

# Performance of Web Service Security

Hongbin Liu, Shrideep Pallikara, Geoffrey Fox  
Community Grids Lab, Indiana University  
Presented by Hongbin Liu

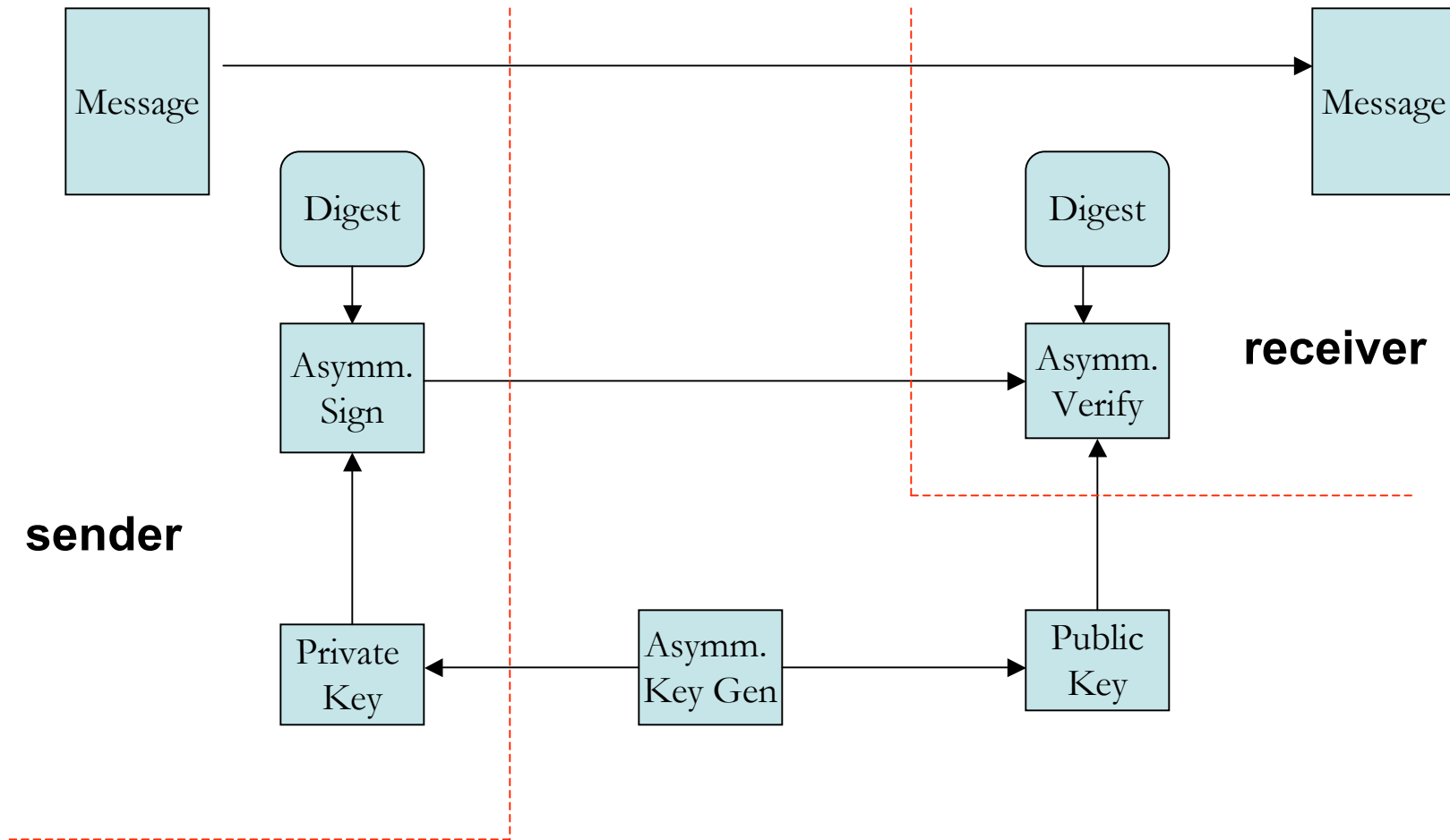
# About the paper

- Address concerns about the performance overheads of security elements in Web services
- Use Java cryptography and open source implementation for test and measurement
- Design experiments that put size and structural complexity of XML document into consideration

# A quick warm-up: general introduction

- Web services: XML messaging, service description, service publish and discovery. Keyword is interoperability and recall RPC, Corba, Java RMI, etc.
- SOAP can be seen as communication layer at the bottom of WS stack
- XML Digital Signature and Digital Encryption
- WS-Security utilizes XML Signature and Encryption, provides flexible and extensible mechanism for SOAP security
- Recall SSL and VPN. Point-to-point solution
- Wss4j package is an Java implementation of WS-Security
- Java cryptographic packages: `java.security.*`, `javax.crypto.*`, `javax.crypto.interfaces.*`, etc.

# Digital Signature



# Sign an XML

```
public Document sign(String input [required],
                    PrivateKey my1 [required],
                    PublicKey my2 [required],
                    PublicKey yourkey [optional],
                    PartsToSign expression [optional],
                    AlgorithmSpecs algoset [optional] ) {
1. parse input string to get a soap Document tree
2. locate nodes PartsToSign
3. apply hash algorithm in AlgorithmSpecs to PartsToSign
4. apply signing algorithm in AlgorithmSpecs to hash value
   4a. generate a symmetric key on the fly
   4b. sign with the symmetric key using Message Authentication Code
   4c. encrypt the the symmetric key with receiver's public key
   4d. insert encrypted key as a node in the tree and discard the symm. key
5. insert signature as a node in the tree
6. insert PublicKey as a node in the tree
7. return tree Document
}
```

# Encrypt an XML

```
public Document encrypt(String input [required],
                        PublicKey yourkey [required],
                        PartsToEncrypt expression [optional],
                        AlgorithmSpecs algoSet [optional] ) {
```

1. parse input string to get a soap Document tree
  2. locate nodes PartsToEncrypt
  3. apply encryption algorithm in AlgorithmSpecs to PartsToSign
    - 3a. generate a symmetric key on the fly
    - 3b. encrypt PartsToSign with the symmetric key to get encrypted data
    - 3c. encrypt the symmetric key with receiver's public key
    - 3d. insert encrypted key as a node in the tree
  4. replace PartsToEncrypt with encrypted data in the tree
  5. return tree Document
- ```
}
```

```

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Header> <wsse:Security soapenv:mustUnderstand="1"
    xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
    <xenc:EncryptedKey>
      <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
      <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#"><wsse:SecurityTokenReference>
<ds:X509IssuerSerial><ds:X509IssuerName>
      CN=Werner,OU=Apache WSS4J,O=Home,L=Munich,ST=Bayern,C=DE </ds:X509IssuerName> <ds:X509SerialNumber>2
</ds:X509SerialNumber> </ds:X509IssuerSerial> </wsse:SecurityTokenReference> </ds:KeyInfo>
      <xenc:CipherData> <xenc:CipherValue>
        hbRtS5lo0OFetpdgJIG+TpYu8ozBh+X1lJ5d99iXzjlo5CxJi8EVnY/ADSQ9U5ToDI34NHsK1q52
        uVc49ecWAMB9Lg1OuROZxemHm2tSnjA+Xh2d9Sbdc7ymxxL3KpLTIFWyHHBHQry6QU37sv2J6gBM
        dCXAvGKYgOG9Jfd583A= </xenc:CipherValue>
      </xenc:CipherData>
      <xenc:ReferenceList> <xenc:DataReference URI="#EncDataId-1340650"/> </xenc:ReferenceList>
    </xenc:EncryptedKey>
  </wsse:Security>
</soapenv:Header>

<soapenv:Body> <xenc:EncryptedData Id="EncDataId-1340650" Type="http://www.w3.org/2001/04/xmlenc#Element">
  <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc"/>
  <xenc:CipherData> <xenc:CipherValue>
    xyc2j6FOum3jIYVfelcs1MAInG5D8HVjcaShLh1E6oq2wxjSNsjYQgWstJPbjCik7l/aCww+OZk
    +DA86nmPLV9e+vecb6C7+6tk0i4+RNL8xHVvvgbJhcVm3jBkHxG7eJeCr5u8fPeT0OshX6an89fSH
    V9gSzOnn3S20qScWA00slw1lOGTKYPSjetQFtLf2zuAR5dYkGrQyXoVHTNp1ndF0BIFKGCRuTiZx
    v4tykHK5NpIKei+1H9tlMOWDkCznyA6yVf0DqLhM7gCWtSrQGoU9/F+33Ba1b7AedsDOYGKiU1u
    QY5mxoLKy1EoivsBd5+5v1bCLr7ulSE7DzHkQJnbKArpxmbghQnc5oy0jzPjh0CyxsSkp/vfVbi9
    adtx </xenc:CipherValue></xenc:CipherData>

  </xenc:EncryptedData>
</soapenv:Body> </soapenv:Envelope>

```

# Important issues

- Document Object Model consumes a large memory but allows random access
- XML canonicalization addresses the fact that logically equivalent XMLs can have different physical representations in bytes
- Speed differences in cryptographic algorithms and their implementations
- Input configuration: 25 soaps, [1-5] size, [6-15] parallel sibling nodes, [16-25] nested sibling nodes

# Theory in the design of experiments

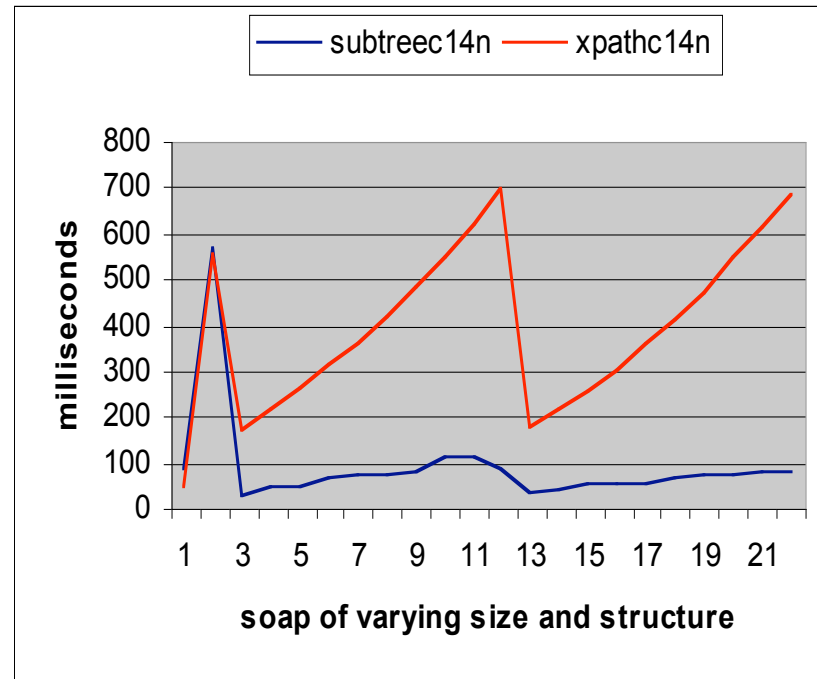
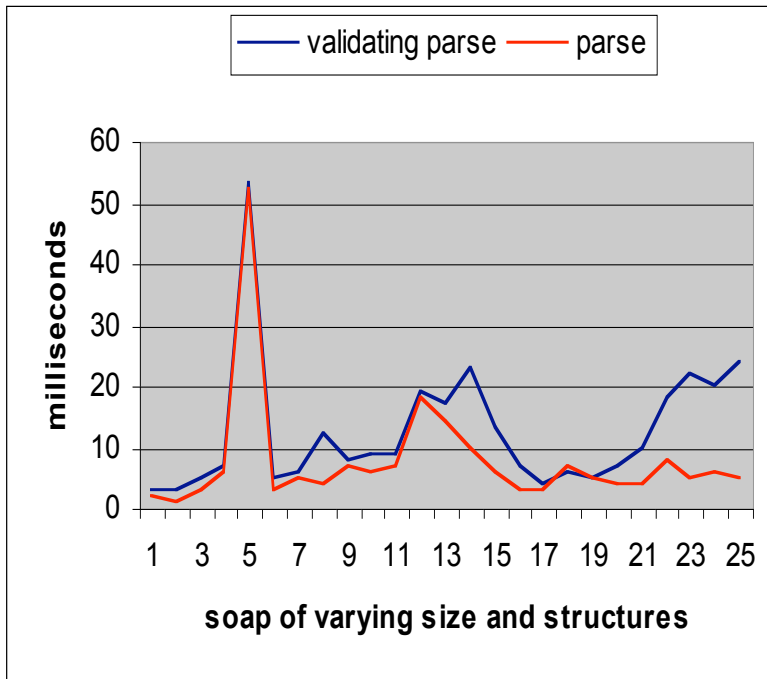
- Experiments are local so no data about latency or throughput is collected as transmission overhead is a separate issue and predictable
- Time is conceptually spent on cryptographic operations such as encryption and decryption, hashing, signing and verification, and XML processing including parsing, validation, transformation, canonicalization, etc
- Separate size documents from structural documents so that we can know if overheads are dependent on difference in sizes and structures

# Measurement results

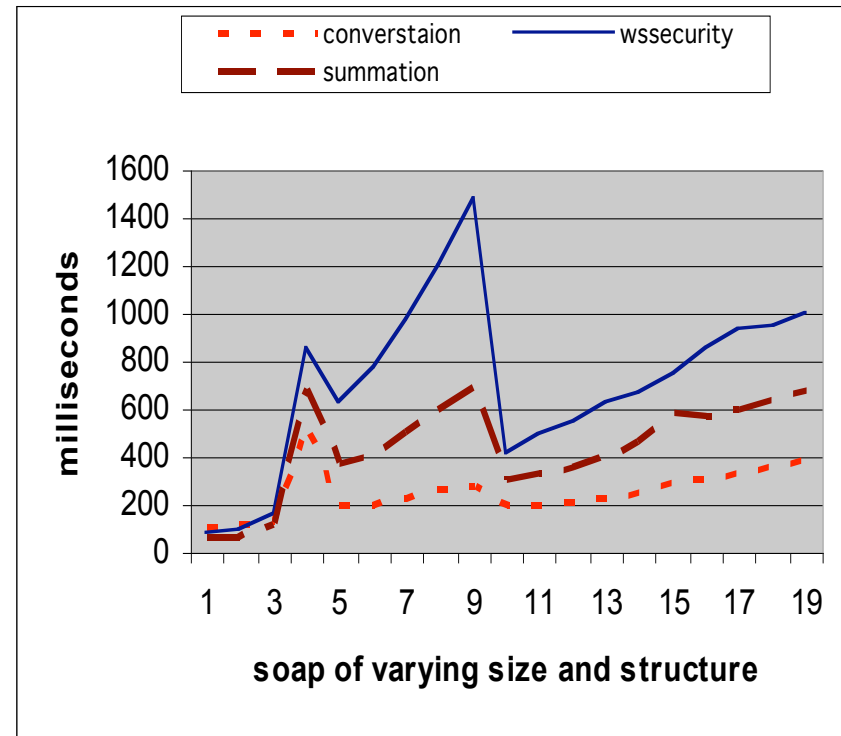
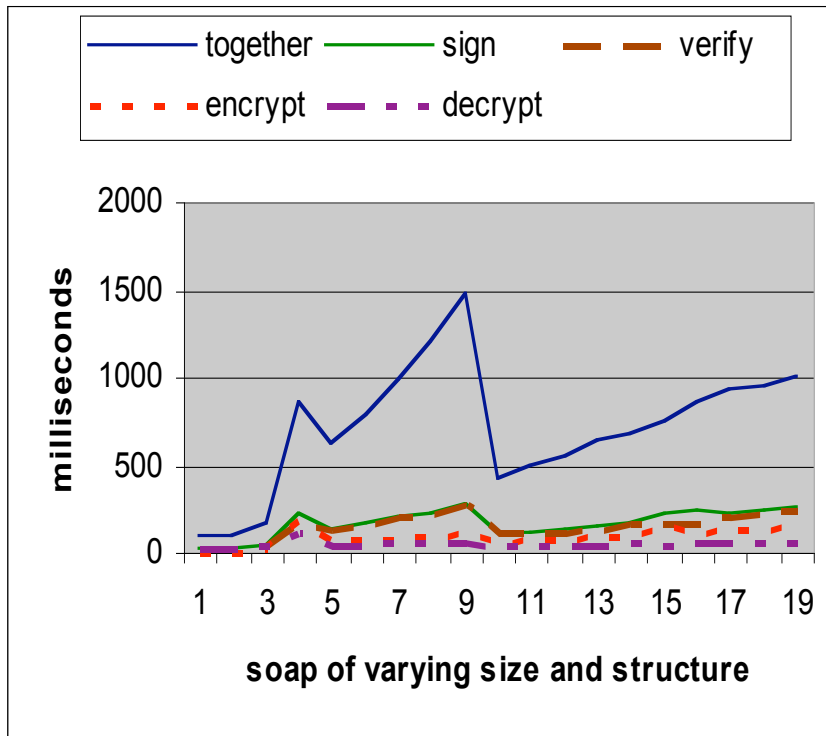
| size        | 100   | 1000  | 10000 | 1e+05 | 1e+06 | 1e+07  |
|-------------|-------|-------|-------|-------|-------|--------|
| rsa<br>sha1 | 20.09 | 19.16 | 19.15 | 18.87 | 27.96 | 109.88 |
| aes         | 0.15  | 0.16  | 0.47  | 3.12  | 37.13 | 426.73 |

- Java cryptography
- Soap processing
- WS-Security and WS-SecureConversation

# SOAP processing



# WS-Security results



# Bottleneck

- DOM implementation: speed and memory leak
- C14n, the excerpt from W3C specification demonstrates
  - Empty element conversion to start-end tag pair
  - Normalization of whitespace in start and end tags
  - Relative order of namespace and attribute axes
  - Lexicographic ordering of namespace and attribute axes
  - Retention of namespace prefixes from original document
  - Elimination of superfluous namespace declarations
  - Addition of default attribute

## Input Document

```
<!DOCTYPE doc [<!ATTLIST e9 attr CDATA "default">]>
<doc>
  <e1 />
  <e2 ></e2>
  <e3 name="elem3" id="elem3" />
  <e4 name="elem4" id="elem4" ></e4>
  <e5 a:attr="out" b:attr="sorted" attr2="all" attr="I'm"
    xmlns:b="http://www.ietf.org"
    xmlns:a="http://www.w3.org"
    xmlns="http://example.org"/>
  <e6 xmlns="" xmlns:a="http://www.w3.org">
    <e7 xmlns="http://www.ietf.org">
      <e8 xmlns="" xmlns:a="http://www.w3.org">
        <e9 xmlns="" xmlns:a="http://www.ietf.org"/>
      </e8>
    </e7>
  </e6>
</doc>
```

## Canonical Form

```
<doc>
  <e1></e1>
  <e2></e2>
  <e3 id="elem3" name="elem3"></e3>
  <e4 id="elem4" name="elem4"></e4>
  <e5 xmlns="http://example.org" xmlns:a="http://www.w3.org" xmlns:b="http://www.ietf.org"
  attr="I'm" attr2="all" b:attr="sorted" a:attr="out"></e5>
  <e6 xmlns:a="http://www.w3.org">
    <e7 xmlns="http://www.ietf.org">
      <e8 xmlns="">
        <e9 xmlns:a="http://www.ietf.org" attr="default"></e9>
      </e8>
    </e7>
  </e6>
</doc>
```

# Recent XML standardization

- Binary xml, Fast Infoset and ASN.1
- Mixed content solutions: SwA, DIME, MTOM/XOP
  - One angle looking at those activities: embedding binary data within XML. These data can be audio/video or scientific arrays
  - Conversion back and forth floating point and ascii.

# Current and future work

- Implement WS-SecureConversation in a way that makes it multi-party capable
- Secure multicast (group) communication that is built upon an overlay network

Thanks!

Questions?