コンピュータセキュリティ　18－1
(2002. 7. 18)

エージェントプラットフォーム「FIPA-OS」における セキュリティ機能の実装

清本　晋作†　田中　俊昭†　中尾　康二†

†KDDI 研究所 〒356-8502 埼玉県上福岡市大原 2-1-15

E-mail: †kiyomoto@kddilabs.jp

あらまし　ネットワークコンピューティングにおけるソフトウェア技術の一つとしてエージェント技術が上げられる。エージェント技術の標準化団体の 1 つである FIPA(Foundation for Intelligent Physical Agents)では、エージェントの入出力インターフェース及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に必要なセキュリティ要件の詳細検討、及び仕様は明確化されていない。オープンなネットワークでの相互接続を考えた場合、認証や暗号化等のセキュリティは必要不可欠な機能である。そこで我々は、FIPA のシステムアーキテクチャに準拠する形でのセキュリティ機能の設計を行い、FIPA 準拠のプラットフォーム・ソフトウェアのひとつである“FIPA-OS”上に実装及び評価を行った。その結果、メッセージ送受信に関わるセキュリティ機能の処理時間がメッセージ送受信時間全体にしめる割合は、約 12%であるなどが判明した。従って、我々は提案した方式はセキュリティ要件を満足し、かつ非常に効率的な実装を実現できたものといえる。

キーワード　エージェント, セキュリティ要件, FIPA, エージェントプラットフォーム

Implementation of security functions for “FIPA-OS” Agent Platform

Shinsaku KIYOMOTO† Toshiaki TANAKA† and Koji NAKAO†

†KDDI R&D Laboratories Inc. 2-1-15 Ohara, Kamihukuoka-shi, Saitama, 356-8502 Japan

E-mail: †kiyomoto@kddilabs.jp

Abstract 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 standardized documents have been already produced in many areas of agent technologies in FIPA, their security functions have not been satisfactorily discussed. In this paper, we design security functions of FIPA architecture, implement to “FIPA-OS”, and finally evaluate their feasibility. We believe that our proposed security functions and their implementation method can be a valuable security framework for FIPA activities.

Keyword Agent, Security Requirement, FIPA, Agent Platform

1. はじめに

ネットワークの爆発的な普及に伴って、ネットワークコンピューティングにおけるソフトウェア技術が注目されてきている。そうしたソフトウェア技術の一つとしてエージェント技術が上げられる。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が進むにつれて、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、エージェントの入出力インターフェース及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に必要なセキュリティ技術に関する仕様は明確化されていない（2002 年 5 月現在）。オープンなネットワークでの相互接続を考えた場合、認証や暗号化等のセキュリティは必要不可欠な機能である。そこで我々は、FIPA におけるセキュリティ要件及び 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の実装方法を検討し、FIPA 準拠のプラットフォーム・ソフトウェアのひとつである“FIPA-OS”[2]上に実装を行った。また、各機能について処理時間を計測し、実装手法についての評価を行った。

2. セキュリティ要件

FIPA の FIPA Security Work Group(以下 Security WG)では、セキュリティ機能の要求仕様の検討を進めている。Security WG が 2001 年 7 月に発行した Request for Information（RFI）[3]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、秘匿性、否認防止、エージェントの権限管理などを挙げているが、詳細なセキュリティ要件の検討はなされていない。

2.1. FIPA アーキテクチャ

FIPA のアーキテクチャは図 1 のようになっており、エンティティとしては、利用者のエージェント、共通機能を提供するエージェントプラットフォーム、サービス提供者のエージェント（エージェントサービス）の 3 つのエンティティに大別される。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 Cannnel)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。利用者の Agent や、エージェントに対してサービスを提供する Agent Service は、エージェントプラットフォーム上で動作するエージェントプログラムとして実装される。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。

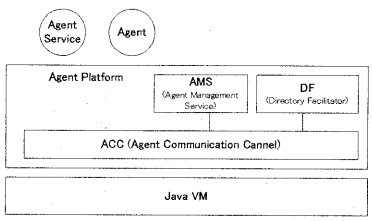

図 1　FIPA アーキテクチャ

2.2. セキュリティ要件の検討

FIPA においては、エージェントが外部に対して行う動作はメッセージの送受信であり、メッセージの送受信のすべてをエージェントプラットフォームが行うことから、エージェントプラットフォームは信頼できるエンティティであることが前提となっている。

“エージェントプラットフォーム”、“エージェント”、“エージェントサービス”の 3 つのエンティティごとにセキュリティ要件を検討し、必要なセキュリティ機能の抽出を行う。

2.2.1. 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

エージェントプラットフォームの機能は、エージェントから依頼されたメッセージの転送を行うメッセージ送受信機能と、AMS や DF などのメッセージの送受信を行うための情報を管理するマネージメント機能とに大別される。

エージェントと、エージェントプラットフォームの間に信頼関係が確立されている環境下においてメッセージ送受信に関して必要なセキュリティ機能は、外部者の攻撃を受ける可能性のあるプラットフォーム間で安全な通信路を構築する機能（完全性、秘匿性）である。また、アクセス制御という意味においては、不正な外部エージェントからのメッセージを受信しないこと、不正な外部エージェントへメッセージを送信しないことが重要であり、セキュリティポリシーに従ってメッセージを送受信可能となるようにフィルタリング機能が必要であると考えられる。否認防止については、メッセージ送信者に対するセキュリティ要件であることから、エージェントに対しての機能と考える。

一方、マネージメント機能に関するセキュリティとしては、不正なサービスやエージェントの動作を許さないために登録時に認証を行うことが重要であり、特にモバイルエージェントのように外部プラットフォームからの移動を考慮した場合には、その認証は必須の機能であると考えられる。

2.2.2. エージェントにおけるセキュリティ要件

エージェントに対しては、利用者の権限に基づいてサービスを利用できるようにアクセス制御を行わなければならない。従って、エージェント自身が権限を証明する機能を持ち、エージェントサービスは、その権限を確認する機能を有する必要がある。また、サービスを利用するのはエージェントであることから、サービス利用に対しての否認防止機能を有する必要があり、エージェントのなりすましを防止するためのメッセージ送信元保証もする必要がある。従って、エージェントに署名/署名検証機能を持たせるべきである。エージェントとエージェントプラットフォームの間に信頼関係が確立しているため、メッセージ転送における秘匿性を実現する機能については特にエージェントに対し

ては必要ないと考えられる。また、完全性の保証については、署名/署名検証機能によってメッセージの完全性を同時に確認することが可能である。

2.2.3. エージェントサービスにおけるセキュリティ要件

エージェントサービスには、エージェントがサービス利用権限を有するか判断する機能が必要である。またエージェント同様、否認防止及びメッセージ送信元保証をする必要がある。従ってエージェント同様署名/署名検証機能を実現する。

2.3. 実装するセキュリティ機能

以上の議論により、本稿では以下の機能の実装を行うこととした。

- プラットフォーム間で安全な通信路を確立するためのプラットフォーム間相互認証機能、メッセージ秘匿機能、メッセージ認証機能
- プラットフォーム上で不正メッセージをフィルタリングするメッセージフィルタリング機能
- エージェント及びエージェントサービスに対して提供する署名/署名検証機能
- エージェントの権限証明機能及び権限確認機能

マネージメント機能に関するセキュリティ機能は、本稿で実装する対象としたFIPA-OSがモバイルエージェント機能を実現していないため、今回は議論の対象外とした。

3. セキュリティ機能の設計

第2章では、FIPAにおけるセキュリティ要件の検討を行った。本章では、各セキュリティ機能を満足するための詳細な設計を進める。設計は、以下の方針に従って行う。

1. FIPAのフレームワークを可能な限り変更せずに実装可能であること。
2. 各機能がセキュリティ要件を満足するように実装できること
3. 各機能の実装によって、処理負荷、処理時間等が著しく増加しないこと

3.1. プラットフォーム・セキュリティ機能の設計

FIPA-OSにおいてエージェント間のメッセージ転送は、プラットフォーム機能のひとつである ACC を介して実現される。ACCの構成を図2に示す。

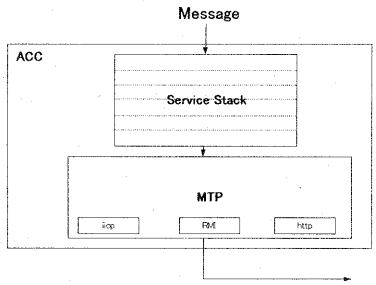

図2　ACCの構成

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、メッセージのEnvelopeに従ってメッセージを送受信するMTP (Message Transport Protocol)から構成される。ACC内でのメッセージ処理の流れを図3に示す。

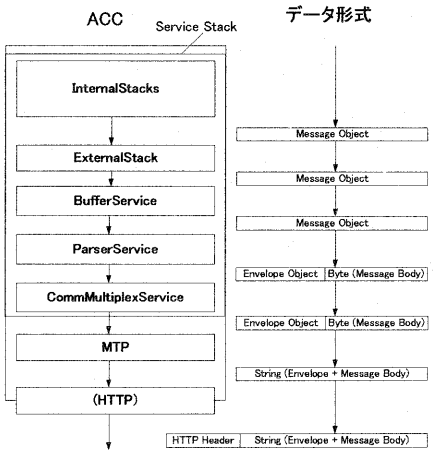

図3　ACC内でのメッセージ処理の流れ

メッセージは、InternalStackからMessageオブジェクトとして渡され、PerserServiceによってbyte列に変換される。MTPによって、Envelopeオブジェクトからデータが取得され、Envelopeとメッセージを1つのString列に変換し、通信機能を利用して送信先のACCに送信する。MTPには、具体的な通信手段として、iiop、RMI、http 等の複数の通信機能が利用可能であるが、本稿においては最も汎用性の高いhttpに対して設計を行った。プラットフォームに対して要求されるセキュリティ機能はメッセージの送受信に関連するものであるため、セキュリティ機能の実装はこのACCを拡張することがもっとも効率的であると考えられる。そこで、プラットフォームに実装する各セキュリティ機能はACC上のService Stack及びMTPを機能拡張する設計

を行うこととした。

3.1.1. プラットフォーム間相互認証機能

プラットフォーム間で相互認証及び鍵共有を実現するセキュリティ機能の設計を行う。本機能は、エージェントのメッセージ転送とは独立したプロトコルを持ち、アウトバンドでの通信を行う必要があることから、MTP から呼び出す外部機能として設計する。また、本機能を MTP から呼び出し可能なように MTP 自身の拡張も行う。プラットフォーム間相互認証が発生するタイミングは、あるプラットフォームに対して初めてメッセージを送信する場合とし、同一プラットフォームに対しての 2 度目以降のメッセージ送信時には認証処理は行わないものとする。ただし、メッセージ送信失敗等の送信エラーとなった場合には、一度初期モードに状態が遷移し、次のメッセージが送信されるときに再び認証モードに遷移して認証処理を行う。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。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。こうした要求を実現するために、本機能モジュール内でプラットフォーム毎の認証ステータスを管理する。認証状態としては、図 4 のように 3 種類の状態を定義する。

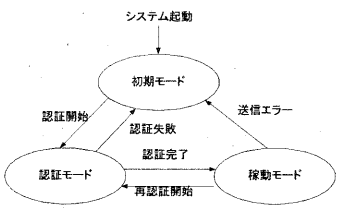

図 4　認証状態の遷移

認証のプロトコルはエージェントメッセージの送受信とは独立のアウトバンド通信とするが、状態が初期モード及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。認証を開始する側を Initiator（i）、受信側を Responder（r）とした時の認証プロトコルを以下に示す。

i → r: ID_i, Sig_Si(ID_i), Enc_Pr(R)
i ← r: ID_r, Sig_Sr(ID_r), Enc_Pi(R'), m, Mac_Kir(m)
i → r: m', Mac_Kir(m')

Kir = H(R||R')

Px は、x の公開鍵、Sx は、Px に対応する x の秘密鍵を表す。Kxy は、x と y で共有された共有鍵を表す。Sig_X(Y)は、鍵 X で Y を署名することを表し、Enc_X(Y)は、鍵 X で、Y を暗号化することを表す。H(X)は、X のハッシュ値を取ることを表し、||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。

本プロトコルによって作成された共有鍵 Kir は、メッセージ秘匿機能及びメッセージ認証機能の鍵として使用される。また、この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。この鍵更新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 → r: R
i ← r: R'
i → r: key-sequence

K'ir = H(Kir||R||R')

K'ir が新たに更新された共有鍵であり、各鍵には鍵シーケンス番号（key-sequence）が一意に定義される。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、key-sequence 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（3.1.2 で記述）の key-sequence 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。

3.1.2. メッセージ秘匿及びメッセージ認証機能

プラットフォーム間に流れるメッセージを暗号化することによってメッセージ秘匿機能を実現する。また、MAC (Message Authentication Code)をメッセージに付加することによりメッセージ認証機能を実現する。メッセージの秘匿及び認証は、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有鍵を使用して行う。Service Stack 内における個々のメッセージサービスの間は、決められたインターフェースを持っていて暗号化されたデータを解釈することが不可能であり、また Envelope も暗号化及びメッセージ認証の対象データとする必要があることから、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 に対して設計を行う。共有鍵は、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、MTP 内に格納される。さらに、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/なしを選択可能とするために、Envelope に表 1 のフィールドを追加する。

表 1　追加 Envelope フィールド（1）

フィールド名	値	意味
Code	true/false	暗号化の有無

Mac	True/false	メッセージ認証有無
key-sequence	数値	鍵シーケンス番号

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 と key-sequence からなるセキュリティヘッダ（SH）を付加して送信する（図５）。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 及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。

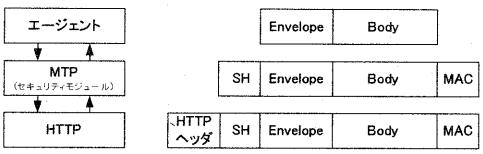

図５　プラットフォーム間通信データフォーマット

3.1.3. 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、エージェント名及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及びプラットフォーム名とする。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、Envelope からの情報取得がより容易なオブジェクトの形で扱うことができるほうが実行処理速度の点で有利であることを考慮し、Service Stack の１サービスとして設計を行う。具体的には、図６のように、Envelope をオブジェクトとして取得できる CommMultiplexService と ParserService との間に FilterService を追加する。

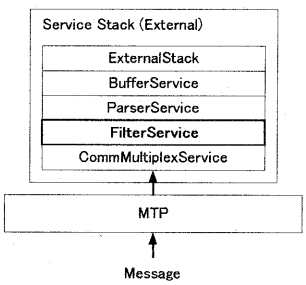

図６　FilterService の設計

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。

[通信方向(IN/OUT)] from:[送信元アドレス] to:[送信先アドレス] [アクション(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。図７にフィルタリングルールの記述例を示す。ServiceStack に実装することによって、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可能となる。

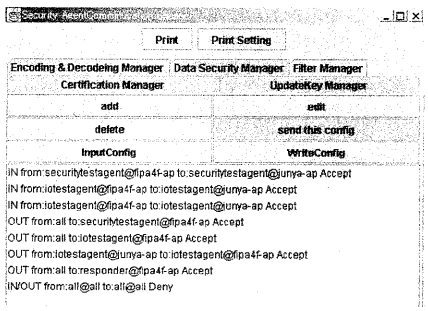

図７　フィルタリングルールの記述例

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 で定義されメッセージ受信拒否（もしくは受信不能）を意味する Reject Proposal を返送するかを決定する。表２に設定例を示す。フィルタリングの対象となる Communicative Act 及びフィルタリング時の動作については、設定ファイルを用いて変更可能とした。

表２　フィルタリング時の動作設定例

Communicative Act	動作
Accept Proposal	破棄
Agree	破棄
Failure	破棄
Inform	破棄
Inform If	破棄
Inform Ref	破棄
Not Understood	破棄
Query	Reject Proposal を返送
Query Ref	Reject Proposal を返送
Refuse	破棄
Reject Proposal	破棄
Request	Reject Proposal を返送
Request When	Reject Proposal を返送
Request Whenever	Reject Proposal を返送

3.2. エージェントへの署名/署名検証機能

エージェントへの署名/署名検証機能は、エージェントに対して同機能を提供する単一クラスとして設計を行う。各エージェントは、送信メッセージ生成後、本クラスのオブジェクトを呼び出すことで署名を行う。また、署名付きメッセージを受信したエージェントは、本クラスを呼び出すことで署名の検証を行う。署名の対象データは、ACL の先頭行(performative 記述)から最終行までとし、署名の有無及び電子署名を格納するために Envelope に表３のフィールドを追加する。

表３　追加 Envelope フィールド（２）

フィールド名	値	意味
Sig	True/False	電子署名の有無
Sig-Data	バイト列	電子署名データ

3.3. エージェントの権限証明機能及び権限確認機能

また、我々は、これまで利用権限に基づくアクセス制御の実現手法として、セキュリティポリシーを管理するポリシーファイル管理サーバが、エージェントの権限を確認し、エージェントに対して動的にチケットを発行し、そのチケットをエージェントサービスに提供することによって権限確認を行う手法を提案し実装を行ってきた[4]。従って、本稿ではエージェントとポリシーファイル管理サーバとの通信を仲介する機能の実現することによって、すでに開発済みであるポリシーファイル管理サーバを FIPA-OS 上で利用可能とし、エージェントサービスが権限確認を行うことのできるようにする。ここで通信仲介機能は、非エージェントソフトウェアのサービスを登録・仲介する ARB(Agent Resource Broker)とエージェントから非エージェントソフトウェアのサービスを起動する WA(Wrapper Agent)で構成される。ARB 及び WA は FIPA-OS 上で動作するエージェントとして設計する。WA では、エージェントから送られてきた ACL から必要情報を取得しポリシーファイル管理サーバが解釈可能なフォーマットに変換し、HTTP プロトコルに格納して送信する。エージェントは、通信仲介機能を経由してポリシーファイル管理サーバと通信を行い、権限を有することを保証するアクセスチケットの発行を受ける。エージェントは、取得したチケットを他のエージェントやサービスに提示することで権限を有することを証明する。エージェント(a)とポリシーファイル管理サーバ(s)とのチケット取得プロトコルを以下に示す。

a → s: Agent_Name
a ← s: R
a → s: Enc_Kas(R||R'||Service_Name)
a ← s: Access_Ticket
Kas = Mac_Km(R)

Agent_Name はポリシーファイル管理サーバに登録しているエージェント名、Access_Ticket は発行されるアクセスチケット、Km は事前にエージェントとポリシーファイル管理サーバとの間で共有されている共有鍵、R 及び R'は乱数を表す。

4. 実装

３章で検討した内容に従って実装を行う。セキュリティ機能のモジュール構成は図８のようになる。

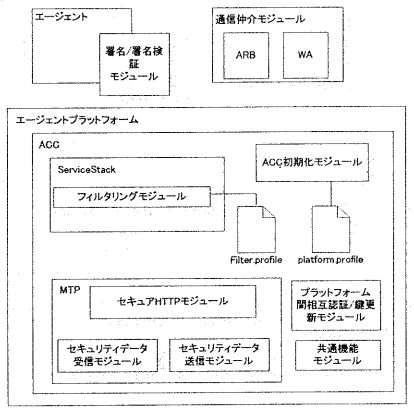

図８　セキュリティ機能のモジュール構成

ACC 初期化モジュールは、起動時に platform.profile を読み込むことによってセキュリティ機能を使用するか否かを決定する。ServiceStack には FilterService を提供するフィルタリングモジュールを実装する。MTP には、MTP の HTTP 通信機能を拡張し、暗号化/復号化及び MAC 値生成/検証の機能を実装したセキュア HTTP モジュールと、SH を作成し、HTTP のペイロードに格納するセキュリティデータ送信モジュール、受信したデータを分解し、SH、暗号化メッセージを取得し、その情報をセキュア HTTP モジュールに返すセキュリティデータ受信モジュールの３つのモジュールを実装する。プラットフォーム間相互認証/鍵更新モジュールでは、認証/鍵更新処理に加えて認証ステータスの更新などの管理を行う。共通機能モジュールは、暗号アルゴリズム等の各機能で使用する共通のセキュリティ機能をパッケージしたものであり、認証ステータス、鍵シーケンス番号及び共有鍵を格納する。各モジュールの実装サイズを表４に示す。

表４　各モジュールの実装サイズ

モジュール名	実装サイズ (KByte)

ACC 初期化モジュール （ACC 本体を含む）	25.8
プラットフォーム間相互認証/鍵更新モジュール	18.3
セキュア HTTP モジュール	19.0
セキュリティデータ受信モジュール	10.0
セキュリティデータ送信モジュール	10.0
共通機能モジュール	8.35
フィルタリングモジュール	13.3
署名/署名検証モジュール	11.8
通信仲介モジュール	137.0

MTP を拡張および外部機能として実装した各機能のクラス構成とその関係を図９に示す。図中のクラスは、fipaos.mts.security パッケージとして実装している。

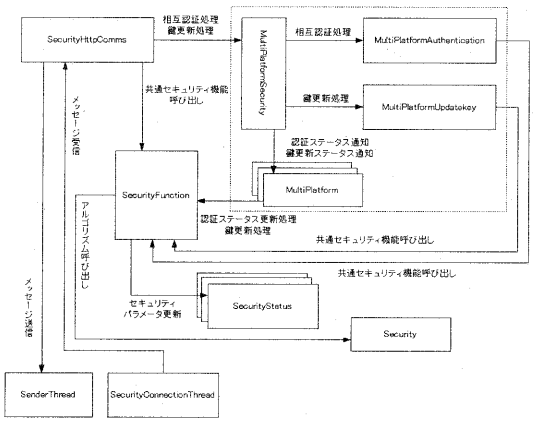

図９　セキュリティ機能を提供するクラスの構成

セキュリティ HTTP モジュールに相当するのが、fipaos.mts.httpMTP.HttpComms を拡張した SecurityHttp Comms クラスである。プラットフォーム間相互認証/鍵更新モジュールは、認証ステータスの管理及び MTP とのインターフェースを提供する MultiPlatformSecuri ty クラス、相互認証処理を行う MultiPlatformAuthenti cation クラス、鍵更新を行う MultiplatformUpdateKey クラス、MultiPlatformSecurity のサブクラスとして実装され実際に認証ステータスの管理を行う MultiPlatform クラスから構成される。共通セキュリティ機能は、共通セキュリティ機能のインターフェースを提供し、実際の処理を行う SecurityFunction クラスと、KSL セキュリティプロバイダ経由で暗号アルゴリズムを呼出すインターフェースである Security クラス、認証ステータス、鍵シーケンス番号及び共有鍵を格納する SecurityStatus クラスで構成される。

5. 評価

Pentium Ⅲ 1 GH z 、Memory512Mbyte の PC2 台を用意し、おのおのにセキュリティ機能実装した FIPA-OS 及びエージェントをインストールした。そして、その間を 100Base-T のネットワークで接続し以下の評価を行った。

5.1. プラットフォーム間相互認証/鍵更新機能

プラットフォーム間相互認証及び鍵更新機能についてその処理時間を計測した。表５に測定結果を示す。

表５　プラットフォーム間相互認証/鍵更新処理時間

評価項目	処理時間（msec）
プラットフォーム間相互認証処理時間	712.3
鍵更新処理時間	164.0

相互認証処理時間は 1 秒以下、鍵更新処理時間は、200msec と高速な処理を実現できていることを確認した。ZHANG らによって FIPA Security WG に提出されたデータ[5]では、本稿と同様に FIPA-OS 上に実装したプラットフォーム間相互認証の実測値は、Authentication Protocol に 4377msec、鍵共有を行う Negotiation Protocol に 27359msec となっている。認証プロトコルが異なるので、一概に比較することはできないが、MTP の外部機能として実装し、アウトバンドで相互認証処理を行う本方式の方が効果的な実装形態ではないかと考えられる。

5.2. メッセージ秘匿及びメッセージ認証機能

メッセージ秘匿及びメッセージ認証機能の評価を行うため、送信データサイズを変化させた場合の、送信開始から受信終了までの処理時間の変化を計測した。図１０に測定結果を示す。暗号アルゴリズムは、AES(Advanced Encryption Standard)、メッセージ認証を行うための MAC 生成アルゴリズムは HmacMD5 を使用している。鍵長はともに 128Bit である。50KByte 程度までは、処理時間はほぼ一定値となり、セキュリティ機能を使用しない場合で約 6 秒、秘匿及び認証機能を使用した場合で約 6.5 秒とセキュリティ機能を使用することによって 500msec 程度の処理時間の増加が見られる。セキュリティ機能がメッセージ送受信処理時間全体に占める割合は、約８％であることからメッセージ秘匿及びメッセージ認証機能の実装は十分な処理性能を達成していると考えられる。なお、暗号化/復号化及びメッセージ認証において暗号アルゴリズムの処理を除いた処理時間は、一様に 50msec 程度であった。従って、セキュリティ機能の処理時間のうち 90%程度がアルゴリズムの処理に使われており、残り 10%程度がセキュリティ機能によるオーバーヘッドである。従ってより高速なアルゴリズムを使用することでセキュリティ機能にかかる処理時間を減らすことが可能となる。ここで再び ZHANG らの実測結果と比較してみると、10KByte のメッセージを送受信する場合のセキュ

リティ機能（暗号化/復号化及びメッセージ認証）の処理時間は 9494msec であり、我々の方式は約 20 倍程度高速である。従って、本機能についいても効果的な実装を行えたと考えられる。

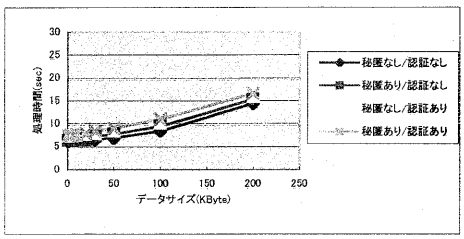

図１０　メッセージ秘匿/認証機能処理時間の変化

5.3. フィルタリング機能

フィルタリング機能については、フィルタリングルールを追加した場合の処理時間の変化を計測することによってその性能を評価した。表６に測定結果を示す。フィルタリングルール 500 の場合で処理時間 458msec と十分な処理性能を実現している。300 ルール程度を記述した場合のメッセージ送受信処理時間全体に占める割合は約 5%であり、秘匿機能、メッセージ認証機能を含めたメッセージ送受信に関わるセキュリティ機能全体では約 12%である。

表６　フィルタリング機能の処理時間

ルール数	処理時間(msec)
1	187
100	328
200	375
300	391
400	453
500	458

5.4. 署名/署名検証機能

署名/署名検証モジュールについても署名するメッセージのデータサイズを変化させた場合の、処理時間の変化を計測した。署名アルゴリズムは、SHA-1withRSA を使用している。図１１に測定結果を示す。署名生成に約 150msec、署名検証に約 50msec の処理時間を要し、合計で約 200msec となった。

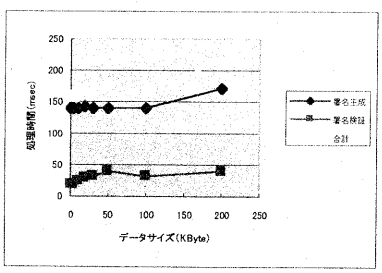

図１１　署名/署名検証処理時間の変化

5.5. エージェントの権限証明機能及び権限確認機能

エージェントの権限証明機能及び権限確認機能の評価として通信仲介機能の処理時間を計測した。具体的にはエージェントがチケットを要求してからアクセスチケットを取得するまでの処理時間を要求するチケットの枚数を変化させて測定した。表７に測定結果を示す。600～1000msec 程度がポリシーファイル管理サーバ上での処理時間であり、約 200msec が DF(Directory Facilitator)に ARB の所在を問い合わせる処理である。従って、エージェントとのプロトコル処理時時間を含む通信仲介機能の処理時間は 1600～2000msec であると考えられる。

表７　アクセスチケット取得処理時間

チケット枚数	処理時間(msec)
1	2953
2	2922
4	3011
8	6750

6. おわりに

本稿では、FIPA 準拠のエージェントプラットフォームである“FIPA-OS”に対して、既存の FIPA のフレームワークを極力変更することなく実装可能な方式を検討し、その実装サイズ、処理性能を評価した。プラットフォーム間の相互認証処理に要する時間は約 700msec となった。またメッセージ送受信に関わるセキュリティ機能の処理時間は、メッセージ送受信時間全体に占める割合は約 12%となった。以上より、本稿で提案・実装した方式は、2 章で検討したセキュリティ要件を満足し、かつ効率的な実装を実現していると考えられる。今後は、暗号化処理の高速化、ACC 全体のカスタマイズによるメッセージ処理の効率化などを検討していきたいと考えている。

なお本研究は、通信放送機構（TAO）による「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に関する研究開発」の一環として行われた。

文　献

[1] FIPA, http://fipa.org

[2] Nortel Networks, FIPA-OS, http://www.nortelnetworks.com/products/announcements/fipa/

[3] FIPA Security WG, FIPA Security Work Group Request For Information (RFI), 2001.

[4] 清本, 田中, 中尾, “モバイルエージェント環境に適用するポリシーファイル管理サーバの設計”, SCIS2002, pp.793-798, 2002.

[5] Min ZHANG and Ahmed KARMOUCH, Adding Security Features to FIPA Agent Platforms, Security WG at the 23rd FIPA meeting, 2001.