FIPA準拠エージェントプラットフォームへのセキュリティ機能の実装とDSRC環境下での評価

清本晋作† 田中俊昭† 中尾康二†

エージェント要素技術の標準化団体であるFIPAでは，ACL（Agent Communication Language）などの，エージェント通信におけるインターオペラビリティを確保するための様々な仕様が策定されている．しかしながら，エージェントサービスの安全性を保証するために必要なセキュリティに関しては，その要件を検討しているにすぎず，具体的な方式はほとんど議論されていない．そこで本稿では，FIPAにおけるセキュリティ要件およびFIPAのシステムアーキテクチャに適用するセキュリティ機能を設計し，FIPA準拠のプラットフォーム上に実装を行った．また，本実装方式の実現性を立証するため，エージェントの適用分野の1つとして注目を集めているITSサービスの実証実験環境下において評価を行った．その評価結果から，提案方式がセキュリティ要件を満足し，実用的な実行時間で処理可能であること，実サービスへの適用が可能であることを示した．

Design and Implementation of a Secure Agent-platform Based on FIPA and Its Evaluation under DSRC Environments

Shinsaku Kiyomoto,† Toshiaki Tanaka† and Koji Nakao†

Recently, mobile agent technologies have been actively discussed and researched. FIPA (Foundation for Intelligent Physical Agents) is one of organizations, which standardize agent technologies in general. Although several standardizations have been already carried out in FIPA, their security functions have not been satisfactorily designed. In this paper, we design security functions of FIPA architecture, implement them over "FIPA-OS", and finally evaluate their feasibility on DSRC environments. We believe that our proposed security functions and their implementation method can be a valuable security framework for FIPA activities and apply to many kinds of agent services discussed in ITS.

1. はじめに

ネットワークの爆発的な普及にともなって，ネットワーク上で自律分散的に処理を行うソフトウェア技術が注目されてきている．そのような技術の1つとしてエージェント技術があげられる．エージェントは，ネットワークコンピューティングの大きな特性である高柔軟性，高信頼性，耐故障性を実現するミドルウェアの構築を可能とする要素技術であり，これまで多くの研究がなされている．エージェント技術の研究の進展にともない，異なる業者が提供するエージェントや，既存のエージェントを自由に組み合わせて使うことのできるインターオペラビリティの確立が求められてきた．そこで，インターオペラビリティを確立する要素技術の標準化団体としてFIPA（Foundation for Intelligent Physical Agents）[1]が設立された．FIPAでは，エージェント間の通信用言語であるACL（Agent Communication Language）などのエージェントの入出力インタフェースおよびエージェント間の通信や管理を実現するためにエージェントが持つべき機能を規定している．しかしながら，エージェントサービスの安全性を保証するためのセキュリティ機能については，十分な検討がなされていない．オープンなネットワークでの相互接続を考えた場合，認証や暗号化などのセキュリティは必要不可欠な機能である．そこで本稿では，必要なセキュリティ機能を設計し，実装を行う．さらにその実現性を立証するために，エージェント技術の適用分野の1つとして注目を集めている[5)~8)]ITSサービスを想定し，実証実験環境上で評価を行う．

2. FIPAエージェントプラットフォームにおけるセキュリティ要件

FIPAのアーキテクチャは，共通機能を提供するエー

† KDDI研究所
KDDI R&D Laboratories Inc.

ジェントプラットフォームと，その上で動作するプログラムであるエージェントから構成される．エージェントプラットフォームは，プラットフォーム上のエージェントすべてのメッセージ転送を行う ACC（Agent Communication Channel）と，エージェントの登録，ライフサイクルの管理などを行う AMS（Agent Management System），いわゆるイエローページの役割を果たし，エージェントサービスの検索に使用される DF（Directory Facilitator）の 3 つの機能によって構成される．エージェントは，ACL（Agent Communication Language）と呼ばれるメッセージフォーマットに従ってメッセージを作成し，Envelope という宛名書きを付加した Letter に入れてメッセージを ACC に渡す．ACC は，ACL のパージングを行い，Envelope を解釈して，送信先プラットフォームの ACC へとメッセージを転送する．そして送信先 ACC は，送信先エージェントに対してメッセージを受け渡す．

オープンなネットワークでのエージェントプラットフォーム間の相互接続を考えた場合，認証や暗号化などのセキュリティは必要不可欠な機能である．FIPA の FIPA Security Work Group（以下 Security WG）においても，セキュリティ機能の要求仕様の検討が進められており，2001 年 7 月に発行した Request for Information（RFI）[2] には，エージェントシステムに求められるセキュリティ要件として，アクセス制御，メッセージ転送時の完全性および秘匿性，エージェントの認証および管理などをあげているが，具体的なセキュリティ機能の検討はなされていない．エージェントシステムに対するセキュリティ機能は，エージェントのセキュリティ機能，エージェントプラットフォームのセキュリティ機能，の 2 つに大別される．FIPA のアーキテクチャモデルは，各プラットフォームがその上で動作するエージェントにとって信頼できるものであるという前提で構築されており，また，各プラットフォームは不正なエージェントを排除するために自プラットフォーム上のエージェントを管理する方式をとっている．一方，他のプラットフォームとの通信を行う場合には，その通信路上で盗聴や改ざんなどの攻撃が想定される．したがって，エージェントプラットフォームには，自プラットフォーム内のエージェントの認証とともに，他プラットフォームと安全な通信路を構築する，というセキュリティ機能が求められる．また，アクセス制御については，アクセスされたくないリソースを管理するエージェントサービスに対してのメッセージ受信を拒否することが必要である．以上のことからエージェントプラットフォームには，以下のセキュリティ機能が求められる．

- メッセージ秘匿およびメッセージ認証機能
  通信路でのメッセージ盗聴・改ざんを防止し，安全な通信路を確立する．
- アクセス制御（フィルタリング）機能
  特定のエージェントを保護するためのメッセージのフィルタリングを行う．
- エージェント認証機能
  自プラットフォーム上にエージェントを生起する際に，適切な認証を行うことによって不正なエージェントを排除する．

本稿では以上の機能のうち，特にメッセージ送受信に関するセキュリティ機能である，メッセージ秘匿およびメッセージ認証機能，アクセス制御機能の 2 つの機能について設計・実装を行い評価する．エージェント認証機能については，ID, パスワードベースの簡易的なものを実装したが，本稿においては議論を行わない．

3. セキュリティ機能の設計と実装

本章では，各セキュリティ機能を具備するための詳細な設計を進める．既存のエージェントプラットフォームを拡張することによりセキュリティ機能の実装を行うが，実装後も FIPA の仕様に準拠したエージェントがこのプラットフォーム上で動作可能である必要がある．すなわち，エージェント間で送受信される ACL のフォーマットを変更することなくセキュリティ機能を実装しなければならない．また，各セキュリティ機能は，エージェントサービスに対して適用可能な処理時間で動作する必要がある．また処理の効率化を目的として，重要なメッセージに対してのみセキュリティ機能を提供するといったように，エージェントが送信するメッセージに応じてセキュリティ機能を取捨選択できることが望ましい．さらに，エージェントサービスは，非同期通信，オフライン処理といった特性から，モバイル環境などの通信の信頼性が低い環境への適用も想定されている．したがって，設計は以下の方針にしたがって行う．

方針 1 各機能がセキュリティ要件を満足すること
方針 2 FIPA 準拠のエージェントが動作可能なプラットフォームであること
方針 3 各機能の実装によって，処理負荷，処理時間が著しく増加しないこと
方針 4 セキュリティ機能をエージェントが取捨選択可能であること
方針 5 プラットフォーム間の通信の信頼性が低い

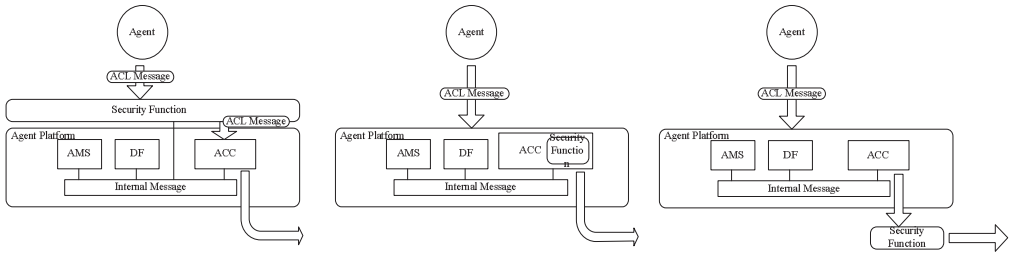

図 1 セキュリティ機能実装方式
Fig. 1 Implementation methods of security architecture.

環境下においても実行可能であること

実装は，FIPA 準拠のプラットフォーム・ソフトウェアの 1 つとして最も一般的である FIPA-OS [3] に対し行う．

3.1 実装方式の検討

FIPA において，エージェント間のメッセージ転送は非同期通信であり，プラットフォーム機能の 1 つであり，ACC を介して実現される．エージェントは，ACL にメッセージの宛先などを記述する Envelope を付けて ACC に渡し，ACC は，Envelope にしたがってメッセージの転送を行う．FIPA-OS において ACC は，主に ACL のパージングや，メッセージのバッファリングの機能を提供する Service Stack と，メッセージを送受信する MTP（Message Transport Protocol）の 2 つから構成される．エージェントから送信されたメッセージは，Service Stack によりバッファリングされ，変換処理の後 MTP に渡され順次送信される．セキュリティ機能の実装方式としては，以下の 3 方式が考えられる（**図 1**）．

(a) 既存プラットフォームの上位に実装する方式
(b) 既存プラットフォーム内の ACC にセキュリティ機能を拡張する方式
(c) 既存プラットフォームの下位の通信機能に組み込む方式

(a) の方式においては既存プラットフォームに変更を加えることなく実装可能である．FIPA の Security WG において，ZHANG らは，FIPA-OS に対する (a) のような実装方式[4] とその評価結果を発表したが，相互認証処理に 4377 msec, 10 KByte のメッセージ送受信におけるセキュリティ処理に 9494 msec の処理時間を要している．これは，メッセージ送受信時に，セキュリティ機能がエージェントからの ACL メッセージをいったん受取り，パージングをしてセキュリティ処理を行い，その後，再度 ACL メッセージを作成し送信しているためと考えられる．実サービスへの適用を考慮した場合，方式 (a) は処理速度の面で問題がある．

(b) の方式は，メッセージの送受信を行う ACC を機能拡張するため，オーバヘッドが少なく (a) と比べてより高速な処理が期待できる．しかしながら，ACC 内への実装に際しては，FIPA の仕様との整合性を保ちつつ，既存モジュールを拡張しなければならないという課題がある．(c) の方式としては，既存の SSL [13] や IPSec [14),15)] などの暗号通信路構築技術に対し，機能拡張を行うことで実現する方式が考えられる．しかしながら，認証ステータスの管理はエージェントプラットフォーム単位で行うことが必要である．したがって，アプリケーション層で管理する必要があり，SSL などのセション層での管理は困難である．また，方針 4 にあげているエージェントが送信メッセージ単位で暗号化，メッセージ認証のセキュリティ機能を取捨選択できる機能を実現する場合，方式 (b) による実装のほうがより効率的に実装できると考えられる．さらに，メッセージのフィルタリングを行う機能は，アプリケーション層以上に実装するほうが効率的である．そこで本稿では，より高速な処理が期待でき，エージェントプラットフォームとして必要なセキュリティ機能が実現可能な方式 (b) について，上記課題を解決する実装方式の検討を行うこととする．

3.2 メッセージ秘匿およびメッセージ認証機能

メッセージ秘匿機能を実現する手段としては，公開鍵暗号アルゴリズムを使用する方式と共通鍵暗号アルゴリズムを使用する方式の 2 通りが考えられるが，公開鍵暗号アルゴリズムは共通鍵暗号アルゴリズムと比べて，100〜1000 倍程度処理時間を必要とするため，すべてのメッセージを公開鍵暗号アルゴリズムで暗号化することは効率的ではない．そこで提案方式では，共通鍵暗号アルゴリズムを使用してメッセージ秘匿機能を実現する．メッセージ認証機能を実現する手段も同様に，公開鍵で検証する署名を付加する方式と，共有した秘密鍵で改ざんの検証を行うメッセージ認証子（MAC: Message Authentication Code）を使用する方式が考えられるが，より高速な処理が可能で

ある後者の方式を用いる．共通鍵を使用した暗号化およびメッセージ認証を実現するためには，両エンティティ間で安全に鍵を共有する必要がある．メッセージの秘匿および認証は，プラットフォーム間相互認証機能によって作成された共通鍵を使用して行う．“暗号化”，“MAC 値生成” の順序は，まずメッセージの暗号化を行い，次いで暗号化されたメッセージに対して MAC 値の計算を行うという手順とする．FIPA-OS では，Service Stack 内における個々のメッセージサービスは，決められたインタフェースを持っており，暗号化されたデータを解釈するためには，大規模な拡張が必要となる．また，宛先などのデータが格納される Envelope も暗号化およびメッセージ認証の対象データとする必要がある．したがって，メッセージ秘匿機能およびメッセージ認証機能は Envelope とメッセージを 1 つのデータとして扱うことのできる MTP に対して実装を行う．また，プラットフォーム間相互認証機能は，MTP を拡張して実装する．それは，MTP より上位のモジュールに対して実装した場合，ACL に従って，相互認証・鍵共有のためのメッセージを送受信しなければならず，データサイズは冗長となり処理時間も増加するためである．メッセージ秘匿・認証に使用する共通鍵は，プラットフォーム間相互認証機能によって，認証ステータス情報とともに格納される．プラットフォーム間相互認証機能は，MTP 本体とは独立の通信インタフェースを持つことで，エージェントメッセージの送受信とは独立して認証プロトコルを実行できるように実装する．したがって，鍵更新と平行してエージェントメッセージを送受信することを可能となる．さらに，エージェントがセキュリティ機能を選択可能なように，Envelope に Code，Mac というの 2 つのフィールドを追加する．このことにより，1 メッセージごとにエージェント自身が暗号化あり/なし，メッセージ認証あり/なしの選択が可能となる．適切にメッセージが復号できなかった場合や，メッセージ認証に失敗した場合は，MTP が通信エラーとして扱う．送信時には，エージェントが Envelope の各フィールドに値を格納し ACC に渡す．ACC の MTP では，各フィールドから値を取り出し，暗号化などの要求された処理を行った後，Code，Mac，$Kseq$ 値からなるセキュリティヘッダ（SH）を付加して送信する．受信時においては，MTP が SH から値を取り出し，セキュリティ処理を行った後，送り先のエージェントへと受け渡す．メッセージ認証の対象データは SH およびエージェントメッセージとし，受信時にメッセージ認証子（MAC 値）の確認に失敗した場合にはメッセージを破棄する．

3.3 プラットフォーム間相互認証および鍵共有機能

通信先プラットフォームを認証し，メッセージ秘匿およびメッセージ認証機能で使用する共通鍵を交換するために，プラットフォーム間での相互認証機能および鍵共有機能を設計する．

3.3.1 プラットフォーム間相互認証および鍵共有プロトコル

プラットフォーム間相互認証および鍵共有プロトコル（以下，相互認証プロトコル）は，公開鍵暗号方式を基とした相互認証方式とする．認証を開始する側を Initiator（i），受信側を Responder（r）としたときの認証プロトコルを以下に示す．c は，認証局のような互いのエンティティが信頼した第 3 者機関である．

$$i \rightarrow r : e_i, D(h(e_i|URL_i), d_c)$$
$$i \leftarrow r : e_r, D(h(e_r|URL_r), d_c), E(R, e_i)$$
$$i \rightarrow r : E(R', e_r), f(Kseq|R, K_{ir})$$
$$i \leftarrow r : f(Kseq|R', K_{ir})$$
ただし $K_{ir} = h(R|R')$

e_x は，x の公開鍵，d_x は対応する x の秘密鍵を表す．URL_x は，x のプラットフォームアドレスを表す．K_{xy} は，x と y で共有された共通鍵を表す．各鍵には鍵シーケンス番号（$Kseq$）が一意に定義される．$E(a, b)$ は公開鍵 b による a の暗号化処理，$D(a, b)$ は秘密鍵 b による a の復号化処理を表す．$D(a, b)$ は，第 3 者機関から各プラットフォームに対して，事前に配布されているものとする．$f(x, y)$ は，x に対する共通鍵 y による暗号処理を表し，$h(X)$ は，X のハッシュ値をとることを表す．|はデータの結合を表す．また，R，R' はその場で生成される乱数を表す．$e_i, D(h(e_i|URL_i), d_c)$ や，$e_r, D(h(e_r|URL_r), d_c)$ の送信は，相手のプラットフォームに対して事前登録しておくことにより省略可能とし，その情報は当該プラットフォームの URL と対応付けて管理される．本プロトコルにおいては，送られてくる暗号化された R，R' を復号化し，正しい値であることを確認することで認証を行う．また，作成された共通鍵 K_{ir} は，メッセージ秘匿機能およびメッセージ認証機能の鍵として使用される．

3.3.2 鍵更新プロトコル

プラットフォームごとに共有される共通鍵は，一定間隔ごとに以下に示す鍵更新プロトコルを使用して更新される．また，このプロトコルを使用することにより，相互認証処理と比べてより短い時間で鍵の更新を

行うことができる．

$i \rightarrow r : R$
$i \leftarrow r : R', f(R|R', K'_{ir})$
$i \rightarrow r : Kseq, f(Kseq, K'_{ir})$
ただし $K'_{ir} = h(K_{ir}|R|R')$

K'_{ir} が新たに更新された共通鍵である．鍵更新処理は，エージェントメッセージの送受信とは独立に行われ，Initiator は $Kseq$ を送信した時点で鍵を更新し，以後の送信メッセージについては新しい鍵を使用して処理を行う．Responder は，送信メッセージの SH（3.2 節で記述）の $Kseq$ を参照し，シーケンス番号が Initiator より通知されたシーケンス番号と一致した時点で鍵の更新を行う．この鍵更新プロトコルは，相互認証プロトコルによって鍵が事前に共有されていた場合の簡易的な認証プロトコルとしても利用可能である．このプロトコルによる認証処理は，相互認証プロトコルよりも高速な処理が期待できる．

3.3.3 認証ステータスの管理

相互認証プロトコルおよび鍵更新プロトコルは，エージェントのメッセージ送受信とは独立したプロトコルとし，相互認証・鍵更新モジュールで行う．エージェントメッセージの送受信は非同期通信であり，エージェントサービスはモバイル環境など常時接続でない通信環境への適用も検討されている．したがって，上記のようなエージェント通信の特性を考慮した認証ステータスの管理を行う．あるプラットフォームと初めて通信を行う場合，認証ステータスは初期モードから認証モードへと遷移して，認証終了後稼動モードへと遷移する．そして，メッセージ送信失敗などのエラーとなった場合やその他のエラー処理によって，初期モードに状態が回帰する機構を設ける．また，一定時間が経過した場合には，稼動モードから認証モードへと遷移して，再度認証処理を行う．状態が初期モードおよび認証モードであるプラットフォームに対してのメッセージ送信は行わない．初期モードへの回帰条件や，認証を行う時間間隔についてはプラットフォーム管理者のセキュリティポリシーに応じてカスタマイズ可能とする．たとえば，頻繁に通信断が起こるネットワーク環境においては，通信断によってメッセージ送信失敗となった場合でも，認証状態を初期状態に回帰させず，稼働状態のままロックする．認証ステータスや共通鍵および鍵シーケンス番号は，プラットフォームごとに一括管理する．なお，認証失敗などによって送信できなかったエージェントメッセージは，MTP により通信エラーとして扱われる．

3.4 アクセス制御（フィルタリング）機能

プラットフォームでのアクセス制御を実現するため，エージェント名およびプラットフォームアドレスによるフィルタリング機能を設計する．フィルタリング対象データは Envelope に含まれている送信先/送信元のエージェント名およびプラットフォーム名である．FIPA-OS において，MTP など Service Stack より下位のモジュールに本機能を実装した場合には，バイト列から Envelope データを構築しなければならない．一方 Service Stack 中，もしくはその上位に実装した場合には Envelope をオブジェクトとして取得できるため，処理が容易となる．フィルタリング機能自体はメッセージデータに変更を加えることがないこと，そして Service Stack は，どこにでも同一のインタフェースを持った任意のサービスを追加できること，以上 2 点を考慮し，Service Stack の 1 サービスとして実装を行う．具体的には，Service Stack のうち最も下位である CommMultiplexService と ParserService との間に FilterService を追加する．また，Service Stack は個々のメッセージ送受信に対して適宜実行されるため，フィルタリングルールはエージェントプラットフォームが稼動中においても動的に変更することが可能となる．FilterService が参照するフィルタリングルールは，以下のように記述する．

[IN/OUT] from:[送信元アドレス]
to:[送信先アドレス] [accept/deny]

アドレスは，“エージェント名@プラットフォームアドレス” のように記述し，“all” ですべてのエージェントもしくは，すべてのプラットフォームを記述できる．フィルタリングルールを読み込んだ際には，まず IN，OUT に分けて新たなリストを生成する．さらに，Java 言語の LinkedList クラスを利用して，送信元エージェント名，送信元プラットフォーム名，送信先エージェント名，送信先プラットフォーム名，コマンド名（accept/deny）に分割した Map を作成する．フィルタリングの際には，メッセージの送受信アドレスも同様にして，LinkedList に変換し照合を行う．フィルタリングによって受信もしくは送信拒否された場合，当該メッセージは破棄される．そして，送信メッセージの種別（Communicative Act）によって，送信元のエージェントに対して FIPA で定義されたメッセージ受信拒否を意味する Reject Proposal を返送するかを決定する．このことにより，送信元のエージェントが，レスポンス待ち状態のままでいることを防止する．フィルタリングの対象となる Communicative Act お

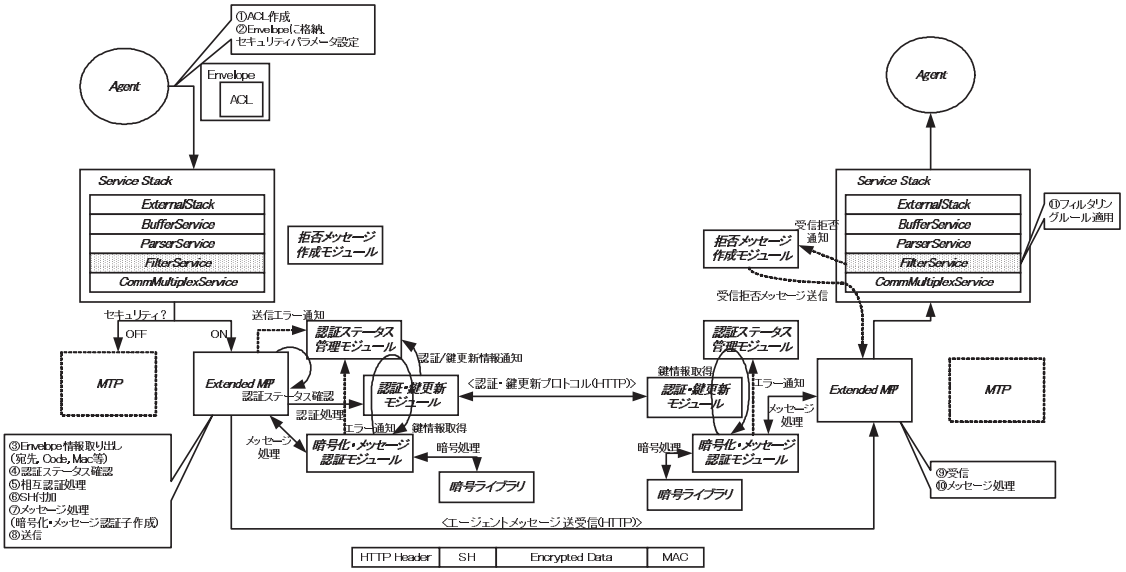

図 2 エージェントメッセージ送受信処理の流れ
Fig. 2 Transportation flow of agent messages.

よびフィルタリング時の動作については，設定により変更可能とする．

3.5 実装結果

実装結果を以下に示す．既存 FIPA-OS に対して追加したプログラムの実装サイズは，約 90 KByte である．

プログラムのほとんどは，FIPA-OS と同様に Java 言語により作成したが，暗号アルゴリズムについては，処理速度の観点から C 言語による実装とし，Java からの呼び出しが可能なようにインタフェースを作成した．公開鍵暗号アルゴリズムは RSA [9]，共通鍵暗号アルゴリズムは AES [10]，メッセージ認証アルゴリズムは HmacSHA-1 [11),12)] を使用した．エージェントプラットフォームは，セキュリティ機能を実装した MTP を使用するかどうかを選択可能とした．**図 2** にエージェントがメッセージを送受信する際の処理手順を示す．実線が正常系の処理を表し，破線がエラー処理を表す．まずエージェントは，送信したい ACL を Envelope に格納し，Messsage Object とする．Envelope には，Code，Mac のセキュリティパラメータの設定を行う．エージェントは，Service Stack を利用して，Message Object を MTP に受け渡し，送信を依頼する．MTP では，まず Envelope から宛先，Code，Mac など必要なパラメータを呼び出す．その後，認証ステータス管理モジュールに対してステータスの確認を行い，宛先が認証されていないプラットフォームであった場合には，認証・鍵更新モジュールに相互認証処理を依頼する．次に Envelope と Message 本体を String 列に変換し，Envelope 情報を基に SH を生成し付加する．SH を付加されたメッセージは，暗号化・メッセージ認証モジュールに渡され，SH に従って暗号化，メッセージ認証子作成の処理を行う．その後，MTP はヘッダ情報を付加し，送信先プラットフォームに送信する．受信側では，逆の手順で処理が行われた後，Service Stack 経由でエージェントにメッセージが送られる．このとき Filter Service では，フィルタリングルールに基づいてアクセス制御が行われる．受信を拒否した場合，特定のメッセージ種別に対しては，拒否メッセージ作成モジュールが Reject Proposal メッセージを作成し返信する．

4. 評価

セキュリティ機能を実装したエージェントプラットフォームについて，実サービスへの適用性を評価するため，エージェントサービスの 1 つの例として ITS サービスへの適用を想定し，DSRC [16)~21)] を利用した ITS サービス実証実験環境にて評価を行う．DSRC 基地局は，離散的に配置されることも想定されるため，サービスの利用中に，ネットワークの接続/切断が頻繁に発生する可能性がある．そこで，車載端末上と，サービスを提供するサーバ上に各々エージェントプラットフォームを構築する．そして，プラットフォームがネットワークレベルの接続/切断を吸収し，さらに通信が切断したとしても，その上で動作するエージェントが，サーバ側で自律的に処理を継続することで，サー

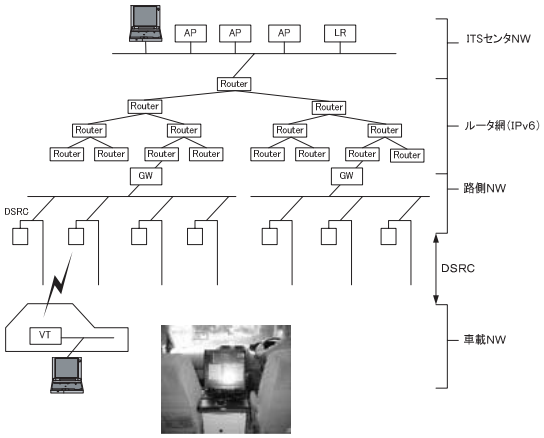

図 3 実証実験ネットワーク
Fig. 3 Network configuration of a DSRC test course.

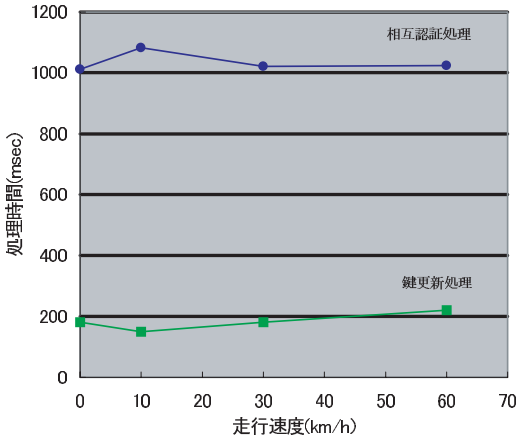

図 4 相互認証・鍵更新処理の処理時間
Fig. 4 Transaction time of mutual authentication and refreshing key.

ビスの連続性を保証するというアプローチが考えられる．したがって本稿では，車載端末 PC と，DSRC 通信網と接続されているサービス提供 PC を想定し，以下の実験環境を構築して評価試験を実施した．

4.1 実証実験環境

実験環境は，道路沿いに設置された 7 本の DSRC 基地局と，ゲートウェイサーバから構成される路側ネットワーク（路側 NW），階層的に構成された IPv6 ルータ網，車載器の位置情報を管理する LR（Location Resister）や様々なアプリケーションサーバ（AP）からなる ITS センタ NW，そして車載端末（VT）と車載 NW から構成される（**図 3**）．測定においては，車載 NW と，ITS センタ NW それぞれに CPU：Pentium3-800 MHz，メモリ：512 MByte のノート PC を接続して行った．

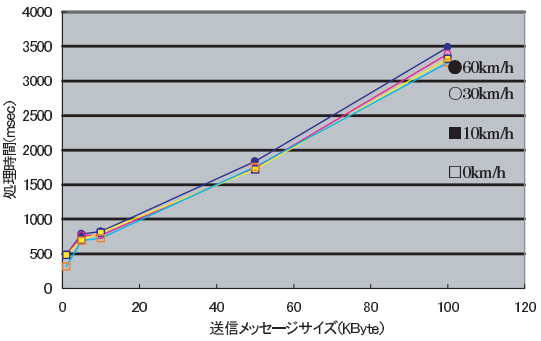

図 5 メッセージ転送処理時間
Fig. 5 Transportation time of messages using security functions.

4.2 評価結果

まず，相互認証機能および鍵更新機能の測定結果を**図 4** に示す．Inisiator 側において，相互認証プロトコル，鍵更新プロトコル，それぞれの処理開始から終了までの時間を計測した．各 20 回計測し，その平均値をプロットした．

時速 60 km までの走行速度では，速度の増加にともなって処理時間が大きく増加することはない．相互認証の処理時間の平均は 1027 msec に対して，鍵更新の処理時間の平均は 181 msec であり，約 5 分の 1 の処理時間で実現できている．同じ PC を使用し，クロスケーブルで PC 間を接続して行った測定結果との比較から，このうち 100 msec 前後がネットワークによる処理遅延であると考えられる．

次に，セキュリティ機能を使用した場合のメッセージ転送時間の測定結果を**図 5** に示す．メッセージサイズを変化させながら，送信側のメッセージ転送開始時刻，受信側のメッセージ受信終了時刻を計測し，その差分からメッセージ転送処理時間を求めた．測定に使用した 2 台の PC は，NTP（Network Time Protocol）を使用して時刻同期した．本測定結果は，センタ PC から車載 PC へのメッセージ転送，車載 PC からセンタ PC へのメッセージ転送をそれぞれ 20 回測定し，合計 40 回の平均値をプロットしたものである．

メッセージサイズ 5 KByte 以上の領域においてはほぼ線形に処理時間が増加する．車の速度増加による影響は若干のスループット低下が見られる程度であり，非常に軽微である．別にセキュリティ機能を使用しない場合の処理時間についても測定を行ったが，メッセージ秘匿およびメッセージ認証機能を利用した場合の処理時間との処理時間の差は，100 KByte 程度のメッセージサイズまででは変化はなく，ほぼ 150 msec 前後であった．

表 1 フィルタリング処理時間
Table 1 Transaction time of the filtering function.

ルール数	処理時間 (msec)
1	105
100	103
500	105
1000	108
2000	128
5000	188
10000	316
20000	488
30000	849

最後にフィルタリング機能の評価結果を**表 1** に示す．結果は，40 回計測した計測値の平均値である．最後でテストメッセージが許可されるようフィルタリングルールを記述し，フィルタリングモジュールの前後で時刻を計測してフィルタリングにかかった時間を求めた．ルールを 2000 までは 100 msec 前後で処理時間の増加は見られず，実用上問題ないと考えられる．2000 以上のルール設定においては，ルールの増加に従ってほぼ線形に増加する．以上から非常に効率的な実装を行うことができているものと考えられる．

5. 考　　察

必要なセキュリティ要件は，各機能の実装によって満足されている．したがって方針 1 を満足する．プラットフォーム間相互認証・鍵更新処理は MTP の拡張機能として実装し，メッセージ秘匿およびメッセージ認証機能は，MTP のうち，最もネットワーク I/F 処理に近い部分に実装している．また，フィルタリング機能は，FIPA-OS の仕様上拡張可能な Service Stack に追加している．追加 Envelope 部を定義しない通常の ACL も送受信可能となっているため，FIPA-OS 上で動作可能なエージェントついては，改変を加えることなくそのまま動作させることが可能である．MTP については，セキュリティ機能付き MTP と通常の MTP との選択が可能となっており，オリジナルの FIPA-OS に対しては後者による通信を選択することで，通信可能である．以上より，本セキュリティ機能を具備したエージェントプラットフォームは，エージェントメッセージの送受信に関して FIPA および FIPA-OS に対する互換性を持っている．したがって方針 2 を満足する．100 KByte までのメッセージ転送に関わるセキュリティ機能の処理時間はメッセージサイズ，フィルタリングルールによらずほぼ一定で，暗号化およびメッセージ認証機能で最大約 150 msec，2000 程度のルールを設定したフィルタリング機能で最大約 100 msec の計 250 msec 程度である．実サービスへの適用例として ITS サービスを考えると，DSRC 基地局 1 台のスポット長を実験環境と同様の 30 m とし，速度が増加しても処理時間は大きく変わらないと仮定した場合，相互認証の場合は，時速約 110 km の速度まで，鍵更新の場合には時速 180 km 以上の速度まで，1 台の DSRC 基地局で処理を終えることが可能である．また，ある 1 台の DSRC 基地局を車が時速 60 km で通過するとした場合，鍵共有による相互認証処理の後，約 43 KByte のメッセージを送受信可能である．エージェントサービスは，本来リアルタイム性を要求しないサービスであり，またエージェントが送信するメッセージサイズは通常 10 KByte 以下であることから，提案方式は ITS サービスに適用可能な処理性能を有していると考えられる．さらに，同じ FIPA-OS に実装した ZHANG らの実装結果よりも処理速度の面で優れている．したがって方針 3 を満足する．また，暗号化あり/なし，メッセージ認証あり/なし，についてメッセージを送信するエージェント自身が設定可能であるため，暗号化やメッセージ認証を必要としないメッセージに関しては，その処理を省略できる．したがって方針 4 を満足する．最後に，認証ステータスの管理や，相互認証，鍵更新という 2 つのプロトコルの使い分けによって信頼性の低い通信環境下においても安全な通信路の利用が可能となっている．たとえば ITS サービスにおける利用例としては，サービス提供終了以前に，DSRC スポット外に車が移動したなどの理由によりメッセージ転送に失敗した場合には，認証ステータスをロックすることで稼動状態のまま保持し，サービス提供終了と判断した場合には，認証ステータスのロックを解除して初期モードに回帰させる，といった認証ステータスを利用した柔軟なサービス運用も可能である．以上のことから，提案方式は方針 5 を満足すると考えられる．方針 5 については，今後，さらにモバイル環境などのサービスに適用して，実用性を評価していく予定である．

6. おわりに

本稿では，求められるセキュリティ要件を満足するセキュリティ機能を具備した標準準拠のエージェントプラットフォームの設計および実装を行った．さらに，DSRC を用いた実証実験環境下において評価を行った．実証したセキュリティ機能は，設計要件を満足し，また ITS 環境での利用が検討されているエージェントサービスに対して，適用可能であることを示した．今後は，AMS，DF と連携したフィルタリングルール

の動的設定方式の検討やモバイル環境などより信頼性の低い通信環境での詳細な評価を行う予定である．

謝辞 本研究は，通信放送機構からの委託研究「走行支援システム実現のためのスマートゲートウェイ技術の研究開発」に基づいて行われた．

参 考 文 献

1) FIPA. http://FIPA.org/
2) FIPA Security WG: FIPA Security Work Group Request For Information (RFI) (2001).
3) Nortel Networks: FIPA-OS. http://www.nortelnetwo- rks.com/products /announcements/FIPA/
4) Zhang, M. and Karmouch, A.: Adding Security Features to FIPA Agent Platforms, *Security WG at the 23rd FIPA Meeting* (2001).
5) 加藤直樹：ITS におけるエージェント技術，情報処理学会高度道路交通システム研究グループ研究報告，情報処理学会研究報告，Vol.99 (1999).
6) 池谷直紀，加藤直樹，大須賀昭彦：エージェント技術を利用したヒューマンナビゲーションシステム，情報処理学会 ITS 研究会研究報告，pp.97–104 (1999).
7) Hattori, M., Kase, N., Ohsuga, A. and Honiden, S.: Agent-Based Drivers' Information Assistance System, Computing Paradigms and Computational Intelligence, *New Generation Computing*, Vol.17, No.8, pp.359–367 (1999).
8) 西山 智，中尾康二，小花貞夫：ITS 通信におけるエージェントを用いたメッセージ移譲サービスの提案，情報処理学会 ITS 研究会研究報告，pp.7–12 (2000).
9) Rivest, R.L., Shamir, A. and Adleman, L.M.: A Method for Obtaining Digital Signatures and Public-key Cryptosystems, *Comm. ACM*, 21, pp.120–126 (1978).
10) Daemen, J. and Rijmen, V.: AES Proposal "Rijndael", *AES submission* (1998).
11) Krawczyk, H., Bellare, M. and Canetti, R.: "HMAC" Keyed-Hashing for Message Authentication, RFC 2104 (1997).
12) NIST: Secure Hash Standard, FIPS PUB 180-1 (1995).
13) Freier, A., Karlton, P. and Kocher, P.: The SSl Protocol Version 3.0. http://netscape.com/eng/ssl3/draft302.txt
14) Atkinson, R.: RFC 1825, RFC 1826, and RFC 1827, Internet Engineering Task Force (1995).
15) Metzger, P., et al.: RFC1828, and RFC1829, Internet Engineering Task Force (1995).
16) ARIB: Dedicated Short Range Communication for Transport Information and Control Systems ARIB STANDARD, ARIB STD-T75 (2001).
17) ITS Team: A Study on the Development of Dedicated Short Range Communication System for ITS, ETRI report (1999).
18) Iwata, et al.: DSRC Communication System, *5th ITS World Congress* (1998).
19) prTR2/204/15: DSRC Application Layer, ISO TC204 WG15 Document (1999).
20) Hasegawa, Y., et al.: An Application Study of The ETC/DSRC Technologies, *7th ITS World Congress* (2000).
21) Tokuda, K., et al.: Activity on ROF-RVC Working Group of ITS Joint Research Group at YRP, *1st workshop on ITS Telecommunications* (2000).

(平成 15 年 4 月 1 日受付)
(平成 15 年 9 月 5 日採録)

清本 晋作（正会員）

2000 年筑波大学大学院工学研究科物質工学専攻前期博士課程修了．同年 KDD（株）入社．現在，(株) KDDI 研究所ネットワークセキュリティグループ研究員．暗号プロトコル，モバイルセキュリティの研究に従事．日本物理学会，電子情報通信学会各会員．

田中 俊昭（正会員）

1986 年大阪大学大学院工学研究科通信工学専攻前期博士課程修了．同年 KDD（株）入社．現在，(株) KDDI 研究所ネットワークセキュリティグループリーダー．暗号プロトコル，著作権保護，モバイルセキュリティ，次世代 IDS の研究に従事．電子情報通信学会会員．

中尾 康二（正会員）

1979 年早稲田大学教育学部数学科卒業．現在，(株) KDDI 研究所，KDDI (株) に勤務．KDDI (株) 情報セキュリティ室室長．早稲田大学，電気通信大学の非常勤講師を兼務．ネットワーク技術，セキュリティ技術の研究に従事．1987 年度情報処理学会研究賞受賞．電子情報通信学会会員．