

Enforcing and Validating WS-SecurityPolicy for Web Service Firewalls

Nils Gruschka

Ralph Herkenhöner

Norbert Luttenberger

Christian-Albrechts-University in Kiel, Germany

IEEE / IST Workshop on
„Monitoring, Attack Detection and Mitigation“
28 / 29 September, 2006, Tübingen, Germany

Overview

- Attacks on Web Services
- Web Service Security
- Attack Prevention
- WS-SecurityPolicy:
 - Enforcement
 - Decision
 - Token Mapping
- Summary & Outlook

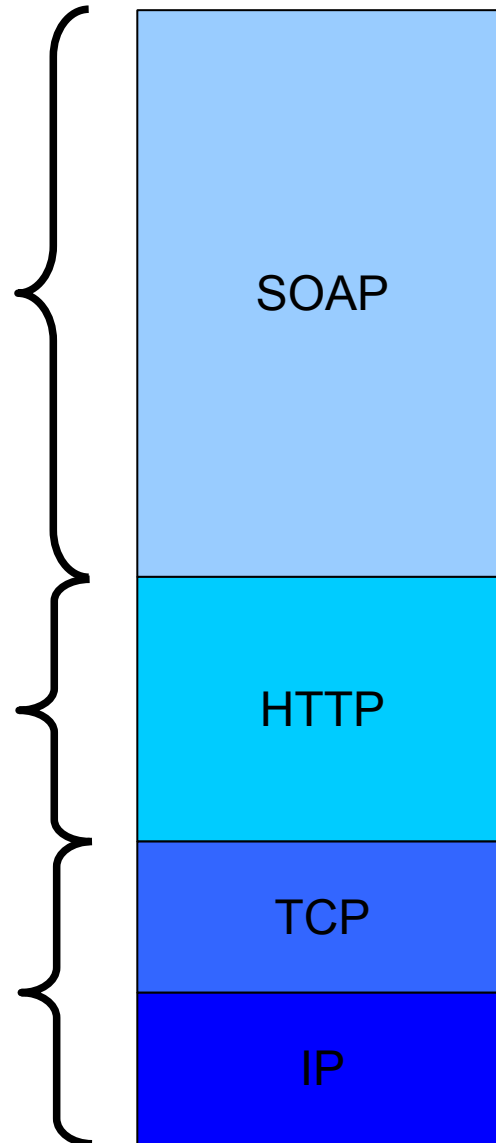
Web Service Protocol Stack

Attacks

?

e.g. Cookie Poisoning

e.g. SYN Flooding



```
<Envelope>  
  <Body>  
    <add>  
      <x>12</x>  
      <x>38</x>  
      <x>27</x>  
    </add>  
  </Body>  
</Envelope>
```

```
POST /Service.asmx  
Content-Type: text/xml  
Content-Length: 523
```

```
Destination Port: 80
```

```
Destination Address  
10.1.1.1
```

Common Attacks (1)

- Oversize Payload:
 - Attack using large SOAP message
 - .NET: 4 MB payload → 9 MB memory usage
 - Axis2: 3 MB payload → 15 MB memory usage
6 MB payload → 51 MB memory usage
 - Oracel BPEL Process Manager:
8 MB payload → 250 MB memory usage

Common Attacks (2)

- Incorrect message structure: .NET

Correct Message:

```
<Envelope>  
  <Body>  
    <twoParams>  
      <a>1</a>  
      <b>2</b>  
    </twoParams>  
  </Body>  
</Envelope>
```

Message accepted

Result: a = 1, b = 2

Incorrect Message:

```
<Envelope>  
  <Body>  
    <twoParams>  
      <a>  
        <b>1</b>  
      </a>  
      <b>2</b>  
    </twoParams>  
  </Body>  
</Envelope>
```

Message accepted

Result: a = 1, b = 0

Common Attacks (3)

- Coercive Parsing: Axis2
 - Extremely deep nested XML structure
 - Sending a non-ending sequence of opening tags (e.g. `<x> <x> <x> <x> <x> ...`)
 - 100% CPU load without any connection break-up
 - Attack bandwidth: 150 Bytes/s
- ➔ (DoS) protection for Web Services required!

Security for SOAP: WS-Security

- For Web Services transport security systems like SSL or IPSec are not suitable
- WS-Security extends SOAP to ensure message *Confidentiality, Integrity and Authenticity*
- Prevents attacks: eavesdropping, message modification, etc.
- Application of
 - XML Encryption
 - XML Signature

for SOAP messages

```
<Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:wss="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
  xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
  xmlns:xenc="http://www.w3.org/2001/04/xmlenc#" xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
  <Header>
    <wss:Security>
      <wsu:Timestamp Id="Timestamp">
```

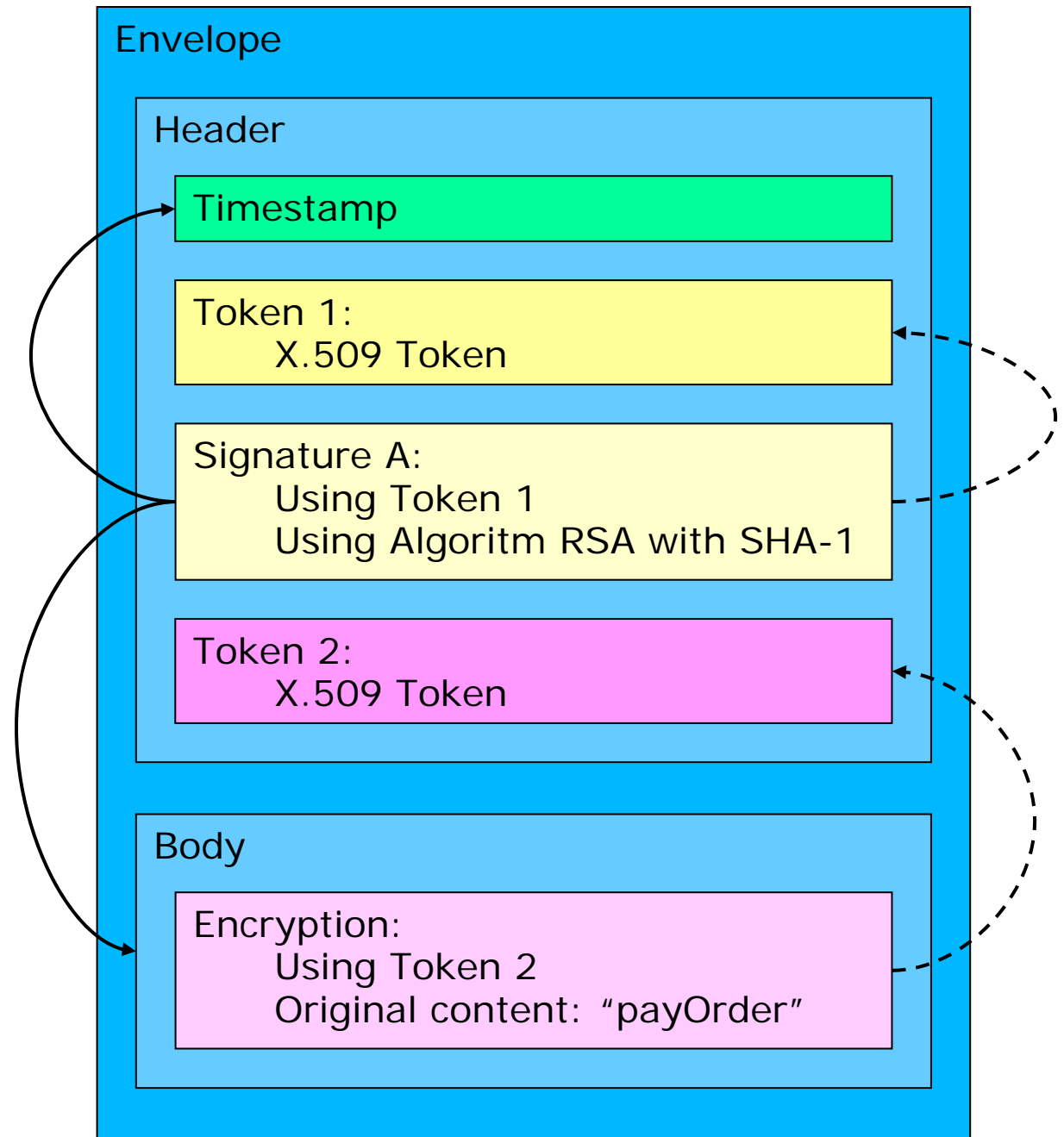
Example: WS-Security in SOAP

```
<ds:Signature>
  <ds:SignedInfo>
    <ds:CanonicalizationMethod Algorithm="...xml-c14n#" />
    <ds:SignatureMethod Algorithm="...#rsa-sha1" />
    <ds:KeyInfo>
      <xenc:EncryptedData Id="enc">
        <xenc:EncryptionMethod Algorithm="...xmlenc#tripleDES-cbc" />
        <xenc:CipherData>
          <xenc:CipherValue>
            <wsse:BinarySecurityToken ValueType="wsse:X509v3" Id="Token1">
              MIICKDCCAZECEBEKcSaowDQYJ...
            </wsse:BinarySecurityToken>
          </xenc:CipherValue>
        </xenc:CipherData>
      </xenc:EncryptedData>
    </ds:KeyInfo>
  </ds:Signature>
  ...
  <Body wsu:Id="body">
```

```
<xenc:EncryptedData Id="enc">
  <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#tripleDES-cbc">
    <xenc:CipherData>
      <xenc:CipherValue>d2FpbmdvbGRfE0lm4byV0MfZ2kQLXDJbchm5gKNHstSmowR0NVy6g5LDkLFkDCCAZECEBEKc...</xenc:CipherValue>
    </xenc:CipherData>
  </xenc:EncryptedData>
</Body>
</Envelope>
```

Web Service Security

Example:
Abstract Security Elements



Security Requirements: WS-SecurityPolicy

- A Web Service server must specify its security requirement
- WS-SecurityPolicy defines allowed/required WS-Security elements
- Example:
 - Abstract requirements:
 - Integrity for message body
 - Confidentiality for element “payOrder” inside message body
 - Policy requirements:
 - Signing message body
 - Encrypting element “payOrder”
 - Use X.509 for security tokens
 - Include timestamp in message

Web Service Security

Example: *WS-SecurityPolicy*

Endpoint Policy

```

<wsp:Policy>
  <sp:SymmetricBinding>
    <wsp:Policy>
      <sp:ProtectionToken>
        <wsp:Policy>
          <sp:X509Token>
            <wsp:Policy>
              <sp:WssX509V3Token10/>
            </wsp:Policy>
          </sp:X509Token>
        </wsp:Policy>
      </sp:ProtectionToken>
      <sp:AlgorithmSuite>
        <wsp:Policy>
          <sp:TripleDesRsa15/>
        </wsp:Policy>
      </sp:AlgorithmSuite>
      <sp:Layout>
        <wsp:Policy>
          <sp:Strict/>
        </wsp:Policy>
      </sp:Layout>
      <sp:EncryptBeforeSigning/>
      <sp:IncludeTimestamp/>
    </wsp:Policy>
  </sp:SymmetricBinding>
</wsp:Policy>

```

Message Policy

```

<wsp:Policy>
  <sp:EncryptedElements>
    <sp:XPath>
      /Envelope/Body/payOrder
    </sp:XPath>
  </sp:EncryptedElements>
  <sp:SignedParts>
    <sp:Body/>
  </sp:SignedParts>
</wsp:Policy>

```

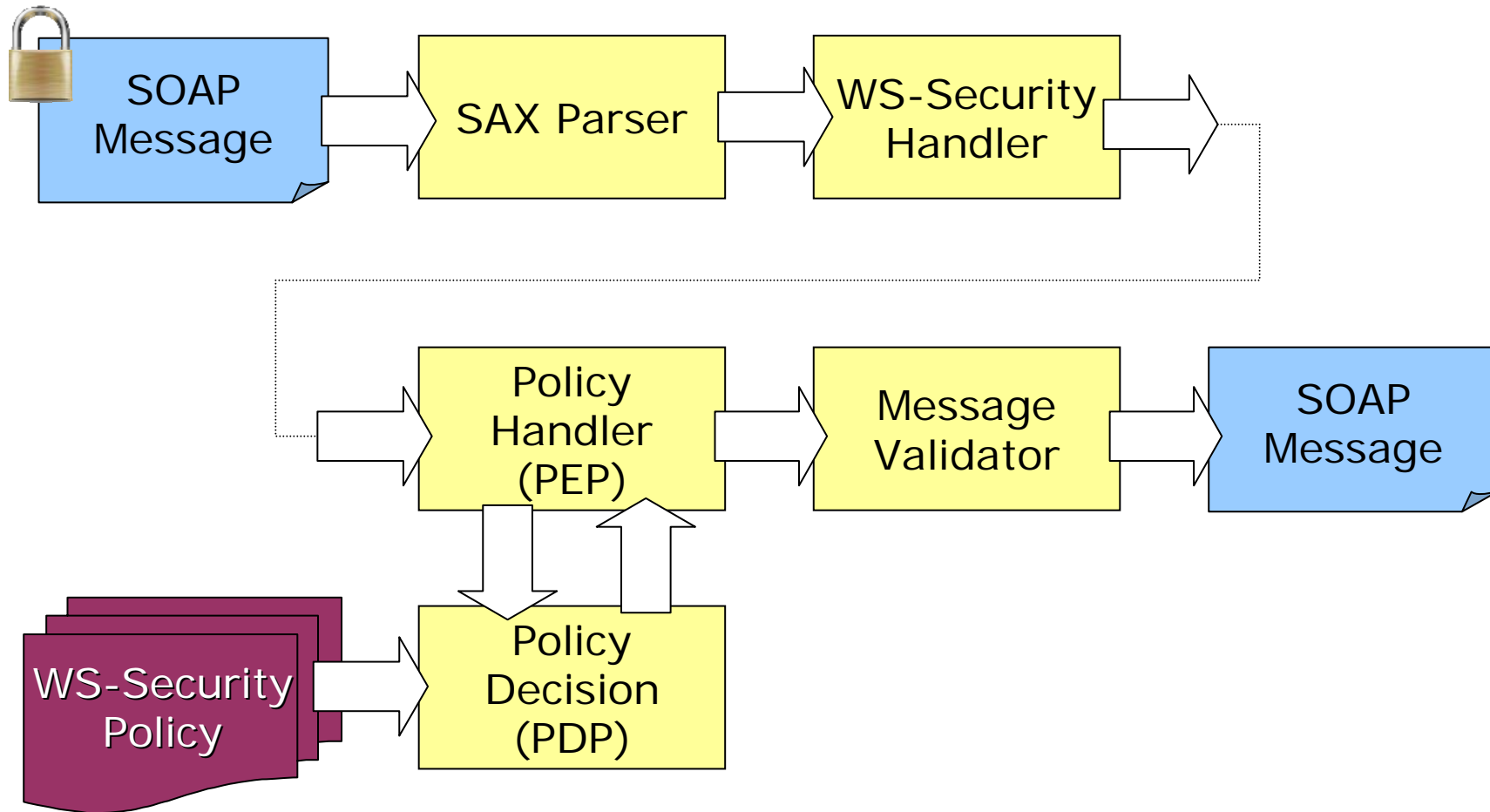
Limits of WS-Security

- No protection against attacks like DoS
- In contrast: new possibilities for attacks
 - Non-policy conforming message
 - Missing security tokens
 - Oversize payload by weak maximum requirements
 - No upper bound for number of security tokens
 - Unnecessary cryptographic operations
 - Encrypted large blocks
 - No syntactic or semantic checks possible
 - Can mask prior mentioned attacks

Strategy for Attack Prevention

- Defining allowed messages:
 - Message structure and data types from “hardened” WSDL
 - Security elements from WS-SecurityPolicy
- Checking messages in a gateway:
 - Efficient (event-based) XML/SOAP processing
 - Validating Message against Schema
 - WS-Security element validation
 - Policy decision and enforcement

Architecture for an event-based Firewall

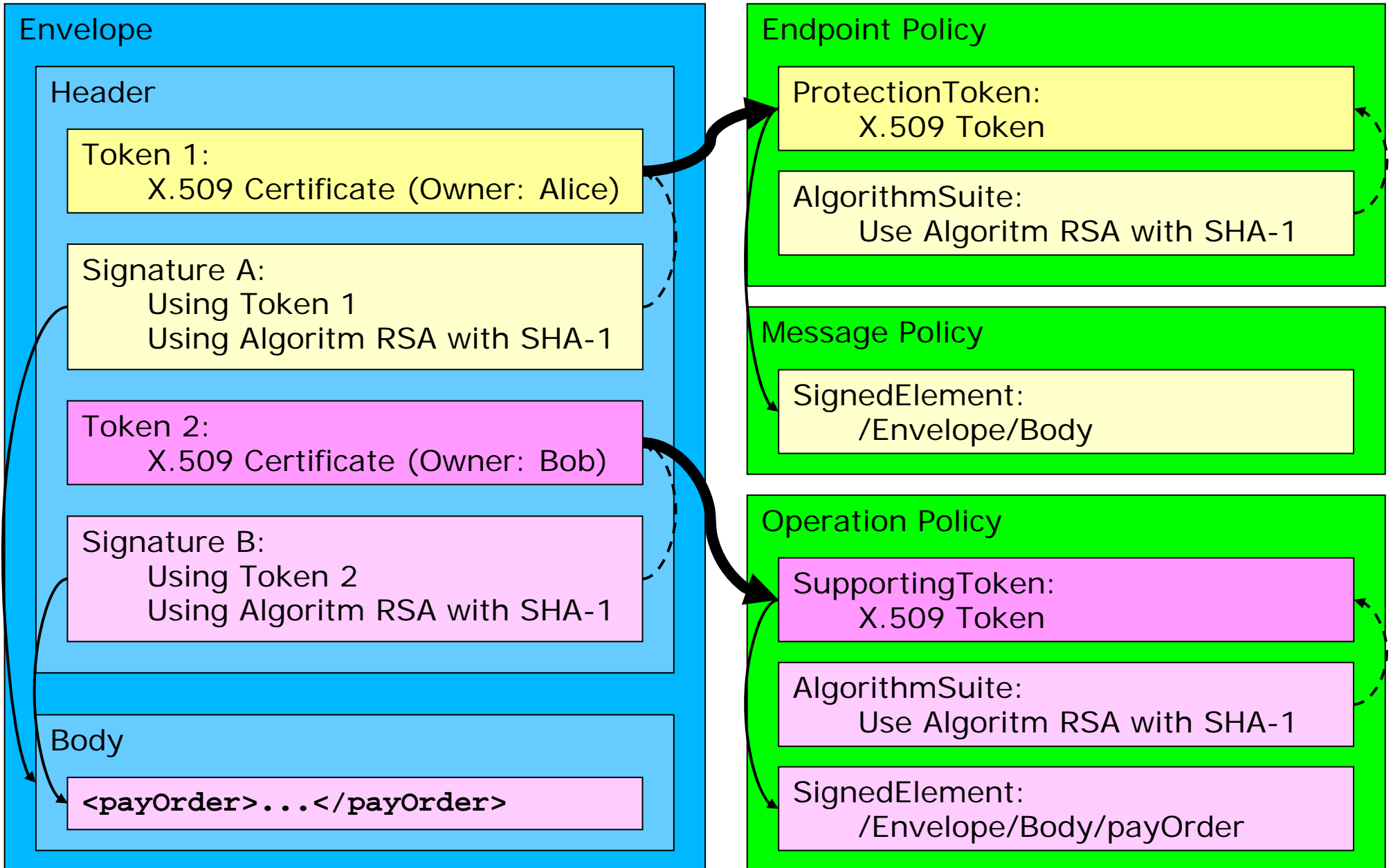


WS-SecurityPolicy Enforcement and Decision

- Policy Decision: “validating” message to policy
 - Token Mapping
 - Protected content check (policy references map to encrypted content in message)
 - Integrity check (policy references map to signed content)
- Policy Enforcement:
 - WS-SecurityPolicy defines minimum requirements but only weak maximum requirements
 - Strong maximum requirements needed for DoS protection → minimum = maximum

Token Mapping

Example: Token Mapping



Basic Idea

- Problem:
 - Tokens in policy given by token type
 - Token mapping ambiguous
 - additional related information (algorithm and usage) required
- Proposed solution:
 - Mapping by token type, algorithm and usage (AlgorithmSuite, SignedElement, etc.)
 - Modeling policy requirements as sets of triple
 - Strict Policy interpretation (hard maximum requirements)

Modelling Policy Requirements

- Create sets Sig^P and Enc^P from the WS-Security-Policy:
 - $Sig^P = \{ (p, r^P, a^P) \}$ the signatures postulated by the policy,
 - $Enc^P = \{ (p, r^P, a^P) \}$ the required encrypted blocks,
with
 - p the token type,
 - r^P the signed/encrypted reference and
 - a^P the used algorithm suite

Modelling SOAP Message Tokens

- Create sets Sig^M and Enc^M from the SOAP message:
 - $Sig^M = \{ (m, r^M, a^M) \}$ the signatures,
 - $Enc^M = \{ (m, r^M, a^M) \}$ the encrypted blocks,with
 - m the security token,
 - r^M the signed/encrypted reference and
 - a^M the used algorithm

Policy Token Mapping Algorithm (PTMA)

Idea:

- Find a bijection $\pi: T^M \rightarrow T^P$ with Sig^M maps to Sig^P and Enc^M maps to Enc^P
- Test every combination π using backtracking:
- For every $m \in T^M$ let $p := \pi(t^M)$
 - check if m is of type p
 - check if $Sig^M|_m \equiv Sig^P|_p$ and $Enc^M|_m \equiv Enc^P|_p$, i.e. check
 - $|Sig^M|_m| = |Sig^P|_p|$
 - $|Enc^M|_m| = |Enc^P|_p|$
 - if all (r^M, a^M) match (r^P, a^P)

Runtime: $O(|Sig^P + Enc^P|^4)$

Using PTMA event-based

- Creating Sig^M and Enc^M successively while reading SOAP message
- While reading Header:
 - Check $|T^M| \leq |T^P|$ (token maximum requirement)
- At the end of Header and if reading reference:
 - Run PTMA (omitting references targeting body elements)
- At message end:
 - Run PTMA

Summary

- Protection of Web Services essential
- WS-SecurityPolicy decision and enforcement is an important “brick”
 - Assisting DoS prevention in firewall systems
 - Strict policy interpretation
 - PTMA solving token mapping problem
 - WS-Addressing for message signature identification
- Outlook:
 - Message originator authentication for DoS protection
 - Protection for stateful Web Service protocols (BPEL)

Enforcing and Validating WS-SecurityPolicy for Web Service Firewalls

Thank you for your attention!

Contact: ngr@informatik.uni-kiel.de