XML Consortium

～ 第8回 XMLコンソーシアムDay ～

sPlatプロジェクト活動報告

2006年12月12日

XMLコンソーシアム セキュリティ部会
中山　弘二郎（株式会社 日立製作所）

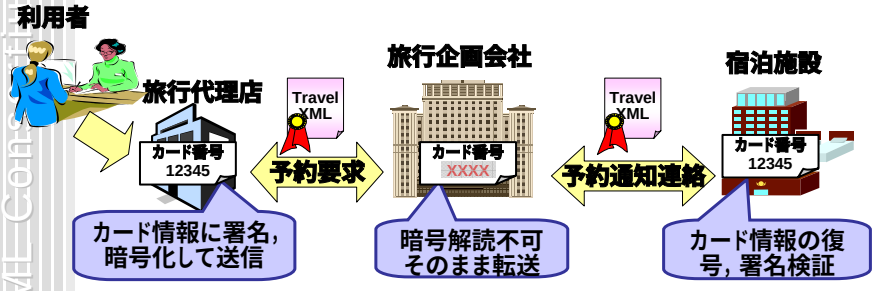
目次

目次

- 背景と目的
- 暗号化データ処理方法
  - ローレベルAPIを使う場合
  - データバインディングを使う場合
- スキーマ変換方式の詳細
- まとめと今後の課題

Webサービスのセキュリティ

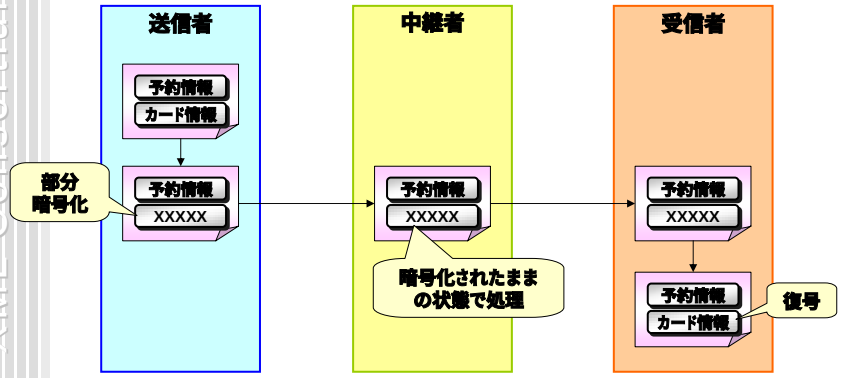
- 想定するシーン
  - 中継者を介したマルチホップのWebサービス
  - End-to-Endのセキュリティを確保



※ 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公開資料」より
http://www.xmlconsortium.org/koukai/travelxml-p/

暗号化処理の流れ

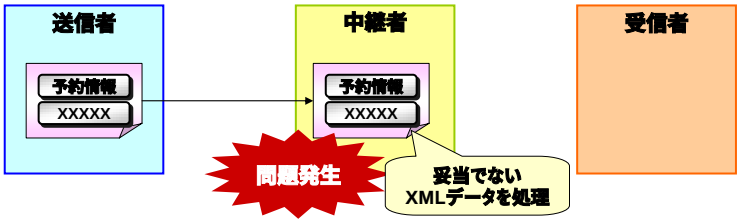
- XML暗号によるメッセージの暗号化
  - 中継者に開示したくないデータを部分暗号化
  - 中継者に対する秘匿性を確保



課題と目的

- 課題
  - XMLデータの構造はスキーマにより事前に定義されている
  - XMLデータを部分暗号化することで、スキーマに対する妥当性が失われる
    - 中継者はスキーマに対して妥当でないXMLを処理する必要がある
  - XMLプロセッサはXMLデータが妥当であることを期待していることが多い

中継者の処理において問題が発生



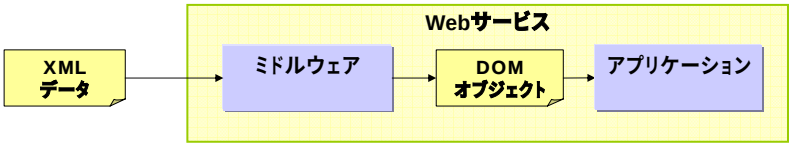
- sPlatプロジェクトの目的
  - Webサービスの中継者における適切な暗号化データの処理方式を検討、開発、提案する

目次

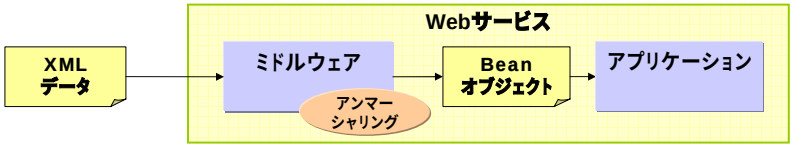
- 背景と目的
- 暗号化データ処理方法
  - ローレベルAPIを使う場合
  - データバインディングを使う場合
- スキーマ変換方式の詳細
- まとめと今後の課題

Webサービスの開発方法

- ローレベルAPIを使う場合
  - DOM、AXIOM (Axis2) などのローレベルAPIを用いてデータにアクセス
  - アプリではXMLを意識したコーディングをする



- データバインディングを使う場合
  - WSDL2Javaなどのツールで生成されたAPIを用いてデータにアクセス
  - 実行時にXMLをBeanオブジェクトにマッピング (アンマーシャリング)
  - アプリではXMLを意識しない

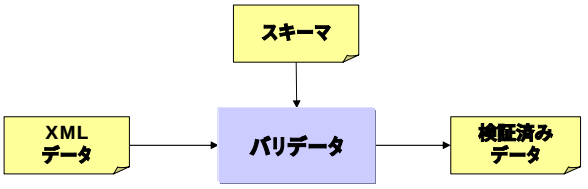


目次

- 背景と目的
- 暗号化データ処理方法
  - ローレベルAPIを使う場合
  - データバインディングを使う場合
- スキーマ変換方式の詳細
- まとめと今後の課題

妥当性検証に関する課題

- 妥当性検証の目的
  - XMLデータの構造がスキーマ定義に従っていることを確認する
  - 信頼性の確保、セキュリティの向上が期待できる
- XMLデータが暗号化されている場合
  - 妥当性検証を実施することができない

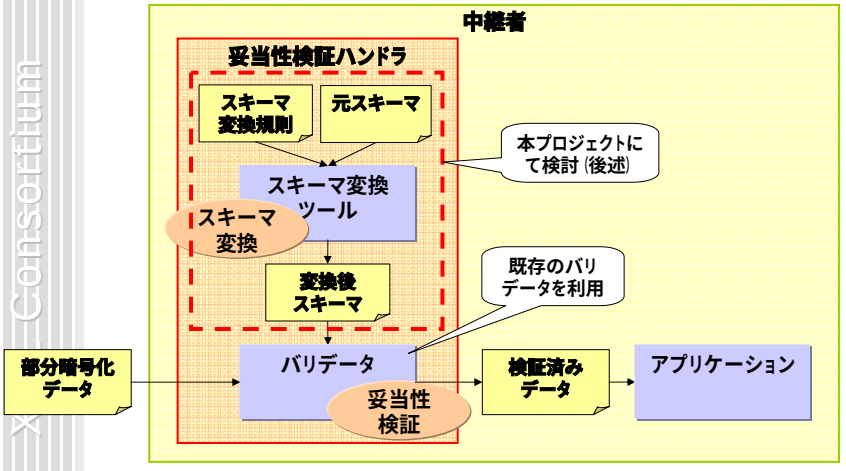


対策方式の比較

方式	処理の概要	課題
バリデータ方式	・バリデータを改変し、スキーマに対して妥当でないXMLデータの妥当性検証を可能にする	・既存のバリデータを利用することができない ・対策方法がバリデータの種別に依存する
XMLデータ変換方式	・スキーマに対して妥当になるように受信したXMLデータを変換する (e.g. 暗号化箇所を取り除く)	・暗号対象が必須要素の場合、対応が困難 ・取り除いた部分の妥当性検証ができない
スキーマ変換方式	・スキーマを変換し、暗号化に対応したスキーマを作成する	・スキーマの変更作業が煩雑でミスが発生しやすい ⇒スキーマ変換ツールを検討 (後述)

スキーマ変換ツールによる対策

- 暗号に対応したスキーマを生成 (開発時 or 実行時)
- Webサービスハンドラにより妥当性検証を実施

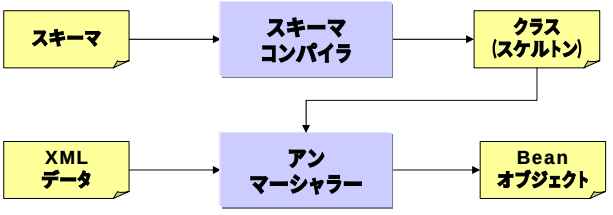


目次

- 背景と目的
- 暗号化データ処理方法
  - ローレベルAPIを使う場合
  - データバインディングを使う場合
- スキーマ変換方式の詳細
- まとめと今後の課題

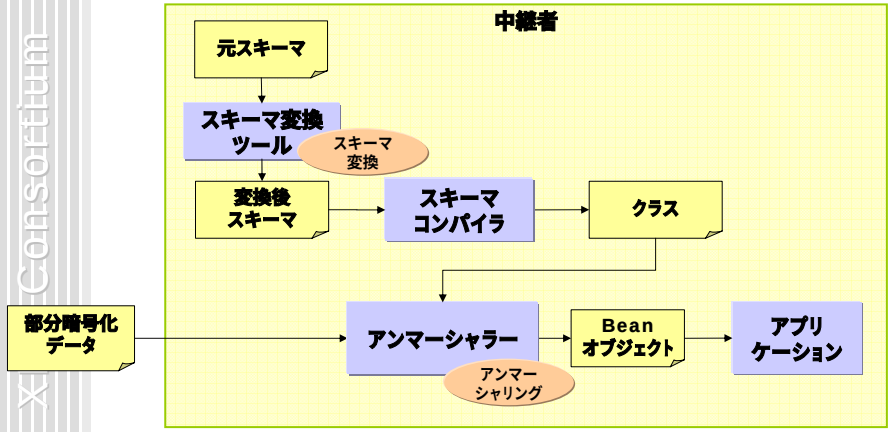
データバインディングに関する課題

- データバインディングの目的
  - XMLデータとBeanオブジェクトのマッピング
  - 簡単なAPIを使ってXMLデータにアクセス
- データバインディングの処理
  - スキーマコンパイル（開発時）：スキーマからクラスを生成
  - アンマーシャリング（実行時）：XMLデータからオブジェクトを生成
- XMLデータが暗号化されている場合
  - アンマーシャリングの処理に失敗する可能性がある



対策方式 (1)

- 変換後スキーマを使ってクラスを自動生成
  - 課題：開発時に暗号箇所が決まっている必要がある

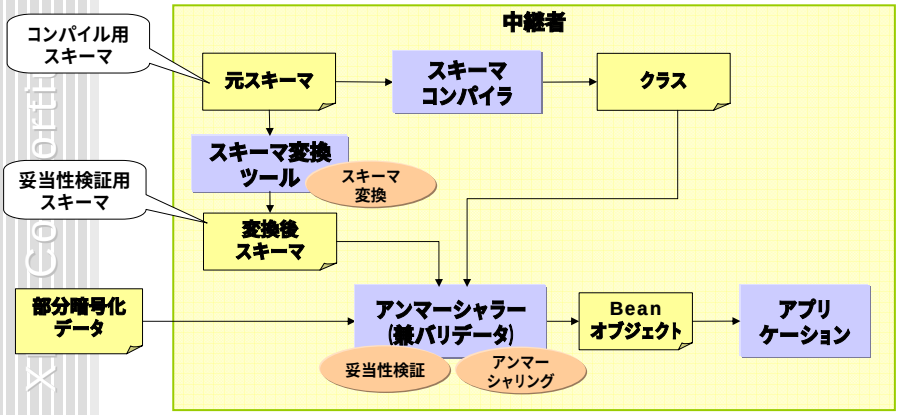


Flexible Unmarshalling

- Flexible Unmarshalling (JAXB 2.0)
  - 妥当性検証とアンマーシャリングの処理を分離
    - 妥当性検証の on / off の指定が可能
  - 妥当でないXMLデータのアンマーシャリングが可能
    - 必須要素が存在しなくても処理を続行する
    - 予期しない要素が出現しても無視する
- 暗号化データのアンマーシャリングへの適用
  - Flexible Unmarshallingを利用することで、中継者のデータバインディングにおけるエラーを回避可能
    - 必須要素が暗号化されていても処理を続行
    - 暗号化データが出現しても無視する
- Flexible Unmarshallingを採用している実装はまだ少ない
  - 今後に期待

対策方式 (2)

- オリジナルスキーマを使ってスキーマコンパイル (開発時)
- 暗号対応スキーマを生成 (開発時 or 実行時)
- 妥当性検証 & Flexible Unmarshalling (実行時)

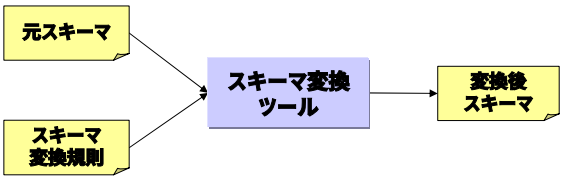


目次

- 背景と目的
- 暗号化データ処理方法
  - ローレベルAPIを使う場合
  - データバインディングを使う場合
- スキーマ変換方式の詳細
- まとめと今後の課題

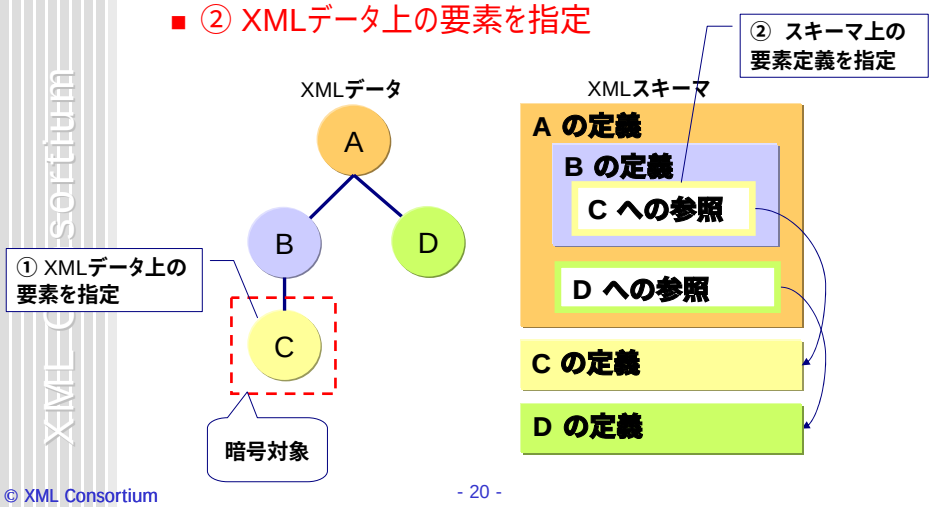
スキーマ変換ツールの概要

- インプット
  - 元スキーマ
  - スキーマ変換規則
    - 暗号化の要求レベル (MUST or MAY)
    - 暗号タイプ (エレメント暗号 or コンテント暗号)
    - **暗号対象要素**
- アウトプット
  - 変換後スキーマ (暗号対応スキーマ)

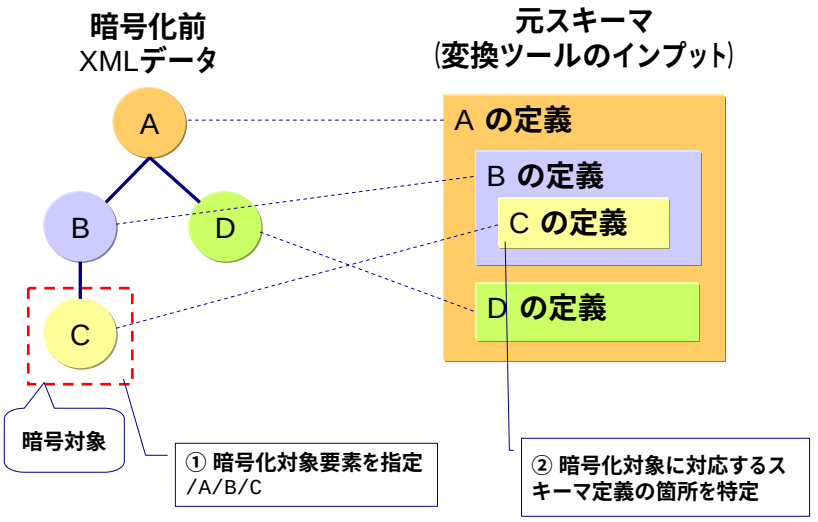


暗号対象要素の指定方法

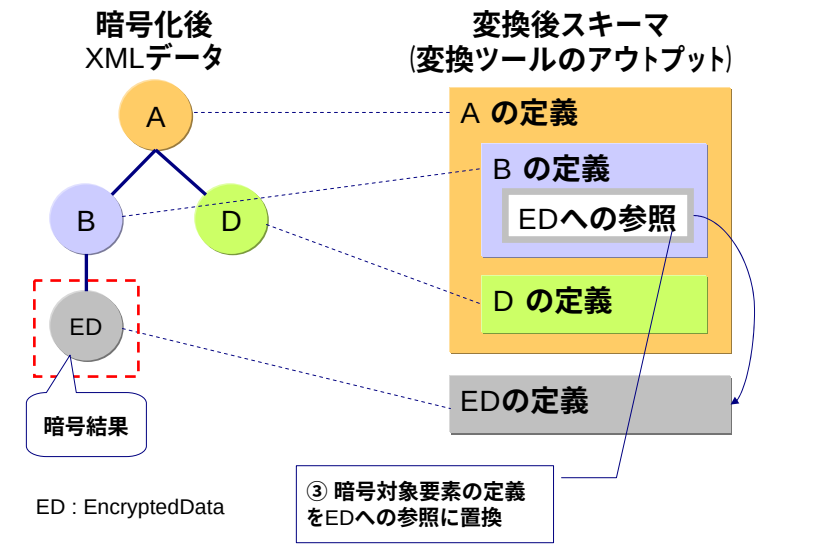
- 2つの暗号対象要素の指定方法
  - ① スキーマ上の要素定義を指定
  - ② XMLデータ上の要素を指定



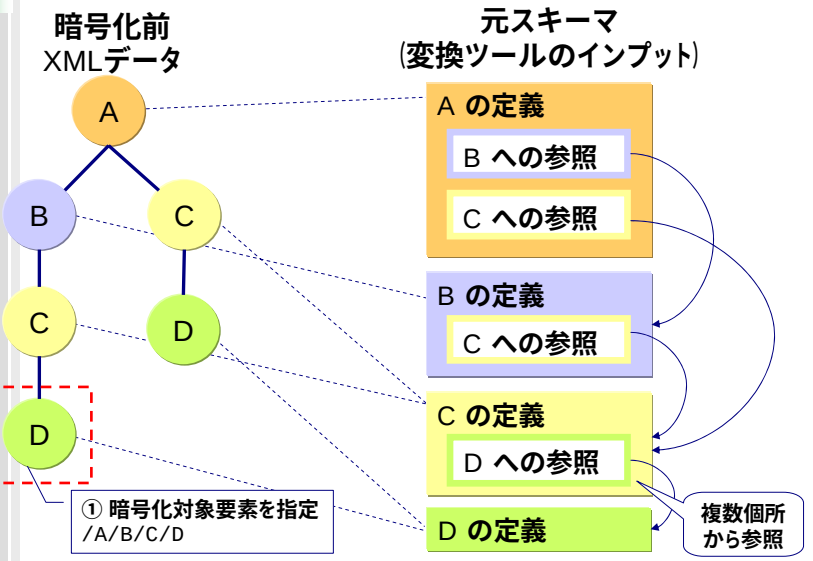
変換例（Russian Dollの場合）



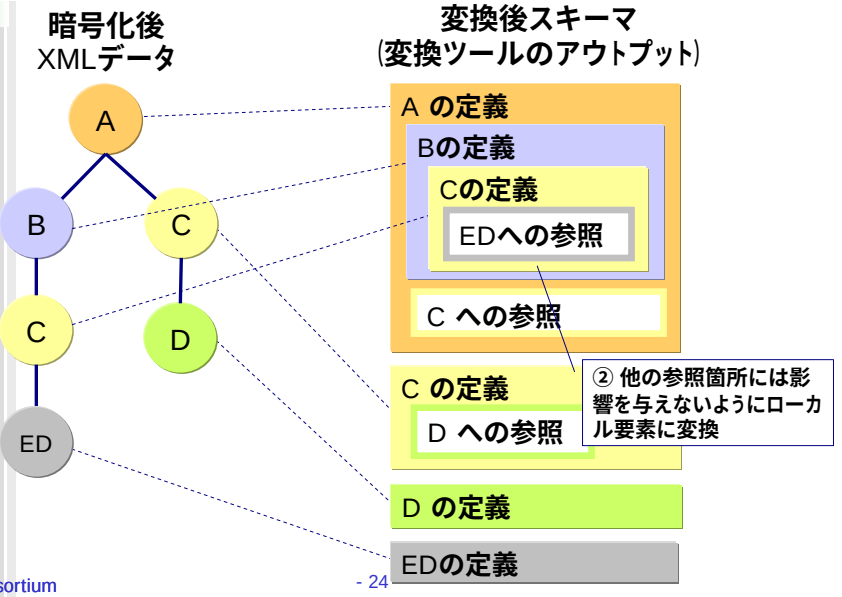
変換例（Russian Dollの場合）



変換例 (Salami Sliceの場合)



変換例 (Salami Sliceの場合)



目次

- 背景と目的
- 暗号化データ処理方法
  - ローレベルAPIを使う場合
  - データバインディングを使う場合
- スキーマ変換方式の詳細
- まとめと今後の課題

まとめ (1)

- Webサービスの中継者における，適切な暗号化データの処理方式について検討
- 暗号化データの妥当性検証を実現する方法として、スキーマ変換方法の詳細を検討
- 今後の課題
  - 暗号化ポリシーとの連携

まとめ (2)