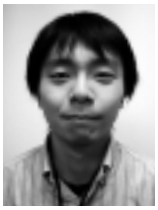
Web Services Light
― あらゆる情報機器にWebサービスを ―

Web Services Light:
Implementing Web Services on All Types of Information Devices

Webサービスおよび Webサービス・セキュリティーの技術を用いることで、アプリケーションはインターネット上で公開された各種サービスと自律的かつ安全に情報をやり取りできるようになる。これらの技術は近年、幾つかのソフトウェア製品に搭載され、サーバーやPCで利用できるようになった。当論文では、モバイル機器、テレマティックス、オフィス機器などの計算資源の限られた各種情報機器上でもWebサービスおよびWebサービス・セキュリティーの技術を実現するために、高速かつ省メモリーの逐次処理による小型実装を紹介する。

Use of the technology underlying Web services and Web service security makes it possible for applications to exchange information independently and securely with the various services available on the Internet. Such technology has in recent years been incorporated into a variety of software and can now be used by servers and personal computers. In this paper, I present ideas for small-scale implementation using high-speed, memory-saving sequential processing with a view to realizing technology for Web services and Web service security on various types of information devices such as mobile devices, telematics, office equipment and other types of information devices with limited computational resources.



日本アイ・ビー・エム株式会社 東京基礎研究所
インターネット・テクノロジー
副主任研究員
Researcher
Internet Technology
Tokyo Research Laboratory, IBM Japan, Ltd.

寺口 正義 Masayoshi Teraguchi

【プロフィール】
2000年、日本アイ・ビー・エム入社。東京基礎研究所にて映像要約の研究プロジェクトを経て、現在モバイルWebサービス・プロジェクトにて各種情報機器向けのWebサービスおよびWebサービス・セキュリティーの研究・開発に従事。

日本アイ・ビー・エム株式会社 東京基礎研究所
インターネット・テクノロジー
副主任研究員
Researcher
Internet Technology
Tokyo Research Laboratory, IBM Japan, Ltd.

山口 裕美 Yumi Yamaguchi

【プロフィール】
2001年、日本アイ・ビー・エム入社。東京基礎研究所にて情報可視化の研究プロジェクトを経て、現在モバイルWebサービス・プロジェクトにて各種情報機器向けのWebサービスおよびWebサービス・セキュリティーの研究・開発に従事。

日本アイ・ビー・エム株式会社 東京基礎研究所
インターネット・テクノロジー
主任研究員
Research Staff
Internet Technology
Tokyo Research Laboratory, IBM Japan, Ltd.

伊藤 貴之 Takayuki Itoh

【プロフィール】
1992年、日本アイ・ビー・エム入社。東京基礎研究所にて科学技術計算、CAD、CAE、情報可視化の研究プロジェクトを経て、現在モバイルWebサービス・プロジェクトのリーダー。博士（工学）。京都大学情報学研究科客員助教授を兼任。

日本アイ・ビー・エム株式会社 東京基礎研究所
インターネット・テクノロジー
主任研究員
Research Staff
Internet Technology
Tokyo Research Laboratory, IBM Japan, Ltd.

西海 聡子 Akiko Nishikai

【プロフィール】
1996年、日本アイ・ビー・エム入社。東京基礎研究所にて高性能液晶ディスプレイの研究・開発などを経て、現在モバイルWebサービス・プロジェクトにて各種情報機器向けのWebサービスおよびWebサービス・セキュリティーの研究・開発に従事。

日本アイ・ビー・エム株式会社 東京基礎研究所
インターネット・テクノロジー
副主任研究員
Researcher
Internet Technology
Tokyo Research Laboratory, IBM Japan, Ltd.

佐藤 史子 Fumiko Satoh

【プロフィール】
2001年、日本アイ・ビー・エム入社。東京基礎研究所にて映像要約、モバイルWebサービスの研究プロジェクトを経て、現在Webサービス・セキュリティー・プロジェクトにてWebサービスおよびWebサービス・セキュリティーの研究・開発に従事。

1. はじめに

Webサービスとは、自己記述型の検索可能な汎用サービスを分散ソフトウェア部品として扱い、サービスの構築方法、プラットフォームを問わず、組織の枠を超えてこれらのサービスを柔軟に連携させる仕組みであり、XML（Extensible Markup Language）技術［参考文献1］を利用した通信プロトコルSOAP（Simple Object Access Protocol）［参考文献2］などの標準技術から成る。

Webサービス・セキュリティー［参考文献3］とは、Webサービスにて通信されるSOAPメッセージの完全性（通信途中で改ざんされていないこと）や秘匿性（通信途中でのぞき見されないこと）をエンド・ツー・エンドのメッセージ・レベルで保証するために、SOAPメッセージに電子署名や暗号化を適用するための仕様であり、現在も着実に標準化が進められている。Webサービスにおけるメッセージ・レベルのエンド・ツー・エンド・セキュリティーの概念図を図1に示す。

これらの技術的流れを受け、各社からサーバーやPC（Personal Computer）などで稼働するWebサービスおよびWebサービス・セキュリティーに対応した製品が公開されている*1が、メディアでも取り上げられているようにWebサービス・セキュリティーによる性能低下が指摘されている。

一方で、モバイル機器、テレマティックス、オフィス機器などの各種情報機器の性能向上に伴い、ユビキタス・コンピューティングの世界［参考文献6］が現実味を帯びてきている。ユビキタス・コンピューティングとは、「利用者が意識することなく、いつでもどこでもネットワークに結合されたデバイスを通じてさまざまな情報にアクセスできる」環境を指しているため、プラットフォームの異なるデバイス同士の自律的な情報交換技術が必要になる。Webサービスはその目的に対して最適な技術である。

そこで、筆者らはサーバーやPC上のみならず、計算資源の限られた各種情報機器上でもWebサービスおよびWebサービス・セキュリティーを実現するために、SAX（Simple API for XML）［参考文献7］を用いた逐次処理による小型実装を試みた。各種情報機器の動作環境がほとんどJ2ME（Java™ 2 Platform Micro Edition）［参考文献8］であることを考慮し、著者らのプロトタイプはJ2MEの最小構成にあたるCLDC（Connected Limited Device Configuration）［参考文献9］上で動作するようにした。また、J2ME上で動作するXMLパーザーは市場に出ていないため、独自のXMLパーザーを利用し、同様に署名および暗号化を行うためのセキュリティーの基本ライブラリーも独自のものを利用した。さらに、DOM（Document Object Model）［参考文献11］を中間構造として利用する実装と比較して、著者らのプロトタイプが高速・省メモリーで動作することを確かめる実験を行った。また、市販の携帯電話上でも実用的な処理時間内で動作することを示す実験も行った。

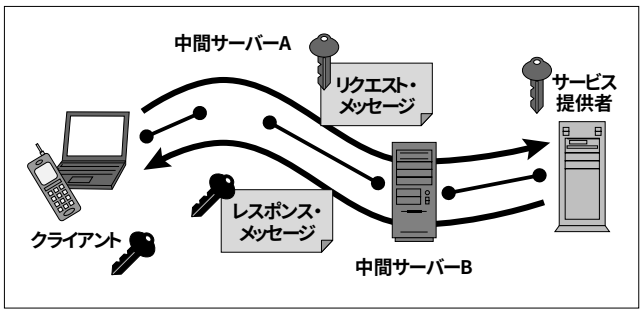


図1. Webサービスにおけるメッセージ・レベルのエンド・ツー・エンド・セキュリティーの概念図

当論文では以降、2章で各種情報機器におけるWebサービスおよびWebサービス・セキュリティーを用いた実用シナリオを例示する。3章ではDOMを用いた一般的な実装を紹介し、4章で著者らのSAXを用いた逐次処理による新しい実装について述べる。さらに、5章で著者らのプロトタイプを用いた実験について述べ、最後に当論文のまとめを6章で述べる。

＊1　IBMではWebSphere® Application Server（WAS）4.0［参考文献4］からWebサービスの導入を、WAS 5.0.2［参考文献5］からWebサービス・セキュリティーの導入を始めている。

2. シナリオ

モバイル機器、テレマティックス、オフィス機器などの各種情報機器にWebサービスおよびWebサービス・セキュリティーが搭載されることで、どのような用途が想定できるだろうか。以下に、具体的なシナリオ案を幾つか例示する。

2.1. 携帯電話・PDAにWebサービス

Webサービスを搭載した携帯電話やPDA（Personal Digital Assistance：携帯情報端末）といった携帯端末を想定する。このとき、企業の従業員が出張先に持参している携帯端末が定期的に勤務先サーバーで公開されているサービスと通信を行い、スケジュールの同期を取ったり、手元の顧客情報を更新することが可能となる。

2.2. テレマティックスにWebサービス

Webサービスを搭載したテレマティックスから各種情報を

発信する乗用車を想定する。このとき、乗用車がメーカーや販売店が公開しているサービスに定期的に部品の消耗情報などを送信することで、各社が乗用車の遠隔メンテナンスを実現できる。また、事故やトラブル発生時に、警察・保険業者・修理業者などが公開しているサービスに即座に事故情報や故障情報を送信することで、各業者が迅速かつ正確な対応を行えるようになる。

2.3. オフィス機器・医療機器にWebサービス

例えば、オフィスで用いるコピー機やファックス、医療現場で用いる撮影／測定機器などにWebサービスが搭載されたと想定する。このとき、各機器がメーカーや販売店が公開しているサービスに定期的に利用履歴や消耗情報などを送信することで、各社は各種機器の遠隔メンテナンスや遠隔課金を実現できる。

2.4. WebサービスおよびWebサービス・セキュリティー実装による利点

2.1～2.3節で例示したどのシナリオにおいても、Webサービスを実装することにより、各種情報機器は動作環境や相互接続性などを気にせず自律的な情報提供・共有が可能となる。また、Webサービス・セキュリティーを実装することにより、通信過程で経由する無線通信業者などのゲートウェイにメッセージをのぞき見されたり、改ざんされたりすることなく、安全に各種情報を通信できる。

3. DOMを用いた一般的な実装のアーキテクチャー

XML文書を木構造で表現するDOMを中間構造として利用するWebサービスおよびWebサービス・セキュリティーの一般的な実装を実現するアーキテクチャーを図2に示す。このアーキテクチャーでは、Webサービスに必要なSOAPメッセージの処理をSOAPエンジンが担当し、Webサービス・セキュリティーに必要な署名・暗号化の処理をWS-Securityエンジンが担当している*2。

一般的な処理手順は以下の通りである。情報機器にはSOAPエンジンおよびWS-Securityエンジンが搭載されているとする。このとき、情報機器にて稼働するアプリケーションが、SOAPエンジンに必要なパラメーターを渡すと、情報機器は以下の手順によりサービスにリクエストSOAPメッセージを送信する。

- SOAPエンジンがパラメーターを包括するリクエストSOAPメッセージをDOMとして構築する。
- WS-Securityエンジンが必要に応じてリクエストSOAPメッセージに署名および暗号化を適用し、DOMを再構築する。
- DOMからSOAPメッセージを生成し、サービスに送信する。

続いて情報機器は、サービスからレスポンスSOAPメッセージを受信すると、以下の手順によりサービスからの回答をアプリケーションに渡す。

- XMLパーザーがレスポンスSOAPメッセージを解析してDOMを構築する。
- WS-SecurityエンジンはDOMを解析しながら必要に応じてレスポンスSOAPメッセージの署名を検証し、暗号を復号化する。復号化されたデータは再帰的に解析され、復号化前のDOMの要素と置換される。
- SOAPエンジンがDOMを解析しながらサービスの回答を抽出し、アプリケーションに渡す。

現在既に、このアーキテクチャー・処理手順に基づくSOAPエンジンおよびWS-SecurityエンジンがWSTKMD（Web Services ToolKit for Mobile Devices）[参考文献11]というalphaWorks®ソフトウェアに搭載されている。WSTKMDはWSDD（WebSphere Studio Device Developer）[参考文献12]という情報機器向けのアプリケーション開発ツール製品の上でプラグインとして稼働することを想定している。筆者らはWSTKMDに対して、WS-Securityエンジンの実装で貢献しており、今後も継続した貢献ができるように研究を進めている。

＊2　WS-Securityエンジンは、SOAPエンジンに対する一種のプラグインであり、不必要な場合には実装しなくてもよい。

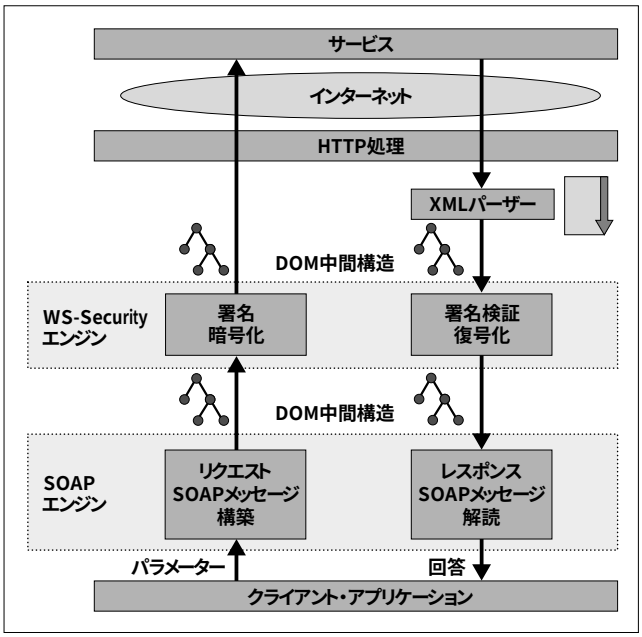


図2. WebサービスおよびWebサービス・セキュリティーのDOMを用いた一般的な実装を実現するアーキテクチャー

4. SAXを用いた逐次処理による新たなWebサービスおよびWebサービス・セキュリティーの実装

筆者らの想定している各種情報機器の中には、携帯電話やPDAなどの小型機器も含まれている。これらの小型機器は、一般の計算機に比べて処理速度が大幅に遅く、また実行時メモリー搭載量も大幅に少ない。そこで、小型機器向けのWebサービスには、高速・省メモリーの新たな実装方法を採用することが望ましい。

一般的にSOAPメッセージを処理するには、中間構造としてDOMを構築して処理する方法と、SAXを用いたイベント列として順に処理する方法の2通りがある。DOMは機能面での汎用性が高く、プログラミングしやすい反面、処理速度・メモリー使用量の点で問題があることが一般的に知られている。一方、SAXは汎用性やプログラミングのしやすさに問題がある反面、処理に必要な情報だけをメモリー上に確保することができるため処理速度やメモリー使用量の点で優れている。

そこで、筆者らは高速・省メモリーを実現するために、SAXを用いた逐次処理による新たな実装を試みた。ただし、逐次処理を考えた場合、Webサービス・セキュリティーの仕様をすべて網羅することは困難であるため、Webサービス・セキュリティーのサブセットであるSOAP Message Security：Minimalist Profile（MProf）[参考文献13]に基づいた制約を考慮している。

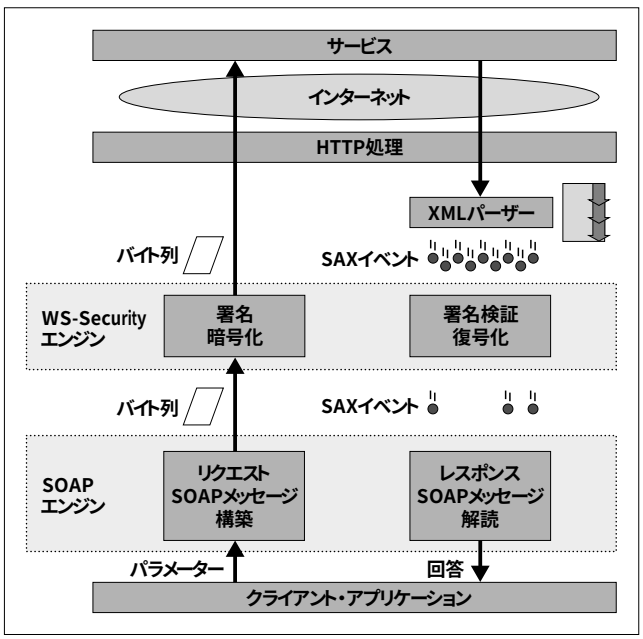


図3. 筆者らが考えるWebサービスおよびWebサービス・セキュリティーの逐次処理による小型実装のアーキテクチャー

4.1. 逐次処理による実装のアーキテクチャー

著者らが考える逐次処理による小型実装のアーキテクチャーを図3に示す。基本的な構成は3章と変わらないが、リクエストSOAPメッセージの構築の際にはバイト列を受け渡し、レスポンスSOAPメッセージの解析の際にはSAXイベントを受け渡す点が大きく異なる。また、WS-SecurityエンジンからSOAPエンジンに渡されるSAXイベントにはWebサービス・セキュリティーに関連するものは含まれない。

4.2. 逐次処理によるSOAPエンジン

SOAPエンジンは必要なパラメーターを受け取ると、DOMを構築せずに、あらかじめ用意されたSOAPメッセージのひな型にパラメーターを埋め込むようにして、リクエストSOAPメッセージをバイト列として構築する。またSOAPエンジンはWS-Securityエンジンから受け取ったSAXイベントのみを解析しながら、サービスの回答を抽出し、アプリケーションに渡す。

4.3. 逐次処理によるWS-Securityエンジン

図4はリクエストSOAPメッセージに対するWebサービス・セキュリティーの逐次処理を示したものである。WS-SecurityエンジンはSOAPエンジンから受け取ったリクエストSOAPメッセージのバイト列を解析しながら、Webサービス・セキュリティーの処理を並行して行い、リクエストSOAPメッセージのバイト列を再構築する。以下にWS-Securityエンジンの処理手順を示す。

- リクエストSOAPメッセージを少しずつ読んではSAXイベントを発生する。
- SAXイベントを解析しながら、あらかじめ指定された署名や暗号化の処理対象かどうかを判別する。署名の対象であるならば、署名に必要なダイジェスト値の算出を行う。暗号化の対象であるならば、対象を暗号化した上で得られた暗号化列と置換する。
- すべてのSAXイベントを処理し終わったら、蓄えられたダイジェスト値を包含する署名情報から署名値を算出する。
- リクエストSOAPメッセージのヘッダー部分に、受信側で署

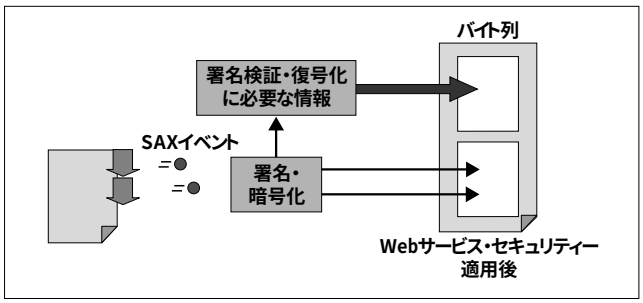


図4. リクエストSOAPメッセージに対するWebサービス・セキュリティーの逐次処理

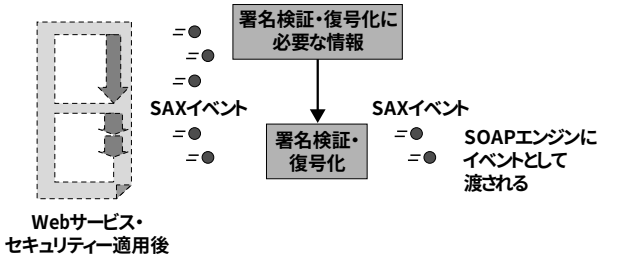


図5. レスポンスSOAPメッセージに対するWebサービス・セキュリティーの逐次処理

名の検証や暗号の復号化に必要な情報（署名検証や復号化の処理対象、署名情報や暗号鍵など）を挿入する。

図5は、レスポンスSOAPメッセージに対するWebサービス・セキュリティーの逐次処理を示したものである。WS-Securityエンジンは受け取ったSAXイベントを解析しながら、レスポンスSOAPメッセージに対するWebサービス・セキュリティーの処理を並行して行う。処理後は回答抽出に必要なSAXイベントだけをSOAPエンジンに渡す。以下にWS-Securityの処理手順を示す。

- 受け取ったSAXイベントを解析しながら、SOAPメッセージに含まれるヘッダー部分を解読し、署名の検証・暗号の復号化に必要な情報（署名検証や復号化の処理対象、復号鍵など）のみを整理してメモリーに蓄える。この際、署名値の検証などの前もって行える処理も併せて行う。
- 引き続き、SOAPメッセージに含まれるボディ部分を解読し、署名検証や復号化の処理対象かどうかを判別する。署名検証の対象であるならば、ダイジェスト値を算出し、署名情報に含まれるダイジェスト値と比較・検証する。復号化の対象であるならば、対象を復号し、得られたメッセージ部分をさらに再帰的に解読する。
- 以上の処理を経た後、レスポンスSOAPメッセージから回答を抽出するのに最低限必要なSAXイベントのみをSOAPエンジンに渡す。

5. 逐次処理による実装のプロトタイプと性能評価

ここでは、筆者らが実装した逐次処理によるSOAPエンジンおよびWS-Securityエンジンのプロトタイプと WSTKMDに搭載されているDOMを用いたSOAPエンジンおよびWS-Securityエンジンの実装の処理時間やメモリー使用量を計測・比較し、筆者らの実装の有効性を実証する。また、市販の携帯電話実機上で実現したSOAPエンジンの処理時間の測定結果を示す。

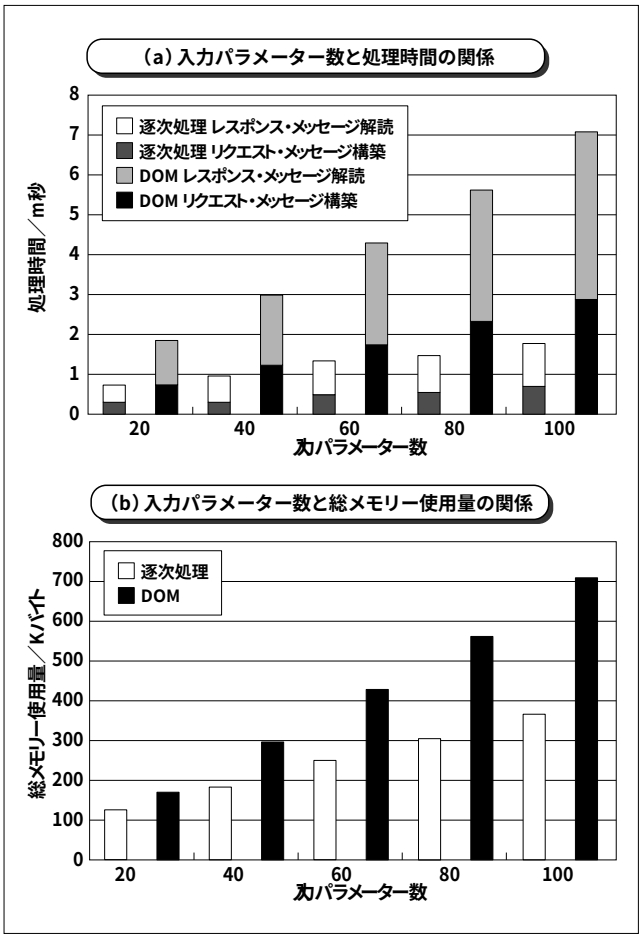


図6. SOAPエンジンのDOMを用いた実装と逐次処理による実装の比較結果

5.1. 逐次処理によるSOAPエンジンのプロトタイプと処理測定結果

筆者らが実装した逐次処理によるSOAPエンジンのプロトタイプのコード・サイズは圧縮jarファイルで71Kバイトである。

筆者らのプロトタイプとWSTKMDに搭載されている中間構造としてDOMを用いるSOAPエンジンについて、処理時間と総メモリー使用量を比較した結果を図6に示す。実験環境はIBM ThinkPad® A31p（CPU：Pentium®4 1.7MHz、メモリー：756Mバイト、OS：Windows®2000）である。本実験では、SOAPエンジンに渡されるパラメーター数を変化させた。

図6（a）のグラフはパラメーター数（X軸）と処理時間（Y軸）の関係を示している。この結果から、リクエストSOAPメッセージの構築およびレスポンスSOAPメッセージの解読のいずれの場合においても、著者らのプロトタイプ（左側の棒グラフ）の方が、DOMを用いた実装（右側の棒グラフ）と比較して、処理速度が2.5～4.0倍に向上していることが分かる。また、図6（b）のグラフはパラメーター数（X軸）と総メモリー使用量（Y軸）の関係を示している。この結果から、著者らのプロトタイプ（左側の棒グラフ）の方が、DOMを用いた実装（右側の棒グラフ）と比較して、総メモリー使用量が14～42%減少していることが

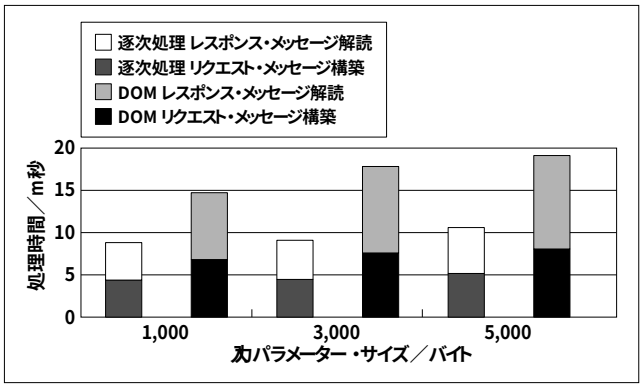


図7. WS-Securityエンジンの DOMを用いた実装と逐次処理による実装の比較結果

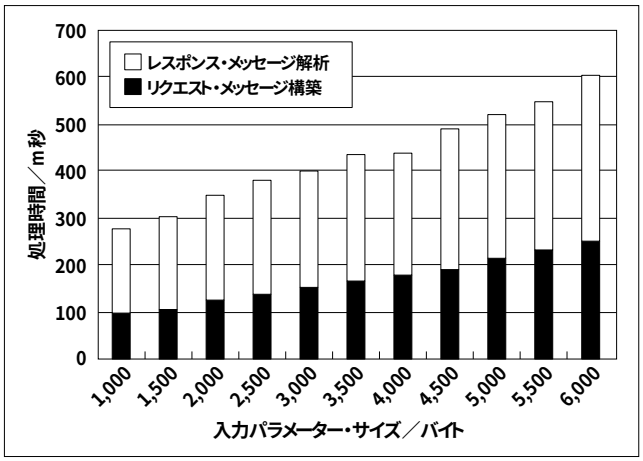


図8. 携帯電話上でのSOAPエンジンの処理時間測定結果

分かる。

5.2. 逐次処理によるWS-Securityエンジンのプロトタイプと処理測定結果

筆者らが実装した逐次処理によるWS-Securityエンジンのプロトタイプのコード・サイズは圧縮jarファイルで83Kバイトである。

筆者らのプロトタイプと、WSTKMDに搭載されている中間構造としてDOMを用いるWS-Securityエンジンについて、処理時間を比較した結果を図7に示す。実験環境は5.1節で示した環境と同一である。本節の実験では、SOAPエンジンに渡されるパラメーターの数を一つだけとし、そのパラメーターのバイト数を変化させた。図7はパラメーターのバイト数（X軸）と処理時間（Y軸）の関係を示している。この結果から、リクエストSOAPメッセージの構築において、著者らのプロトタイプ（左側の棒グラフ）の方が、DOMを用いた実装（右側の棒グラフ）と比較して、処理時間が1.5倍向上していることが分かる。同様に、レスポンスSOAPメッセージの解読においても、処理時間が1.8〜2.0倍に向上していることが分かる。

ここで5.1節と本節での実験の差異について触れておく。5.1節の実験ではSOAPエンジンに渡されるパラメーター数を横軸に取ったが、本節の実験結果では、SOAPエンジンに渡される一つのパラメーターのバイト数を横軸に取った。これはサービス環境の不備によるものであるが、筆者らは今後改善を重ねることで、SOAPエンジンに渡されるパラメーター数がWS-Securityエンジンの性能にどのような影響を及ぼすのかも検証する予定である。

5.3. 携帯電話実機上でのSOAPエンジンの実装と処理測定結果

筆者らはPCのエミュレーター上だけではなく、市販の携帯電話上でもSOAPエンジンの稼働を実験している。

当実験は、5.1節で利用した実装からさらに機能などを減らして作成した35Kバイトの圧縮jarファイルをJavaアプリケーションとして携帯電話にダウンロードして行った。携帯電話上でのSOAPエンジンの処理時間を測定するために、筆者らは同一メッセージをサービスに反復送信し、初回を除いて、2回目以降の処理時間の平均値を算出した。初回の測定時の処理時間を考慮に入れない理由は、処理に必要なライブラリーをメモリー上にロードするのにかかる時間が実験結果に大きく影響すると判断したためである。

SOAPエンジンに渡されるパラメーターの数は一つだけとし、そのパラメーターのバイト数を変化させながら処理時間の変化を測定した結果を図8に示す。図8ではX軸をパラメーターのバイト数とし、Y軸を処理時間とした。この結果から、リクエストSOAPメッセージの構築およびレスポンスSOAPメッセージの解読のいずれの場合においても、処理時間が400m秒に抑えられていることが分かる。これにより、現時点でも市販の携帯電話上でWebサービスを実現するのに実用上問題のない数値が得られることが実証できた。

5.4. 携帯電話実機上でのWS-Securityエンジンの実装

現在市販されている携帯電話上ではWS-Securityエンジンを稼働させることはできない、これは以下の理由によるものである。

- 携帯電話の持つセキュリティー・ライブラリーのAPI（Application Program Interface）が公開されていない。
- ダウンロード可能なアプリケーションのサイズの制約が厳しい。

筆者らは、上記の問題を解決できれば、今後、携帯電話の実機上でWS-Securityエンジンの性能評価を行うことも考えている。

6. おわりに

当論文では、Webサービスおよびwebサービス・セキュリティーをモバイル機器、テレマティックス、オフィス機器などの各種情報機器でも稼働させるために、SAXを用いた逐次処理による小型実装を試みた。また、筆者らのプロトタイプを用いた実験により、中間構造としてDOMを用いた実装と比較して、筆者らの実装の有効性を示した。さらには、市販の携帯電話上で筆者らが実装したSOAPエンジンを動作させることで、現状でもWebサービスを実現する上で問題のない処理時間を示すことができた。これらの結果からも、今後の各種情報機器上でのWebサービスおよびWebサービス・セキュリティーの具現化が大きく期待できる。

[参考文献]
[1] XML 1.0 http://www.w3.org/TR/REC-xml
[2] SOAP Version1.2 http://www.w3.org/TR/soap12/
[3] Web Services Security
http://www-106.ibm.com/developerworks/webservices/library/ws-secure/
[4] WebSphere Application Server 4.0
http://www-6.ibm.com/jp/software/websphere/appserv/
[5] WebSphere Application Server 5.0
http://www-6.ibm.com/jp/software/websphere/wv5/
[6] M. Weiser. Some computer science problems in ubiquitous computing
Communications of the ACM, 36(7):74-85, July. 1993
[7] Simple API for Xml (SAX) http://www.saxproject.org/
[8] Sun Microsystems, Java 2 Platform, Micro Edition(J2ME)
http://java.sun.com/j2me
[9] Sun Microsystems, the Connected Limited Device Configuration(CLDC)
http://java.sun.com/products/cldc/
[10] Document Object Model (DOM) http://www.w3.org/DOM/
[11] Web Services ToolKit for Mobile Devices
http://www.alphaworks.ibm.com/tech/wstkmd/
[12] WebSphere Device Developer
http://www-3.ibm.com/software/wireless/wsdd/
[13] SOAP Message Security: Minimalist Profile
http://lists.oasis-open.org/archives/wss/200303/pdf00002.pdf