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

2007－ＤＤ－62（10）
2007／7／27

End-to-End のセキュリティにおけるポリシー伝達の課題

荒本 道隆†

†アドソル日進（株） 生産技術部 〒108-0075 東京都港区港南 4－1－8 リバージュ品川
E-mail: †Aramoto.Michitaka@adniss.jp

あらまし Web サービス呼び出しでは WS-Security によってメッセージ層でのセキュリティが実現されている．また，WS-Security での暗号化・署名といったセキュリティに関するポリシーを記述には，WS-Policy や WS-SecurityPolicy という仕様がある．しかし，これらを組み合わせてマルチホップで end-to-end のセキュリティによってシステムを運用しようとした場合，特に「ポリシーの伝搬」において多くの課題がある．本研究では，それらの課題について検討を行う．

キーワード XML，Web サービス，セキュリティ，セキュリティポリシー，XML 暗号，

Problem of Policy Transmission in End-to-End Security

Michitaka ARAMOTO†

† Productive Engineering Dept., Ad-Sol Nissin Corp. 4-1-8 Kounan Riverge, Shinagawa, Minato-ku, Tokyo 108-0075 Japan
E-mail: †Aramoto.Michitaka@adniss.jp

Abstract In the Web service call, security in the message layer has been achieved by WS-Security. The policy description concerning security like the encryption and the signature with WS-Security can be done with using the specifications named WS-Policy and WS-SecurityPolicy. However, there are a lot of problems in combining these technologies to operate multi-hop systems with end-to-end security, especially in the "Propagation of the policy". In this research, those problems are examined.

Keyword XML, Web Services, Security, Security Policy, XML Encryption,

1. はじめに

ブラウザを使った Web ブラウズでセキュリティが必要とされる場面では https が広く使われている．https では PKI(Public Key Infrastructure)が使われており，SSL(Secure Socket Layer)/TLS(Transport Layer Security)によって point-to-point のセキュリティが確保されている．また http 以外でも，TCP による telnet や ftp などでも SSL を使うことで安全に通信ができる．

Web サービスのセキュリティでは，WS-Security が使われている．Web サービスで https を使うこともできるが，ブラウザと同様に SSL/TLS となり point-to-point のセキュリティしか実現できない．WS-Security を使った場合は，MLS(Message Level Security)となり，end-to-end のセキュリティが確保される．

WS-Security で使われている XML Encryption や XML Signature では，部分暗号化・部分署名を使って『必要な部分だけにセキュリティをかける』ことができる．また，メッセージに対するセキュリティであるためにトランスポート層の影響を受けないので，http 以外のプロトコルを使うことや，セキュリティをかけたままメッセージを保存することもできる．

セキュリティを確保する上では，「どの部分を暗号化・署名するのか？」「誰の鍵で暗号化・署名するのか？」が重要な要素である．特に，２者間ではなく，３者間以上のマルチホップでやり取りすると，「複数サイトのセキュリティポリシー」「PKI を使う上での課題」も考慮しなければならない．また，マルチホップで通信を行う上で，セキュリティのポリシーをどうやって伝達するかが非常に重要になる．ポリシー自体が改ざんされると，改ざんされたポリシーに従ったセキュリティは無意味だからである．

本稿では，マルチホップの Web サービス呼び出しにおいて，WS-Security を用いて end-to-end のセキュリティを実現する上でのポリシーを伝達する場合の課題について検討する．

セキュリティには，第三者に盗み見られたり改ざんされたりしないようにするための暗号化と，改ざんを防止したり偽造を防止したりするための署名の２つがあるが，本稿では主に PKI による暗号化について扱う．

2. TLS のセキュリティ

TLS によるトランスポート層のセキュリティの例として，https による point-to-point のセキュリティを挙げる．https では，PKI という公開鍵暗号方式を使って暗

号化・署名により，認証をおこない，盗聴や改ざんを防止している．

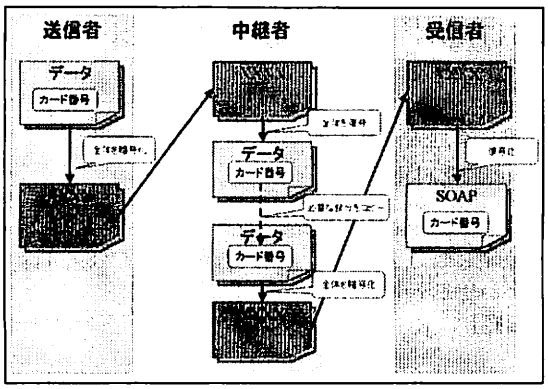

図1　トランスポート層のセキュリティ

https を使って通信するには，通信の開始時にクライアントはサーバからサーバ証明書を受信し，クライアント側で保持しているサーバ証明書や CA 局の証明書を使って受け取った証明書を検証する．次に，必要であればクライアントからクライアント証明書を送付し，サーバ側でも同様に受け取った証明書を検証する．広く利用されているブラウザによるサーバ認証では，図2のようにブラウザにはルート証明機関の証明書があらかじめ入っているので，ルート証明機関から発行された証明書は個別にサーバや CA 局の証明書をインポートする手間無しに認証できる．

証明書を確認することで認証が完了すれば，以降は通信内容に対して証明書に入っている公開鍵を使って，暗号化と署名をおこなう．

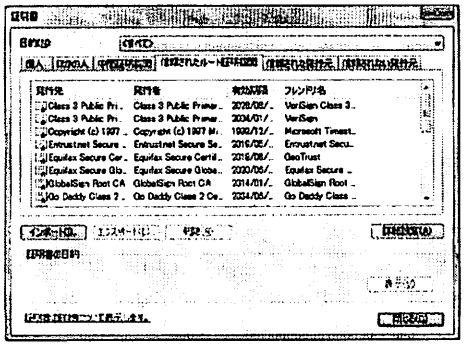

図2 ブラウザのルート証明機関の証明書

TLS で，クライアントとサーバ間でやり取りするセキュリティに関する情報は，以下の通りである．

・使用する暗号化・署名のアルゴリズム
・サーバ証明が必要か？
・サーバの証明書
・クライアント証明が必要か？
・クライアントの証明書

TLS では通信内容全体に対して暗号化・署名をおこなうために，受信した段階で復号・署名検証を行ってしまう．通信とアプリケーションは独立しているので，アプリケーションは暗号化されていることを一切考慮しないで済むが，「必ずすべてを復号してしまう」ので，マルチホップで通信する場合には，各中継者でいったんすべてが復号された状態になる．また，トランスポート層以外では使えないので，「暗号化したまま DB に格納したい」「保存しておいて，あとで署名を確認したい」という用途にも使えず，他のセキュリティソリューションを組み合わせて利用する必要がある．

3. MLS のセキュリティ

TLS は通信経路での盗聴に対しては強固であるが，各サーバ上ではメッセージ全体が復号されて無防備な状態になってしまう．そのため，各サーバ上やデータの保存先からであれば容易にデータが盗み出せてしまう．そこで，MLS によるメッセージ層のセキュリティが必要になる．

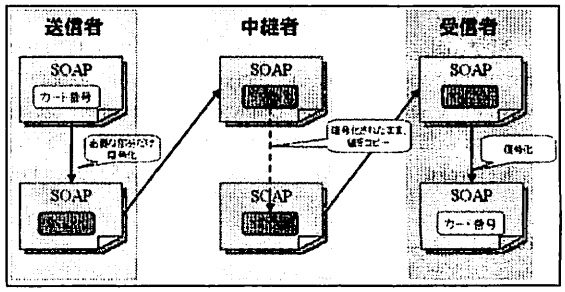

図3　メッセージ層のセキュリティ

MLS によるメッセージ層のセキュリティの例として，SOAP の WS-Security による end-to-end のセキュリティを挙げる．SOAP では http 以外にも ftp,smtp などを利用できるが，WS-Security はトランスポート層に依存しないので，どのような通信方法でも利用できる．

WS-Security では，SOAP メッセージに対する暗号化・署名の方法のみの仕様であり，セキュリティに関するポリシーのやり取りは含んでいない．

セキュリティに関するポリシーを交換するタイミングは，事前と実行時の2つに分けられる．

事前にセキュリティに関するポリシーを交換すると，アプリケーションのスキーマ，ソースコード，設定ファイルなどに，暗号化・署名に関する記述をおこなうことができる．この場合，ポリシーは静的なコードに埋め込まれるので，ポリシーの変更時には再コン

パイルや設定ファイルの修正により，アプリケーションの再起動が必要になる．また，異なるポリシーを要求する相手が複数ある場合，WS-Security ではポリシーの内容を伝えることはできないので，ポリシーの種類ごとに Web サービスのエンドポイントを変更するなどの対応が必要である．

実行時にセキュリティに関するポリシーを交換すると，Web サービスは動的なポリシーの変化に対応する必要がある．そうすることで，ポリシーの変化に強くなるので，異なるポリシー同士での通信も容易に行える．

Web サービスのセキュリティポリシーを定義する WS-Policy と WS-SecurityPolicy では，以下の内容を記述する．

・使用する暗号化・署名のアルゴリズム
・暗号化・署名対象の要素
・セキュリティポリシーの使用条件
・サーバの証明書
・クライアントの証明書

TLS との最大の相違点は「暗号化・署名対象の要素」が指定できる事である．ただし，「この要素に対しては，誰の鍵を使用する」は指定できない．そのため，２者間であれば問題ないが，３者以上のマルチホップの場合，誰の鍵を使うかという課題がある．

4. セキュリティのポリシー

TLS では，通信の開始時にセキュリティポリシーを交換し，双方が満足する方法で通信を行う．point-to-point のセキュリティであるので，送信者と受信者の２者のセキュリティポリシーを満足すれば十分である．

しかし MLS では，end-to-end のセキュリティであるために，送信者は直接通信する相手だけではなく，全ての中継者と受信者のセキュリティポリシーを満足する必要がある．そのため，直接通信する相手とだけセキュリティポリシーを交換するだけでは不十分で，そのメッセージを配信するすべてのサイトのセキュリティポリシーを手に入れる必要がある．しかし，常にリアルタイム通信とは限らないので，すべてのサイトがオンラインでない場合もありえる．

5. 静的なポリシー伝達

事前にセキュリティに関するポリシーをやり取りしておけば，暗号化された状態を前提として開発することができる．仕様書などによってポリシーを決定し，開発者は「その場所は暗号化されていること」を前提とした実装とできるので，自分のところで暗号化・復号しなければ SOAP ヘッダを含めて次に渡せさえすれば良い．

ただし，Java や.net のような，XML をインスタンスにマッピングする方式で開発する場合には，いくつか注意しなければならない．まず，WS-Security のセキュリティに関する情報の一部は SOAP メッセージのヘッダ部に記述されるが，Web サービスの実装によっては SOAP ヘッダを操作するには特別なコードが必要となる．また，異なるポリシーを要求する相手が複数ある場合には，名前空間に関する問題が発生する．セキュリティポリシーをスキーマに組み込んでも，名前空間は変更しない．そのため，「構造の異なるスキーマが同じ名前空間を持つ」ということになるので，XML の名前空間を Java 言語の package などにマッピングすると，構造が違うのに自動生成されるのは図４と図５のように同じクラス名になる．

また，何らかの理由でセキュリティポリシーを変更する場合には，すべての関連サイトのソースコードや設定ファイルに埋め込まれたポリシーを変更しなければならないので，非常にインパクトが大きい．

```
public class CreditCardInformation  implements java.io.Serializable {

    private java.lang.String creditCardAuthority;
    private java.lang.String CreditCardNumber;
    private org.apache.axis.types.YearMonth expireDate;
    private java.lang.String cardHolderName;
     :
```

図４　元の Java ソース

```
public class CreditCardInformation  implements java.io.Serializable {

    private java.lang.String creditCardAuthority;
    private org.w3.www._2001._04.xmlenc.EncryptedDataType encryptedData;
    private org.apache.axis.types.YearMonth expireDate;
    private org.w3.www._2001._04.xmlenc.EncryptedDataType encryptedData2;
    private java.lang.String cardHolderName;
     :
```

図５　セキュリティを反映した Java ソース

6. 動的なポリシー伝達

TLS のように通信の開始時にセキュリティに関するポリシーを伝達して動的に対応できれば，１つの Web サービスで異なるポリシーに対応することも可能であり，より汎用的な使い方ができる．

Web サービスにおけるポリシーの記述方法として，WS-Policy と WS-SecurityPolicy がある．WS-Policy ではポリシーのルールを記述し，WS-SecurityPolicy ではセキュリティに関するポリシーを記述する．図６に WS-Policy と WS-SecurityPolicy の例を示す．すでに使用可能な WS-Policy を使った実装として WSIT（Web Services Interoperability Technologies）がある．WSIT では通信の開始時に WSDL を要求し，通信相手は WS-Policy と WS-SecurityPolicy を含んだ WSDL を返し

ている．

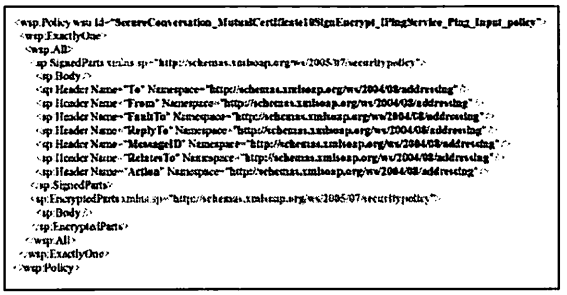

図 6　WS-Policy と WS-SecurityPolicy の例

7. 複数サイト間の動的なポリシー伝達

end-to-end のセキュリティにおいて動的なポリシー伝達をする上で，複数サイト間でおこなうには様々な課題がある．送信者は，すべてのサイトのセキュリティポリシーを満足するために，それぞれのポリシーをマージし，送信者はマージしたポリシーを適用する必要がある．そして，正しいセキュリティポリシーを適用するためには，そのポリシー自体が改ざんされていないかの検証も必要である．ポリシーが改ざんされてしまうと，送信するデータ中の「暗号化は必須」の要素を「暗号化は不要」とされてしまう危険がある．

ポリシー自体の署名の検証には，以下に注意する必要がある．

・署名検証のためのＣＡ局やサーバ証明書
・中継者でのポリシーのマージ方法

送信者はＣＡ局やサーバ証明書によって，中継者や受信者の署名検証ができる状態になっているはずである．そのため，中継者は「送信者が検証できる署名」をすることができる状態であると言える．中継者は，受信者のセキュリティポリシーの署名を削除して自身の署名をすることも可能であるため，「中継時にセキュリティポリシーを改ざんされていない」ということを確認するためには，単に署名を検証するだけではなく，受信者の署名であることを確認する必要がある．

また，Web サービスでは，アグリゲーションやマッシュアップと呼ばれる，複数の Web サービスの中から選択して使用するという利用方法がある．沢山ある受信者から選択するには，送信者が選択する場合と，中継者が選択する場合に分けられる．

送信者が受信者を選択する場合には，送信者が受信者を知っているという事なので，セキュリティポリシーや公開鍵も直接取得できるはずである．そのため，正しいセキュリティポリシーや公開鍵を入手できる．

一方，中継者が受信者を選択する場合には，中継者が自由に受信者を選べるようにするためには，セキュリティ上さまざまな課題が存在する．今後も Web サービスは増え続けていくと予想されるが，そこでセキュリティが必要なデータを扱うためには，この課題を解決しなければならない．

8. ポリシーを中継する場合の課題

中継者が自由に受信者を選べるようにし，各受信者が自由にセキュリティポリシーを定義して異なる公開鍵を持つ場合には，セキュリティ上さまざまな課題が発生する．

受信者が多数ある場合には，送信者は受信者のセキュリティポリシーや公開鍵を事前に入手しておく事ができず，受信者の情報を入手するには中継者に頼らなければならない．

送信者が受信者の情報を中継者経由で得るには，以下の方法が考えられる．

1．受信者のポリシーや公開鍵を受け取る
2．受信者のポリシーや公開鍵の URL を受け取る
3．送信者が公開鍵を渡し，受信者がその公開鍵で暗号化して返したものを中継者経由で受け取る

1は，図７のようにポリシー自体を中継者が経由して伝達すると，中継者にポリシーを改ざんする機会を与えてしまう．ポリシー自体に署名を付けても，その署名が受信者のものであるか検証できなければ，中継者による改ざんを防ぐことができない．

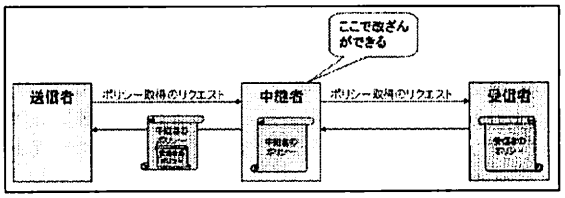

図７　中継者がポリシーを中継

2は，図８のようにポリシーを直接やり取りするので，ポリシー自体を中継者が改ざんすることはできない．しかし，中継者が偽の URL を送付することで，偽のポリシーを送信者に渡す事ができてしまう．

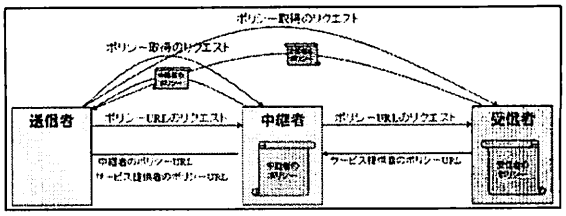

図８　URL を通知

3は，図９のように受信者の公開鍵で暗号化すれば，中継者はポリシーの中身を見ることすらできない．しかし，送信者の公開鍵を送る途中で中継者の公開鍵に

すり替えてしまえば，受信者は偽の公開鍵で暗号化して返してくるので,それを中継者は復号して改ざんし，再び送信者の公開鍵で暗号化すれば，送信者は暗号化された偽のポリシーを受け取ってしまう．

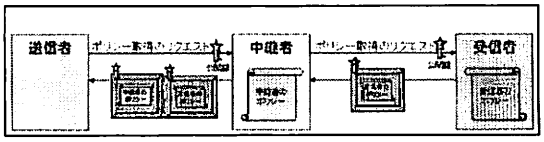

図９　ポリシーを暗号化

どの方法を使っても，送信者が直接やり取りする相手が中継者だけである限り，中継者が悪意をもってポリシーや証明書を改ざんしようとすると出来てしまうという問題がある．

9. 信頼モデル

end-to-end のセキュリティを実現するためには，「どこを信頼するか？」を選択する必要がある．

１．中継者を信頼する
２．受信者を信頼する
３．第三者を信頼する

１は，中継者がポータルサイトで，受信者が出店者の場合などのモデルである．受信者である出展者の数が非常に多いために，送信者がすべての受信者を把握するのは無理がある．受信者は中継者と契約をおこなう．８章で述べたように，証明書や公開鍵を中継者経由で渡そうとすると，中継者で何らかの不正な操作を行える余地ができてしまう．しかし，受信者と中継者はビジネス上の契約を取り交わしているのであれば，それなりに信頼できる状態にあると言える．そこで，「送信者のログ」などによって不正があったか確認できれば，運用上は十分である場合が多いだろう．

２は，受信者がカード会社であるような，受信者の数が限られた場合に有効なモデルである．送信者が全受信者のポリシーと公開鍵を事前に入手しておくことが容易であり，送信者と受信者が契約をおこなう．そのため，受信者のポリシーや公開鍵が途中で改ざんされる心配がない．

３は，Web サービス用の認証局のようなものを新設するモデルである．ブラウザで https が容易に利用できるのは，始めからブラウザにルート証明書が入っており，各サイトはそのルート証明書の認証局から発行された証明書をブラウザに渡し，ブラウザでサイトの証明書の検証ができるからである．しかし、マルチホップの Web サービスの場合は，中継者と受信者の区別が付かない事が混乱の原因になってしまう．その２つを区別できるような認証局があれば，問題が解決できるかもしれない．

10. まとめと考察

WS-Policy や WS-SecurityPolicy の仕様だけでは，動的なポリシー伝達において，中継者の盗聴や改ざんを完全に防止するような end-to-end のセキュリティは実現できない．実システムを構築するにあっては，「どこを信頼するか？」によってポリシー伝達のモデルによっては盗聴を完全に防止できないかもしれないが，盗聴されていないことをログによって証明することで十分かどうかも検討する．

つまり，end-to-end のセキュリティを実現する上では，「中継者が盗聴・改ざんしたくてもできない」「中継者が盗聴・改ざんをしなかったことが証明できる」の２つは区別して考え，最適な方を選択する必要がある．そして，中継者をどこまで信頼するのかについては，契約も含めて技術以外の観点でも注意しなければならない．

WS-Security は一部では使われているが，まだまだ https ほど多くの場面で使われているようにはなっていない．point-to-point のセキュリティであれば TLS の https で十分だが，end-to-end のセキュリティが必要な場合には，特にマッシュアップのように多数の受信者を積極的に利用するには，サイトごとに異なるセキュリティポリシーを安全に解決できれば安心して使えるようになるだろう．

データのスキーマに，最初からセキュリティに関する要素を入れておくことで，ポリシーを統一する事もある程度は可能だろう．しかし，暗号化のための公開鍵をどうやって渡すのか，その公開鍵が改ざんされていないことをどうやって証明するのか，という課題は残る。

また，暗号化されたデータについても，「鍵さえなければ復号する事はできない」のではなく，「鍵がなければ解読には非常に時間がかかるので，実際には悪用できない」だけであるので，中継者は暗号化されたデータといえども取り扱いには十分に注意する必要がある．

11. 謝辞

本検討は，XML コンソーシアム応用技術部会とセキュリティ部会の合同で実施した sPlat プロジェクトにおいて検討したものをまとめたものです．検討には，アドソル日進株式会社，株式会社内田洋行，日本電気株式会社,株式会社ノムラシステムコーポレーション，株式会社日立製作所，PFU アクティブラボ株式会社，富士ゼロックス株式会社，キヤノン株式会社，株式会社ＪＩＥＣ，東京エレクトロン株式会社，株式会社ネット・タイム，富士通株式会社が参加しました．参加された皆さまに心から感謝いたします．

文　献

[1] K.Nakayama, T.Ishizaki, and M.Oba, "Application of Web Services Security Using Travel Industry Model," SAINT 2005 Workshop, pp 358-361, 2005.

[2] 中山弘二郎，植田良一，大場みち子，"XML 暗号による部分暗号化後のスキーマ検証方法"，電気学会 情報システム研究会，IS-06-06，pp.31-36，2006.

[3] 中山弘二郎，荒本道隆，大場みち子，"暗号化 XML データのスキーマ検証方式の提案",情報処理学会研究報告，2006-DD-056，Vol.2006，No.83，pp.31-37 (2006)

[4] 荒本道隆，中山弘二郎，大場みち子，"XSLT を用いた暗号化 XML データのスキーマ検証方式の実装",情報処理学会研究報告，2006-DD-056，Vol.2006，No.83，pp.39-45 (2006)

[5] 中山弘二郎，大場みち子，荒本道隆，藪田慈久，"マルチホップ Web サービスにおける部分暗号化データの処理方式"，電学論 C，Vol.127，No.6，pp.951-956，2007

[6] W3C, "SOAP Version 1.2", W3C Recommendation (2007), http://www.w3.org/TR/soap/

[7] W3C: "XML Encryption Syntax and Processing", W3C Recommendation (2003), http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/

[8] W3C, "Web Services Description Language (WSDL) 1.1", W3C Note (2001), http://www.w3.org/TR/2001/NOTE-wsdl-20010315

[9] OASIS, "Web Services Security: SOAP Message Security 1.0 (WS-Security 2004)", OASIS Standard (2004), http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf

[10] W3C, "Web Services Policy 1.5 - Framework", W3C Working Drafts(2006), http://dev.w3.org/cvsweb/~checkout~/2006/ws/policy/ws-policy-framework.html

[11] OASIS, "WS-SecurityPolicy 1.2", OASIS Committee Specification (2007), http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.2/ws-securitypolicy.html

[12] WSIT, "wsit: Project Tango"(2007), https://wsit.dev.java.net/

[13] XML コンソーシアム，「TravelXML 利用 Web サービス実証実験プロジェクト 成果 公開資料」(2004)，http://www.xmlconsortium.org/koukai/travelxml-p/

[14] XML コンソーシアム，「TravelXML 仕様公開ページ」，http://www.xmlconsortium.org/wg/TravelXML/TravelXML_index.html

[15] XML コンソーシアム，「sPlat プロジェクト プレスリリース」(2006)，http://www.xmlconsortium.org/release/pdf/px060406-security-project-final2.pdf