On the relationship of traditional and Web Services Security protocols

E. Kleiner and A.W. Roscoe

Traditional security protocols

We identify traditional security protocols with the common notation in the literature in the style of Dolev-Yao in which both the concrete syntax and the concrete cryptographic algorithms are abstracted as constructors of a free algebra.

1. A$\rightarrow$B : $\{n_a, A\}_{PK(B)}$
2. B$\rightarrow$A : $\{n_a, n_b, B\}_{PK(A)}$
3. A$\rightarrow$B : $\{n_b\}_{PK(B)}$

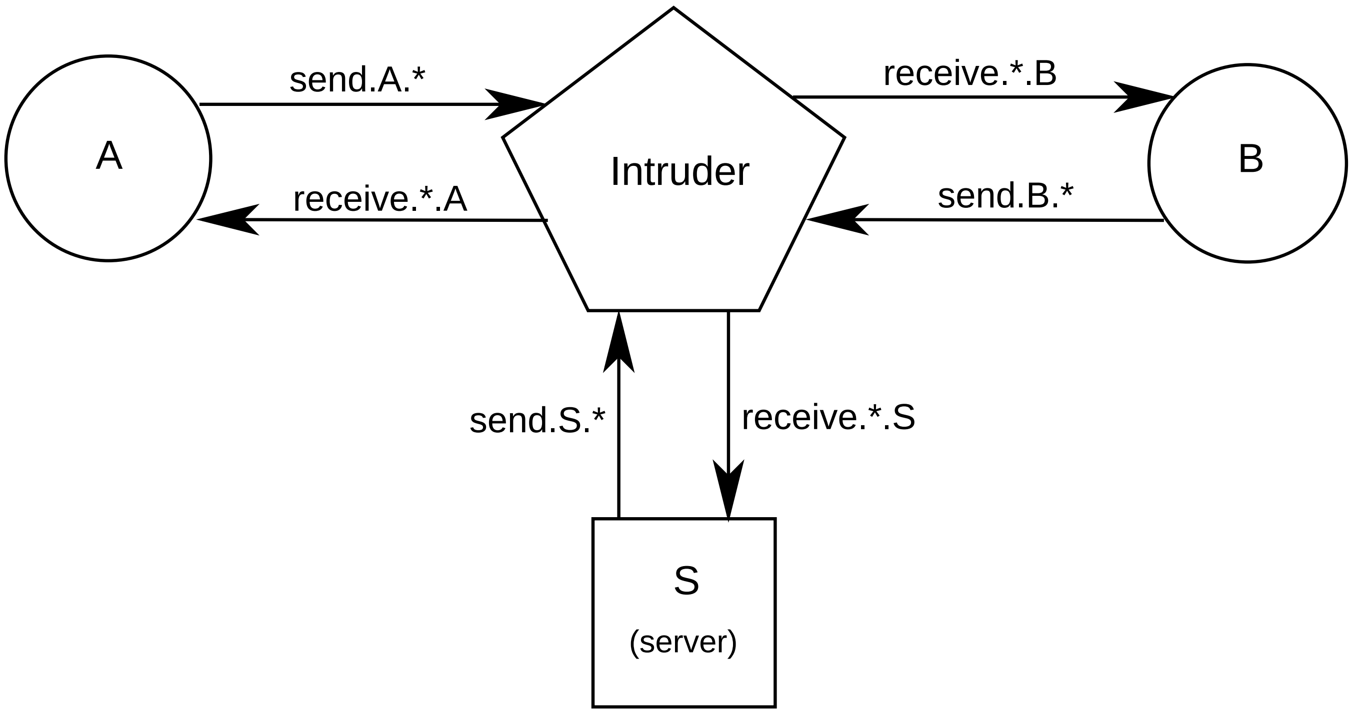
Casper - Compiler for Analysing Security Protocols

Casper adopts this notation as its input.

```
#Protocol description
0.     -> A : B
1.   A -> B : {na, A}{PK(B)}
2.   B -> A : {na, nb, B}{PK(A)}
3.   A -> B : {nb}{PK(B)}
```

Modelling a protocol

$$SYSTEM \mathrel{\widehat{=}} (|||_{A \in Honest} P_A) \parallel INTRUDER(IIK)$$



Web Services Security - Overview

WS-Security was initially proposed by Microsoft in October 2001.

In June 2002 WS-Security was submitted to the Oasis standard body and in March 2004 it became an Oasis Standard.

It defines elements to incorporate security tokens within SOAP messages.

A security token is an XML illustration of a collection of one or more claims.

A claim is a statement made by an entity and can be name, password, identity, key or privileges.

XML-Signature and XML-Encryption are used for achieving integrity and confidentiality of the security tokens.

XML-Signature

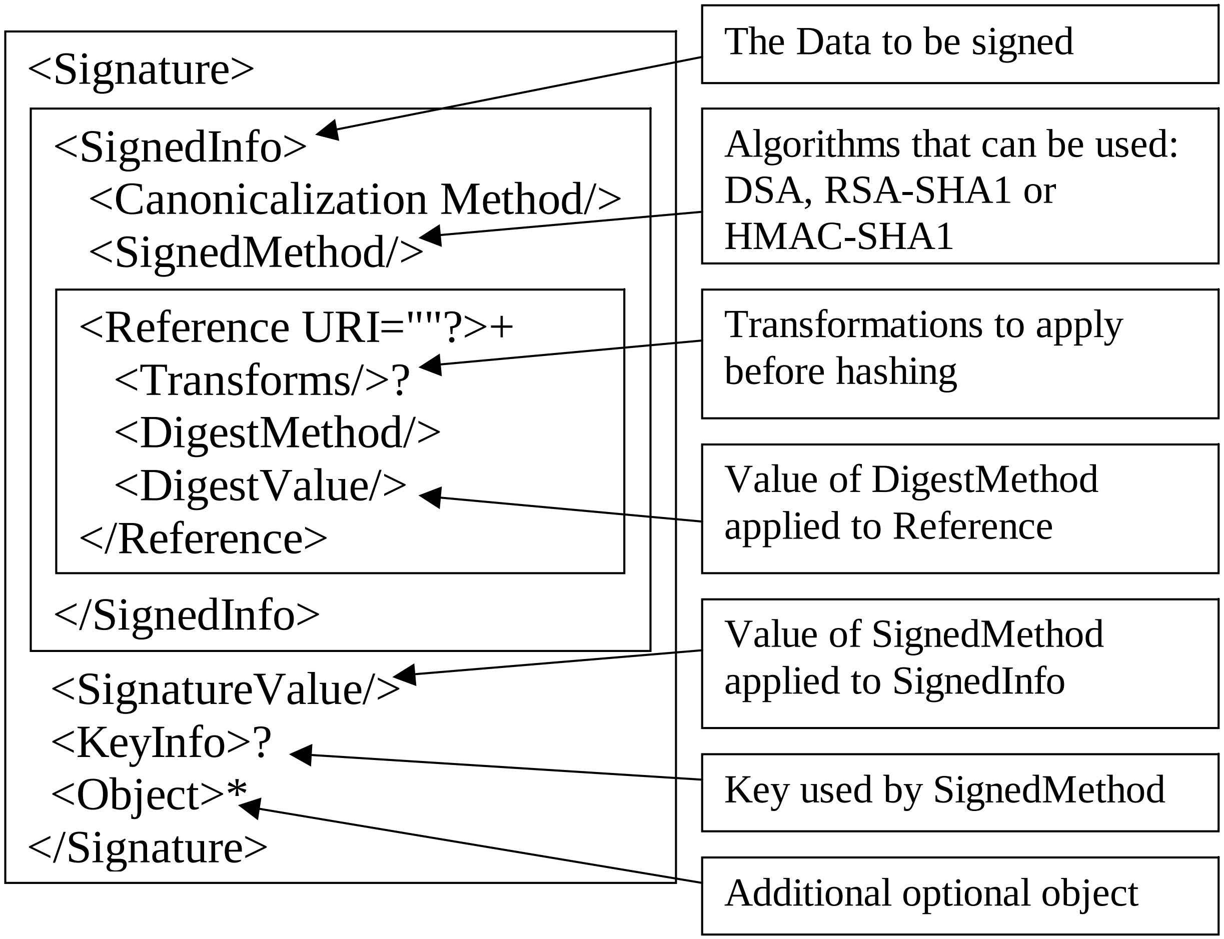


Figure 1: Structure of XML-Signature element

Modelling WS-Security

We construct a mapping ϕ from SOAP messages to Casper input, such that if a WS-security protocol contains the messages $m_1, m_2 ..., m_n$ then,

1. If an attack is found on $\phi(m_1), \phi(m_2)..., \phi(m_n)$ then a corresponding attack can be reproduced on $m_1, m_2..., m_n$.
2. If an attack exists on $m_1, m_2..., m_n$ then it also exists on $\phi(m_1), \phi(m_2)..., \phi(m_n)$. (MFPS05)

More important of the above properties is (2), since we definitely do not want to generate a false "proof" of correctness using the translation.

Any attack found by Casper can be translated back to make sure it is really present in the original protocol.

Message M - taken from an Oasis proposed protocol

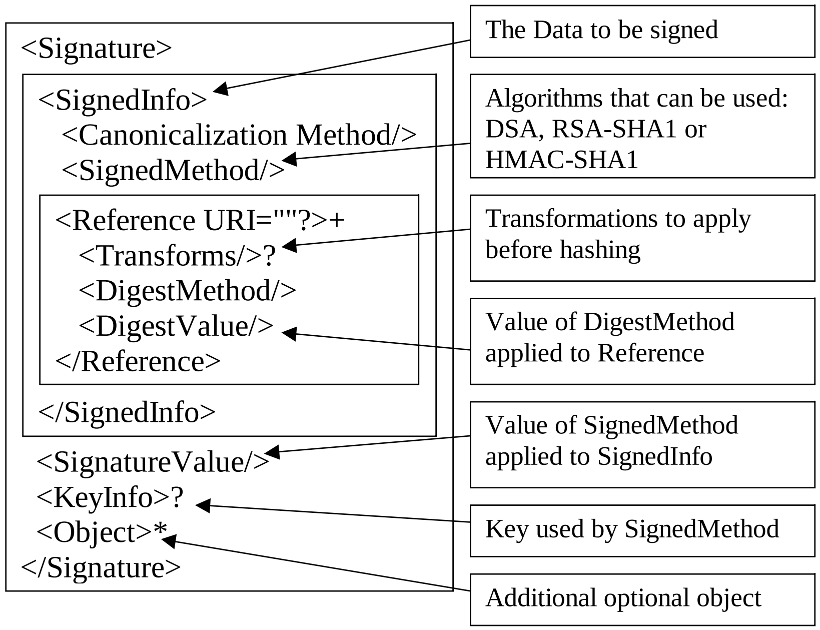
```
<Envelope>
 <Header>
  <Security mustUnderstand="1">
   <BinarySecurityToken ValueType="x509v3" Id="myCert"> BV1
   </BinarySecurityToken>
   <Signature>
    <SignedInfo>
     <CanonicalizationMethod Algorithm=.... />
     <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig\#rsa-sha1"/>
     <Reference URI="#body">
      <Transforms>
       <Transform Algorithm=.... />
      </Transforms>
      <DigestMethod Algorithm=... />
      <DigestValue> BV2 </DigestValue>
     </Reference>
    </SignedInfo>
    <SignatureValue> BV3 </SignatureValue>
    <KeyInfo>
     <SecurityTokenReference>
      <Reference URI="#myCert" />
     </SecurityTokenReference>
    </KeyInfo>
   </Signature>
```

```
      <EncryptedKey>
        <EncryptedMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
        <KeyInfo>
          <SecurityTokenReference>
            <KeyIdentifier ValueType="X509v3"> BV4
            </KeyIdentifier>
          </SecurityTokenReference>
        </KeyInfo>
        <CipherData>
          <CipherValue> BV5 </CipherValue>
        </CipherData>
        <ReferenceList>
          <DataReference URI="#enc" />
        </ReferenceList>
      </EncryptedKey>
    </Security>
  </Header>
  <Body  Id="body">
    <EncryptedData Id="enc" Type="http://www.w3.org/2001/04/xmlenc#content">
      <EncryptedMethod Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-cbc" />
      <CipherData>
        <CipherValue> BV6 </CipherValue>
      </CipherData>
    </EncryptedData>
  </Body>
</Envelope>
```

Applying ϕ on `Security` element

$$\phi(\langle Security\rangle ... \langle/Security\rangle) =$$
$$\phi(\langle BinarySecurityToken\rangle ... \langle/BinarySecurityToken\rangle),$$
$$\phi(\langle EncryptedKey\rangle ... \langle/EncryptedKey\rangle), \phi(\langle Signature\rangle ... \langle/Signature\rangle)$$

Applying ϕ on `Signature` element



$$\phi(\langle Signature\rangle ... \langle / Signature\rangle) = \{\phi(\langle Reference ...\rangle ... \langle / Reference\rangle), \ldots$$
$$\phi(\langle Reference ...\rangle ... \langle / Reference\rangle) ...\}_{\phi(\langle KeyInfo\rangle ... \langle / KeyInfo\rangle, SIG)}$$
$$\phi(\langle Reference ...\rangle ... \langle / Reference\rangle), \ldots, \phi(\langle Reference ...\rangle ... \langle / Reference\rangle) ...$$

Demonstrate the complete derivation of $\phi(M)$

$\phi(M)$

$\Rightarrow \phi(\langle\text{Header}\rangle\ldots\langle/\text{Header}\rangle),\phi(\langle\text{Body}\rangle\ldots\langle/\text{Body}\rangle)$

$\Rightarrow \phi(\langle\text{Security}\rangle\ldots\langle/\text{Security}\rangle),\phi(\langle\text{Body}\rangle\ldots\langle/\text{Body}\rangle)$

$\Rightarrow \phi(\langle\text{BinarySecurityToken}\rangle\ldots\langle/\text{BinarySecurityToken}\rangle)$, $\phi(\langle\text{EncryptedKey}\rangle\ldots\langle/\text{EncryptedKey}\rangle)$, $\phi(\langle\text{Signature}\rangle\ldots\langle/\text{Signature}\rangle)$, $\phi(\langle\text{Body}\rangle\ldots\langle/\text{Body}\rangle)$

$\Rightarrow \phi(\langle\text{EncryptedKey}\rangle\ldots\langle/\text{EncryptedKey}\rangle)$, $\phi(\langle\text{Signature}\rangle\ldots\langle/\text{Signature}\rangle)$, $\phi(\langle\text{Body}\rangle\ldots\langle/\text{Body}\rangle)$

$\Rightarrow \phi(\langle\text{ReferenceList}\rangle\ldots\langle/\text{ReferenceList}\rangle,\{\text{K}\})$, $\{\text{K}\}_{\phi(\langle\text{KeyInfo}\rangle\ldots\langle/\text{KeyInfo}\rangle,\text{ENC})}$, $\phi(\langle\text{Signature}\rangle\ldots\langle/\text{Signature}\rangle)$, $\phi(\langle\text{Body}\rangle\ldots\langle/\text{Body}\rangle)$

$\Rightarrow \phi(\langle\text{DataReference URI=\#enc }/\rangle,\{\text{K}\})$, $\{\text{K}\}_{\phi(\langle\text{KeyInfo}\rangle\ldots\langle/\text{KeyInfo}\rangle,\text{ENC})}$, $\phi(\langle\text{Signature}\rangle\ldots\langle/\text{Signature}\rangle)$, $\phi(\langle\text{Body}\rangle\ldots\langle/\text{Body}\rangle)$

$\Rightarrow \text{Context}(\text{enc},\{\text{K}\})$, $\{\text{K}\}_{\phi(\langle\text{KeyInfo}\rangle\ldots\langle/\text{KeyInfo}\rangle,\text{ENC})}$, $\phi(\langle\text{Signature}\rangle\ldots\langle/\text{Signature}\rangle)$, $\phi(\langle\text{Body}\rangle\ldots\langle/\text{Body}\rangle)$

$\Rightarrow \text{Context}(\text{enc},\{\text{K}\})$, $\{\text{K}\}_{\phi(\langle\text{SecurityTokenReference}\rangle\ldots\langle/\text{SecurityTokenReference}\rangle,\text{ENC})}$, $\phi(\langle\text{Signature}\rangle\ldots\langle/\text{Signature}\rangle)$, $\phi(\langle\text{Body}\rangle\ldots\langle/\text{Body}\rangle)$

$\Rightarrow \text{Context}(\text{enc},\{\text{K}\})$, $\{\text{K}\}_{\phi(\langle\text{KeyIdentifier}\rangle\ldots\langle/\text{KeyIdentifier}\rangle,\text{ENC})}$, $\phi(\langle\text{Signature}\rangle\ldots\langle/\text{Signature}\rangle)$, $\phi(\langle\text{Body}\rangle\ldots\langle/\text{Body}\rangle)$

$\Rightarrow \text{Context}(\text{enc},\{\text{K}\})$, $\{\text{K}\}_{\text{PK(B)}}$, $\{\phi(\langle\text{Reference URI=\#body}\rangle\ldots\langle/\text{Reference}\rangle)\}$,

$\{\phi(\langle \text{KeyInfo}\rangle ... \langle /\text{KeyInfo}\rangle),\text{SIG}\},\ \phi(\langle \text{Body}\rangle ... \langle /\text{Body}\rangle)$

$\Rightarrow \text{Context}(\text{enc},\{\text{K}\}),\ \{\text{K}\}_{\text{PK(B)}},$
$\{\phi(\langle \text{DigestMethod}.../\rangle)(\phi(\text{body}))\}_{\phi(\langle \text{KeyInfo}\rangle ... \langle /\text{KeyInfo}\rangle,\text{SIG})},\ \phi(\langle \text{Body}\rangle ... \langle /\text{Body}\rangle)$

$\Rightarrow \text{Context}(\text{enc},\{\text{K}\}),\ \{\text{K}\}_{\text{PK(B)}},\ \{\text{sha1}(\phi(\text{body}))\}_{\phi(\langle \text{KeyInfo}\rangle ... \langle /\text{KeyInfo}\rangle,\text{SIG})},$
$\phi(\langle \text{Body}\rangle ... \langle /\text{Body}\rangle)$

$\Rightarrow \text{Context}(\text{enc},\{\text{K}\}),\ \{\text{K}\}_{\text{PK(B)}},$
$\{\text{sha1}(\{\text{Body}\}_{\phi(\langle \text{EncryptedData Id=``enc''}\rangle ... \langle /\text{EncryptedData}\rangle)})\}_{\phi(\langle \text{KeyInfo}\rangle ... \langle /\text{KeyInfo}\rangle,\text{SIG})},$
$\phi(\langle \text{Body}\rangle ... \langle /\text{Body}\rangle)$

$\Rightarrow \text{Context}(\text{enc},\{\text{K}\}),\ \{\text{K}\}_{\text{PK(B)}},\ \{\text{sha1}(\{\text{Body}\}_{\text{K}})\}_{\phi(\langle \text{KeyInfo}\rangle ... \langle /\text{KeyInfo}\rangle,\text{SIG})},$
$\phi(\langle \text{Body}\rangle ... \langle /\text{Body}\rangle)$

$\Rightarrow$
$\text{Context}(\text{enc},\{\text{K}\}),\ \{\text{K}\}_{\text{PK(B)}},$
$\{\text{sha1}(\{\text{Body}\}_{\text{K}})\}_{\phi(\langle \text{SecurityTokenReference}\rangle ... \langle /\text{SecurityTokenReference}\rangle,\text{SIG})},$
$\phi(\langle \text{Body}\rangle ... \langle /\text{Body}\rangle)$

$\Rightarrow \text{Context}(\text{enc},\{\text{K}\}),\ \{\text{K}\}_{\text{PK(B)}},\ \{\text{sha1}(\{\text{Body}\}_{\text{K}})\}_{\phi(\langle \text{Reference URI=\#myCert}.../\rangle,\text{SIG})},$
$\phi(\langle \text{Body}\rangle ... \langle /\text{Body}\rangle)$

$\Rightarrow \text{Context}(\text{enc},\{\text{K}\}),\ \{\text{K}\}_{\text{PK(B)}},\ \{\text{sha1}(\{\text{Body}\}_{\text{K}})\}_{\text{SK(A)}},\ \phi(\langle \text{Body}\rangle ... \langle /\text{Body}\rangle)$

$\Rightarrow \text{Context}(\text{enc},\{\text{K}\}),\ \{\text{K}\}_{\text{PK(B)}},\ \{\text{sha1}(\{\text{Body}\}_{\text{K}})\}_{\text{SK(A)}},\ \{\text{Body}\}_{\text{K}}$

$\Rightarrow \{\text{K}\}_{\text{PK(B)}},\ \{\text{sha1}(\{\text{Body}\}_{\text{K}})\}_{\text{SK(A)}},\ \{\text{Body}\}_{\text{K}}$

Oasis proposed protocol

1. A$\longrightarrow$B: `M`
2. B$\longrightarrow$A: `M'`

After applying ϕ to both of the messages we get the following protocol.

1. MSG 1 .A$\rightarrow$B : $\{K\}_{PK(B)}$, $\{sha1(\{Body\}_K)\}_{SK(A)}$, $\{Body\}_K$
2. MSG 2. B$\rightarrow$A : $\{K2\}_{PK(A)}$, $\{sha1(\{Body2\}_{K2})\}_{SK(B)}$, $\{Body2\}_{K2}$

An attack

Using FDR the following authentication attack was found.

1. MSG 1. $I \rightarrow Bob : \{K\}_{PK(Bob)}, \{sha1(\{Body\}_K)\}_{SK(I)}, \{Body\}_K$
2. MSG 2. $Bob \rightarrow I : \{K2\}_{PK(I)}, \{sha1(\{Body2\}_{K2})\}_{SK(Bob)}, \{Body2\}_{K2}$
3. MSG 1. $Alice \rightarrow I_{Bob} : \{K3\}_{PK(Bob)}, \{sha1(\{Body3\}_{K3})\}_{SK(Alice)}, \{Body3\}_{K3}$
4. MSG 2. $I_{Bob} \rightarrow Alice : \{K2\}_{PK(Alice)}, \{sha1(\{Body2\}_{K2})\}_{SK(Bob)}, \{Body2\}_{K2}$

Contribution and Ramifications

- Any WSS protocol can be analysed in the traditional model after it was transformed by ϕ
- We would like to emphasise that although this proof is based on the theory of CSP, it is valid for any tool regardless to its underlying theory.
- ϕ's input is the SOAP messages of the protocol to be analysed. This fact allows the unprofessional user to analyse complex WSS protocols in a few minutes time.
- It was suggested that ϕ can be used for making the semantics of WSS clearer.

Proof of a WS-SecureConversation based protocol

WS-SecureConversation Background

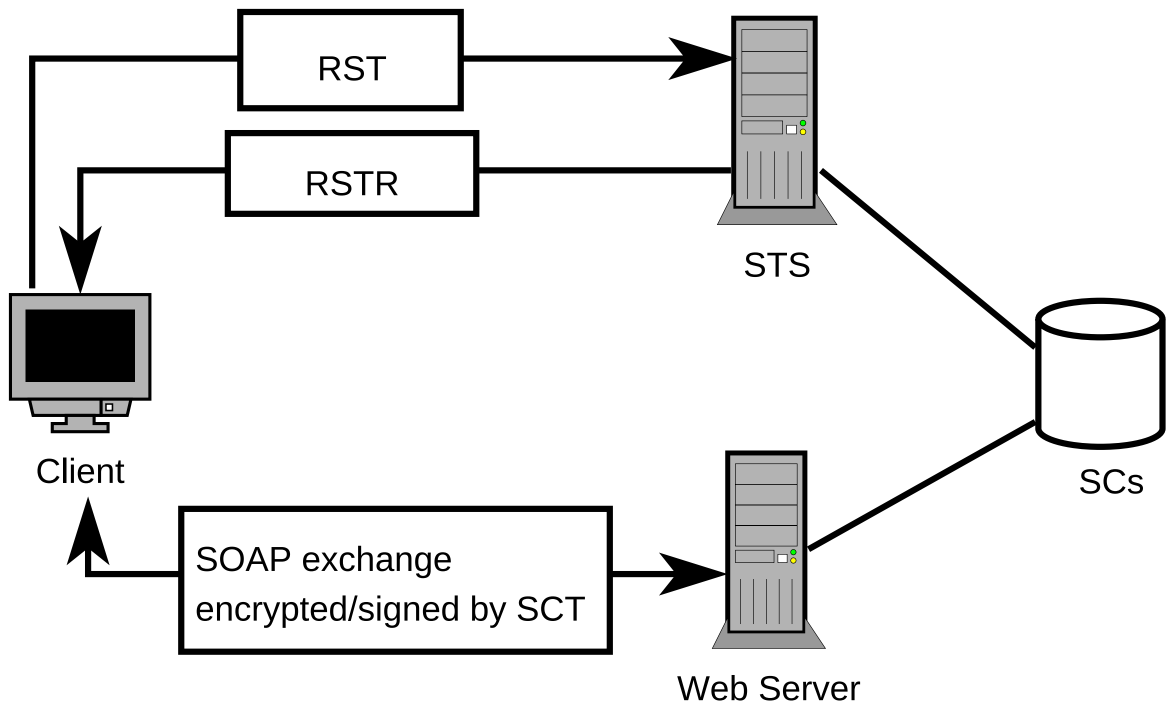
- WS-Security defines security tokens to provide different security properties to the claims that these tokens encapsulate.
- Yet, the process of verifying these tokens against the security policy has to be repeated for each SOAP message.
- WS-SecureConversation addresses this issue. It is built upon WS-Security and WS-Trust to allow a requestor and a Web Service to set up a mutually authenticated *security context*.
- After the *security context* establishment, the parties may use this shared secret with WS-Security for signing and encrypting messages.

Establishing *security context*

WS-Trust defines three ways to establish *security context*

- **Security Context Token is created by a security token service (STS)**
  1. A –> STS : RST (may contain an `Entropy` (e.g. secret key))
  2. STS –> A : RSTR (contains an `Entropy` and `SCT`)

  If MSG 1. includes an `Entropy` then `SCT` $= p\text{–}sha1(k_A, K_{STS})$
  else `SCT` $= K_{STS}$
- **Security Context is created by one of the agents and propagated with a message**
- **Security Context created by negotiation**



Note that WS-SecureConversation and WS-Trust do not define how to secure the `RST`, `RSTR` and `SCT`.

Modelling WS-SecureConversation

1. Extend ϕ to capture the new tokens (e.g. RST, RSTR, SCT etc)
2. Prove by induction that the extension of ϕ does not harm the safety of ϕ

Example - WSE's WS-SecureConversation

The protocol after applied to ϕ:

1. A→B: RST, UMI1, anonymous, B, ts1,
   $\{$sha1(ts1), sha1(SecurityToken-b8...,$\{K_1\}_{PK(B)}$),
   sha1(RST), sha1(UMI1), sha1(anonymous), sha1(B)$\}_{p\text{-}sha1(pass(A),N_A+ts1)}$
   sha1(password(A),N_A,ts1), N_A, ts1,$\{K_1\}_{PK(B)}$
2. B→A: RSTR, UMI2, UMI1, anonymous, ts2,
   $\{$sha1(RSTR), sha1(UMI2), sha1(UMI1), sha1(anonymous),
   (sha1(ts2),sha1(uuid1, ts2', $\{K_2\}_{K_1}$))$\}_{SK(B)}$,uuid1, ts2', $\{K_2\}_{K_1}$
3. A→B: UMI3, anonymous, B, ts3, (uuid1, ts2'), $\{$sha1(UMI3), sha1(anonymous),
   sha1(B), sha1(ts3), sha1(body1)$\}_{p\text{-}sha1(K_1,K_2)}$,$\{$body1$\}_{p\text{-}sha1(K_1,K_2)}$
4. B→A: UMI4, UMI3, A, ts4, (uuid1, ts2'),$\{$sha1(UMI4), sha1(UMI3),
   sha1(anonymous),sha1(ts4),sha1(body2)$\}_{p\text{-}sha1(K_1,K_2)}$,
   $\{$body2$\}_{p\text{-}sha1(K_1,K_2)}$

After applying some simplifying transformations...

MSG 1. A→B: UMI1, $\{$UMI1, B, $\{K_1\}_{PK(B)})\}_{p\text{-}sha1(pass(A),N_A)}$,
sha1(password(A),N_A)), N_A, $\{K_1\}_{PK(B)}$

MSG 2. B→A: $\{$UMI1, UMI2, $\{K_2\}_{K_1}\}_{SK(B)}$,uuid1,$\{K_2\}_{K_1}$

MSG 3. A→B: $\{$UMI3, B, body1$\}_{p\text{-}sha1(K_1,K_2)}$,$\{$body1$\}_{p\text{-}sha1(K_1,K_2)}$

MSG 4. B→A: $\{$UMI3, UMI4, body2$\}_{p\text{-}sha1(K_1,K_2)}$,$\{$body2$\}_{p\text{-}sha1(K_1,K_2)}$

No Secrecy violation was found by FDR

- Therefore the original WSS protocol is correct in term of secrecy

An authentication attack

MSG 1α.	A $\rightarrow$ I_B	: UMI1, {UMI1, B, $\{K_1\}_{PK(B)}$)$\}_{p\text{-}sha1(pass(Alice),N_A)}$, sha1(password(Alice),$N_A$)), N_A, $\{K_1\}_{PK(B)}$
MSG 1β.	I $\rightarrow$ B	: UMI1, {UMI1, B, $\{K_1\}_{PK(B)}$)$\}_{p\text{-}sha1(pass(I),N_A)}$, sha1(password(I),$N_A$)), N_A, $\{K_1\}_{PK(B)}$
MSG 2β.	B $\rightarrow$ I	: {UMI1, UMI2, $\{K_2\}_{K_1}\}_{SK(B)}$,uuid1,$\{K_2\}_{K_1}$
MSG 2α.	I_B $\rightarrow$ A	: {UMI1, UMI2, $\{K_2\}_{K_1}\}_{SK(B)}$,uuid1,$\{K_2\}_{K_1}$
MSG 3α.	A $\rightarrow$ I_B	: {UMI3, B, body1$\}_{p\text{-}sha1(K_1,K_2)}$,{body1$\}_{p\text{-}sha1(K_1,K_2)}$
MSG 3β.	I $\rightarrow$ B	: {UMI3, B, body1$\}_{p\text{-}sha1(K_1,K_2)}$,{body1$\}_{p\text{-}sha1(K_1,K_2)}$
MSG 4β.	B $\rightarrow$ I	: {UMI3, UMI4, body2$\}_{p\text{-}sha1(K_1,K_2)}$,{body2$\}_{p\text{-}sha1(K_1,K_2)}$
MSG 4α.	I_B $\rightarrow$ A	: {UMI3, UMI4, body2$\}_{p\text{-}sha1(K_1,K_2)}$,{body2$\}_{p\text{-}sha1(K_1,K_2)}$

In addition

- UMI1 and N_A are not properly authenticated in the WS-Trust part of the protocol.
- UMI3 and UMI4 are not properly authenticated in the WS-SecureConversation part of the protocol.

Conclusion

- That part of SOAP Message Security which lies outside of any cryptographic operators may be constructed at will by any user, trustworthy or malicious. There is nothing secret about it.
- Since Kerckhoff's Known Design Principle is adopted by most if not all crypto-analysts, the extra information about the structure of the messages given by the XML-tagging is assumed anyway.
- We have already shown in ARS04 that there are some interesting ways in which SOAP can assist security by providing degree of protection against type flaw attacks.

Related work

- Gordon and Pucella proposed a security abstraction to RPC services in which requests and responses are encoded as SOAP messages. This abstraction is modelled using an object calculus which its semantics is defined by pi-calculus. This approach is currently limited to checking authentication properties.
- Bhargavan, Fournet, Gordon and Pucella developed a tool (`TulaFale`) based on the Blanchet's `ProVerif`. The tool compiles a description of SOAP-based security protocol and its properties into the pi-calculus and then runs `ProVerif` to analyse it.
- TulaFale specification language was extended for modelling WS-Trust and WS-SecureConversation based protocols.

Future work

1. We have already automated ϕ, we intend to write a user interface for analysing WS-Security protocols.

2. We are interested in “internalising” potential intermediaries and believe we then be able to model and check protocols with arbitrary number of intermediaries.