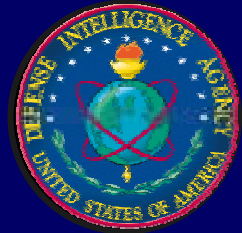


Solutions for Web Services Security



Lessons Learned in a
Department of Defense
Program

Kevin T. Smith
McDonald Bradley, Inc.
Chief Security Architect,
The Virtual Knowledge Base

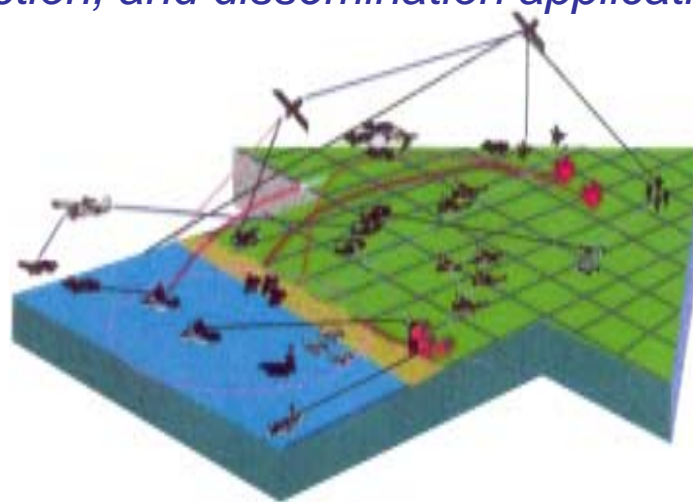


Background – Virtual Knowledge Base



The Mission: To Enable Decision Superiority

“The Virtual Knowledge Base (VKB) is an interoperability framework that will provide seamless, distributed and meaningful access to a virtual enterprise repository of structured, semi-structured and unstructured data for intelligence support. The future VKB data environment will be accessible to all production, collection, and dissemination applications and systems.”



An Interoperability Framework

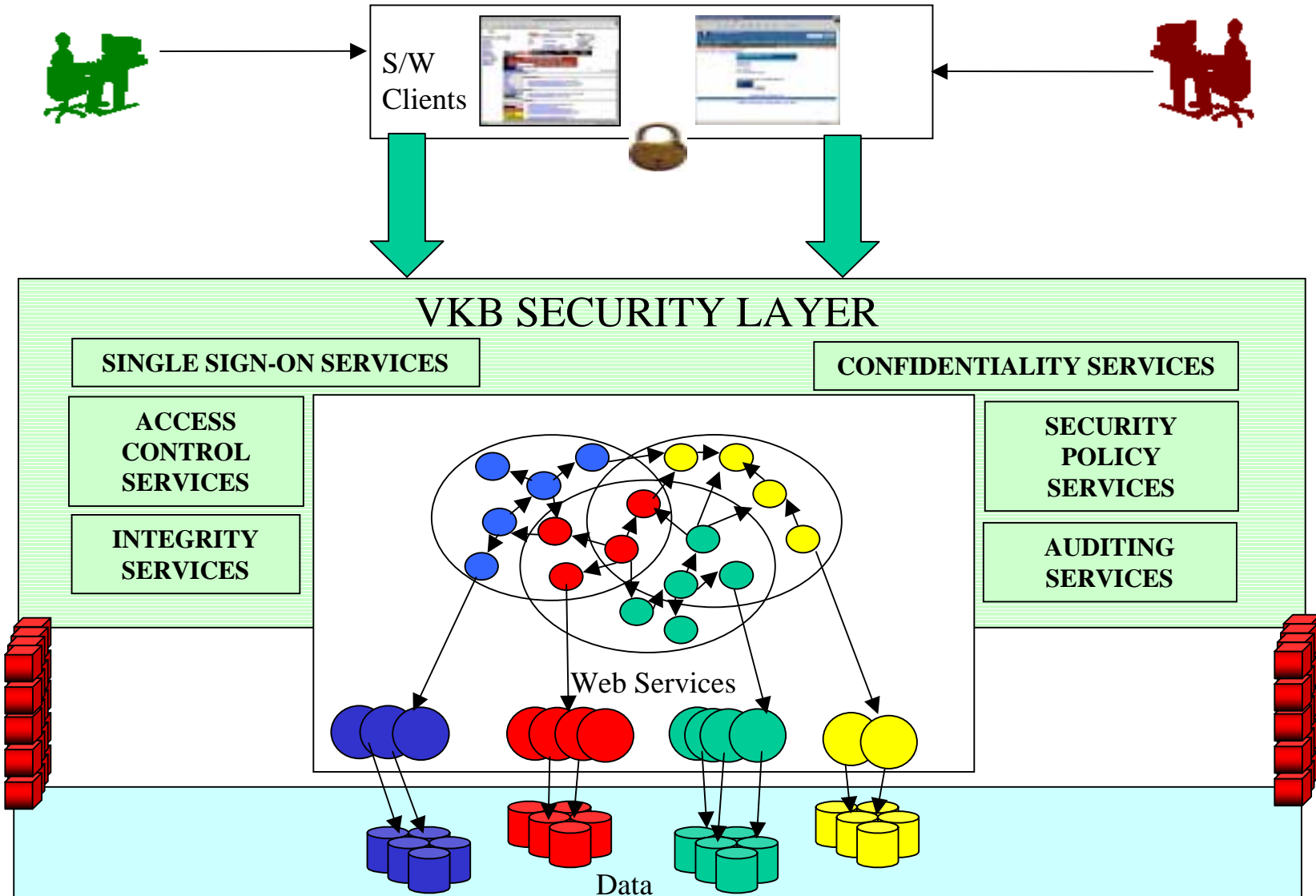
Based on Web Services

Background - What is the VKB?

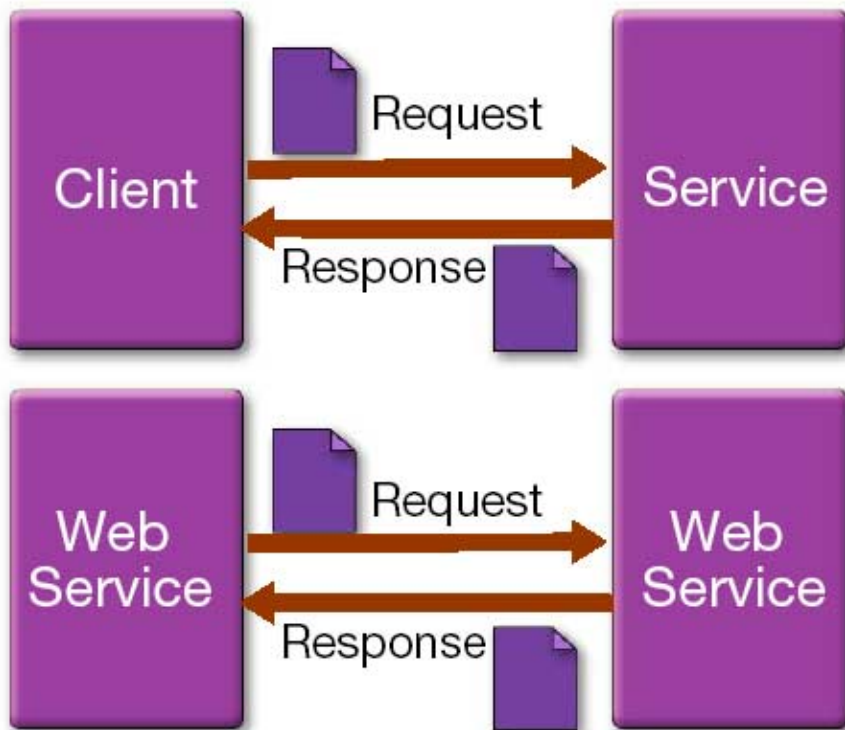


- Virtual Knowledge Base
 - **Virtual** – provides an interoperability framework among data sources
 - Clients are other systems (portals, applications)
 - **Knowledge** – provides a “semantic web” of data sources
 - Catalogued via a registration process.
 - Authoritative systems/resources
 - From discovery to custom data assembly to globally linked, annotated data.
- The VKB exposes data sources as web services
- Because it is a DoD program, there are serious security requirements.

Security and the VKB



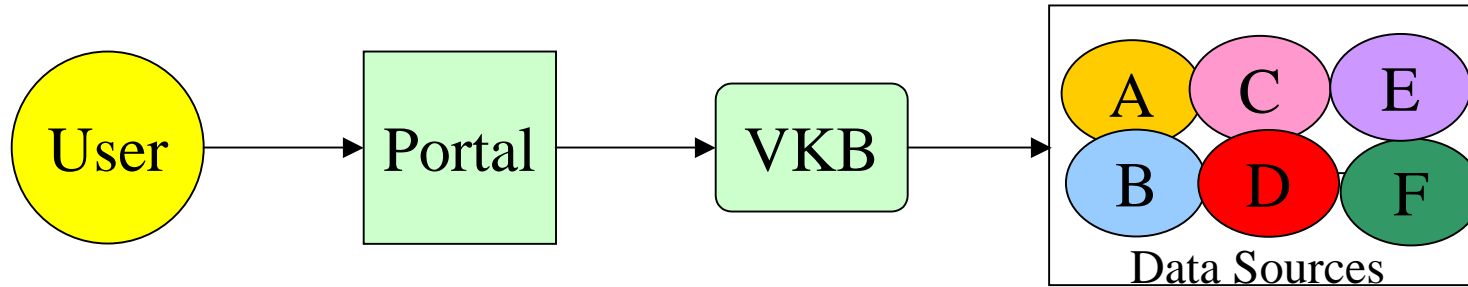
A Security Challenge in Web Services



- In old client-server model, there was a one-to-many relationship between client & servers
- Now, many-to-many model!
 - Users talk to portals who talk to web services that talk to web services that talk to data sources!

Bottom line for security: more nodes to protect, trust propagation issues, Single Sign-On

Problem 1: Trust Propagation and SSO

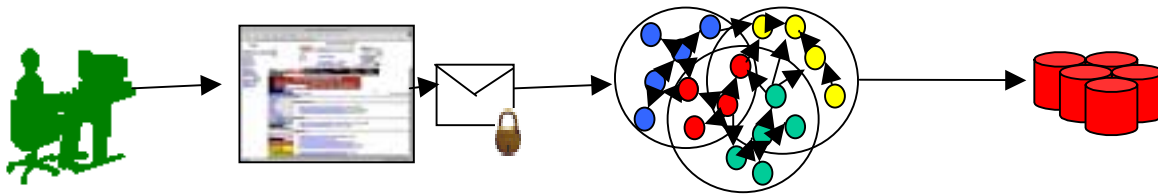


- There are numerous layers between the user and the eventual datasource
 - How does the datasource have assurance that the user authenticated or what his credentials are?
 - If Datasources A, B, C, D, E, and F have different login credentials for the user, we don't want to force the user to log in 6 times!!!
 - The VKB needs high assurance of who the end-user is, and what he/she can do!
 - We would like a standards-based solution!

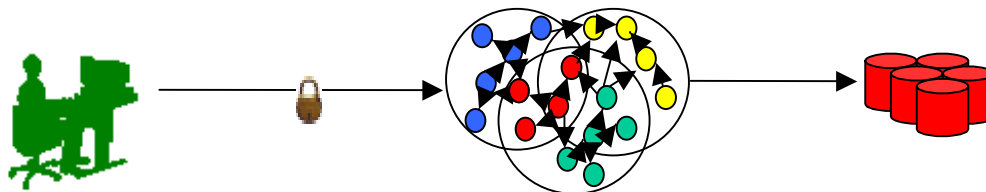
Problem 2: Confidentiality



- 1) In some scenarios, encryption needs to happen from the client to the eventual data source without any decryption along the way (the VKB can't see the confidential data)
 - Point-to-Point Bulk Encryption (I.E SSL between nodes) can not work here – need to encrypt from the user directly to data source, and vice-versa



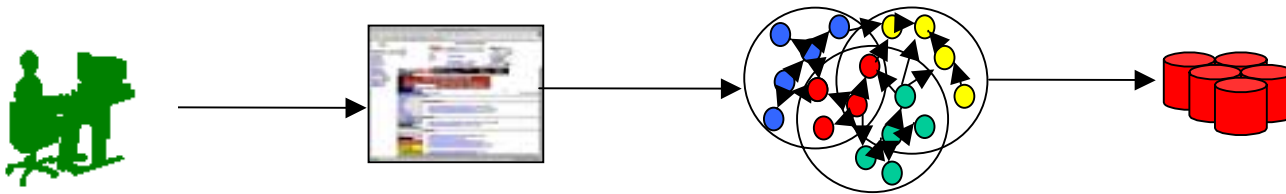
- 2) In scenarios where the network is not trustworthy, all data must be bulk encrypted. (ex: analyst using PDA over an 802.11 network)
 - Point-to-Point Bulk Encryption between nodes is acceptable for this scenario



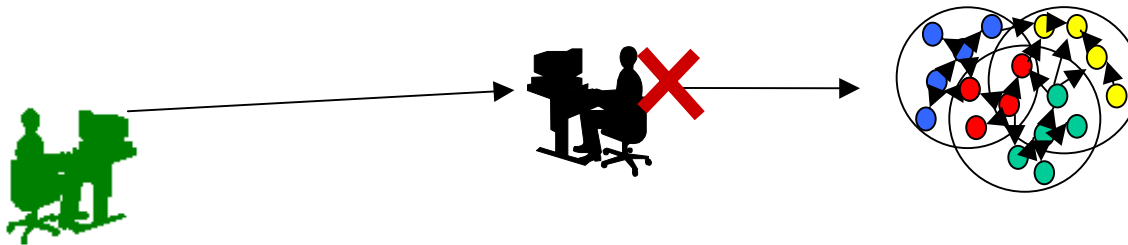
Problem 3: Non-Repudiation & Integrity



- 1) Analyst logs into portal, and the data source needs to have non-repudiated assurance that the user has been authenticated before sending confidential information



- 2) Data source (or user) needs solid assurance that the message has not been altered in transit by malicious agent

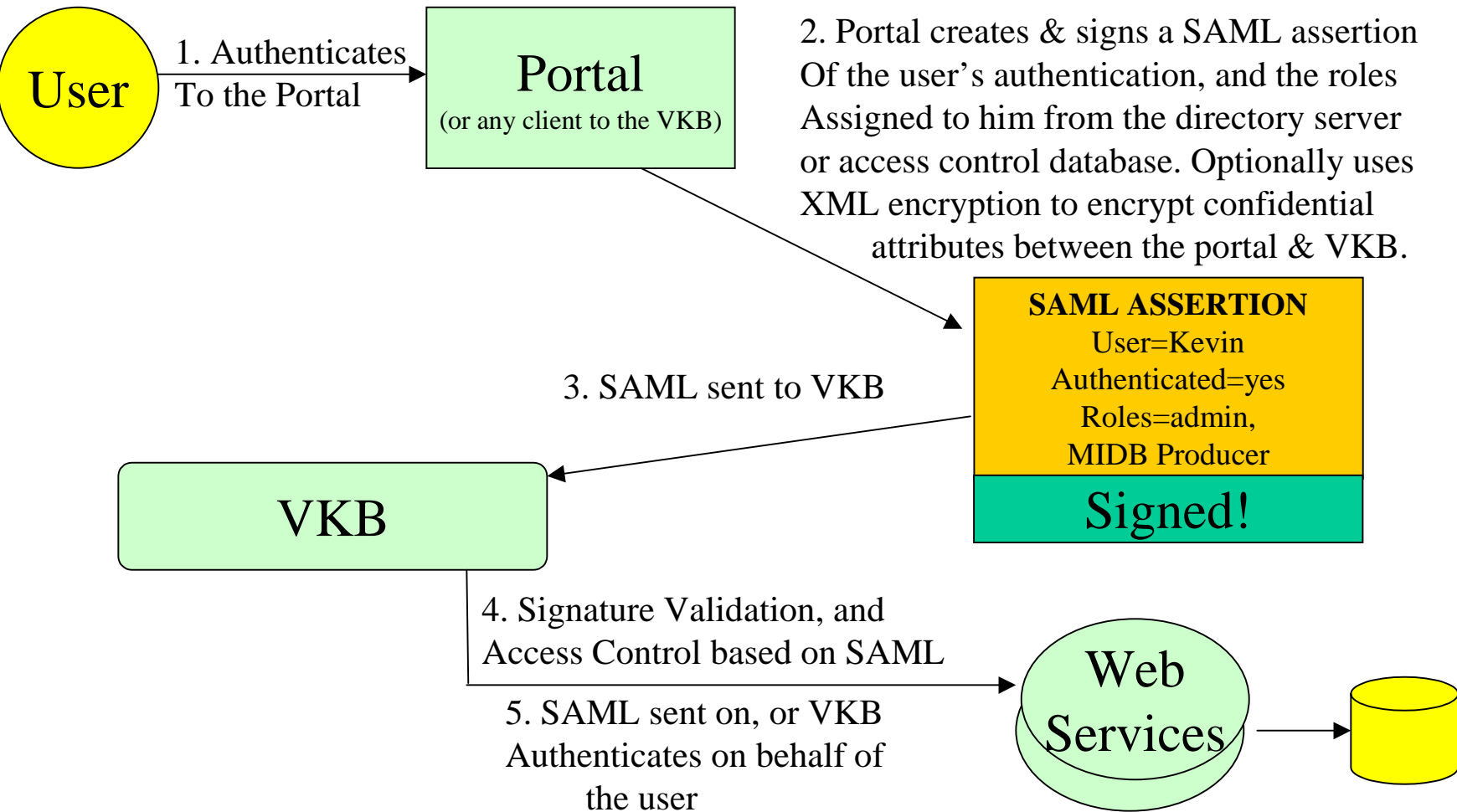


A Technology Enabler: SAML



- Security Assertion Markup Language
 - Uses “assertions” – an application or service can “assert” that the user is who he says he is (authentication assertions), and the application can list that user’s privileges (authorization assertions).
 - Infrastructure can be P2P, or you can have a centralized trust authority..
 - **Enabling Solution for Single Sign-On in Web Services**
- Growing Support for SAML
 - Liberty Alliance embraces SAML in Project Liberty
 - Microsoft and IBM (and their competitors!) have announced SAML support
- Used with XML Signature (signed SAML assertions), you can provide non-repudiation of authentication and authorization credentials.
- As individual SAML attributes may be encrypted with XML Encryption, can provide confidential security attributes

The VKB SAML Solution: Passing Roles & Credentials for Deeper Access Control



Non-repudiation (and Integrity) of the portal's authentication and authorization of the user
Confidentiality of some SAML attributes;

Solves Different Requirements For Different Data Sources



- Requirement: The VKB should control & log access with high assurance
 - Solution: Use this SAML solution that validates digital signature on the VKB end, controls access to the web service, and logs it.
 - Requires no change to existing web services !
- Requirement: The VKB should authenticate to the data source on behalf of the user (with the user's credentials)
 - Solution: The portal passes in username/password (encrypted) in the SAML Header and signs it. The VKB passes the username/password to the web service, and the web service authenticates to the datasource with this information
 - Requires minimal change to web services that need this functionality
- Requirement: The final endpoint (data source) wants the SAML to handle access.
 - Solution: pass SAML right on through..



Prototype Details

Prototype Purpose



- Determine the feasibility of using SAML with SOAP to accomplish Single Sign-On, Trust Propagation, and Role Based Access Control at the Web Service Level
- Determine the performance impact of using XML Signature with SAML and SOAP
- Determine the performance impact of using XML Encryption to encrypt confidential SAML attributes in SOAP messages
- Develop a system that can be enhanced and re-used with the VKB

Software Packages Used in Prototype



- Prototyping Portal to VKB Communication
 - Use Tomcat 4.1.12 LE and JSP pages for Test Portal
 - Use Apache Axis with Tomcat (same version) for Test VKB
- Core Prototype Uses Apache Axis Web Services
 - Uses an Axis Handler to perform
 - Signature & Timestamp Validation
 - Authentication and Authorization Control
 - Decryption, if Appropriate
- Uses Verisign's Trusted Services Integration Kit (TSIK API)
 - Has Beta SAML API
 - Has XML DSIG (XML Digital Signature) capability
 - Has XML Encryption API
- Plugs right into the VKB

Implementation: SAML in The SOAP Header & Modularity



- The VKB had existing web services, and we wanted to add this security without disturbing the existing interfaces
- So we place the SAML Assertion in the SOAP Header!

```
<SOAP:ENV>  
  <SOAP:Header>  
    <SAML:Assertion>...</SAML:Assertion>  
  </SOAP:Header>  
  <SOAP:Body>  
    <vkb call here.../>  
  </SOAP:Body>  
</SOAP:ENV>
```

- Standard Servlet Filters allow us to have a “handler” see the message before the web service does!
- We can use the handler for some or All web services
- Minimal Changes to Web Service Code with this Model

SAML Client (Portal) – Technical Details



- Created JSP pages that:
 - Get user authentication information
 - Get user's roles
 - Get digital certificate information from keystore (its own, and VKB's)
 - May need to encrypt SAML attributes for confidentiality
 - Create and sign SAML assertions
 - Create a SOAP Message with SAML & Send to VKB
- Much of this should be transparent to web developer
 - Solution: Use JSP Header that is imported from main pages
 - Signed SAML:
 - `<%@include file="signHeader.jsp" %>`
 - Signed SAML with Encrypted Attributes:
 - `<%@include file="signEncrypt.jsp"%>`
 - The headers do all the hard work! The main JSP just builds the SOAP messages & focuses on the GUI stuff

Details of Client Code



Header.jsp

jspInit(): loaded at first init()

- Loads Keystore
- Loads Portal Certificate & Private Key, VKB Public Key
- Creates a SAML AssertionGenerator

Body: loaded every time

- Generates SAML assertion from the affirmed credentials of the web container (username, roles, authentication method)
- If special needs (encryption of attributes, etc), does it here
- SAML is signed with Portal's Private Key

Main JSP

- Creates a simple SOAP call to message service
- Puts the Assertion (generated in header.jsp) in the SOAP Header
- Uses JSP IO & SOAP taglibs to make SOAP call
- Prints The Result

Footer.jsp

- Releases resources

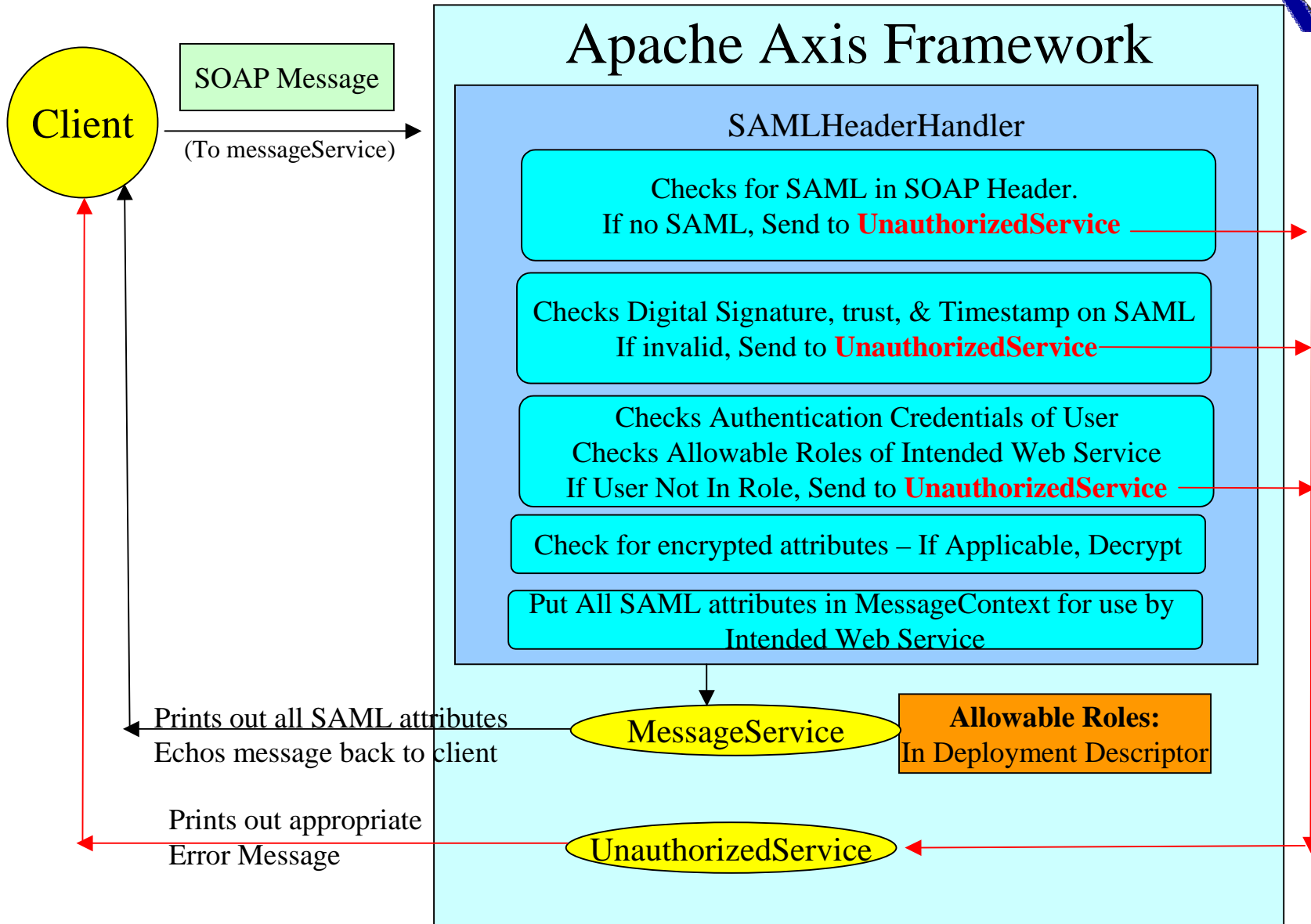
Server Details: Access Control Described in Axis Descriptor



```
<service name="MessageService" style="message">
  <parameter name="className" value="mil.dia.security.saml.axis.MessageService" />
  <parameter name="allowedMethods" value="echoElements" />
  <parameter name="allowedRoles" value="Republican"/> <!-- Our handler will restrict access based on this-->
  <requestFlow>
    <handler type="java:mil.dia.security.saml.axis.SAMLHeaderHandler" provider="java:RPC">
      <parameter name="keystore" value="C:/testkeystore/vkb.ks"/><!-- test keystore for experiment -->
      <parameter name="keystorepassword" value="****"/> <!-- Used to load certs from keystore-->
      <parameter name="vkbpassword" value="****k"/> <!-- used to get VKB private key -->
      <parameter name="timetolive" value="6000000"/>
    </handler>
  </requestFlow>
  <responseFlow>
    <handler type="java:mil.dia.security.saml.axis.SAMLHeaderHandler" provider="java:RPC"/>
  </responseFlow>
</service>
```

- **allowedRoles** parameter – SAML for user must include this role in order to access web service
- SAMLHeaderHandler here provides the security layer

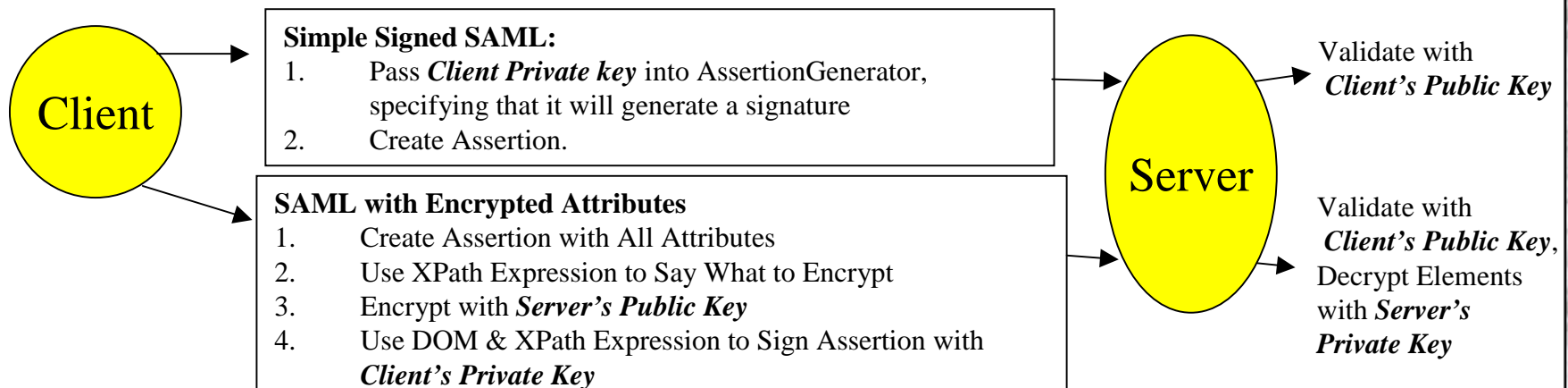
Details of Server Side Code



Details of XML Signature in Solution



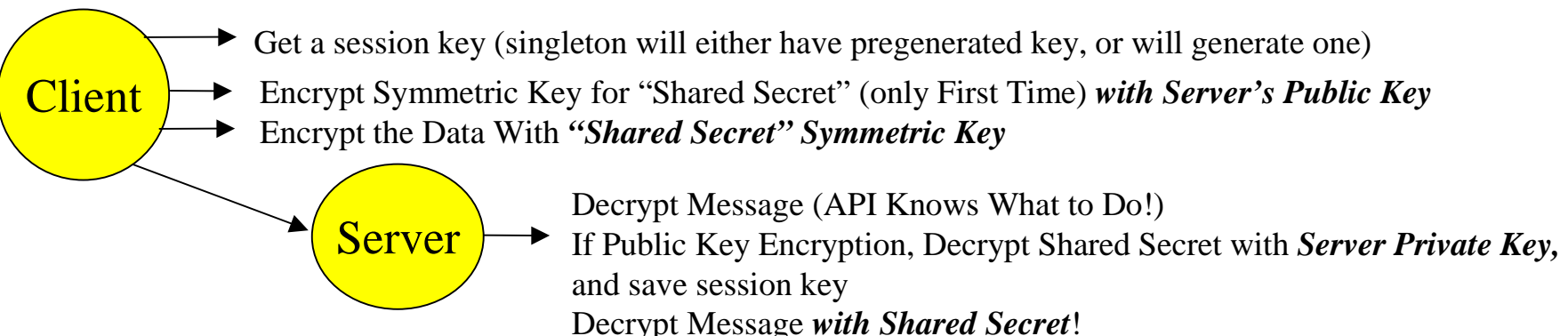
- Used Verisign TSIK 1.5, BouncyCastle APIs
 - RSA Public Key Cryptography (Asymmetric) for Signature
- Performance Much Faster Than Encryption
 - With encryption, the size of the text to be encrypted is proportional to the time it takes to encrypt it!
 - For signature, we only sign a hash that verifies the integrity of a message
- Signature is Built In to the TSIK SAML APIs
 - For SAML messages that simply sign messages, only one method call is done to do a signed assertion
 - For SAML messages where we want to encrypt before we sign, we must do trickier work with Signature, DOM & XPath APIs.



Details of XML Encryption in Solution



- Used Verisign TSIK 1.5, BouncyCastle APIs
 - RSA Public Key Cryptography (Asymmetric) for Encryption
 - Triple DES Encryption (Symmetric)
- Data Must Be Encrypted with Symmetric Key
 - Can Be Randomly Generated
- XML Encryption can:
 - Encrypt the Symmetric Key w/ Public Key Encryption
 - Encrypt the Data with the Symmetric Key
- This Can Be Slow!
 - Using Public Key Encryption Every Time is Slow!
 - Randomly Generating Symmetric Keys is Slow!
- Can do Public Key Encryption ONCE to establish a session key, as a SHARED SECRET between Portal & VKB!



Notes on Signatures & Encryption



- With Digital Signatures, regardless of the amount of data per message, the hash of the message is what is signed. The speed of digital signature processing should be **constant**, regardless of the size of the message.
- With Encryption, the data is encrypted, so the speed of the cryptography is **relative to the size of the data** being encrypted.
- To do accurate performance comparisons between the encryption/decryption processing and digital signature processing, the same data will be encrypted every time for the encryption test.

Performance Experiment

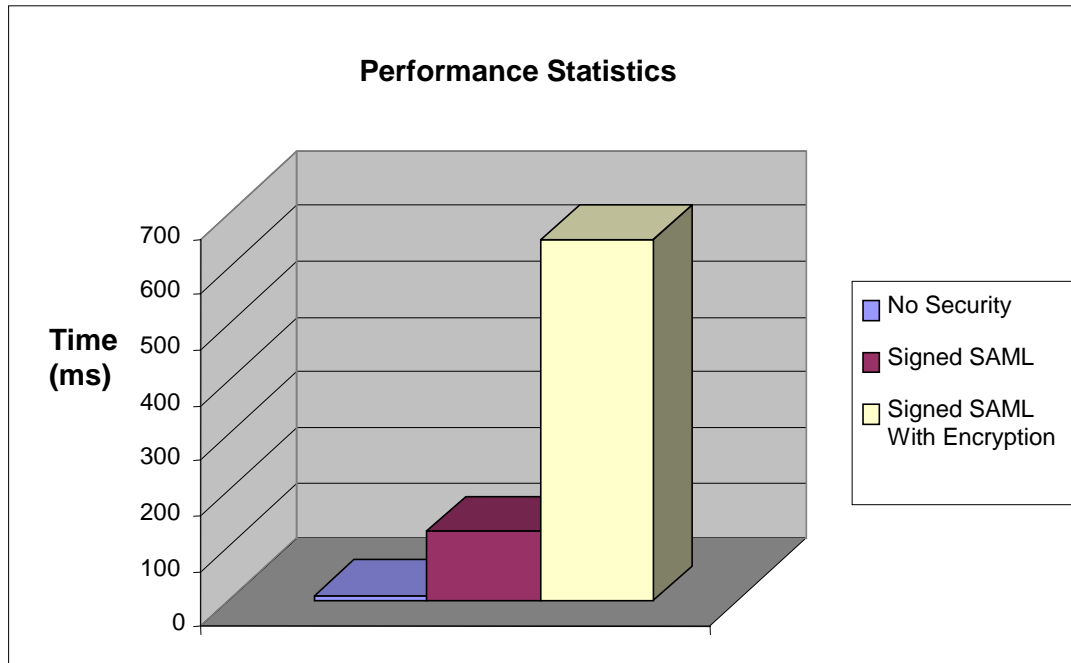


- Configuration Details
 - Pentium 4 Machine 1GHZ, 1 GB RAM, Windows 2000
 - Apache Axis 1.0 Server on Tomcat 4.1.12 LE
 - Portal: JSP-based web pages
 - VKB: Axis Web Services & Axis Handler for SAML
- Experiment Details:
 - Want to measure speed of :
 1. Processing of SOAP Message with No Security (straight to web service)
 2. Processing of SOAP Message with Digitally Signed SAML Header that contains authentication information, roles of the user. For this experiment, the web service will have to validate the signature and restrict access based on role.
 3. Processing of SOAP Message with Digitally Signed SAML Header that contains authentication information, roles for the user, and an encrypted “password” attribute specifically for the data source. For this experiment, the web service will have to validate the signature, restrict access based on role, and decrypt the encrypted attribute.
 - Measured each one alone (not load tested)
 - Ran 3 rounds of 50 messages each

Performance Experiment Findings



1. Regular SOAP messages with No Security:
 - Run-Time Average: **10ms (1/100th of a second)**
 - Note: Time to Compile JSP & Load Classes: 2000 ms
2. Signed SAML with Roles
 - Run-Time Average: **125ms (1/8th of a second)**
 - Note: Time to Compile JSP first time & Load Classes: 10000 ms (10 secs)
3. Signed SAML with Roles & Encrypted Data (#3)
 - Run-Time Average: **650ms (only after shared secret negotiation)**
 - Shared Secret Negotiation (the first time): **1000 ms**
 - Note: Time to Compile JSP first time & Load Classes: 15000 ms



Performance Experiment Findings: Signed SAML



- **The metrics for the metrics of Signed SAML was significant, because:**
 - In 1/8th of a second (125 milliseconds) the following was achieved:
 - Portal capturing user's authentication and authorization credentials
 - Portal creating and signing SAML with Digital Signature
 - Portal Placing SAML in a SOAP message
 - Portal sending SOAP message to web service
 - Web Service Handler validating Digital Signature for proof of user's authentication & roles
 - Web Service Handler validating SAML timestamp to prevent replay attacks
 - Web Service Handler processing SAML information
 - Web Service Handler accepting or denying request to Web Service, based on original user's role (RBAC)
 - Web Service Handler finally sending message to web service
 - Web service processing
 - Web service sending response back to portal
- **Although the speed is roughly 10 times the speed of having no security, the benefits of this are:**
 - Undeniable evidence of the user's original authentication
 - Undeniable evidence of the user's roles and other credentials
 - Security-in-depth, providing trust propagation through the enterprise
 - Preparation for the next stage of web service computing – orchestration
- **Great results for slow experiment machine!**

Performance Experiment Findings: Signed SAML with Encrypted Elements



- **Solves tough issues, but performance is a heavy factor**
 - The first message (initial setup for Portal-VKB communication) is very performance intense, and not used in the 650 millisecond comparison, because it is a one-time penalty:
 - Initial Pseudorandom Session Key Generation (took 1.2 seconds!)
 - Public Key Encryption for Initial Session Negotiation (took .5 seconds!)
 - So including the 650 milliseconds, the first message from the Portal to VKB takes about 2.4 seconds!
 - In more than half a second (650 milliseconds) the following was achieved:
 - Portal capturing user's authentication and authorization credentials
 - Portal Encrypting a confidential attribute, placing it in a SAML attribute
 - Portal creating and signing SAML with Digital Signature
 - Portal Placing SAML in a SOAP message
 - Portal sending SOAP message to web service
 - Web Service Handler validating Digital Signature for proof of user's authentication & roles
 - Web Service Handler validating SAML timestamp to prevent replay attacks
 - Web Service Handler processing SAML information
 - Web Service Handler decrypting the confidential attribute
 - Web Service Handler accepting or denying request to Web Service, based on original user's role (RBAC)
 - Web Service Handler finally sending message to web service
 - Web service processing
 - Web service sending response back to portal
- **SAML with XML Encryption using this technique will not scale.**
- **It is possible that the SAML/WS-Security Combo will achieve better performance (better negotiation, one-time penalties, etc.)**

Performance Results – Scalability Assumptions



- Signed SAML will scale, but need more robust servers
 - Factor of ten vs. no security
- Encrypted SAML Attributes will not scale
 - Factor of 65 vs. no security
 - Accomplishes much, but too much performance hit for doing every transaction this way
- Signed Future specifications may be faster for encrypting confidential attributes
 - Marriage of WS-Security & SAML may solve the key negotiation problems & could create “SAML sessions” per user so that the performance hit is a one-time penalty.
 - Let’s wait and see.



Recommendations

Results: When We Will use Signed SAML



- **We Will Use Signed SAML when**
 - **The data source requires a strong assertion of the user's credentials.** If the data source requires this, it is necessary. (Since the client is the first point in user authentication and authorization, the client needs to sign these credentials as 'proof')
 - **The data source requires logged proof of the authentication and authorization.** Logging the SAML assertion will prove non-repudiation of the portal's trust of the user and the user's credentials.
 - **CYA is necessary.** If the VKB provides access of a user based on a signed assertion from the portal, and the VKB store the signed SAML, it provides legally provable proof (non-repudiation) that the portal checked those credentials with a user and role directory.
 - **We need to provide further access control to web services beyond the portal layer**
 - Because SOAP messages do not carry any credential information, using SAML in the SOAP header will be necessary for points of further access control beyond the portal
 - **We will provide web service orchestration**
 - If you are preparing to combine web services with other web services that have other access control requirements, SAML may be the only way to propagate trust in an orchestration environment.

When we will not use Signed SAML



- **The VKB Doesn't Need Signed SAML when:**
 - **No Access Control is Needed**
 - **When the Portal and VKB are on the same machine.**
 - When the VKB and portal share the same VM, they should both have access to the authenticated user's credentials
 - If logged non-repudiation is necessary, the portal can create signed SAML and log it for the first time the user logs on. If the data source does not need the SAML, it does not have to be passed to the VKB and is only necessary once – for auditing purposes.
 - If the data source needs it, the VKB can grab it from memory & send it on

When Confidential Info Is Passed



- **Consider using SAML within SSL connections as an alternative to Using XML Encryption within SAML:**
 - SSL (TLS – Transport Layer Security) is a proven protocol, Cheap SSL/TLS-specific hardware accelerators can be added for speed.
 - SAML within the SSL accomplish the passing of credentials
 - When you need to pass confidential information to the VKB from the Portal, put it in unencrypted as a SAML Attribute, and send it to the VKB via SSL.
 - If no logging or non-repudiation is necessary, ***SAML doesn't necessarily need to be signed.*** The encrypted connection handles the integrity and confidentiality of the message between points.
- **Only Use SAML with XML Encryption when this alternative won't scale** (multiple nodes, orchestration, etc)

Conclusion



- **This SAML solution accomplishes much**
 - SAML in the SOAP Header provides Single Sign-On for Users in the Community
 - Digital signature provides trust propagation and non-repudiation for solid assurance
 - Enterprise Roles can be placed in SAML as SAML Attributes for RBAC
 - Handlers in the Axis Framework control access to web services. Axis Deployment Descriptor will allow us to place Handlers as a decision point before any web service.
 - The added decision point before web services provides deeper access control
- **Solution is Customizable**
 - We can write different Handlers for web services based on their requirements
 - Some web services can use it, and others don't have to.
- **Know that there is a performance tradeoff**
 - Know when you need SAML, and when you don't

Contact Information



Kevin T. Smith - kevintsmith@attbi.com

Co-authored these books just released:

