..., x_{n_x})$ に対応するように選んだ L_{i,x_i} を L'_i とする. $(i = 1, 2, ..., n_x)$
- $id, \mathcal{C}, L'_1, ..., L'_{n_x}, \bar{K}, \mathcal{U}_z$ を次のホスト H に送る. ただし, $\mathcal{U} = \mathcal{U}_x + \mathcal{U}_z$ と分ける.

Step2　実行ホスト H は以下の操作を行う.

- 自身の秘密情報 $(y = y_1, y_2, ..., y_{n_y})$ に対応するように選んだ $\bar{K}_{i,y_i}$ を $\bar{K}'_i$ とする. $(i = 1, 2, ..., n_y)$
- id と $\bar{K}'_i$ を T に送る. $(i = 1, 2, ..., n_y)$

Step3　信頼できるホスト T は id と添え字 i を確認して, 復号した K'_i を実行ホスト H に返す.$(i = 1, 2, ..., n_y)$

Step4　実行ホスト H は以下の操作を行う.

- $evaluate(\mathcal{C}, L'_1, ..., L'_{n_x}, K'_1, ..., K'_{n_y})$ を計算する.
- 計算結果の $U'_1, ..., U'_{n_x+n_z}$ 出力.
- 計算結果と $\mathcal{U}_z$ から $(z = 1, 2, ..., n_z)$ を入手し, 残りの $U'_1, ..., U'_{n_x}$ をホスト O に送る.

Step5　ホスト O は実行ホスト H から手に入れた $U'_1, ..., U'_{n_x}$ と $\mathcal{U}_x$ から $\xi = \xi_1, ..., \xi_{n_x}$ を入手する

3.4　既存方式の問題点

ここで, モバイルエージェントが持つ暗号回路入力を信頼できるホストの公開鍵によって暗号化することによって, 安全性を高めた. しかし, この提案ではモバイルエージェントの持つ暗号回路入力を実行ホストに渡す方法が明記されていない. このことより, 悪意あるホストがモバイルエージェントの持つ信頼できるホストの公開鍵で暗号化された暗号回路入力を全て入手し, 自由に選択して, 信頼できるホストに復号してもらうことができてしまい, そこから不正に計算を行うことが可能になってしまう. 仮に, 1-out-of-2 紛失通信を持ちいて, 暗号回路入力

を実行ホストに渡そうとしても, 1-bit ごとにパラレルに通信を行うので効率が悪い. そこで, モバイルエージェントが持つ暗号回路入力を入力ごとに信頼できるホストの公開鍵で暗号化するのではなく, 一度に暗号回路入力を全て暗号化し, 各実行ホストは, 一旦, 信頼できるホストに暗号化された暗号回路入力を渡す. そこから, 信頼できるホストと実行ホスト間で紛失通信を用いて, 安全に暗号回路入力を渡す提案をする.

既存の 1-out-of-2 紛失通信と k-out-of-n 紛失通信を比較して計算量, メッセージ数を削減したプロトコルを示す.

4. 提 案 方 式

[5] をもとにして, モバイルエージェントとエージェントを計算するホスト (実行ホスト) の間に信頼できるというホストを備え, モバイルエージェント代わりに信頼できるホストが実行ホストに計算させるまでの通信において, 計算量、メッセージ数を削減したプロトコルを提案する.

4.1 信頼できるホストと実行ホストの通信プロトコル

信頼できるホストと実行ホストにおいて, 紛失通信を行う以前 [5] のプロトコルを示し, 提案方式との違いを明確にする.

ただし, 実行ホストの数を増やしても機能的に変わらないので, モバイルエージェント, 信頼できるホスト T, 実行ホスト O の 3 者間のみのプロトコルを示す.

Step1　ホスト O は名前等記入された id を選び, $construct(C)$ を計算する.

- $construct(C)$ より $(\mathcal{C}, \mathcal{L}, \mathcal{K}, \mathcal{U})$ を出力する.
- 信頼できるホストの公開鍵で $\bar{K}_{i,b} = E_T(id||i||K_{i,b})$ のように暗号化する.$(i = 1, 2, ..., n_y)$, $b \in \{0, 1\}$
- 自身の秘密入力 $x = (x_1, x_2, ..., x_{n_x})$ に対応するように選んだ L_{i,x_i} を L'_i とする.$(i = 1, 2, ..., n_x)$
- $id, \mathcal{C}, L'_1, ..., L'_{n_x}, \bar{K}, \mathcal{U}_z$ を次のホスト H に送る.

Step2　実行ホスト H は以下の操作を行う.

- 自身の秘密情報 $y = (y_1, y_2, ..., y_{n_y})$ に対応するように選んだ $\bar{K}_{i,y_i}$ を $\bar{K}'_i$ とする.$(i = 1, 2, ..., n_y)$
- id と $\bar{K}'_i$ を T に送る.$(i = 1, 2, ..., n_y)$

Step3　信頼できるホスト T は id と添え字 i を確認して, 復号した K'_i を実行ホスト H に返す.$(i = 1, 2, ..., n_y)$

ここで Step2 における H の $\bar{K}_{i,y_i}$, $(i = 1, 2, ..., n_y)$ のとり方が明確ではないので, 以下に原始的な方式として, 詳細なプロトコルを示す.

4.2 原始的な方式

Step2 において信頼できるホスト T と実行ホスト H に n 回, 1-out-of-2 紛失通信を適用する.

4.2.1 プロトコルの詳細

原始的なプロトコルで用いる記号を以下のように定義する

q	:	大きな素数 ;
p	:	$p = 2q + 1$ となるような大きな素数 ;
$K_{i,b}$	:	$G \in \mathbb{Z}_p$ の元 $(i = 1, 2, ..., n_y)$, $b \in \{0, 1\}$;
$\delta^{(i)}$	:	G の元 $(i = 1, 2, ..., n_y)$;
$\alpha^{(i)}$	:	$\mathbb{Z}_q$ の元 $(i = 1, 2, ..., n_y)$, $b \in \{0, 1\}$;
$r_b^{(i)}$	:	$\mathbb{Z}_q$ の元 $(i = 1, 2, ..., n_y)$, $b \in \{0, 1\}$;

ただし, p と g は公開している.

Step1　ホスト O は名前等記入された id を選び, $construct(C)$ を計算する.

- $construct(C)$ より $(\mathcal{C}, \mathcal{L}, \mathcal{K}, \mathcal{U})$ を出力する.
- 信頼できるホスト T の公開鍵で $\bar{K} = E_T(id||1||(K_{1,0}, K_{1,1})||2||..., ||n_y||(K_{n_y,0}, K_{n_y,1}))$ のように暗号化する.
- 自身の秘密入力 $x = (x_1, x_2, ..., x_{n_x})$ に対応するように選んだ L_{i,x_i} を L'_i とする.$(i = 1, 2, ..., n_x)$
- $id, \mathcal{C}, L'_1, ..., L'_{n_x}, \bar{K}, \mathcal{U}_z$ を次のホスト H に送る.

Step2　実行ホスト H は以下の操作を行う.

- id と $\bar{K}_{i,b}$, $(i = 1, 2, ..., n_y)$, $b \in \{0, 1\}$ を T に送る.

Step3　信頼できるホスト T は $\bar{K}$ を復号し, id と添え字 i を確認する. 実行ホスト H と, 以下の通信を行う $(i = 1, 2, ..., n_y)$

(1)　信頼できるホスト T はランダムに $\delta^{(i)}$ を選び, 実行ホスト H に送る.

(2)　実行ホスト H は, 自身の秘密情報 $y = y_1, y_2, ..., y_{n_y}$ に対応するようにランダムに $\alpha^{(i)}$ を選び, $\beta_c^{(i)} = g^{\alpha^{(i)}}$, $\beta_{c\oplus 1}^{(i)} = \delta^{(i)}/\beta_c^{(i)}$ を計算して $\beta_0^{(i)}$, $\beta_1^{(i)}$ を信頼できるホスト T に送る.

(3)　信頼できるホスト T は $\beta_0^{(i)}\beta_1^{(i)} = \delta^{(i)}$ を確認し, 正しければランダムに $r_0^{(i)}$, $r_1^{(i)}$ を選び $(e_0^{(i)}, f_0^{(i)}) = (g^{r_0^{(i)}}, K_{i,0}\beta_0^{(i)r_0^{(i)}})$, $(e_1^{(i)}, f_1^{(i)}) = (g^{r_1^{(i)}}, K_{i,1}\beta_1^{(i)r_1^{(i)}})$ を計算して $(e_0^{(i)}, f_0^{(i)}, e_1^{(i)}, f_1^{(i)})$ を実行ホスト H に送る.

(4)　実行ホスト H は $f_c^{(i)}/e_c^{(i)\alpha^{(i)}}$ を計算して $K_{i,c}$ を入手する. ただし $(i = 1, 2, ..., n_y)$, $c \in \{0, 1\}$

このプロトコルの安全性は 1-out-of-2 紛失通信と同じく CDH 仮定の下に保証されている.

4.3 k-out-of-n 紛失通信の改良

通信量の面で k-out-of-n 紛失通信は 1-out-of-2 紛失通信を n 回行うより効率がよい. しかし, k-out-of-n 紛失通信を [5] の secure function evaluation に適応するには問題が 2 つ挙げられる. 1 つは受信者によって, 受信するメッセージ数が決まってしまうことと, もう 1 つは, 受信者がメッセージ k 個をランダムに選んでしまう, つまり暗号回路入力ペア $(K_{i,0}, K_{i,1})$ $(i = 1,$

2, ..., n_y) の中から, 必ずしも各 1 つずつ入手するのではなく, 無作為にしか入手できない. 前者の問題は実行ホストが必要以上の暗号回路を入手してしまう恐れがあり, 後者は, 実行ホストの秘密情報に対応する暗号回路が入手できないといった問題がある.

上記の問題を解決するために 4,1 の Step1 の暗号回路入力の暗号化方法を変え, Step2 において信頼できるホスト T と実行ホスト H に k-out-of-n 紛失通信を適用する. ここで, 既存の k-out-of-n 紛失通信の Step1 で受信者が Q_{i_j} を選ぶところを, 送信者が選ぶことにより受信者のメッセージ数を固定することができた. 次に, 受信者が A_j を分ける事によりランダムな k をとる事を防ぐ. 以下にメッセージ数を通信量として, [5] の secure function evaluation に適応した既存の方式と改良方式の通信量等を比較する. ただし, 受信者は実行ホスト, 送信者は信頼できるホストにあたる.

表 1　通信量等の比較

	k-out-of-n(既存)	k-out-of-n(改良)
パス数	2	3
送信者→受信者	0	k
受信者→送信者	k	k
送信者→受信者	$1+k+n$	$2+k+n$
総数	$1+2k+n$	$2+3k+n$
受信メッセージ数	受信者が決める	送信者が決める
受信者によるメッセージの制御	できない	できる

4.4　提案プロトコルの詳細

まず最初に, 提案方式で用いる記号を以下のように定義する.

q	:	大きな素数;
p	:	$p=2q+1$ となるような大きな素数;
l	:	$l$bit;
$G=\langle g\rangle$	:	g で生成される巡回群;
g	:	位数 q となる $g\in\mathbf{Z}_p$ 上の原始元;
P	:	G の元;
H_1	:	$\{0,1\}^*\rightarrow G$ となる衝突困難なハッシュ関数;
H_2	:	$G\rightarrow\{0,1\}^l$ となる衝突困難なハッシュ関数;
$K_{i,b}$	:	$K_{i,b}\in\{0,1\}^l$, ($i=1, 2, ..., n_y$), $b\in\{0,1\}$;
s	:	$\mathbb{Z}_q^*$ の元;
t_b	:	$\mathbb{Z}_q^*$ の元 $b\in\{0,1\}$;
a_j	:	$\mathbb{Z}_q^*$ の元 ($j=1, 2, ..., n_y$);

ただし, (P, H_2, G) は公開し, このプロトコルは楕円曲線上で考える.

Step1　ホスト O は名前等記入された id を選び, $construct(C)$ を計算する.

- $construct(C)$ より ($\mathcal{C}$, $\mathcal{L}$, $\mathcal{K}$, $\mathcal{U}$) を出力する.
- 信頼できるホスト T の公開鍵で $\bar{K}=E_T(id||1||(K_{1,0}, K_{1,1})||2||..., ||n_y||(K_{n_y,0}, K_{n_y,1}))$ のように暗号化する.
- 自身の秘密入力 $x=x_1, x_2, ..., x_{n_x}$ に対応するように選んだ L_{i,x_i} を L'_i とする. ($i=1, 2, ..., n_x$)
- id, $\mathcal{C}$, $L'_1, ..., L'_{n_x}$, $\bar{K}$, $\mathcal{U}_z$ を次のホスト H に送る.

Step2　実行ホスト H は以下の操作を行う.

- id と $\bar{K}$ を T に送る.

Step3　信頼できるホスト T は $\bar{K}$ を復号し, id と添え字 i を確認する. 実行ホスト H と以下の通信を行う ($i=1, 2, ..., n_y$)

（1） 信頼できるホスト T は i, ($i=1, 2, ..., n_y$) を選び $H_1(1)=Q_1, ..., H_1(n_y)=Q_{n_y}$ を計算する.

（2） 信頼できるホスト T は秘密にランダムな s を選び, $sQ_1=R_1, ..., sQ_{n_y}=R_{n_y}$ を計算し, $R_1, ..., R_{ny}$ を実行ホスト H に送る.

（3） 実行ホスト H は秘密情報 $y=y_1, y_2, ..., n_y$ に対応して, $I_0=\{i|y_i=0\}$, $I_1=\{i|y_i=1\}$ とする. ただし $i=1, 2, ..., n_y$

（4） 実行ホスト H は, $j\in I_0$ について $A_j=R_j+a_jP$, $j\in I_1$ について $B_j=R_j+a_jP$ とする. $\{A_j|j\in I_0\}$, $\{B_j|j\in I_1\}$ を信頼できるホスト T に送る. ただし, ランダムな a_j を選ぶ. また $j=1, 2, ..., n_y$ とする.

（5） 信頼できるホスト T は実行ホスト H から $j\in I_0$ について $A_j=R_j+a_jP$, $j\in I_1$ について $B_j=R_j+a_jP$ を受け取り, n 個ある事を確認して, ランダムな $t_0\neq t_1$ を選び $P_0=t_0P$, $P_1=t_1P$, $D_j=t_0A_j$, $E_j=t_1B_j$ を計算する.

次に $H_1(k)=Q_k$, ($k=1, 2, ..., n$) をとり st_0Q_k, st_1Q_k を計算し, $C_{k,0}=K_{k,0}\oplus H_2(st_0Q_k, k)$, $C_{k,1}=K_{k,1}\oplus H_2(st_1Q, k)$ を計算する.

P_0, P_1, D_j, E_j, $C_{k,0}$, $C_{k,1}$ を実行ホスト H に送る.

（6） 実行ホスト H は $D_j-a_jP_0$, $E_j-a_jP_1$ を計算して $K_{j,0}=C_{k,0}\oplus H_2(D_j-a_jP_0, j)$, $K_{j,1}=C_{k,1}\oplus H_2(D_j-a_jP_1, j)$ を入手. ただし ($j=1, 2, ..., n_y$) とする.

このプロトコルの安全性は, k-out-ofn 紛失通信と同じく CT-CDH 仮定 [9] の下に保証されている.

5. 評価方法

本章では, 前章で提案した k-out-of-n 紛失通信を用いた方法と既存とされる 1-out-of-2 紛失通信を用いた方法 (原始的なプロトコル) との計算コストを比較し, 評価を行う.　原始的, 提案方式では G の生成元を用いてプロトコルの計算, 通信を行っている. 1 つの生成元を 1 メッセージとして考え, 信頼できるホスト T と実行ホスト H 間の通信量をメッセージ数で測り, 原始的, 提案方式を比較する.

また, 計算量コストを考えるにあたり, 有限体上のべき乗算及

び, 逆演算における回数で原始的, 提案方式の計算量を比較する.

表 2　通信量の比較

	原始的な方式	提案方式
パス数	3	3
$T \rightarrow H$	n	n
$H \rightarrow T$	$2n$	n
$T \rightarrow H$	$4n$	$2+3n$
総数	$7n$	$5n+2$

表 3　計算量の比較

	原始的な方式	提案方式
計算量 (乗算)	$\frac{21n}{2}(\log p)^3 + 2n(\log p)^2$	$(9n+3)(\log p)^3$

提案方式 (k-out-of-n 紛失通信) と原始的方式 (1-out-of-2 紛失通信) についてラウンド数は変わらないが, 提案方式のメッセージ数は原始的方式に比べ $2n$ 近くのメッセージ数が削減された. また, 提案方式と既存方式の計算量を比較して, 提案方式の計算量が原始的方式に比べ n 近く減っている.

6. ま と め

本稿では k-out-of-n 紛失通信を用いてモバイルエージェントとその実行ホストにおいて安全性を備えたプロトコルを提案した. 提案方式では [8] より, モバイルエージェント, 信頼できるホスト, 実行ホストの安全性を満たしている. また, 1-out-of-2 紛失通信を用いた場合と比較してメッセージ数, 計算量が削減できている. さらにモバイルモバイルエージェントに適応できるように, k-out-of-n 紛失通信を送信者側がメッセージ数を決める事ができるように改良した.

文　献

[1] Tomas Sander and Christian F. Tschudin,"Protecting Mobile Agents Against Malicious Host" in Mobile Agents and Security (G.Vigna,ed.), LNCS 1419, 1998.

[2] M. Naor and B. Pinkas "Oblivious Transfer and Polynomial Evaluation" Proc. of the 31st Symp. on Theory of Computer Science pp. 245-254, 1999.

[3] D. Boneh, "The decision Diffie-Hellman problem." in ANTS-III vol. 1423 of LNCS, pp. 48-63, Springer-Verlag, 1998.

[4] Christian Cachin, Jan Camenisch, Joe Kilian, and Joy Müller,"One-Round Secure Computation and Secure Autonomous Mobile Agents" in International Colloquium on Automata, Language and Programming, Lecture Notes in Computer Science, vol. 1853, Springer, July 2000, pp. 512-523.

[5] Joy Algesheimer, Christian Cachin, Jan Camenisch and Gunter Karjoth,"Cryptographic Security for Mobile Code", Proceedings of IEEE Security and Privacy 2001.

[6] A. C. Yao, "How to generate and exchange secrets" in Proc.27th FOCS, pp162-167, 1986.

[7] Donald Beaver, Silvio Micali and Phillip Rogaway "The round complexity of secure protocols", Proc of the 22nd annual ACM symposium on Theory of computing(1990) pp 503 - 513.

[8] Cheng-Kang Chu and Wen-Guey Tzen "Efficient k-out-of-n Oblivious Transfer Schemes with Adaptive and Non Adaptive Queries ", Cryptology ePrint Archive: Report 2004/041

[9] Alexandra Boldyreva "Threshold Signatures, Multisignatures and Blind Signatures Based on the Gap-Diffie-Hellman-Group Signature Scheme", Public Key Cryptography 2003: 31-46

[10] Rabin, M. O "How to exchange secrets by oblivious transfer", Tech memo, TR-81 Aiken Comp. Lab., Harvard University (1981).