

Providing Web Services Security: A Case Study on Multiparty Service Delivery

Joana Sequeira Torreira da Silva

Roch Glitho

Ferhat Khendek

Agenda

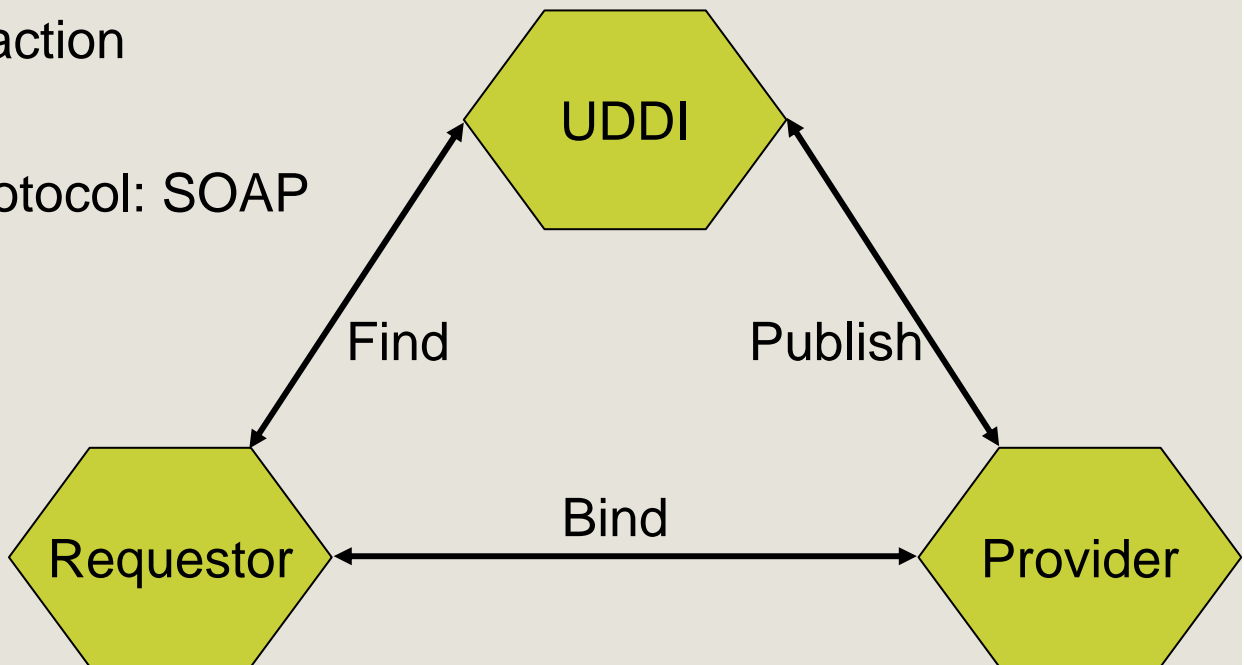
- Introduction
- Background and State of the Art
- Security Web Services
- Case Study & Prototype
- Tests, Results & Analysis
- Lessons Learned
- Conclusion
- Future Work

- 3G networks
 - Networks open to 3rd party providers
- Security
 - authentication, authorization, data integrity & confidentiality, non-repudiation, mitigation of Denial-of-Service threats, policies, key management



Web Services

- Definition
 - software system designed to support interoperable machine-to-machine interaction over a network [W3C]
- Properties
 - High level of abstraction
 - Light coupling
 - Communication protocol: SOAP





Security

- Security Levels
 - Transport vs. Message vs. Application
- WSS
 - Data integrity and confidentiality
 - XML-Dsig & XML-enc

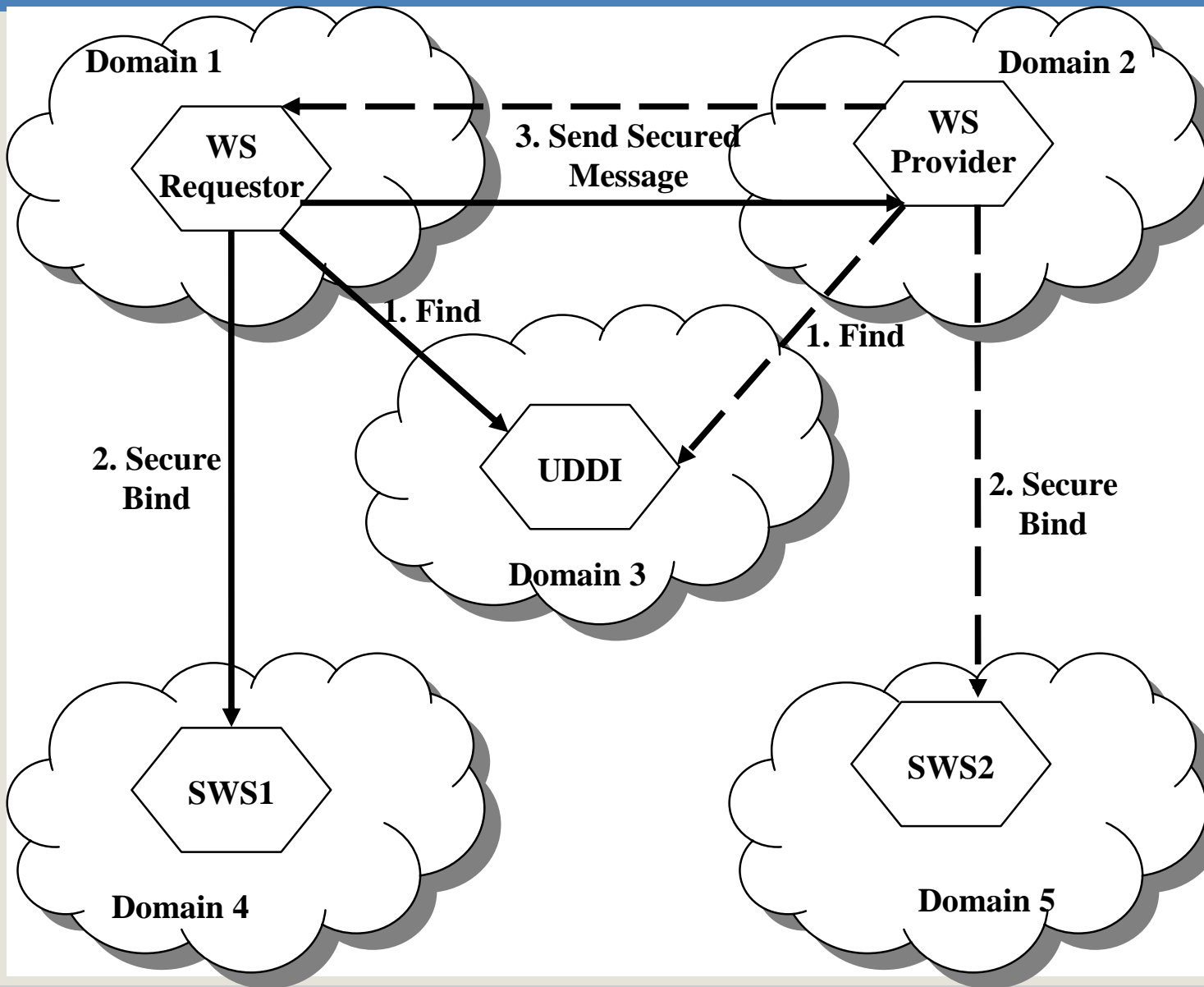
Libraries

- High resource
- Low level abstraction
- Knowledge required

Proxies

- Lack of flexibility
- Lack of developer control
- Pre-configuration

Security Web Services (SWS) Our Solution



- 2 new WS interfaces
 - No pre-configuration
 - Necessary information is passed on the moment
- elementSecurity interface
 - 1 SOAP element at a time
 - Enforces or checks security
 - Timestamps, signs, encrypts, or any combination
- secureSendRemove interface
 - Secures 1 element (stamps/signs/encrypts)
 - Sends request
 - Receives response
 - Checks security on response (stamp/signature/decrypts)
 - Returns Response in the clear

- Providing WSS through high level WS

Pros

- +Domain independent
- +Non-local resources
- +Little security knowledge required
- +No preconfiguration
- +Low coupling
- +New player in the business model

Cons

- Secure Bind (not addressed)
- Time Delay
- Network Load

Case Study and Prototype

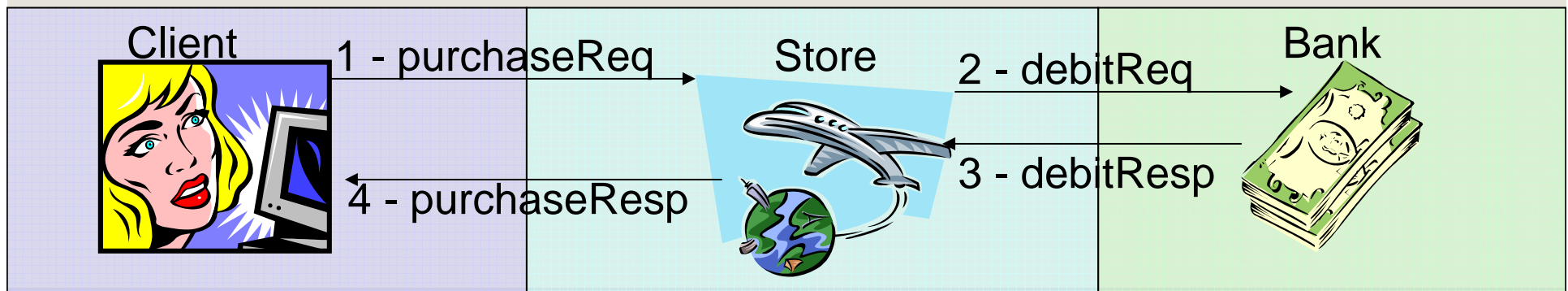
The Case Study

Scenario

- A Client wishes to buy from a store
- She must send her credit card information to the store
- She does not want the Store to see/change this information
- The Store wants long term proof of client approval
- The Bank wants long term proof of both the Client's and the Store's approval

Context

- All in the same domain
- Store, Bank and SWS already known, no need for UDDI



The Prototype

- **Computers:**
 - P4 DELL: Windows XP, 60 GB HD, 512 MB RAM, 2.2GHz
- **BEA WorkShop 8.1.2**
 - Issues: in-out parameters & security only at lower level
- **Libraries:**
 - javax.crypto, javax.xml.soap, java.Security, java.net.URL, java.util.Iterator,
 - weblogic.jws.proxies, webserviceclient.jar

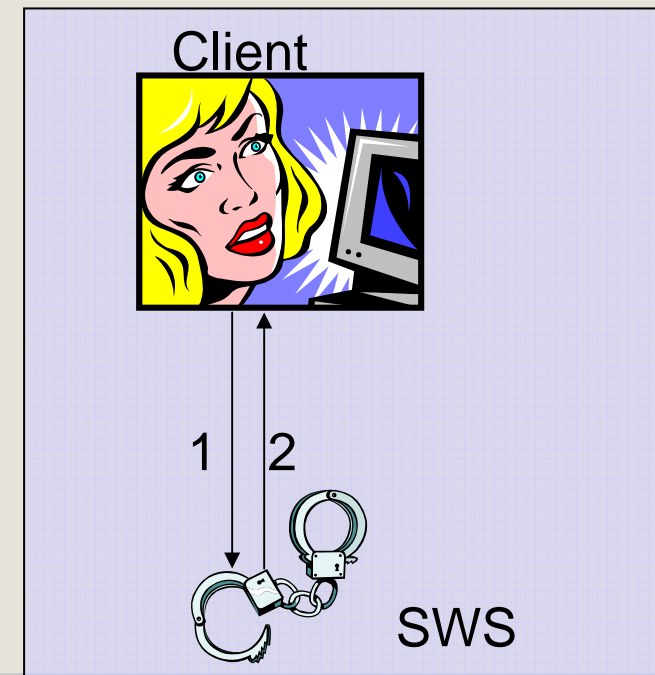
Tests, Results and Analysis

The Tests

- Individual Test
- Chain Test (secured requests)
- Full Chain Test (secured requests and responses)

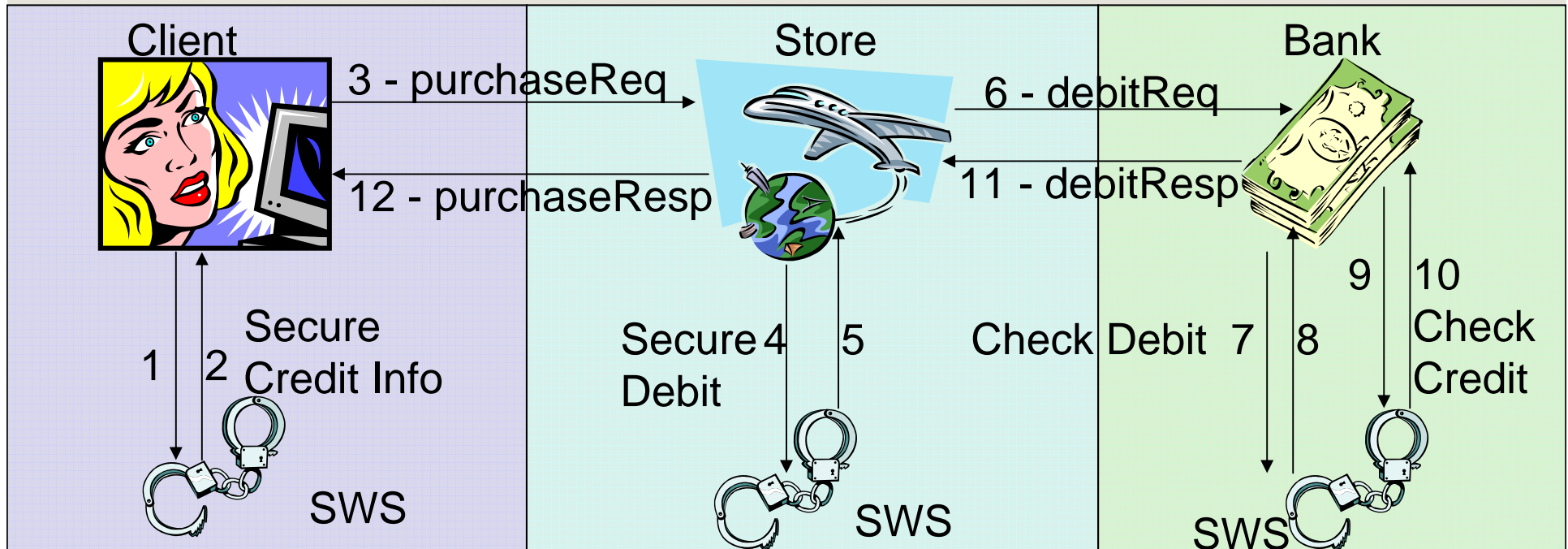
Individual Testing

- The client secures the Credit Card Information
- Uses elementSecurity only



Chain Testing using elementSecurity

- Only the requests are secured

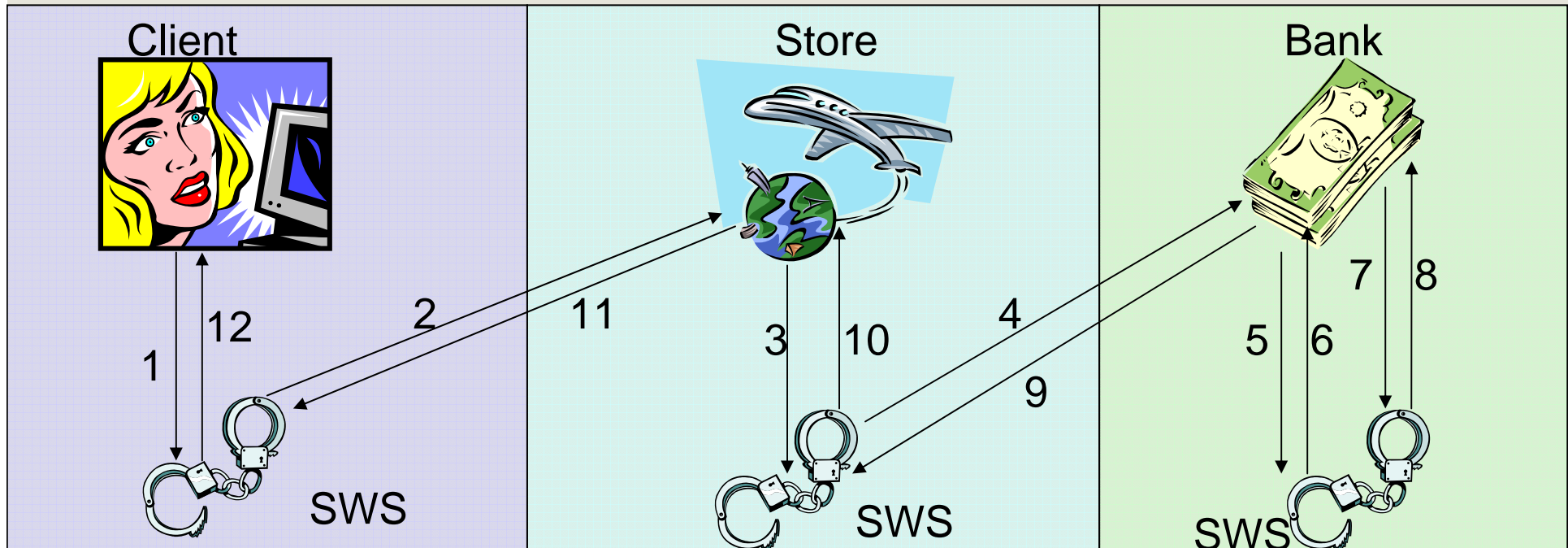


Full Chain Testing using elementSecurity

- 8 more messages required

Chain Testing using secureSendRemove

- Only the requests are secured



Full Chain Testing using secureSendRemove

- Requests and responses are secured
- 4 more messages would be required

	Lib (ms)	SWS (ms)	SWS – Proxy (ms)	Overhead (ms)	
				SWS	SWS - Proxy
Time Stamp (in)	~0.1	38.7	N/A	38.6	N/A
Time Stamp (out)	2.7	37.1	N/A	34.4	N/A
Encode	10.5	50.5	N/A	40	N/A
Decode	17.5	N/A	N/A	N/A	N/A
Sign	7.3	41.9	N/A	34.6	N/A
Check Signature	20.8	N/A	N/A	N/A	N/A
Timestamp (in), Encode And Sign	20.4	53.5	N/A	33.1	N/A
Chain Time Stamp	147.1	300	325	152.9 > 146.2 = 38.7*2 + 34.4*2	177.9 > 146.2
Full Chain Time Stamp	166.9	507.2	446	340.4 > 296.6 = 38.7*4 + 34.4*4	279.1 < 296.6



	Lib (KB)	SWS (KB)	SWS – Proxy (KB)	Overhead (KB)	
				SWS	SWS – Proxy
Time Stamp (in)	0	4.7	N/A	4.7	N/A
Time Stamp (out)	0	5.2	N/A	5.2	N/A
Encode	0	5.5	N/A	5.5	N/A
Decode	0	N/A	N/A	N/A	N/A
Sign	0	N/A	N/A	N/A	N/A
Timestamp (in), Encode And Sign	0	N/A	N/A	N/A	N/A
Chain Time Stamping	5	35.5	34	30.5 > 19.8 = 4.7*2+5.2*2	29 > 19.8 = 4.7*2+5.2*2
Full Chain Time Stamp	6.8	54	48.8	47.2 > 39.6 = 4.7*4+5.2*4	42 > 39.6 = 4.7*4+5.2*4

- Time delay decreases as number of features applied increases
 - Message transmission delay
 - Parsing SOAP message to String and back
- elementSecurity
 - Most advantageous when not all messages are secured
- secureSendRemove interface
 - Most advantageous when requests and responses are secured

- Currently, the SWS API's cannot be used
 - Many of today's Application Servers do not fully support in-out parameters
 - Many of today's Application Servers do not fully support the use of XML-enc and XML-Dsig in the upper layers
- The time delay introduced is not prohibitive
 - Less than 400ms to secure a 3-party service
 - Decreases as number of security features applied in one call increase
 - NB: Services in prototype were “dummy/empty” services
- Network load is not prohibitive for small messages

- A new way of providing WSS
 - Architecture
 - 2 new WS definitions
 - Case Study
 - Tests and Results Analysis
 - Time delay overhead
 - Network load overhead

- SWS in a non-trusted domain (SecureBind)
- More accurate measurements when:
 - Support of in-out parameters
 - Support of XML-enc and XML-Dsig at high levels
- Impact on network load
 - as need for secured WS increases

