［論文］

オープンシステムにおける ミドルウェアがもつべきセキュリティ機能の分析

広沢元†

要 旨

オープンシステム開発では，ソフトウェア間の緩衝層としてミドルウェアを配置し，その上で機能要件に合致するソフトウェアを選択することが一般的である。しかし，セキュリティ要件に注目した場合，機能要件を満たすようなソフトウェアの組み合わせは必ずしもセキュリティ要件を満たしているとは言えず，両者を満たすソフトウェアの組み合わせはソフトウェア間の依存度が高く，保守性が低下する問題がある。

本問題に対して，ミドルウェアがもつべきセキュリティ機能を明確にすることで各ソフトウェアに求められるセキュリティ要件が減り，プロダクト間の依存度が下げられると考えた。本論文ではミドルウェアが保持すべきセキュリティ機能について分析し，その実現性および実効性について問題がないことを確かめた。

Abstract

In the open system development, it is general to arrange middleware as the buffer layer between software, and to select the software that agrees with the functional requirement on that. When the security requirement is paid attention, the combination of software that fills the function requirement cannot necessarily be said that it will fill the security requirement, and however, the combination of software that fills both has high dependency between software, and has the problem that maintainability decreases.

It was thought that the security requirement requested from each software by clarifying the security function that middleware had to have this problem decreased, and the dependency between products was lowered. It was confirmed that the security function that middleware had to maintain it was analyzed, and there was no problem about the feasibility and the effect in this thesis.

1 はじめに

オープンシステムとは一般的に「様々なベンダのソフトウェアやハードウェアを組み合わせて構築されたコンピュータシステム」と言われている。つまり，オープンシステムの開発は，汎用機のようにシステム内全てのソフトウェアを単一のベンダが開発を行うのではなく，業務要件に合致するソフトウェアを選択し，システムを構築することである。このようにソフトウェアの開発を異なるベンダが行うメリットの 1 つに，求める機能を専門のベンダが実装することで，成熟したテクノロジを組み合わせることができ，システム全体の信頼性を高めることができる[1]ということが挙げられる。

このようなメリットの反面，あるソフトウェアのバージョンアップが発生した場合や，使用するソフトウェアを変更した場合，その都度関連する全てのソフトウェアについて影響調査を行う必要がある。変更となるソフトウェアが OS の場合，この上で動くアプリケーションソフトウェアへの影響は大きい。また，大規模システムでは，１箇所の変更が，影響を与える範囲が広がる。そのため，OS とアプリケーションソフトウェアの依存度を下げることを目的に，ミドルウェアを配置し，その上で機能要件に合致するソフトウェアを選択することが一般的である。

次に，機能要件とセキュリティ要件の充足について考える。セキュリティ要件は非機能要件

Analysis of security function of middleware in open system

Hajime Hirosawa†

†USOL Tokyo Co., Ltd.

［論文］2007 年 06 月 30 日受付

© 情報システム学会

の一つである[2]。つまり，機能要件を満たすソフトウェアの組み合わせは必ずしもセキュリティ要件を満たしていない。機能要件とセキュリティ要件の両方を満たすようなソフトウェアの組み合わせは，セキュリティ要件を考慮しなかった場合に比べ，ソフトウェア選択の選択肢が狭まる。それによりソフトウェア間の依存度が上がり，それによる保守性の低下が問題となる。

これに対し，本研究ではソフトウェア間の依存度を下げる役割をもつミドルウェアの考え方[3]がこの問題に適用できると考えた。つまり，ミドルウェアがもつべきセキュリティ機能を明らかにすることで，各ソフトウェアに求められるセキュリティ要件が減り，プロダクト間の依存度が下げられるからである。

本論文は次の構成を採る。2 章ではシステムを「OS 層」「ミドルウェア層」「アプリケーションソフトウェア層」「保守サービス層」に層別し，システムへの攻撃に対してどの層が防御層となるのかを分析し，ミドルウェアに求められるセキュリティ機能を定義する。3 章では 2 章で定義した機能に対し，各機能の実現性について検証を行い，4 章ではその実効性を評価する。5 章では本論文の考察を行い，今後に残された課題について述べる。

2 セキュリティ機能の分析

本章でははじめに本研究におけるミドルウェア層の位置づけを明確にしておき，その上でセキュリティ環境を分析する。オープンシステムにおいて，ミドルウェアが保持すべきセキュリティ環境の分析と必要な機能を分析し，システムに対する攻撃としてどのようなものがあるのかを整理する。さらに，それらの攻撃に対して，どの層で防御可能であるかの分析を行い，ミドルウェアで保持すべきセキュリティ機能を定義する。

2.1 ミドルウェア

システム全体におけるミドルウェアの役割とは一般的に「OS とアプリケーションの中間に位置し，特定の分野でしか使われないが，その分野では必ず必要とされるような機能を提供するソフトウェアである」と言われている。しかし，本研究ではオープンシステム環境（マルチベンダ環境）に注目しているため，このような一般的な定義に加え，ベンダ間の中間層という側面も考える。そこでミドルウェアについて次のように定義する(図 1 参照)。

ミドルウェアとは，異なるベンダが提供する OS とアプリケーションの中間に位置し，OS でカバーできない機能およびアプリケーションの汎用機能を提供する基盤ソフトウェアを指す。

2.2 セキュリティ脅威の分類

ミドルウェアの保持すべきセキュリティ機能を分析するため，まずオープンシステムへの攻撃の種類としてどのようなものがあるのかを分類する必要がある。Sun Microsystems[4]はネットワークシステムへの攻撃を表 1 のように分類している。

表 1 では考えられる攻撃者と，攻撃に対応する被害の種類で分類を行ったものである。これらの攻撃を防御する場合，同じ防御方法でもいくつかの層で防御することができると考える。例えば，「(10)悪意のあるコードを意図的に使用」におけるデータ改ざんという脅威を防御する場合，データ使用時に改ざんチェックを行うことで，攻撃を検知することができる。当該機能を保持することができる層としては，「ミドルウェア層」「アプリケーション層」が考えられる。そこで，攻撃の種類と，攻撃に対する回避方法および，攻撃に対し防御可能な層を表 2 のように整理した。

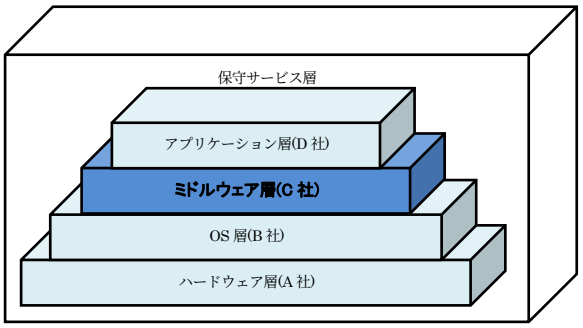

図 1 システムレイヤ概念図

表 1 オープンシステムへの攻撃の種類

被害の種類	考えられる攻撃者の種類					
	内部者				外部者	
	オペレータ	プログラマ	データ入力者	社内ユーザ	外部ユーザ	侵入者
物理的な破壊	(1)爆破，回路のショート					
情報の破壊	(2)ディスクの消去	(3)悪意のあるソフトウェア	(4)データの削除		(5)悪意のあるコードを偶然に使用	(6)悪意のあるコードを意図的に使用
データ改ざん		(7)悪意のあるソフトウェア	(8)うそのデータ入力	(9)うそのデータ入力		(10)悪意のあるコードを意図的に使用
サービスの無断使用		(11)ユーザとして無断使用		(12)許可のない使用	(13)許可のない使用	(14)許可のない使用
ブラウジング				(15)許可のないアクセス	(16)許可のないアクセス	
情報を盗む	(17)メディアを盗む			(18)許可のないアクセス	(19)許可のないアクセス	(20)許可のないアクセス

表 2 攻撃に対する防御層の対応

項番	攻撃の種類	回避方法	保守サービス層	OS 層	ミドルウェア層	アプリケーション層
1	(1)爆破，回路のショート	マシンルーム入口での ID 検査	○			
2	(3)(7)悪意のあるソフトウェア	適切なコード管理	○			
3	(8)(9)うそのデータ入力	入力データの監査	○			
4	(11)ユーザとして無断使用	適切なパスワード運用	○			
5	(17)メディアを盗む	マシンルーム出口での手荷物検査	○			
6	(2)ディスクの消去	適切なファイルアクセス権		○		
7	(10)悪意のあるコードを意図的に使用	データ改ざんチェック機能			○	○
8	(12)(13)(14)許可のない使用	ユーザ認証			○	○
9	(15)(16)許可のないアクセス	サービス認証機能			○	○
10	(18)(19)(20)許可のないアクセス	通信経路暗号化			○	○
11	(4)データの削除	適切な実行権限チェック			△	○
12	(5)悪意のあるコードを偶然に使用	SQL インジェクションチェック等				○
13	(6)悪意のあるコードを意図的に使用	SQL インジェクションチェック等				○

○：直接機能提供可能，△：間接的に機能提供可能

表 2 の項番 1～5 に分類した攻撃の種類は，物理的な破壊および不正な運用による攻撃である。そのため防御層は保守サービスとなる。また項番 6 は誤操作または意図的な不正ファイル操作であることから OS のファイルシステムによる防御が必要となる。同様に項番 7～10 は，

許可のないサービス使用または盗聴・改ざんによる攻撃である。これらの攻撃を回避するためには認証機能および通信経路の暗号化が必要である。しかしこれらの機能はミドルウェア層またはアプリケーション層で防御が可能である。この場合，ミドルウェアの定義で示した「アプリケーションの汎用機能を提供する」の原則から，ここではミドルウェアで提供することとしている。また，項番 11 はデータ入力者の誤操作または意図的なデータ削除である。そのためアプリケーション層での防御が必要となるが，ミドルウェア層として直接機能提供することができないが，権限チェックに必要な暗号・認証ライブラリを提供することで，特殊な暗号知識等を隠ぺいできるため AP 層の負担軽減が期待できる。項番 12〜13 は業務アプリケーションのバグを狙った攻撃であるため，アプリケーション層による防御が必要となる。

2.3 ミドルウェアに求められるセキュリティ要件

2.2 節では攻撃の種類に対する防御層の分析について述べた。分析の結果，ミドルウェアに求められるセキュリティ要件としてミドルウェアが直接的に備えるべき機能と，ミドルウェアが間接的に使用者に提供すべき機能に大別することができる。ミドルウェアが直接的に備えるべき機能として具体的には表 3 の４つを挙げることができる。表 3 に挙げた機能は汎用性が高く，どのような業務アプリケーションでも適用できる機能である。

また，ミドルウェアが直接的に提供する機能ではないが，使用者に対して間接的に提供すべき機能を表 4 に示す。これらの機能はアプリケーション層でセキュリティ機能を実装する上で必要となるライブラリを提供することで，基盤要素技術の隠ぺいを行うことができる。これによりアプリケーション層は業務ロジックに注力することができ，ソフトウェア品質の向上が期待できる。

これらのセキュリティ機能を各システムのセキュリティ要件に合わせて組み合わせることで，柔軟なシステム設計を行うことができる。

表 3 ミドルウェアが備えるべきセキュリティ機能

項番	機能名	説明
1	ユーザ認証機能	ユーザ名/パスワードによる認証を行い，クライアントからの接続チェックを行う。本機能により，外部からの不正なアクセスの多くを防御することが可能となる。
2	サービス認証機能	IP アドレスまたは，ユーザ認証機能によって認証されたユーザ単位で，許可されたサービスであるかどうかの認証を行う。本機能により，許可されていないサービスの使用を防御することが可能となる。
3	データ改ざんチェック機能	ネットワーク中のデータが改ざんされたことを検知する機能。本機能により，改ざんされたデータの拒否を行い，不正操作等を防御することが可能となる。
4	通信経路暗号化機能	ネットワーク中のデータの暗号化を行う機能。本機能により，N/W 中の通信内容が保護され，盗聴や情報漏えいを防御することが可能となる。

表 4 ミドルウェアが提供すべき機能

項番	機能名	説明
1	共通暗号化ライブラリ	暗号･復号および疑似乱数生成･ハッシュ値取得などのセキュリティ機能実装に必要な API を提供する。本機能により，アプリケーション層でのセキュアプログラミングのサポートを行うことが可能となる。

3 ミドルウェアセキュリティ機能の実現性

2 章ではセキュリティとミドルウェアの関係について分析し，ミドルウェアに求められる機能を洗い出した。本論文では，これらの機能の妥当性を検証するため，各機能の実現性と実効性の確認を行った。

まず，本章ではミドルウェアセキュリティ機能の実現性を確認するため，2 章で洗い出した機能を日本ユニシス㈱が販売するミドルウェア「MIDMOST®」に実装を行い，ミドルウェア上に実装するセキュリティ機能として問題がないかを検証した。以下では「MIDMOST®」に実装した各機能の具体的な実装モデルを示す。

3.1 ユーザ認証機能

本機能はクライアントノードがサーバノードと接続する際にユーザ名とパスワードで認証を行う機能である。本機能を実装する際，パスワードを平文でネットワーク上に送信してしまうと，盗聴などによってパスワードが漏えいしてしまう恐れがある。このためパスワードの暗号化を行う必要がある。この場合，一般的に公開鍵方式または，共通鍵方式によりパスワードの暗号化を行い送信することが考えられる。しかし，公開鍵/共通鍵方式は次に示す 2 点の問題が発生する。

(1) 鍵を交換しなければならない
(2) 暗号文を盗聴された場合，同一パケット再送によりなりすましが行われる恐れがある

これらの問題を解決するため，認証情報が暗号化される CHAP（Challenge Handshake Authentication Protocol）の技術を利用することにした。図 2 に本機能の実装モデルを示す。

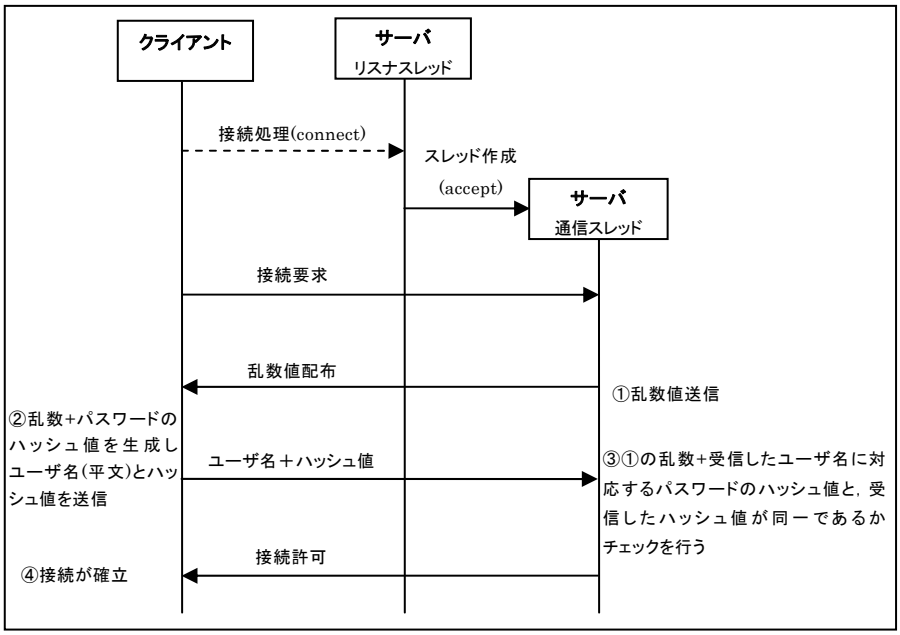

図 2 ユーザ認証プロトコル

本モデルでは認証情報を接続毎に異なる乱数情報①を基にパスワードをハッシュ化②したデータを渡す。認証側では②と同じ処理を行い③，認証情報が一致した場合のみ接続を許可している④。これにより不正な使用者のアクセスおよび盗聴による攻撃を防御することができる。

3.2 サービス認証機能

本機能は接続してきたクライアントを IP アドレスまたは，ユーザ認証機能によって認証されたユーザをもとに使用可能なサービス(機能)を制限することができる機能である。システムの多くは，通常権限や管理権限などのように権限毎に使用できるサービス(機能)を提供している。このような機能はアプリケーション層で実装することが多いが，ミドルウェア層で実装することで，アプリケーション層へのセキュリティ機能の隠ぺいを行うことができる。

本機能では通常ネットワークリソースやファイルリソースのアクセス権管理を行うための手法である ACL（Access Control List）の考えを応用した。ACL はファイルリソースなどの物理オブジェクトのアクセス管理を行うものであるが，本機能ではサービスという論理オブジェクトに対して応用した。また，動的な運用や，運用補完ツール等での編集を可能とするために，ACL 定義ファイルを外部フラットファイルとした。図 3 に本機能の実装モデルを示す。

本モデルでは，接続時のネゴシエート処理にて IP アドレスによる認証を行う①。ACL 管理ファイルを確認し，許可されていれば接続許可を返している②。その後，使用者がサービスの使用要求を行った場合は，サービス内でユーザ権限チェック③を行い①同様, ACL 管理ファイルを確認し，ユーザの権限チェックを行い，許可されていれば使用許可④を返している。

3.3 データ改ざんチェック機能

本機能は，ネットワーク中を流れている電文の一部または，全てが攻撃者によってデータが改ざんされた場合に改ざん検知を行う機能である。一般的に HTTP 層におけるデータ改ざんに対するセキュリティとしては一般に SSL(Secure Socket Layer)を使用することが多いが, ミドルウェア層では HTTP 等の既存プロトコルではなく独自のプロトコルにより通信が行われることが多く，SSL を使用することができない。このため，SSL の内部で使用されている MAC（Message Authentication Code）の技術を使用することで，セキュリティ強度を落とすことなくミドルウェア層独自の通信プロトコルに対応する。図 4 と図 5 に MAC を使用した場合と, しなかった場合の実装モデルを示す。

図 4 のように Message のみを受取側に渡した場合，ネットワーク中でデータが改ざんされた場合に受取側は正しい Message であるか検証する術がないため，改ざんを検知することができない。それに対し，MAC を使用した改ざん検知モデルを図 5 に示す。

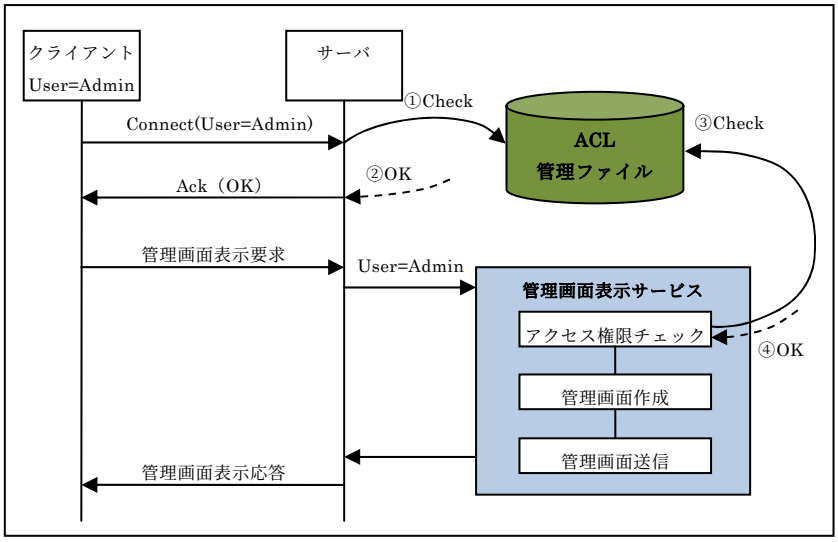

図 3 サービス認証機能概念図

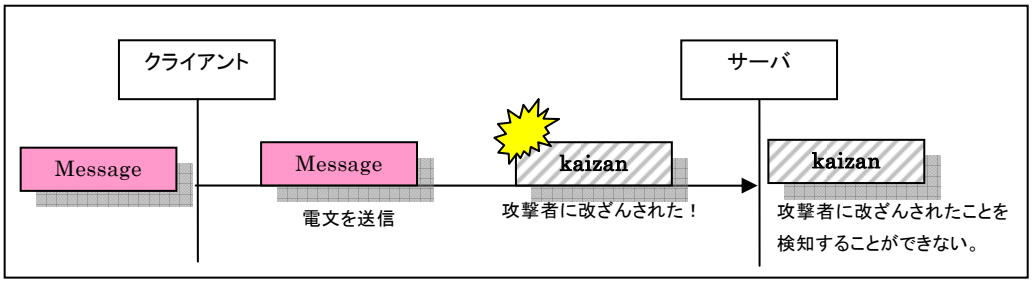

図 4 MAC を使用しない場合

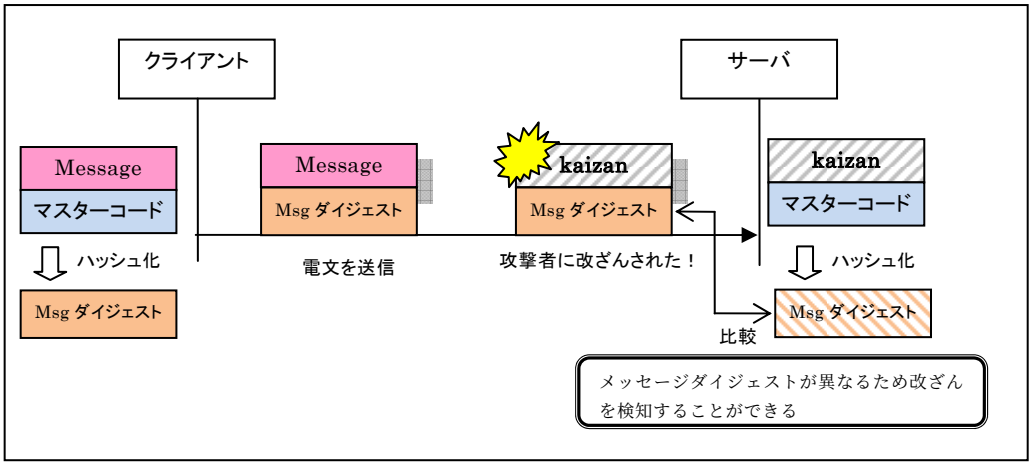

図 5 MAC を使用した改ざん検出例

本モデルでは事前に交換してあるマスターコードを用い, 正当な Message のダイジェスト(ハッシュ値)を作成し，このダイジェストを Message に添付して送付することで, 受取側も送り側と同様にマスターコードからダイジェストを作成し，作成したダイジェストと受け取ったダイジェストの比較により改ざんを検知している。

3.4 通信経路暗号化機能

本機能は使用者のメッセージをサーバとやり取りする際に，盗聴を防ぐ機能である。盗聴防止機能を実装するためにミドルウェア層で考慮しなければならない点は，パフォーマンス劣化を最小限に抑えること，および使用者への基盤技術を隠ぺい(運用負荷の低減)することを挙げることができる。これらのことを踏まえ，本機能を次の方針で実装することにした。

(1)暗号方式はハイブリッド暗号方式を使用する

通信システムにおいて暗号時に使用する鍵の交換を安全に行えるかどうかがセキュアな通信を行う上で重要である。そこで，処理速度は遅いがセキュリティ面で優れている公開鍵方式で共通鍵を暗号化して共通鍵の交換を行うハイブリッド暗号方式を採用した。

(2)通信で使用する鍵は使用者から隠蔽し，必要に応じて鍵を内部的に生成する

わずらわしい鍵の運用から使用者を解放するため，それぞれの鍵は必要に応じて，最低限の数だけ内部的に生成する。

・公開鍵：プログラム起動時に自分用の公開鍵/秘密鍵のペアを 1 つ生成する
・共通鍵：サーバと接続する毎に乱数を交換し，交換した乱数を元に生成する

※内部的に生成した鍵は全て内部で生成・破棄されるため，使用者が証明書や鍵を意識する必要は全くない。また，公開鍵/秘密鍵ペアはプロセス起動毎に生成し直すため，より一層セキュリティ強度を高めることができる。

上述の点を考慮し，鍵を自動生成し，安全な鍵交換が行える実装モデルを図 6 に示す。

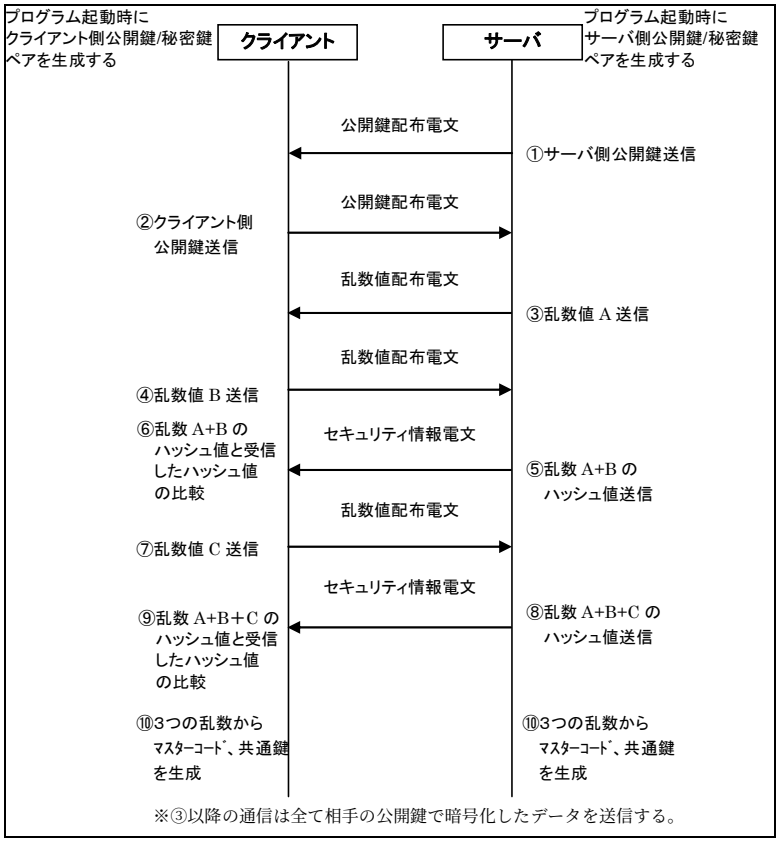

図 6 鍵交換プロトコル

本実装モデルは自動生成した共通鍵を安全に交換するため，交換の際は公開鍵方式で暗号化を行い，かつ接続先の確認をお互いが行うため 3 ウェイハンドシェイクで 3 つの乱数を交換している。これにより，乱数から求められた共通鍵が安全に鍵交換を行うことが可能となる。

4 ミドルウェアセキュリティ機能の実行性

3 章ではミドルウェア層におけるセキュリティ機能の実現性について述べたが，実現性が確認できたところでシステムのパフォーマンスの劣化が著しい場合，正しい実装モデルであると評価できない。そのため，各機能のパフォーマンスを計測し，セキュリティ機能を使用した場合と，使用しなかった場合でパフォーマンス劣化の度合いを測定し，実運用に耐えることができるかどうかという点で実効性の評価を行う。

4.1 評価指針

一般にセキュリティレベルの向上とパフォーマンス追求はトレードオフの関係にある。また，3 章の実装モデルで示した通り，CPU 時間や IO 時間などの処理コストに関してもセキュリティ機能を使用した場合，セキュリティ機能を使用しなかった場合に比べ，セキュリティ機能に付帯して処理コストが増加している。このことからパフォーマンス測定ではパフォーマンス劣化が発生することは明白であるが，以下で示す一般的なシステムのアプリケーションモデルを想定し，想定環境におけるアプリケーションの処理パフォーマンス評価を行う。

一般的にクライアントサーバシステムのアプリケーションモデルの場合，クライアントからサーバへの接続は処理コストが大きいため，システム起動時にあらかじめ想定範囲の接続コネクションを作成して使いまわす「コネクションプール」方式(図 7 参照)を使用することが多い。このため,「コネクションプール」方式のクライアントサーバシステムのアプリケーションモデルを想定して評価する。

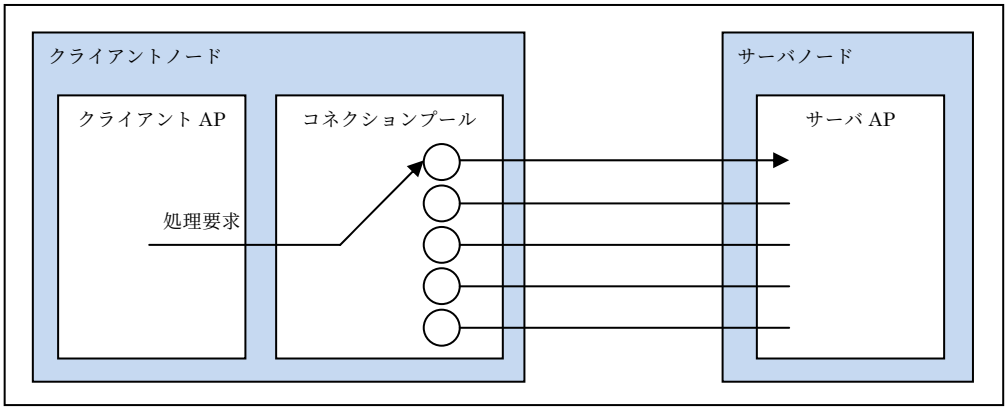

図 7 コネクションプール概念図

4.2 評価指針

本章で行うパフォーマンス評価で使用するマシン環境は表 5 の通りである[5]。暗号化アルゴリズムおよび鍵長によるパフォーマンスの変化が考えられるが，本論文はアルゴリズムによるパフォーマンス測定を行うものではなく，3 章で示した実装モデルの評価を行うため，一般的に広く使用されるアルゴリズムと鍵長で測定を行った。

4.3 パフォーマンス分析

本章で測定する処理時間は非常に小さい値であるため，誤差が出やすく，他アプリケーションの影響を受け易い。そのため各処理は 100 回実行し，平均値を求めることとする。また，機能 ON にした場合と OFF にした場合で交互に各 3 回測定することで機能 ON/OFF を比較した場合に，より正確なパフォーマンス測定を行う。

分析項目は 4.1 で示した指針に従う。評価指針ではコネクションプール方式のシステムを想定しているため，パフォーマンス分析の観点としては，以下の 2 点について分析する。

(1) コネクション確立時の処理時間
(2) コネクション確立後の AP ターンアラウンドタイム

以下では各分析結果について述べる。

(1) コネクション確立時のパフォーマンス分析

本項目は，システム起動時のコネクション作成時処理を想定している。表 6 は大きく「認証」と「通信経路暗号化」機能に大別し，各機能を使用した場合と，使用しなかった場合でコネクション確立にかかる処理時間を測定した結果である[5]。

表 5 パフォーマンス測定マシン環境

マシン	IBM IntelliStation M Pro
CPU	Pentium4 3.6GHz
メモリ	1GB
LAN	100BASE-T
暗号化アルゴリズム	AES（鍵長 256bit）/ RSA（1024bit）

表 6 コネクション確立時のパフォーマンス測定結果

	平均（msec） **認証**	平均（msec） **通信経路暗号化**	平均（msec） **認証＋通信経路暗号化**
機能 ON	253.331	493.769	560.626
機能 OFF	82.544	82.544	82.544
パフォーマンス対比	3.069	5.981	6.791

表 6 より，ユーザ認証機能では，クライアントーサーバ間のラウンドトリップが 3 往復行われるため，認証機能を OFF にした場合に比べ著しくパフォーマンス劣化していることがわかる。通信経路暗号化ではユーザ認証よりパフォーマンスが劣化した。その理由として，公開鍵の生成は処理時間が多くかかること，ラウンドトリップが 4 往復半である（ユーザ認証に比べ 1 往復半多い）ことがあげられる。

以上のことから，システム起動時に発生するコネクション確立時の処理パフォーマンスは「認証機能」「通信経路暗号化機能」のどちらを使用しても，セキュリティ機能を使用しなかった場合と比較し，著しくパフォーマンス劣化が発生することがわかる。

(2) コネクション確立後の AP ターンアラウンドタイム分析

本項目は，システム起動時のコネクションプール作成処理が完了した後の AP 処理を想定している。クライアント AP が処理を開始して，サーバ AP から処理を受け取るまでの処理時間について分析する。コネクション確立後は通信経路暗号化を行った場合に通信電文サイズに応じてパフォーマンス劣化が発生することが想定できる。このため，通信経路暗号化を使用した場合と，しなかった場合，さらに通信電文のサイズを 1 KB～10MB の 5 段階で測定した[5]。その際，サーバ AP の処理がパフォーマンス測定の結果に影響を与えないようにするため，サーバ AP では何も処理せずに受信したメッセージをそのままクライアント AP に送り返している。

表 7 のように通信経路の暗号化を行った場合，1KB 程度のメッセージサイズの場合，パフォーマンスの劣化が発生しなかった。しかし，メッセージサイズを 100KB 以上にした場合，通信経路暗号化を行わなかった場合に比べ，2.5 倍～2.7 倍の処理時間がかかることがわかった。処理時間と通信電文のデータサイズの間には正比例の傾向が見られ，10MB 以上のデータサイズの送受信時間も推定することができることがわかった。

4.4 パフォーマンス評価

パフォーマンス測定を行った結果，システム起動時に行うコネクション確立処理においては，著しいパフォーマンス劣化が発生し，その後の AP ターンアラウンドタイムに関しては 1KB 程度の通信電文であればパフォーマンス劣化がほぼ発生しないということがわかった。これらの結果に対し「コネクションプール」方式のクライアントサーバシステムのアプリケーションモデルを想定して評価した結果を以下に示す。

(1) コネクション確立処理のパフォーマンス

評価指針で示したコネクションプール方式のアプリケーションモデルにおいて，コネクション確立処理はシステム起動時に発生するため，システム起動時のオーバヘッドが大きくなるのみであり，アプリケーションへの影響は少ないと考える。

(2) コネクション確立後の通信パフォーマンス

コネクション確立後の AP ターンアラウンドタイムは，通信電文が 1KB の場合，ほぼ劣化は発生していない。通信電文が 10KB 以上の場合のパフォーマンス劣化に関しては，暗号アルゴリズムの限界であるが，セキュリティ強度とパフォーマンスはトレードオフの関係にあるため，このパフォーマンス劣化が許容できない場合は，セキュリティ強度が低くパフォーマンス性能が高い暗号アルゴリズムを選択することで対応することができる。

これらのことから，「2 セキュリティ機能の分析」で定義した機能の性能が満たされていると言える。

表 7 コネクション確立後の AP ターンアラウンドタイム測定結果

	平均（msec） **1KB**	平均（msec） **10KB**	平均（msec） **100KB**	平均（msec） **5MB**	平均（msec） **10MB**
通信経路暗号化 ON	25.866	40.864	199.415	1059.444	2110.455
通信経路暗号化 OFF	25.001	26.589	79.740	393.268	781.510
パフォーマンス対比	1.034	1.536	2.500	2.693	2.700

5 考察と今後の課題

本研究はオープンシステムにおけるミドルウェアがもつべきセキュリティ機能の分析を行い，本分析に対し，実現性検証と実効性検証を行った。まず，実現性検証として日本ユニシスが販売するミドルウェア「MIDMOST®」に実装を行い，各機能の実装モデルを示すことで実現性の確認を行った。また，実効性検証として各機能のパフォーマンス検証を行い，パフォーマンス面での実効性の確認も行った。

これらに加え，本研究で分析したセキュリティ機能を実装した「MIDMOST®」を用いた大規模フルバンキングシステムが平成 19 年 5 月から H 銀行にて稼働を開始している。このことから，実運用面においても妥当性が証明された。

本研究ではミドルウェアがもつべきセキュリティ機能の実現性および実効性の確認として「MIDMOST®」による確認を行ったが，今後は「MIDMOST®」以外のミドルウェアでの検証も行い，より本研究の妥当性を高める必要があると考えられる。

謝辞

本論文を執筆する題材を与えていただいた，日本ユニシス㈱MIDMOST®プロジェクトチームおよび，執筆指導・ご協力していただいた関係各位に対し，厚くお礼を申し上げます。

参考文献

[1] Douglas Heintzman,An introduction to open computing, open standards, and open source, IBM Technical library, July, 2003.
[2] 大津留史郎, IT セキュリティ要件の抽出手法 , Provision No.54, pp.81, Summer, 2007.
[3] 菅谷俊朗・白木和彦, ミドルウェアに期待される役割と特性－MIDMOST® for .NET からみた考察, UNISYS TECHNOLOGY REVIEW 第 89 号, pp.4, May, 2006.
[4] SunMicrosystems, 深刻化するセキュリティ攻撃とその解決へ向けて, Sun Inner Circle, Vol.21, April, 2004.
[5] 広沢元, MIDMOST を例にした TCP 通信におけるセキュリティ確保の実例, 日本ユニシス(株)TechnicalSymposium2005, 2005.

著者略歴

2002 年日本ユニシス・ソフトウェア(株)(現 USOL 東京(株))入社。共通利用技術部にて Windows ベースでは世界初のオープンフルバンキングシステム『BankVision®』を支えるミドルウェア『MIDMOST®』の開発を行う。現在は，『MIDMOST®』を含む Windows 基盤システムの導入サポートを行っている。