【TravelXML利用Webサービス実証実験プロジェクト成果資料】

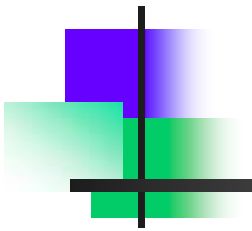
TravelXMLを利用したWebサービス実証実験デモ

デモ④：セキュリティ

東京エレクトロン株式会社　松永豊
株式会社日立製作所　中山弘二郎
アドソル日進株式会社　荒本道隆
日本アイオナテクノロジーズ株式会社　片山　良雄
日本IBM株式会社　吉田 忠行

アジェンダ

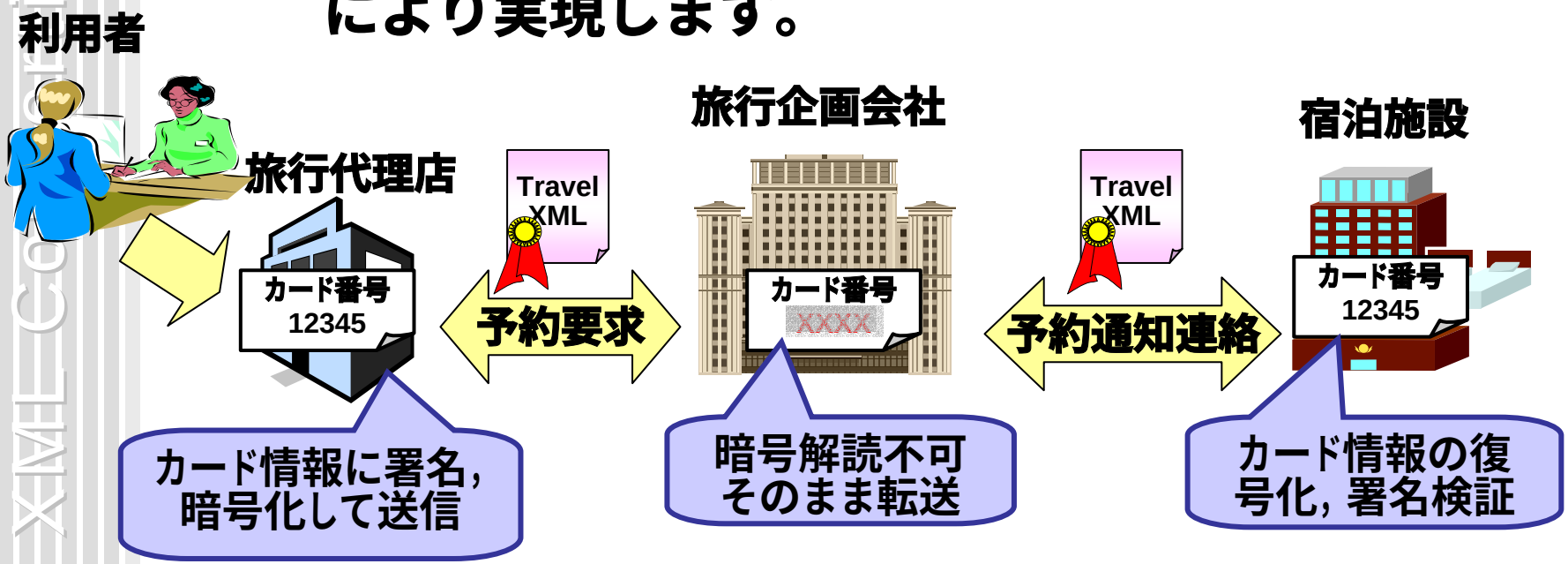
- 実証実験概要
- デモ④セキュリティ検証パターン
- 参加企業による実装詳細説明
  - 東京エレクトロン株式会社
  - 株式会社日立製作所
  - アドソル日進株式会社
  - アイオナテクノロジーズ株式会社
  - 日本IBM株式会社



実証実験概要

デモ4：セキュリティの利用

- 予約情報・個人情報の安全性の確保
  - 旅行の予約では個人の情報を扱う場合が多く、取引する情報の高い安全性の確保が必要になる。
  - 電子商取引に必要な否認防止、情報改竄防止、秘匿性の確保を署名認証技術や部分暗号化技術により実現します。



デモ④セキュリティ の目的

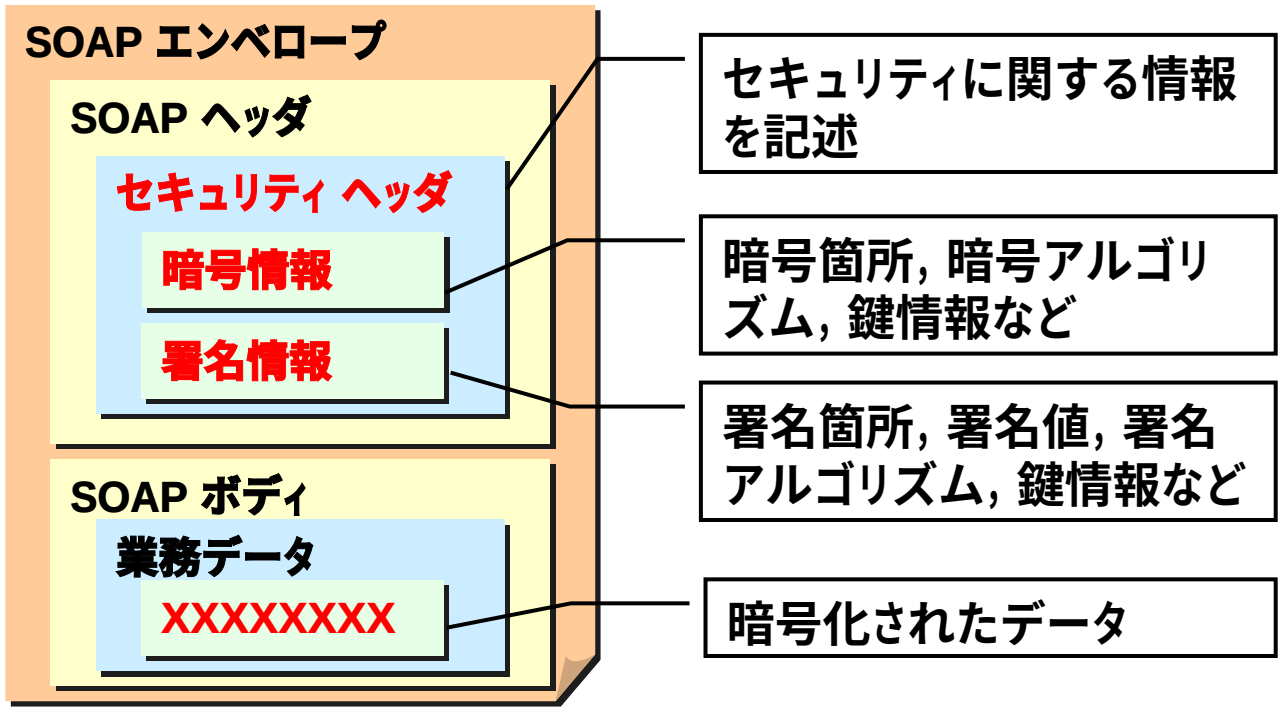
- ビジネスシナリオベースでWebサービスのセキュリティを検証
- Webサービスにおける主なセキュリティ要件
  - 秘匿性確保，認証，改ざん検出，否認防止
- SSLなどの既存技術を利用することで・・・
  - 通信路のセキュリティを確保することはできる
  - ただしEnd to Endのセキュリティは確保できない
    - 中継者に全ての情報が開示されてしまう
    - 宿泊施設が旅行代理店を認証することができない

本実証実験では，WS-Securityを用いた
End to Endのセキュリティの検証を実施

SSL：Secure Socket Layer，WS-Security：Web Services Security

WS-Security概要

- Webサービスのメッセージレベルのセキュリティに関する仕様
- SOAPメッセージに対する暗号，署名の方法を規定

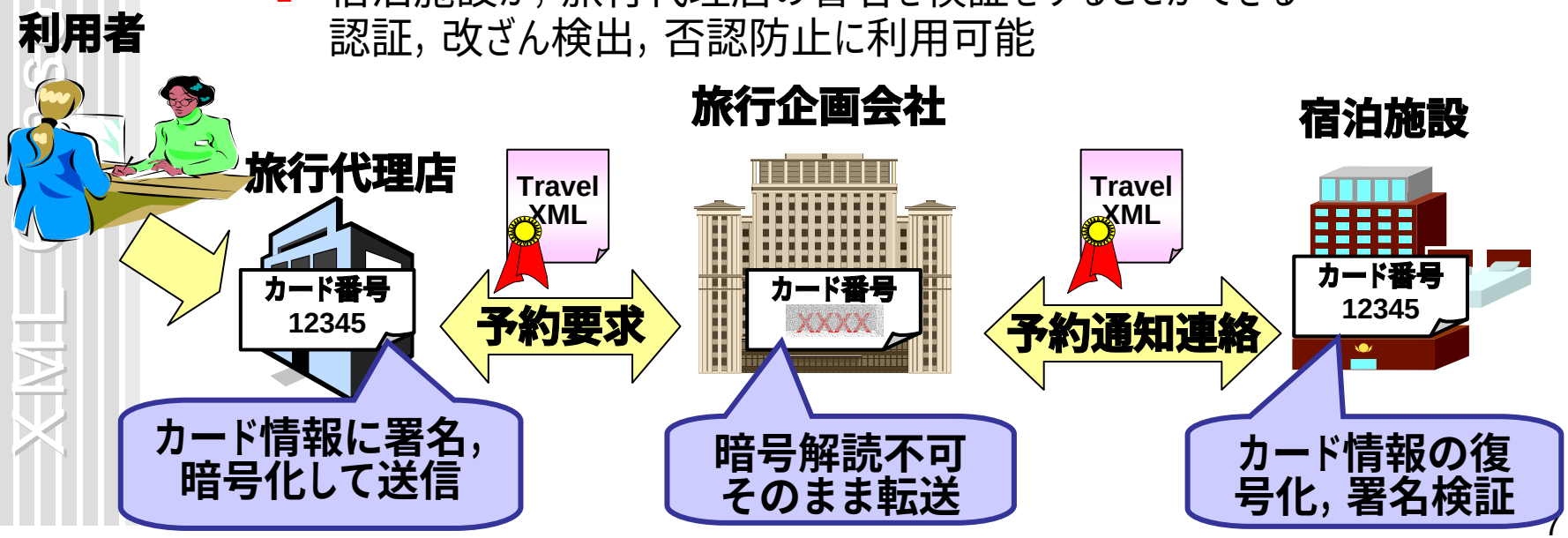


- 標準化団体OASISで仕様を策定
  2004年4月にv1.0が標準仕様として承認

OASIS：Organization for the Advancement of Structured Information Standards

セキュリティ・シナリオの概要

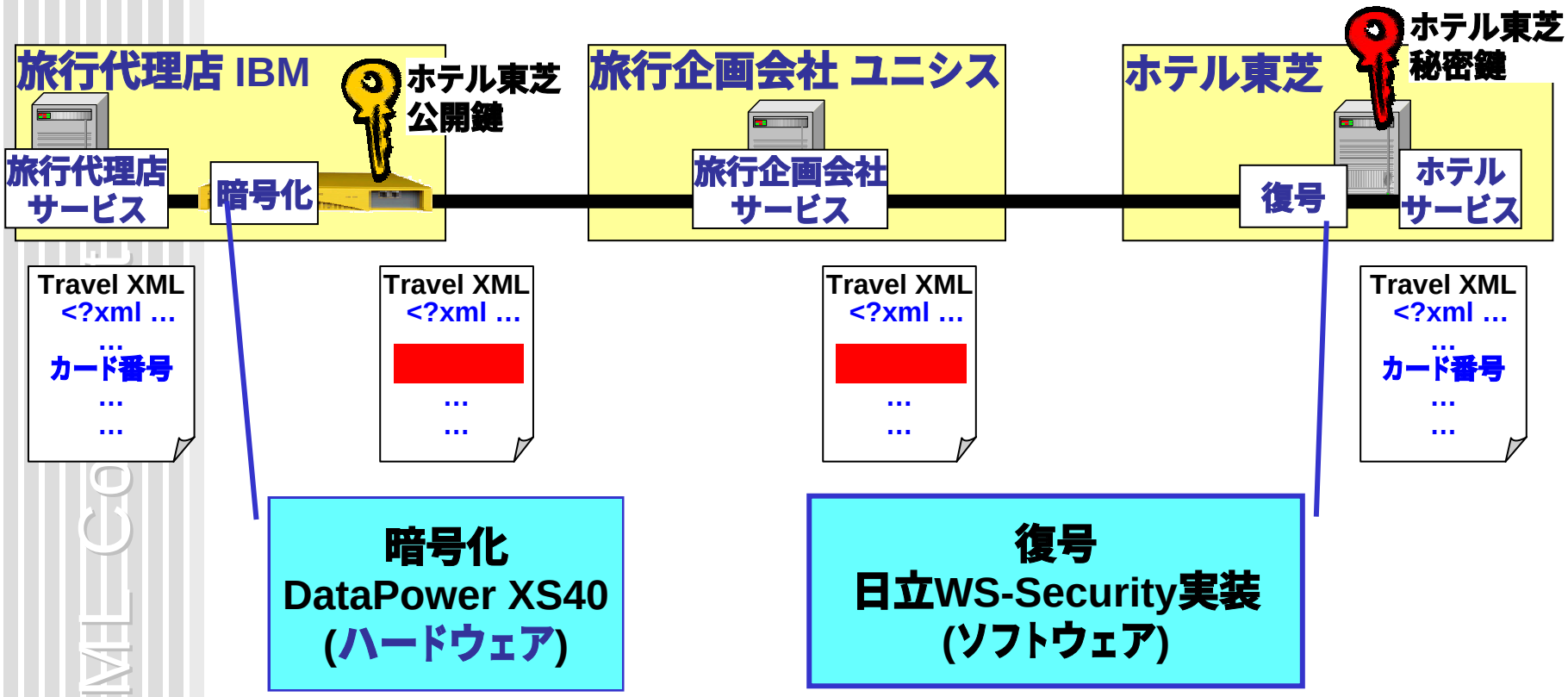
- **デモシナリオの想定**
  - エステ料金の支払い保証の為にクレジットカードを利用
    旅行代理店は，旅行企画会社を介して宿泊施設にカード情報を送信
    旅行企画会社はカード情報を知る必要はない
  - カード番号の部分暗号，部分署名を行なう
- **メリット**
  - 宿泊施設だけにカード情報を開示することが可能
    旅行企画会社は不要な機密情報の管理責任を負う必要がない
  - 宿泊施設が，旅行代理店の署名を検証をすることができる
    認証，改ざん検出，否認防止に利用可能



デモ④セキュリティ検証パターン

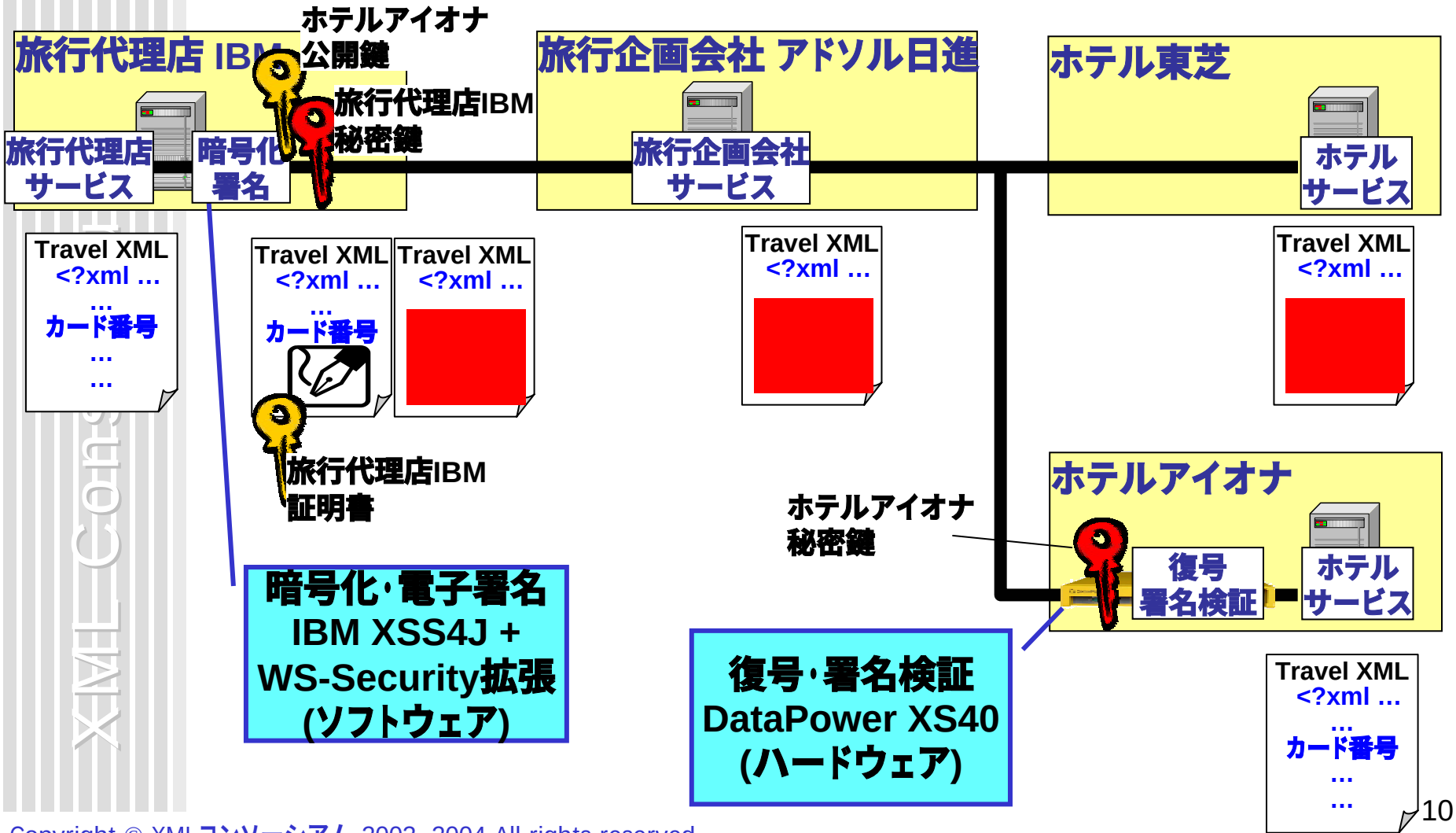
デモ④セキュリティ その1

- ハードウェア装置で暗号化しソフトウェアで復号



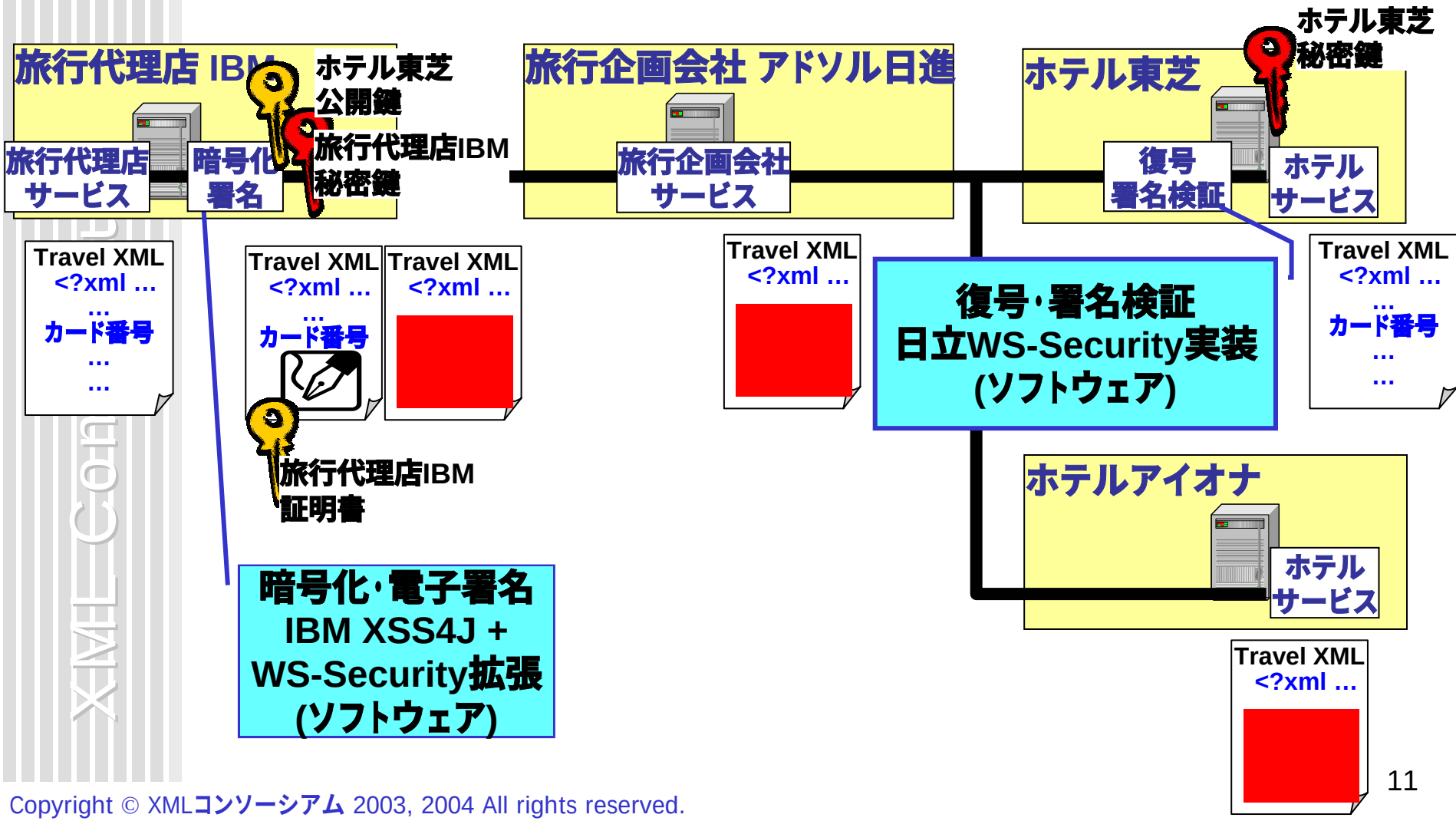
デモ④セキュリティ その2

- ソフトウェアで暗号化・署名しハードウェアで復号・署名検証



デモ④セキュリティ その3

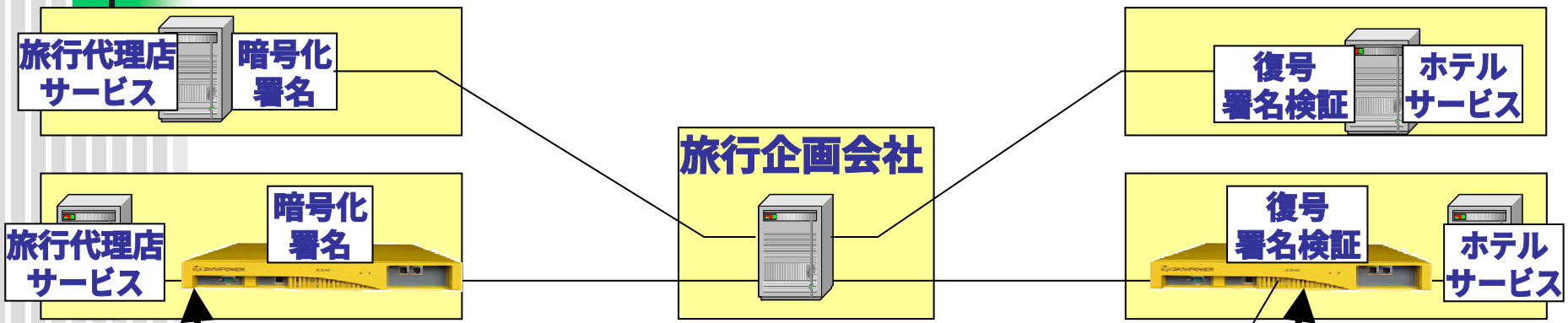
- ソフトウェアで暗号化・署名しソフトウェアで復号・署名検証



東京エレクトロン株式会社

松永 豊

東京エレクトロン – セキュリティ



セキュリティ機能をハードウェアで提供

- **暗号化･復号と電子署名･検証**
- **ネットワーク上のゲートウェイとして機能**

DataPower XS40
XMLセキュリティ･ゲートウェイ

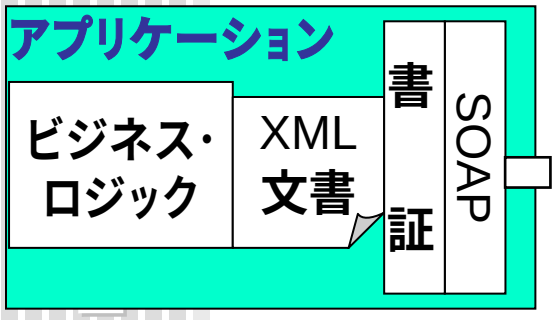
- **XMLファイアウォール**
- **XML暗号化･復号**
- **XML電子署名･検証**
- **スキーマ検証**
- **XSLT変換**

- 開発の記録
  - アプリケーション開発は無し
  - 調査、設定ファイルカスタマイズ、接続テスト
    (3/25～ 約5人日?)
- 苦労した点
  - **WS-Securityバージョンや実装上相違点の調整**
    (ゲートウェイは設定ファイルのカスタマイズで対応)

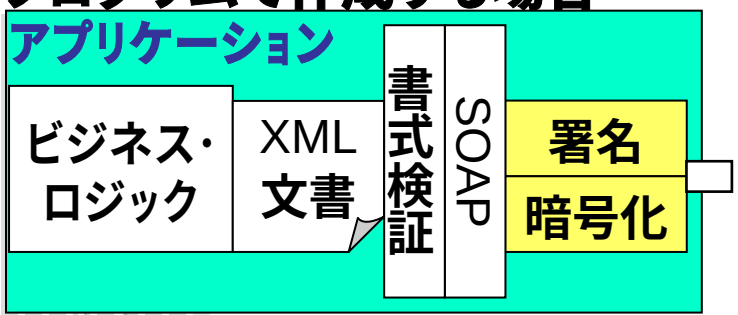
東京エレクトロン

ゲートウェイによるセキュリティ

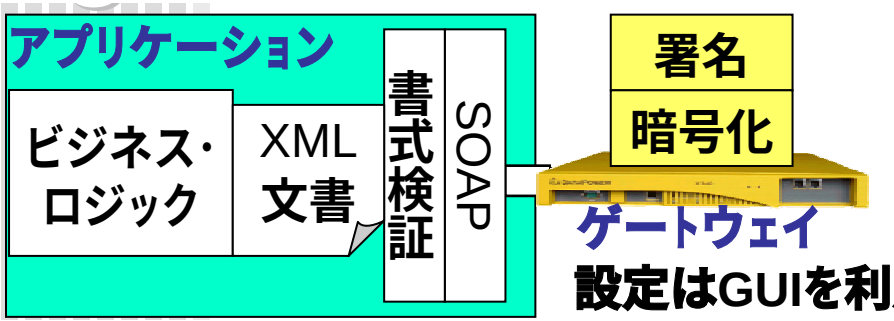
●セキュリティ無しの場合



●プログラムで作成する場合

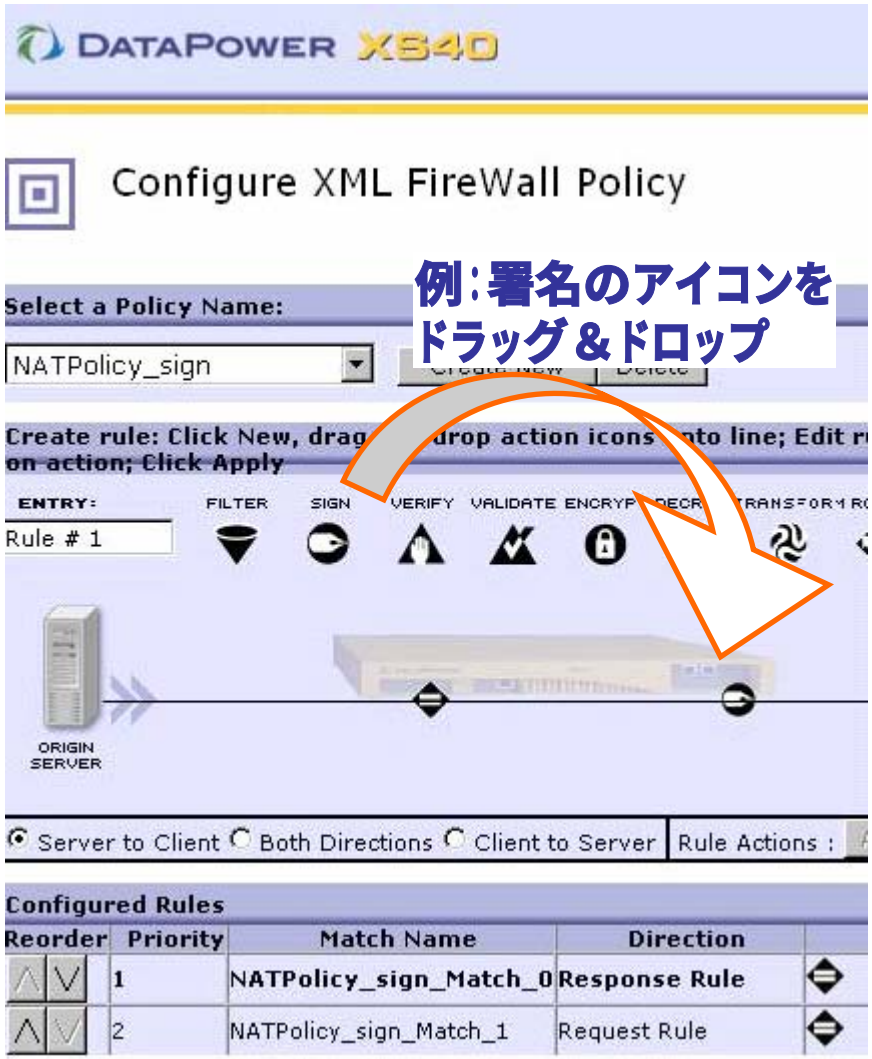


●ゲートウェイを利用する場合



ゲートウェイ

設定はGUIを利用→



株式会社日立製作所

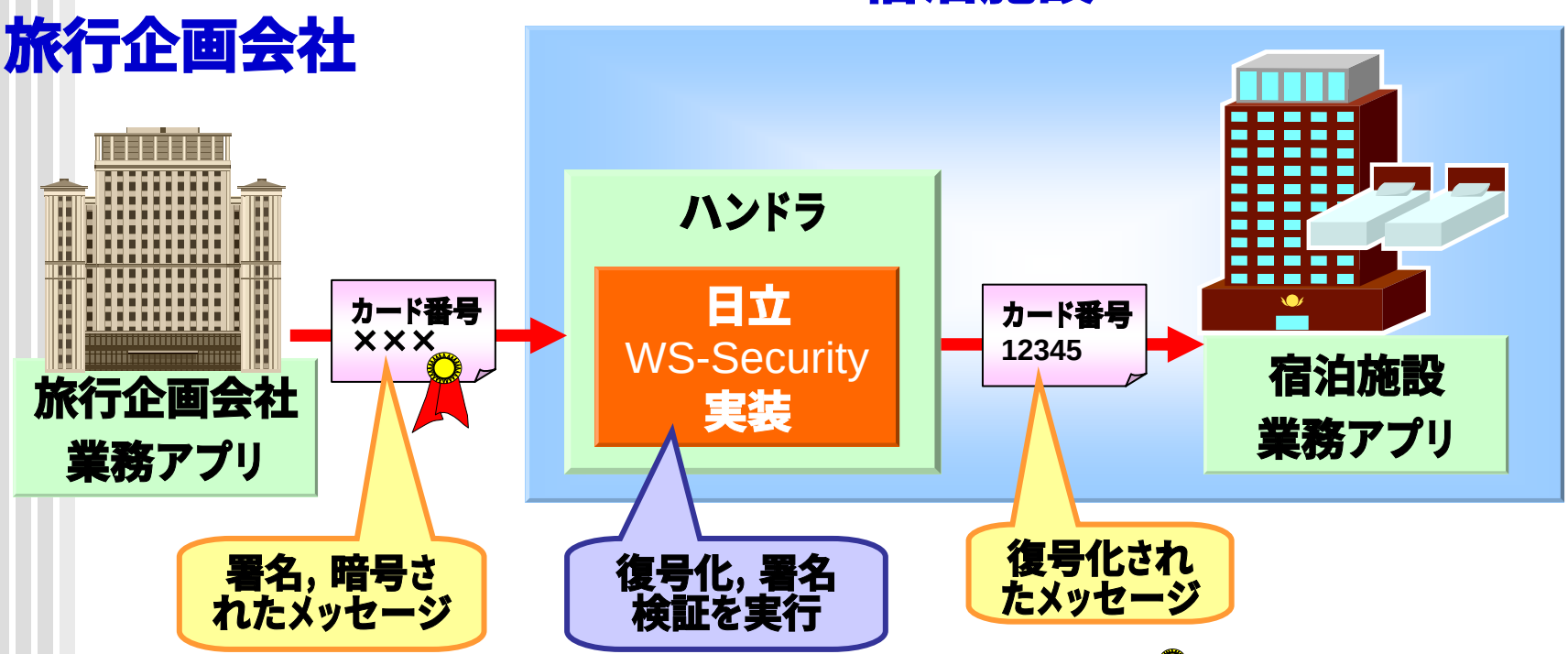
中山 弘二郎

概要

- WS-Security処理を行なうライブラリを実装
  - 部分暗号/復号
  - 部分署名/検証
  - X.509証明書付与/取得
- “Apache-XML-Security-J 1.0.5D2”を利用
  (XML署名，暗号ライブラリ)
- 独自APIを用いて実装
  (WS-Securityの標準APIはまだ存在しない)
- 開発工数　：　約 0.5人月
- ステップ数　：　約 1.2 k ステップ (コメント除く)

宿泊施設での利用例

- SOAPハンドラに組み込んで利用
- 業務アプリの処理前にメッセージを取得し，復号化，署名検証を行なう
- 復号化されたメッセージを業務アプリに渡す



成果・感想等

- 苦労した点
  - Apache-XML-Security-J 1.0.5D2 のXML暗号実装は未完成
    ソースコードに手を入れる必要あり
  - JDKのバージョンなど，環境によって動かないことがあった
- 成果
  - 他実装との相互接続に成功
  - End to End のWS-Security処理に成功
  - WS-Securityに関するノウハウを蓄積
- 感想
  - 業務アプリ開発とセキュリティ実装を分離することができた
  - 今後，WS-Security実装ガイドライン，標準API，対応製品の整備が進むはず

⇒ WS-Securityを利用したセキュアなWebサービスが，より簡単に実装可能になると期待

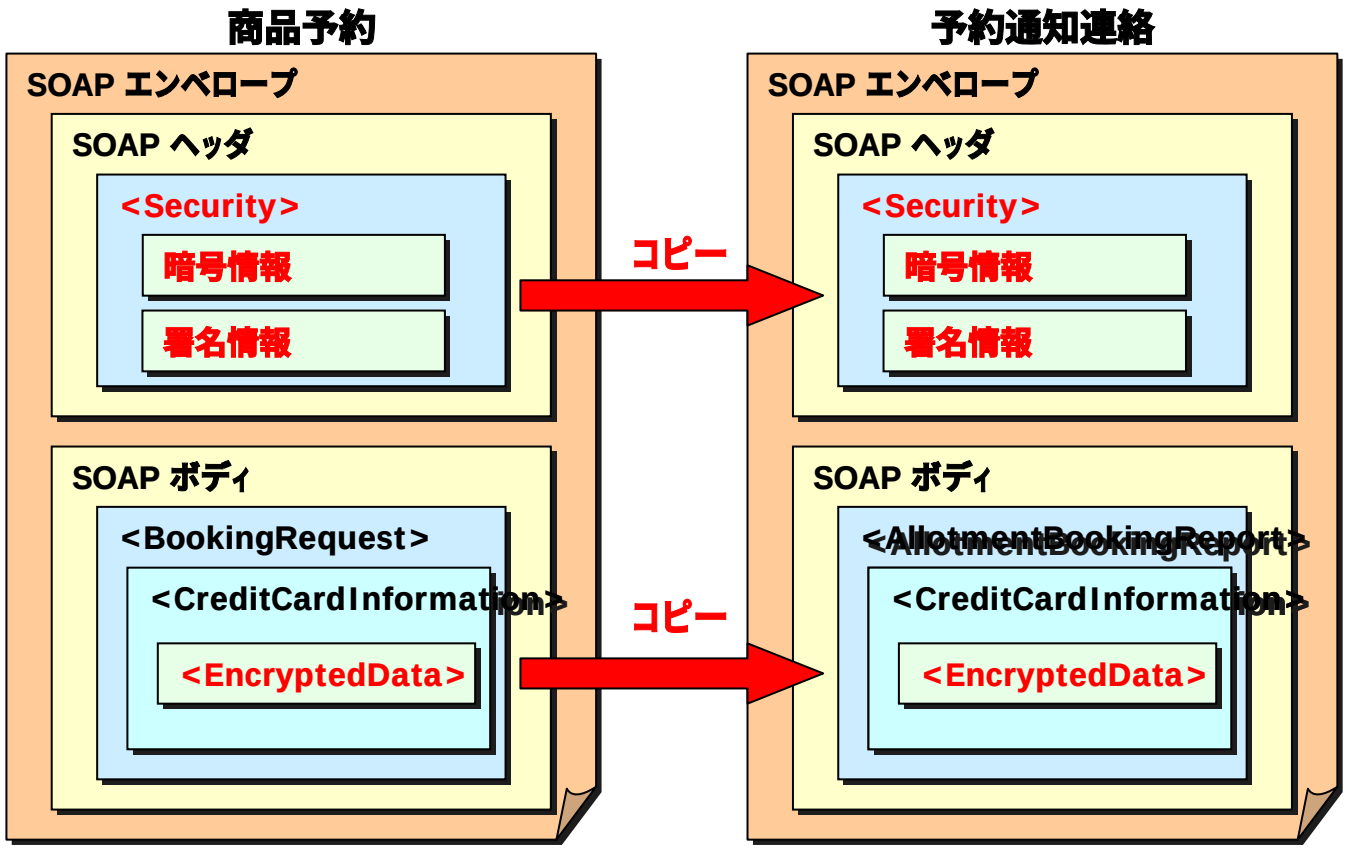
アドソル日進株式会社

荒本 道隆

概要 (旅行企画会社)

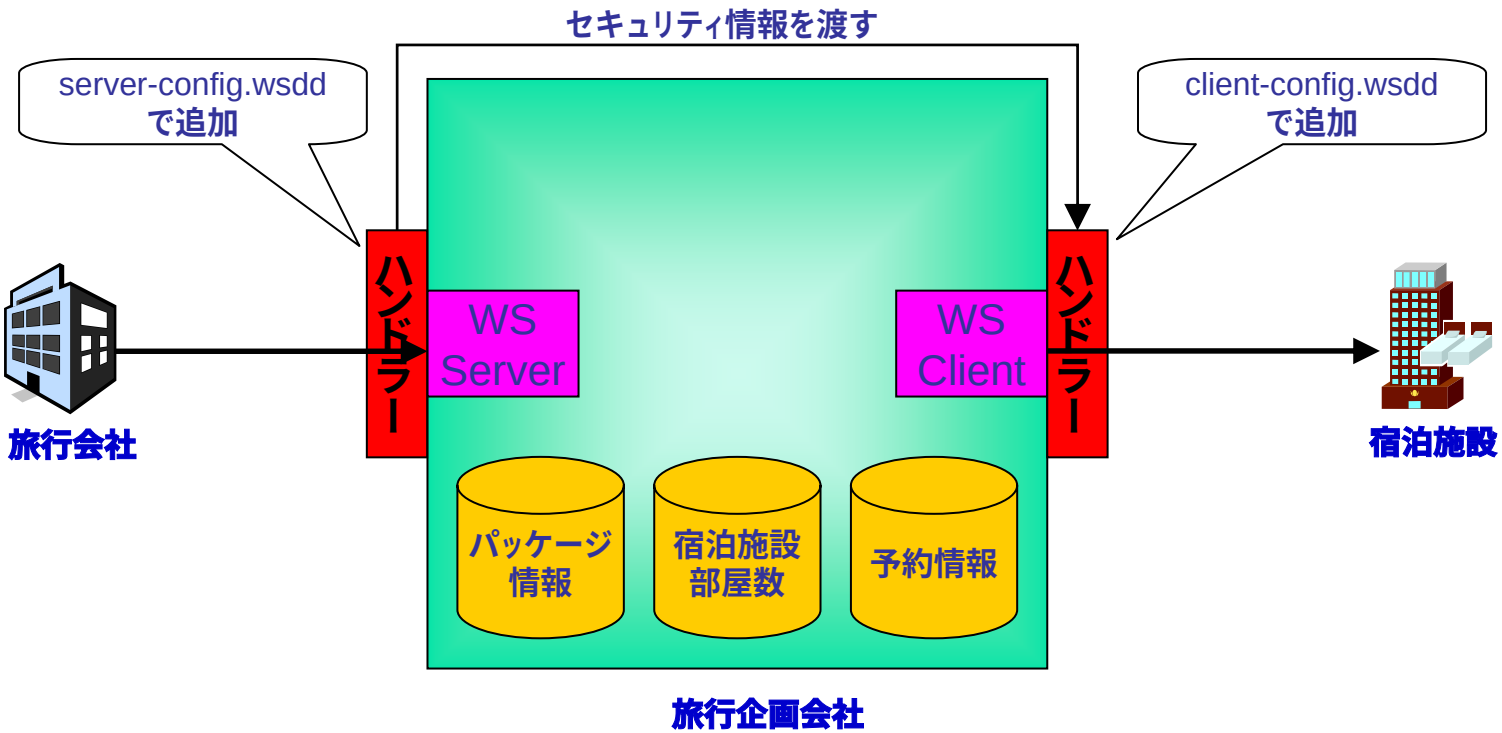
- End to Endのセキュリティを実現するために、旅行会社からのリクエスト中のセキュリティ情報をホテル／旅館に渡す
  - SOAPヘッダの、<Security>タグをコピー
  - 署名・暗号化された<CreditCardNumber>の上位の、<CreditCardInformation> をコピー
- 今回、旅行企画会社は暗号・復号も署名・署名検証もしない
  - セキュリティに関するミドルウエアは使用していない
  - 旅行会社⇔旅行企画会社や、旅行企画会社⇔ ホテル／旅館間のセキュリティは、WS-Security以外にも、SSLなどを使うことでも実現可能
- 開発工数 ： 約 0.3人月
  - セキュリティに関する部分：0.1人月
- ステップ数 ： 約 1.9 k ステップ (コメント除く)
  - セキュリティに関する部分：0.3 k ステップ (コメント除く)

処理内容



システム構成

OS	Windows XP Professional
開発言語	Java 2 SDK 1.4.2
Webサーバ	Tomcat 4.1.24
Webサービス	Apache AXIS 1.1



成果・感想等

- 苦労した点（主にAXIS1.1の問題）
  - 名前空間のprefixの書き方によっては、prefixが消えてしまい、XMLとしての意味が変わってしまう
    - →AXISでprefixが消えてしまわないような書き方にする
  - ハンドラでSOAPMessageをそのまま使うと、余計な改行やスペースがはいり、署名検証に失敗してしまう
    - →SOAPMessageをString化してからDocumentに再構築することで、回避
  - AXIS1.1ベースの製品を使う場合でも、同様の注意が必要？
- 成果
  - WS-Securityを使ってみて、実際に使用する上でのノウハウを蓄積
  - 署名検証については、中継する側でも注意が必要
- 感想
  - ハンドラを使用する事で、アプリのソースを一切変更する事なく、セキュリティに関する機能を追加することができた
  - スキーマを設計する段階で、セキュリティを考慮する必要性がある
    - スキーマには、<EncryptedData>などは一切記述されていない
    - コピー先の<CreditCardInformation>が同じElementでないと困る

日本アイオナテクノロジーズ株式会社

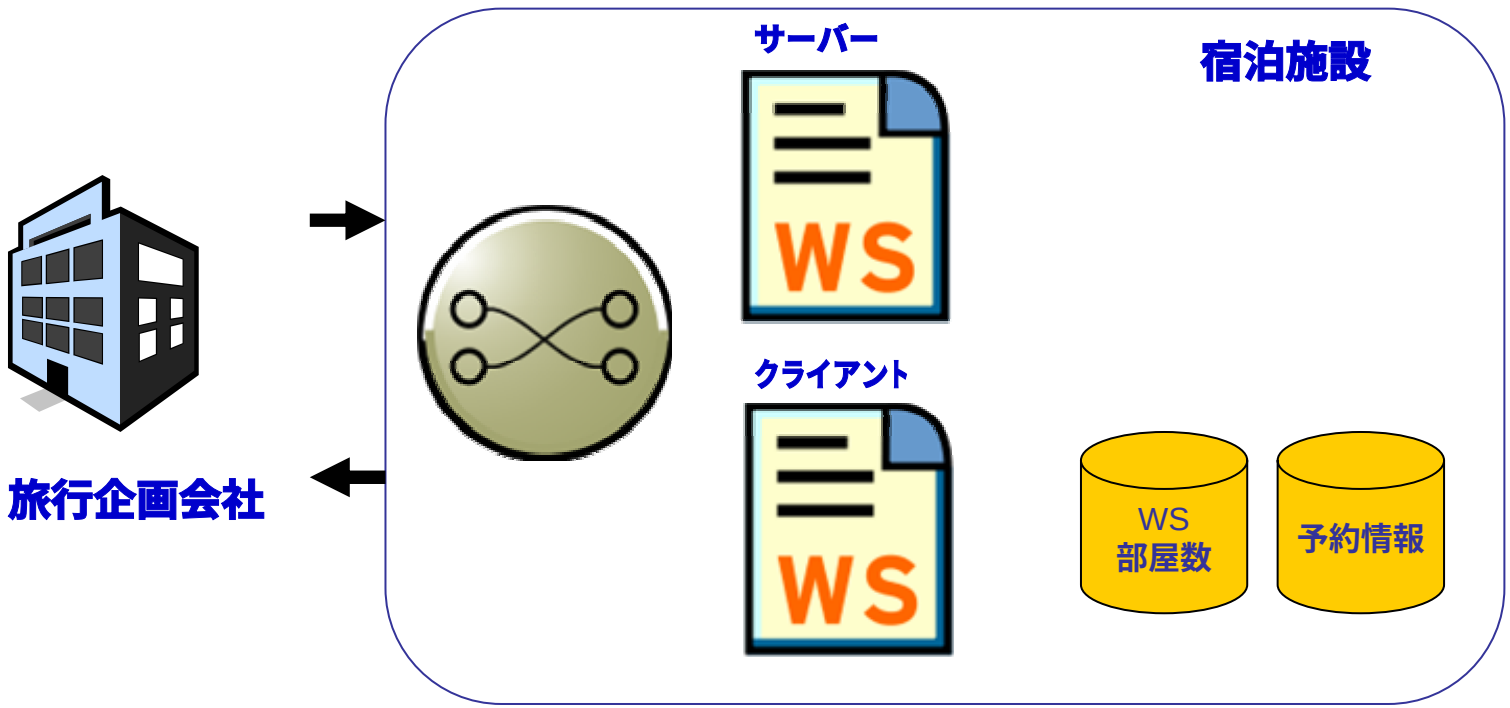
片山　良雄

概要(宿泊施設)

- セキュリティ・ゲートウェイによる復号、署名検証済みメッセージを受信
- 今回、宿泊施設としては、セキュリティに関する実装やミドルウェアの利用は行わない
- 開発工数　：　約 0.5人月
- ステップ数　：　約 1.2k ステップ (コメント含む)

システム構成

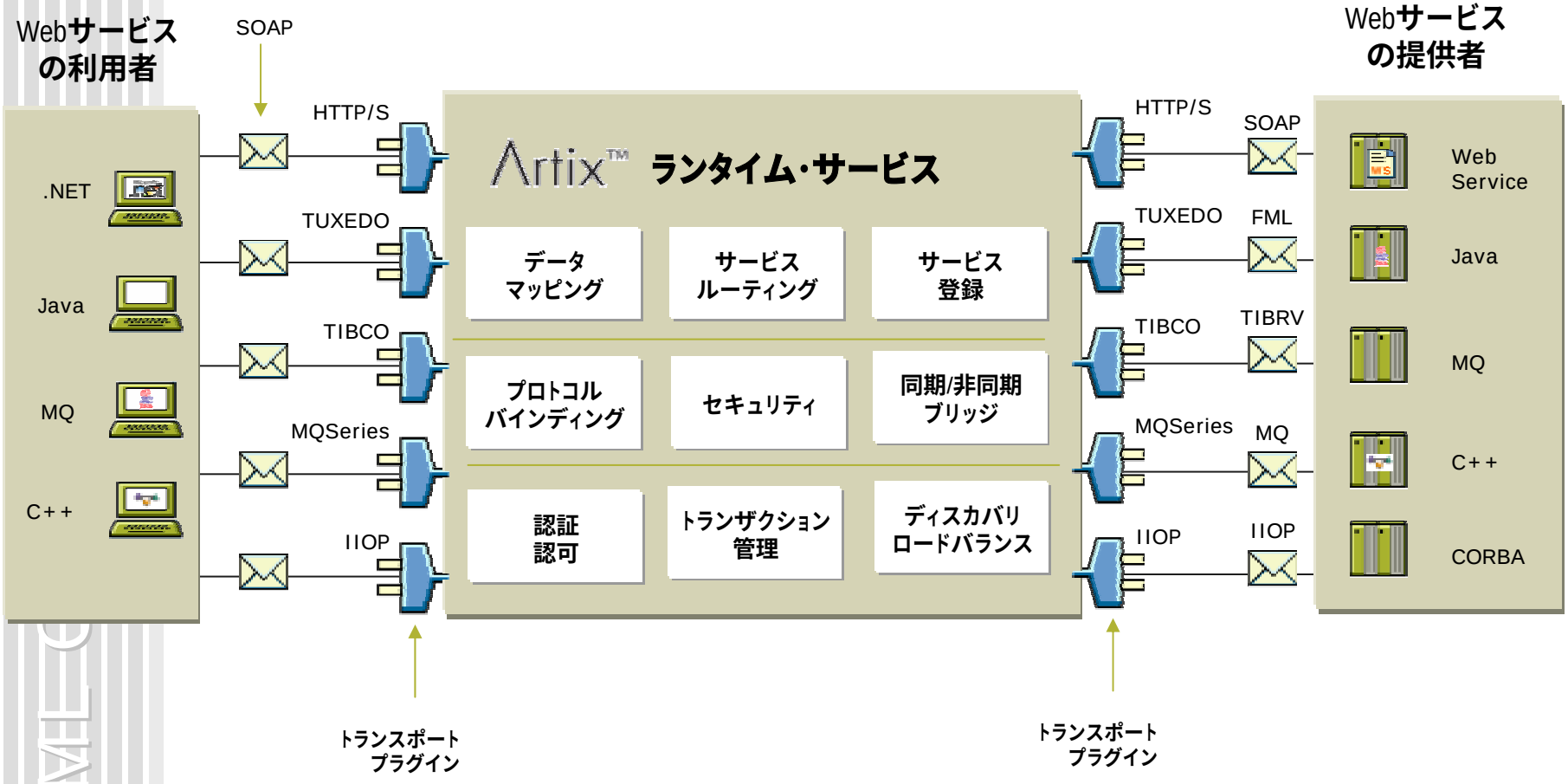
OS	Windows 2000 Professional
開発言語	Java 2 SDK 1.4.2
Webサービス	IONA Artix 2.0.3
DB	Microsoft Access 2000



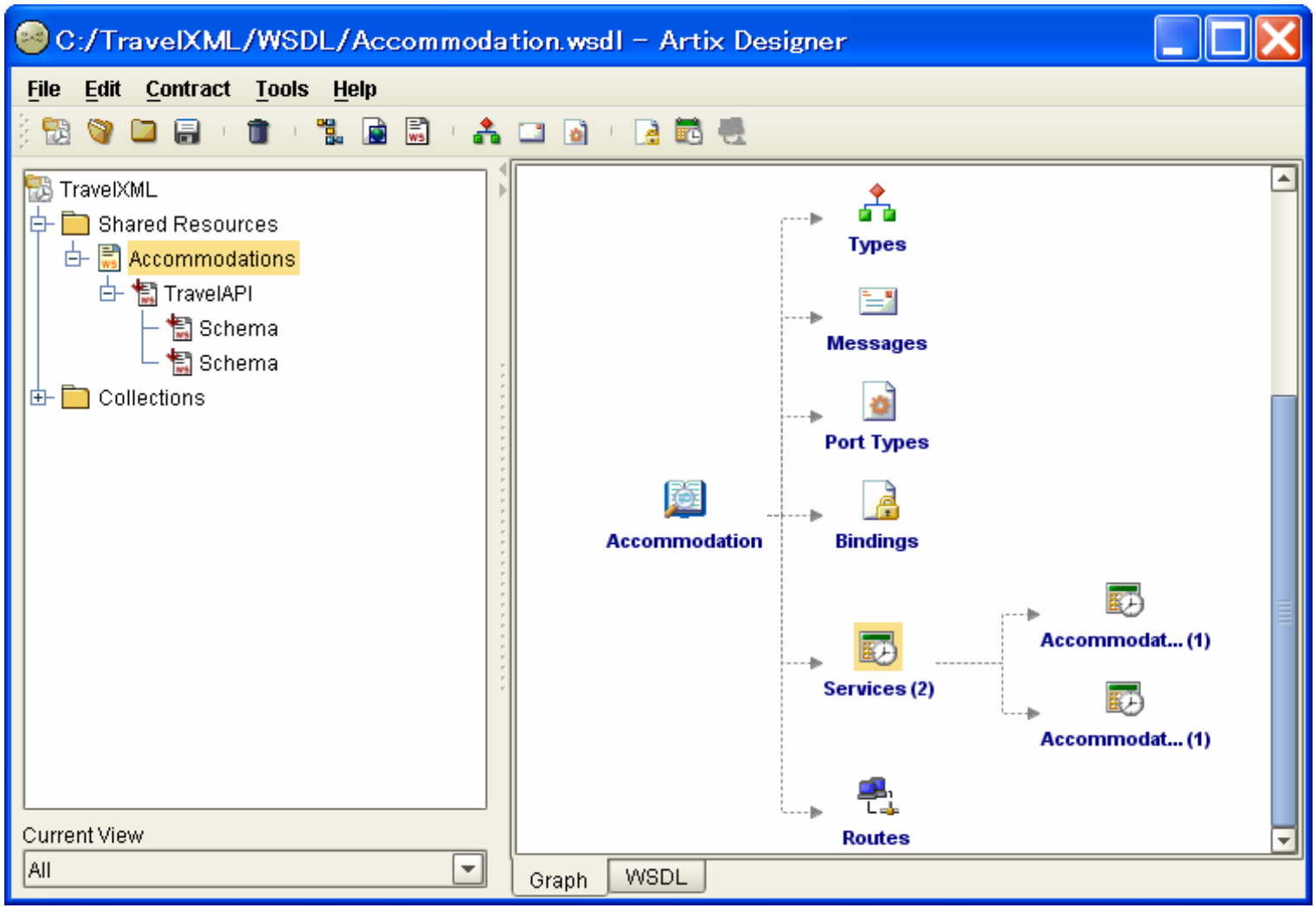
成果・感想等

- 苦労した点
  - XMLスキーマの内容の理解
  - 他社との接続検証
- 成果
  - 製品の完成度や、相互接続における問題点を知ることができた
- 感想
  - セキュリティに関して、製品のセキュリティ対応の、相互接続時の動作検証が課題

IONA Artixのご紹介



Artix Designer



日本IBM株式会社

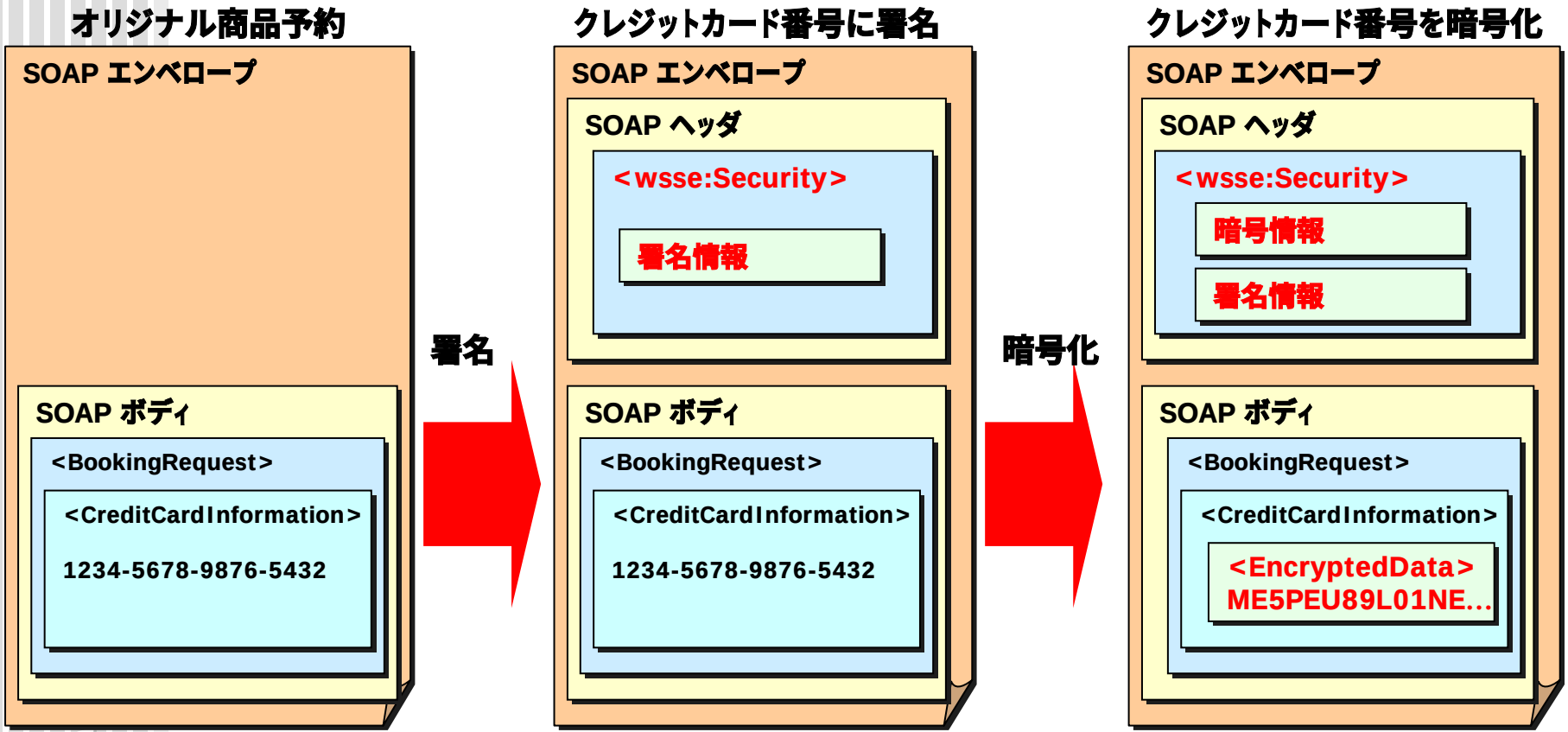
吉田 忠行

概要(旅行代理店)

- 予約の際にクレジットカード番号に対して，署名を行ったのちに，暗号化を行う
  - 署名には，自分の秘密鍵を使用
  - 暗号化には，受け取る相手(東芝，アイオナ)それぞれの公開鍵をシナリオにあわせて使用する。同時にactorを受け取る相手にセットする。
  - Apache Axis のハンドラとして実装
- XMLセキュリティのライブラリはXSS4J (XML Security Suite for Java: IBM alphaworks[*1]にて配布中)を使用
  - XSS4Jに対するWS-Security(OASIS 2004/04版)対応のWrapperを作成
  - XSS4Jを用いて署名/暗号化したあと，WS-Securityに準拠したセキュリティのSOAPヘッダを作成
- 開発工数 ： 約 0.4人月
- ステップ数 ： 約 1.3 k ステップ (コメントおよびXSS4Jのコードを除く)

*1 http://www.alphaworks.ibm.com/tech/xmlsecuritysuite

処理内容(署名→暗号化)



受け取る相手(東芝，アイオナ)それぞれの公開鍵をシナリオにあわせて使用すると同時にactorを指定する。

成果･感想等

- **苦労した点**
  - XSS4Jの最新版(2003/01/27版)では，2004/04版のOASISによるWS-Security仕様には完全に準拠していないため，今回作成したWrapperによって，最新仕様に準拠したメッセージの組み立てが必要になった
  - 署名のあと，暗号化する段階で，署名時に対象になったXML要素に変更を加えることなく暗号化を施す実装
    - Axis が改行文字や空白文字のノードを追加してしまうので，それを回避
  - 復号化･検証する相手の実装によって，署名･暗号化対象に変更を行う必要がある
    - 名前空間の宣言(xmlns:wsu="http://…"など)を署名･暗号化対象のXML要素の属性として含める必要が生じる可能性がある
  - 暗号化単独･署名単独･署名/暗号化の組み合わせにおいて，重ね合わせが正しく動くこと

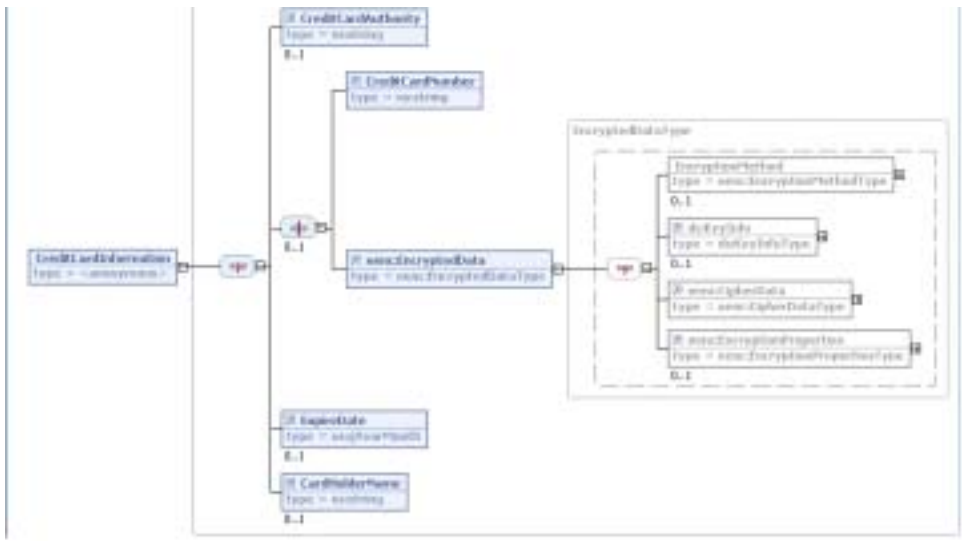
成果･感想等

- **感想**
  - WS-Securityに対応したヘッダを作成する部分は，仕様をよく読み，正確に実装する必要があり，接続実験等を行わなければ，実装の正確さを検証することが難しい
    - 署名検証･復号化をする相手が同一の場合，wsse:Securityの中に，署名情報および暗号化情報を含める必要がある
    - 使用した鍵の情報を渡す部分は，いくつかの方法があり，実装でカバーしつつ，署名･暗号化するエンティティと，検証･復号化するエンティティ間で，鍵情報のフォーマットについて合意がとれている必要がある

課題：セキュリティ実装とインターオペラビリティの問題

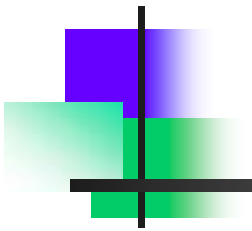
- AXISのハンドラーやDataPowerを利用することで以下が可能であることが分かった
  - 業務アプリ開発とセキュリティ実装を分離することができる
  - アプリのソースを一切変更する事なく、セキュリティに関する機能を追加することができる
- しかし、セキュリティ実装用のスキーマは業務アプリ用とは異なる
  - スキーマには、<EncryptedData>などは一切記述されていない
  - スキーマを拡張してセキュリティ実装用を作るべきかどうかは各社の判断に任せられた

IBMのセキュリティ実装用スキーマ：
CreditCard NumberとEncryptedDataが選択（Choice）できる



まとめ

- End-to-Endのセキュリティ検証を実際に旅行企画会社を挟んで検証できた
- 様々なセキュリティ製品間で検証できた
  - Apache-XML-Security (日立) ⇔DataPower (東京エレクトロン) ⇔XSS4J (IBM) で成功
  - Apache-XML-Security (日立) ⇔DataPower (東京エレクトロン) ⇔ ActiveGlobe WebOTX (NEC) で成功



つづく