WS-SX Specifications

Martin Gudgin
Microsoft Corp.

Agenda

- WS-Trust
- WS-SecureConversation
- WS-SecurityPolicy

Agenda

- WS-Trust
  - Introduction
  - Requesting and returning tokens
  - Token Scope
  - References
  - Keys and Entropy
  - Returning Multiple Tokens
  - Negotiations and Challenges

WS-Trust Requirements

- Uniform semantics for token exchange
- Integration of existing negotiation protocols
- Extensible and customizable
- Focus point for challenges

WS-Trust

- A protocol framework
  - Supports different exchange patterns and topologies
- Builds on Web Services Security
- Defines mechanisms for brokering trust
  - Still need to bootstrap trust
- Introduces the Security Token Service
  - Anyone can be an STS

WS-Trust

- Is token-type agnostic
- Client doesn’t need token specific knowledge
  - Decouples client from token-type
  - Implies certain token parameters will need hoisting
- Defines common patterns
- Can be extended and customized

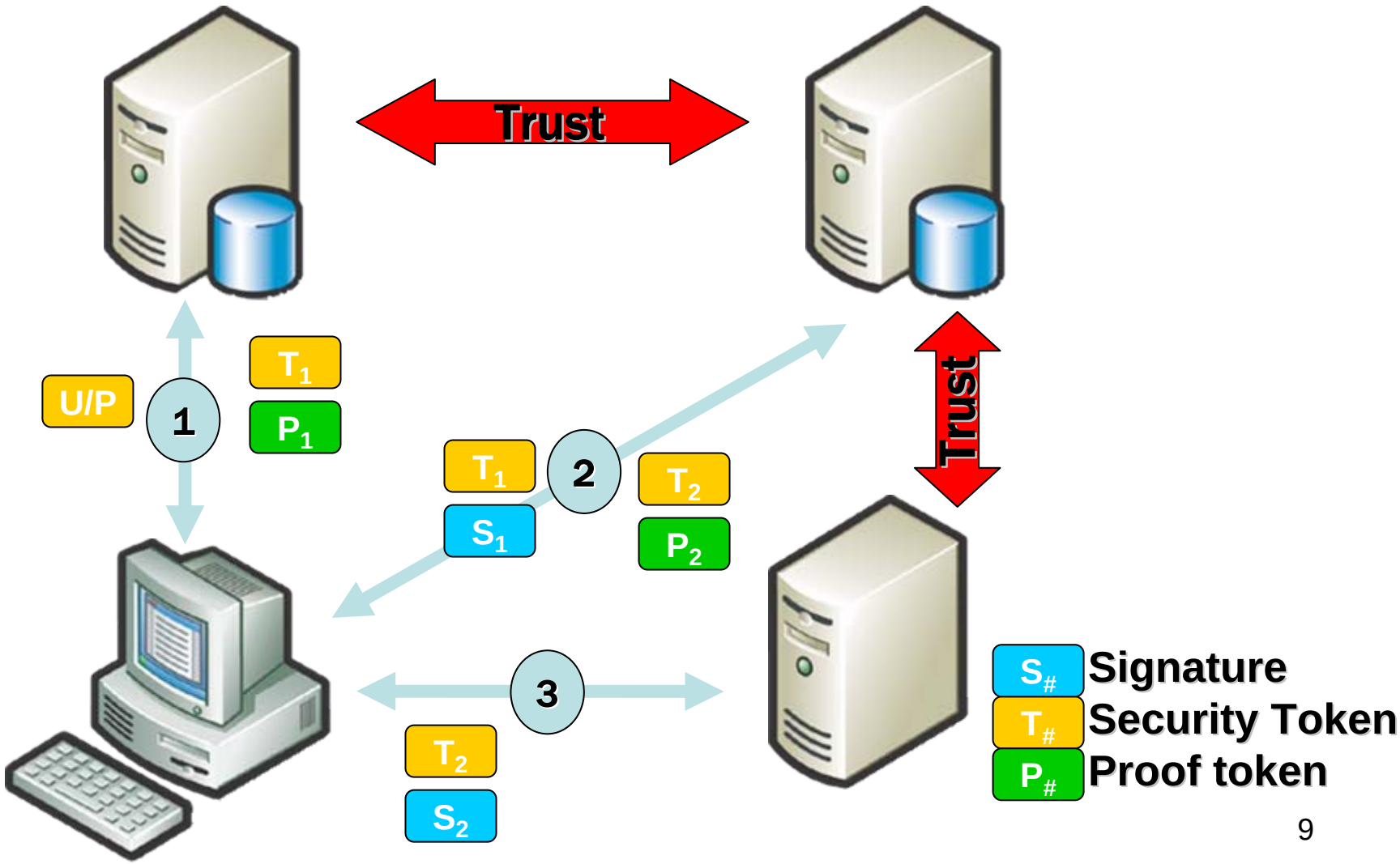
Common Patterns

- Issuance
  - Defines mechanisms for requesting a new token
- Renewal
  - Defines mechanisms for renewing previously issued tokens
- Validation
  - Defines mechanisms for verifying validity of tokens

Common Patterns

- Cancellation
  - Defines mechanisms for cancelling a previously issued token
  - Cancelled tokens can no longer be used
- Challenges/Negotiations
  - Defines mechanisms for secure multi-leg challenges and negotiations prior to token issuance

Example


Trust
Trust
U/P
1
T_1
P_1
T_1
S_1
2
T_2
P_2
3
T_2
S_2
$S_\#$ Signature
$T_\#$ Security Token
$P_\#$ Proof token

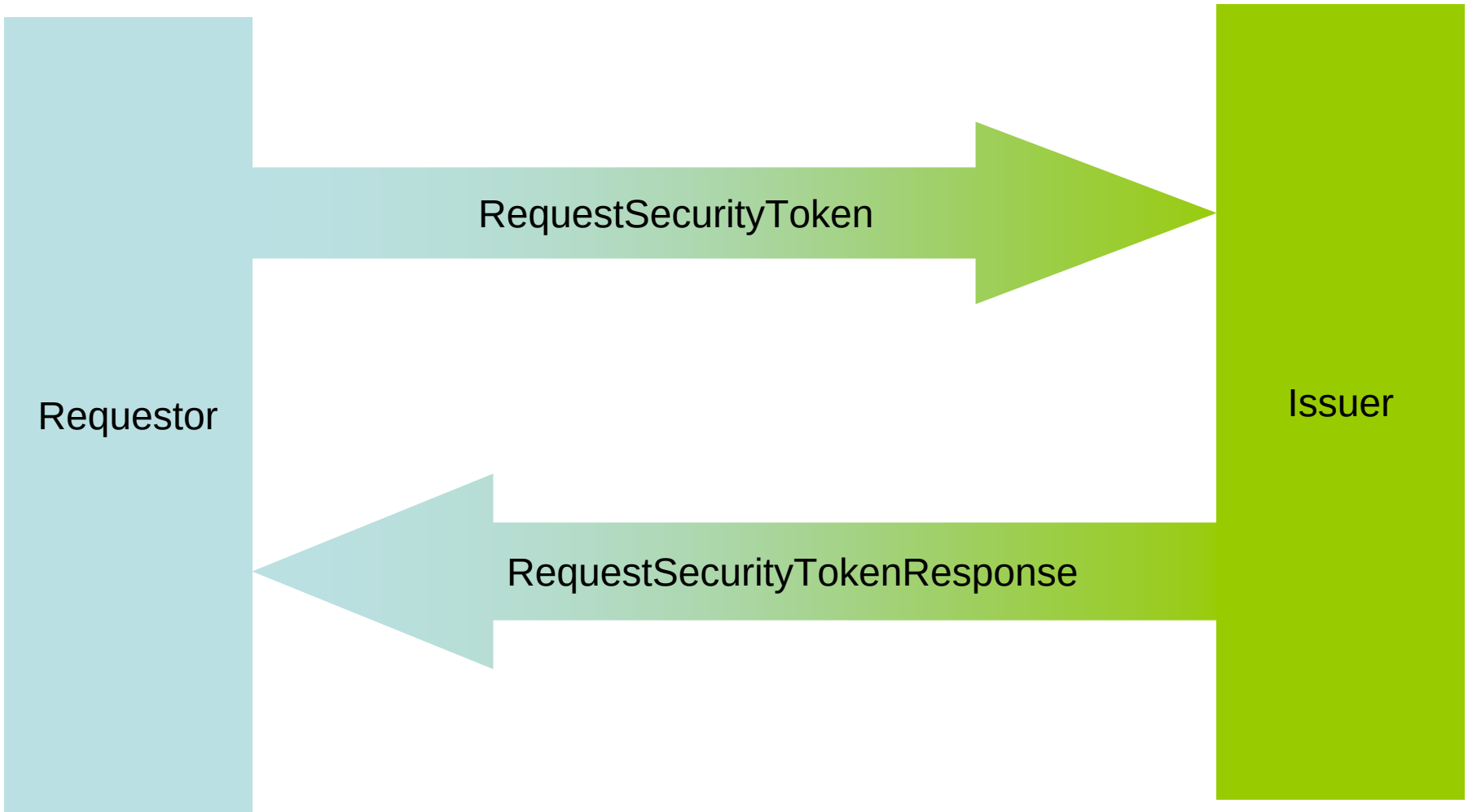
Protocol messages - 1

- Requests sent as RequestSecurityToken (RST) messages
- Always specify operation type
  - Issue, Validate etc.
- May also specify requested token-type
  - SAML, X509, Kerberos etc.

Protocol messages - 2

- Responses returned as RequestSecurityTokenResponse (RSTR) messages
- Return the requested token
  - Or a reference thereto
  - May also specify token-type returned
- Typically also returns a proof-of-possesion token

Protocol Messages



Simple RST Example

```
<wst:RequestSecurityToken>
 <wst:TokenType>
 http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-
1.1#SAMLV1.1
 </wst:TokenType>
 <wst:RequestType>
 http://schemas.xmlsoap.org/ws/2005/02/trust/Issue
 </wst:RequestType>
</wst:RequestSecurityToken>
```

Simple RSTR Example

```
<wst:RequestSecurityTokenResponse>
 <wst:TokenType>
 http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-
1.1#SAMLV1.1
 </wst:TokenType>
 <wst:RequestedSecurityToken>
  <saml:Assertion ... >
  ...
  </saml:Assertion>
 </wst:RequestedSecurityToken>
 <wst:RequestedProofToken>
  <xenc:EncryptedKey>...</xenc:EncryptedKey>
 </wst:RequestedProofToken>
</wst:RequestSecurityTokenResponse>
```

Observations

- Identity of requestor determined per normal authentication mechanisms
- Returned requested token considered opaque to requestor
- Returned proof token NOT opaque to requestor
- Scope of returned token implicit

Making scope explicit

- RST supports wsp:AppliesTo
  - Allows requestor to specify the required scope for the requested token
- Scope can be any domain expression
  - e.g. wsa:EndpointReference
- RSTR can also indicate scope of returned token
- Token independent

wsp:AppliesTo

Requestor wsp:AppliesTo	**Issuer wsp:AppliesTo**	**Results**
Absent	Absent	OK. Implied scope.
Present	Absent	OK. Issued token has scope specified by requestor.
Absent	Present	OK. Resulting token has scope specified by issuer.
Present	Present and matches Requestor	OK.
Present	Present and specifies a scope greater than specified by the requestor	OK.

Requested References

- Token issuer can provide Security Token References for referring to returned token
  - STRs opaque to requestor
- Requested Attached Reference
  - For referring to the token when it appears in a message
- Requested Unattached Reference
  - For referring to the token when it does not appear in a message

Requested References Example

```
<wst:RequestSecurityTokenResponse>
 <wst:TokenType>
 http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-
1.1#SAMLV1.1
 </wst:TokenType>
 <wst:RequestedSecurityToken>
  <saml:Assertion ... >
  ...
  </saml:Assertion>
 </wst:RequestedSecurityToken>
 <wst:RequestedProofToken>
  <xenc:EncryptedKey>...</xenc:EncryptedKey>
 </wst:RequestedProofToken>
 <wst:RequestedAttachedReference>
  <wsse:SecurityTokenReference>...</wsse:SecurityTokenReference>
 </wst:RequestedAttachedReference>
</wst:RequestSecurityTokenResponse>
```

Entropy and Keys

- Often proof token directly specifies key material
- WS-Trust also allows one or both parties to provide key material
- Proof token then specifies algorithm for computing resulting key

Entropy and Keys

Requestor	Issuer	Results
Provide Entropy	Uses requestor entropy as key	No proof-of-possession token is returned.
	Provides entropy	No keys returned, key(s) derived using entropy from both sides according to method identified in response
	Issues own key (rejects requestor's entropy)	Proof-of-possession token contains issuer's key(s)
No Entropy provided	Issues own key	Proof-of-possession token contains issuer's key(s)
	Does not issue key	No proof-of-possession token

Requestor Provided Entropy Example

```
<wst:RequestSecurityToken>
 <wst:TokenType>
 http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-
1.1#SAMLV1.1
 </wst:TokenType>
 <wst:RequestType>
 http://schemas.xmlsoap.org/ws/2005/02/trust/Issue
 </wst:RequestType>
 <wst:Entropy>
  <wst:BinarySecret>WmPjJkStecgGm0SlT7ORuQ==</wst:BinarySecret>
 </wst:Entropy>
</wst:RequestSecurityToken>
```

Issuer Provided Entropy Example

```
<wst:RequestSecurityTokenResponse>
 <wst:TokenType>
 http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-
1.1#SAMLV1.1
 </wst:TokenType>
 <wst:RequestedSecurityToken>
  <saml:Assertion ... >
  ...
  </saml:Assertion>
 </wst:RequestedSecurityToken>
 <wst:RequestedProofToken>
  <wst:ComputedKey>
  http://schemas.xmlsoap.org/ws/2005/02/trust/CK/PSHA1
  </wst:ComputedKey>
 </wst:RequestedProofToken>
 <wst:Entropy>
  <wst:BinarySecret>Q98y+DzgED9KvWF/QOVaQA==</wst:BinarySecret>
 </wst:Entropy>
</wst:RequestSecurityTokenResponse>
```

Token lifetime

- Requestor can specify a desired token lifetime
- Issuer can denote actual lifetime in response
  - No need for requestor to parse token

Requesting Lifetime Example

```
<wst:RequestSecurityToken>
 <wst:TokenType>
 http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-
1.1#SAMLV1.1
 </wst:TokenType>
 <wst:RequestType>
 http://schemas.xmlsoap.org/ws/2005/02/trust/Issue
 </wst:RequestType>
 <wst:Lifetime>
  <wsu:Created>2005-11-28T11:00:00Z</wsu:Created>
  <wsu:Expires>2005-11-28T23:00:00Z</wsu:Expires>
 </wst:Lifetime>
</wst:RequestSecurityToken>
```

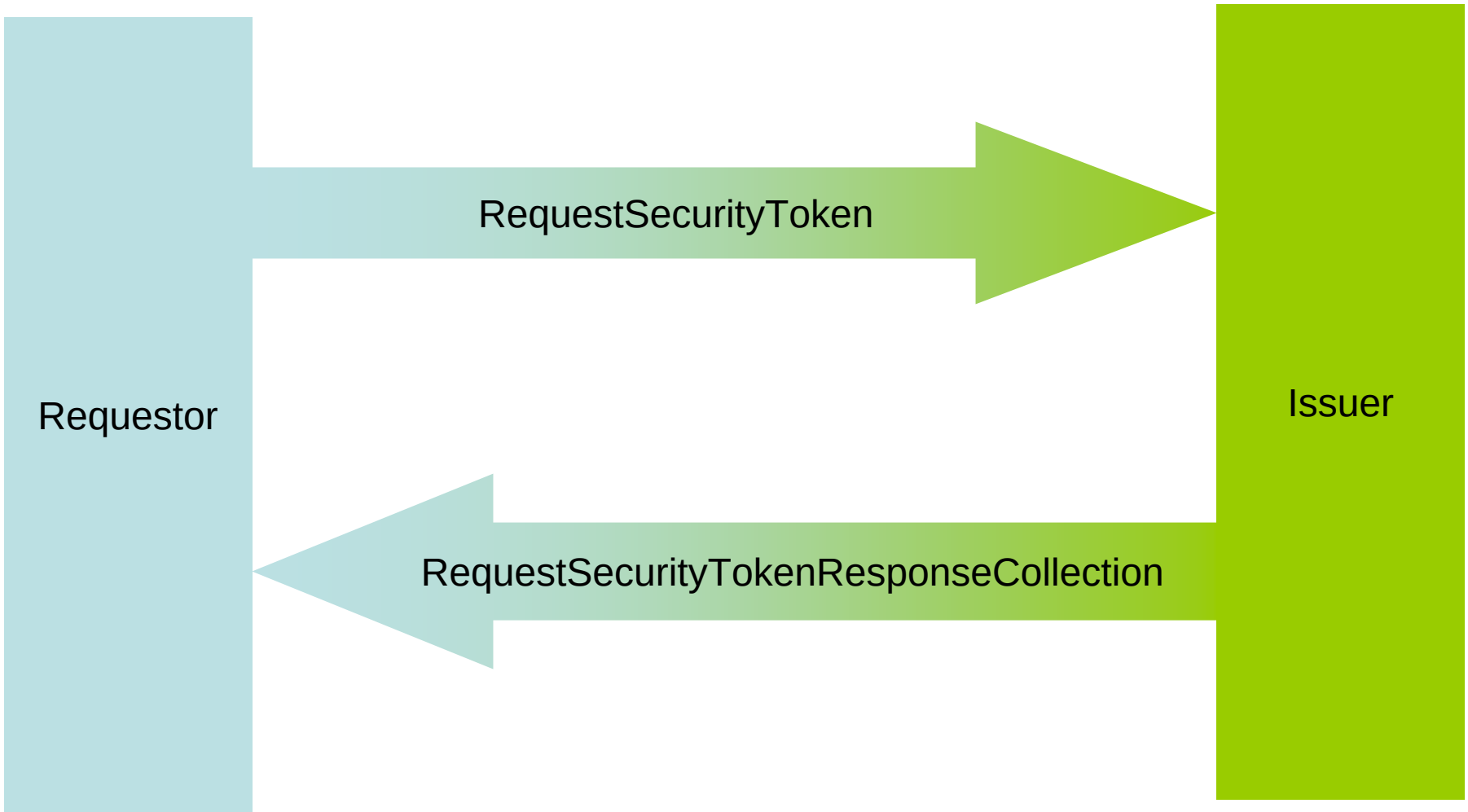
Issuing Lifetime Example

```
<wst:RequestSecurityTokenResponse>
 <wst:TokenType>
 http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-
1.1#SAMLV1.1
 </wst:TokenType>
 <wst:RequestedSecurityToken>
  <saml:Assertion ... >
  ...
  </saml:Assertion>
 </wst:RequestedSecurityToken>
 <wst:RequestedProofToken>...</wst:RequestedProofToken>
 <wst:Lifetime>
  <wsu:Created>2005-11-28T11:00:00Z</wsu:Created>
  <wsu:Expires>2005-11-28T17:00:00Z</wsu:Expires>
 </wst:Lifetime>
</wst:RequestSecurityTokenResponse>
```

Returning multiple tokens

- Issuer may return multiple tokens
  - RequestSecurityTokenResponseCollection
- Can also pass tokens out-of-band
  - IssuedTokens header

Protocol Messages



RSTRC Example

```
<wst:RequestSecurityTokenResponseCollection>
 <wst:RequestSecurityTokenResponse>
 ...
 </wst:RequestSecurityTokenResponse>
 <wst:RequestSecurityTokenResponse>
 ...
 </wst:RequestSecurityTokenResponse>
 ...
</wst:RequestSecurityTokenResponseCollection>
```

IssuedTokens Header

- Allows issuance of tokens outside of RST/RSTR
  - Typically as part of some other protocol
- Carries same content as RSTRC

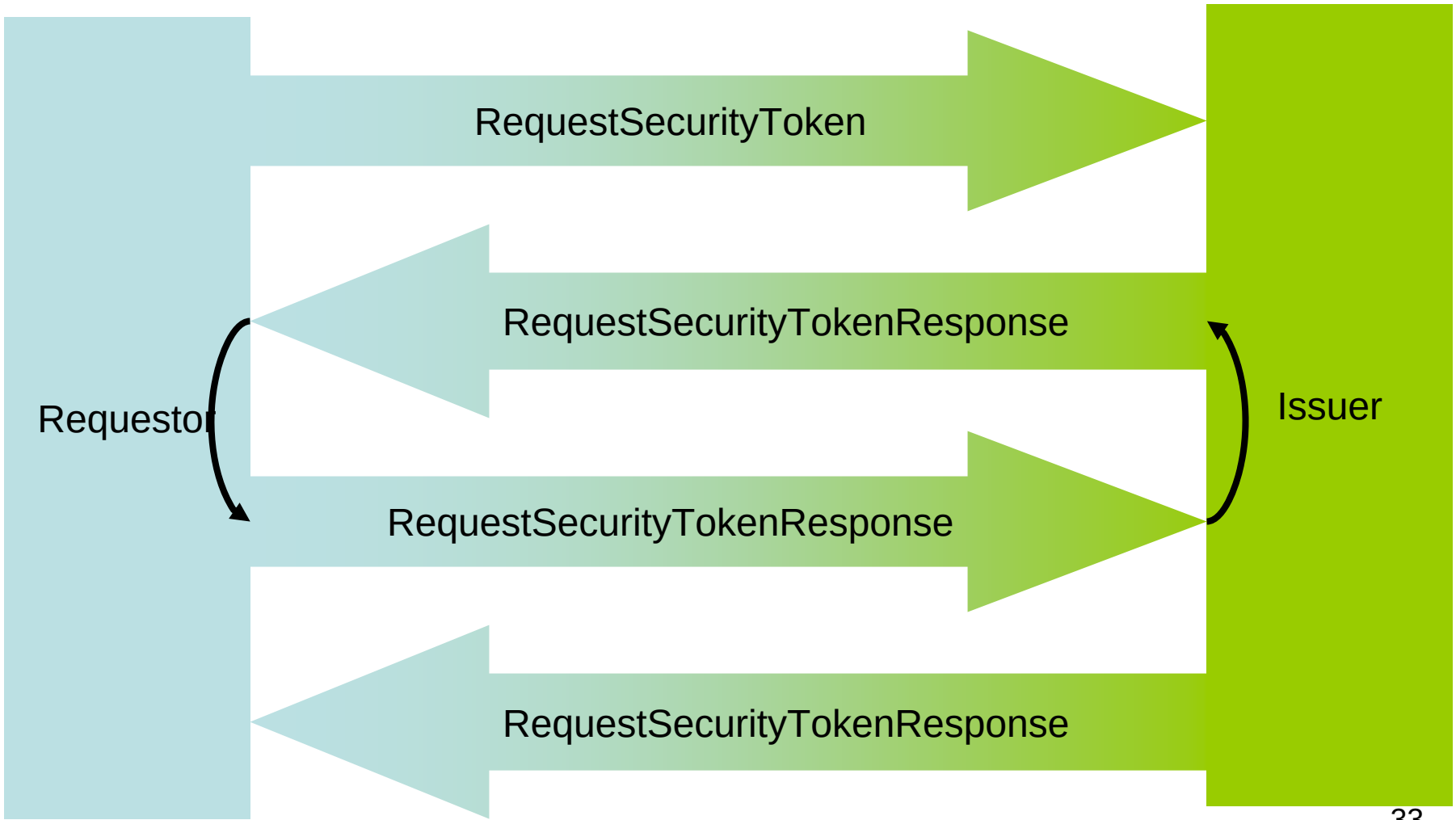
IssuedTokens Example

```
<soap:Envelope>
 <soap:Header>
  ...
  <wst:IssuedTokens>
   <wst:RequestSecurityTokenResponse>
   ...
   </wst:RequestSecurityTokenResponse>
   <wst:RequestSecurityTokenResponse>
   ...
   </wst:RequestSecurityTokenResponse>
   ...
  </wst:IssuedTokens>
  ...
 </soap:Header>
 <soap:Body>
 ...
 </soap:Body>
</soap:Envelope>
```

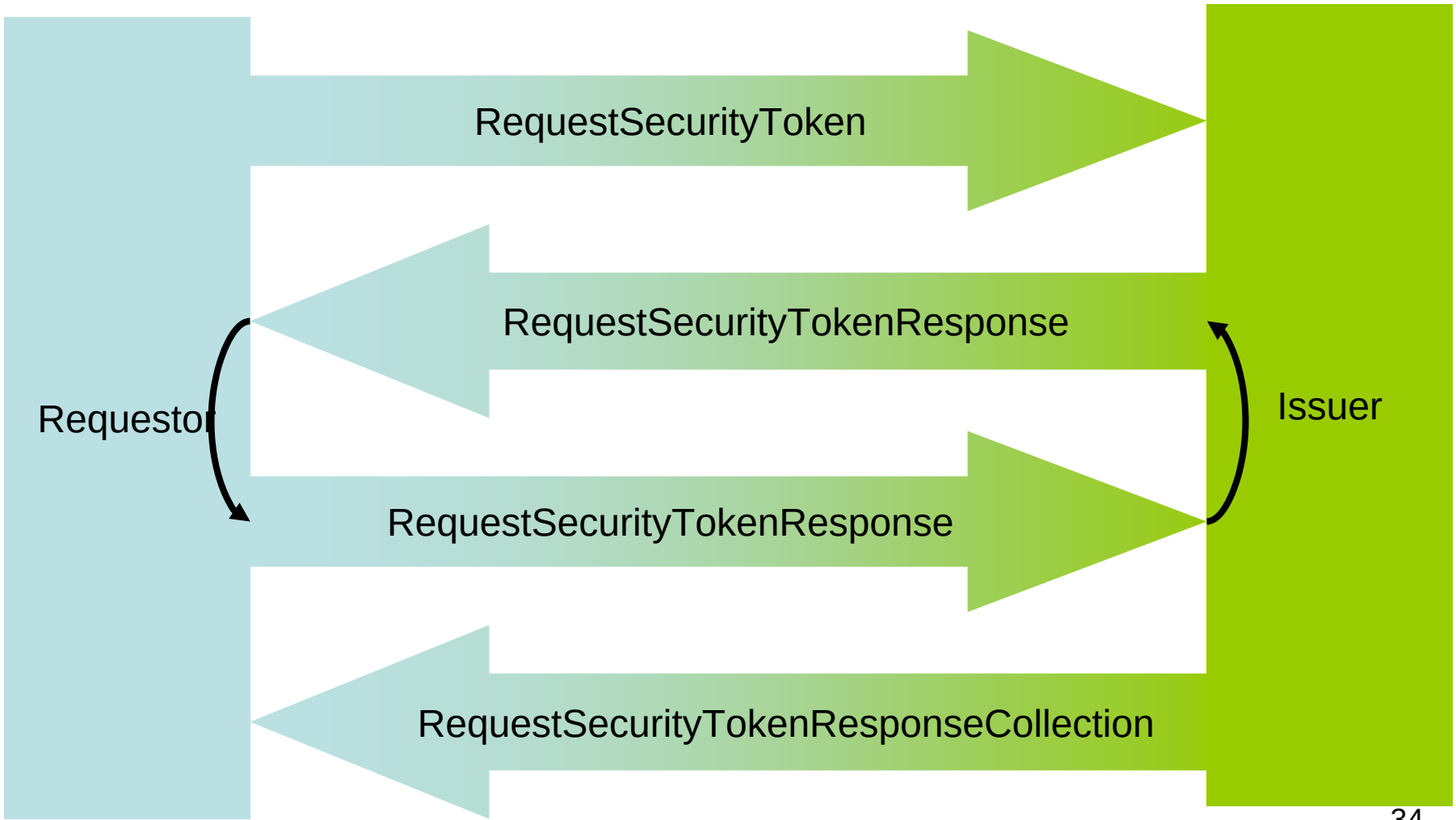
Negotiations and Challenges

- Framework supports multi-leg protocols
  - Challenges
  - Binary protocols
  - Key exchange tokens
- Intermediate legs are RSTR messages
- Final message contains issued token, tokens or token references

Protocol Messages



Protocol Messages



Protecting Exchanges

- Exchanges can be protected by using keys cryptographically bound to the exchange
- Algorithm defined for deriving key from hash of all exchanged message bodies

Protecting Exchanges

- Signature confirmation can be used in leg n to confirm legs 1 to n-1
- STS can include an authenticator in the RSTRC to the requestor
  - proves the key computed for the issuance
- Requestor can validate exchange without having to transmit data using the key

Authenticator Example

```
<wst:RequestSecurityTokenResponseCollection>
 <wst:RequestSecurityTokenResponse Context='Ctx1' >
 ...
 </wst:RequestSecurityTokenResponse>
 <wst:RequestSecurityTokenResponse Context='Ctx1'>
  <wst:Authenticator>
   <wst:CombinedHash>HHR70SK/Ps/WqOyJ69+6cw==</wst:CombinedHash>
  </wst:Authenticator>
 </wst:RequestSecurityTokenResponse>
</wst:RequestSecurityTokenResponse>
```

Bindings and profiles

- Issuance, Renewal et.al are bindings
  - General usage pattern
- Profiles constrain an existing binding
  - Specific token type
  - Challenge protocol

Customization

- Additional parameters
  - Many defined
    - E.g., Key type, size, etc.
  - Open model allows custom parameters
- Responses can contain custom data
  - May indicate anything it thinks important

Agenda

- WS-SecureConversation
  - Security Contexts and Sessions
  - Security Context Tokens
  - Derived Key Tokens

WS-SecureConversation

- Establishes a shared security context/session
  - Context contains keys/secrets and other information (e.g. claims)
- Context established using WS-Trust
  - Defines a separate profile of issuance, amendment, renewal, cancellation

WS-SecureConversation

- Defines two new token types
- SecurityContextToken
  - Light-weight token, carries an identifier
  - Associated with key material
  - Content unconstrained but no defined semantics
  - Can support farm scenarios
- DerivedKeyToken
  - Allows specification of derived keys

Requesting SCT Example

```
<soap:Envelope>
 <soap:Header>
 ...
 </soap:Header>
 <soap:Body>
  <wst:RequestSecurityToken>
   <wst:TokenType>
   http://schemas.xmlsoap.org/ws/2005/02/sc/sct
   </wst:TokenType>
   <wst:RequestType>
   http://schemas.xmlsoap.org/ws/2005/02/trust/RST/SCT
   </wst:RequestType>
  </wst:RequestSecurityToken>
 </soap:Body>
</soap:Envelope>
```

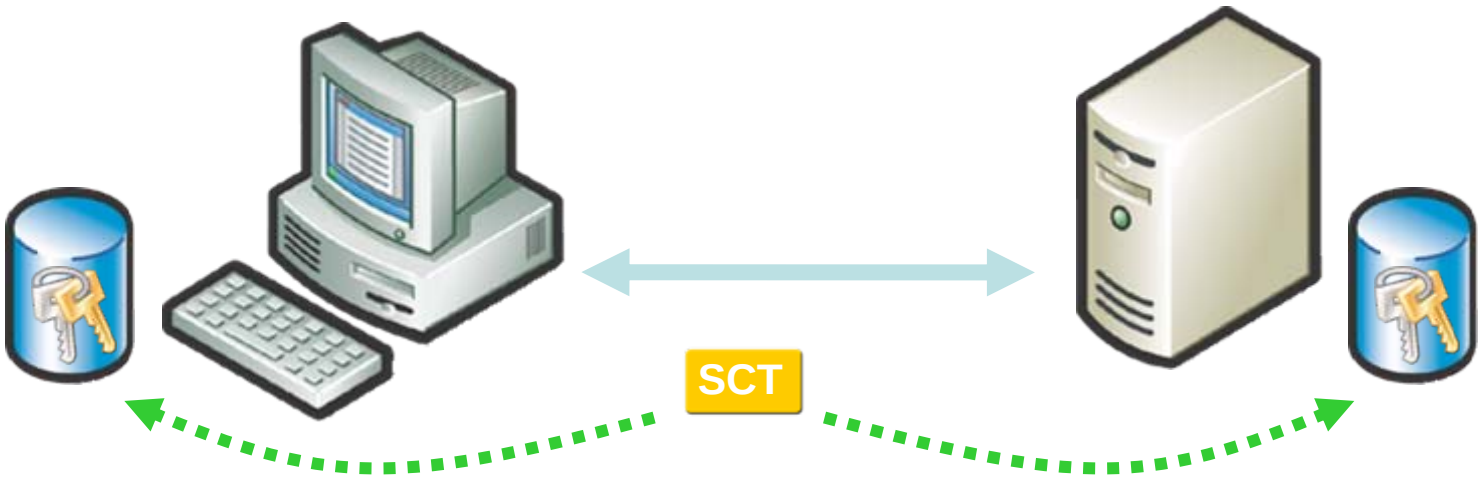
Returning SCT Example

```
<soap:Envelope>
 <soap:Header>
 ...
 </soap:Header>
 <soap:Body>
  <wst:RequestSecurityTokenResponse>
   <wst:RequestedSecurityToken>
    <wsc:SecurityContextToken>
     <wsc:Identifier>
     uuid:ed5ef37f-5822-4436-ad37-2e7c23d19b4c
     </wsc:Identifier>
    </wsc:SecurityContextToken>
   </wst:RequestedSecurityToken>
   <wst:RequestedProofToken>...</wst:RequestedProofToken>
  </wst:RequestSecurityTokenResponse>
 </soap:Body>
</soap:Envelope>
```

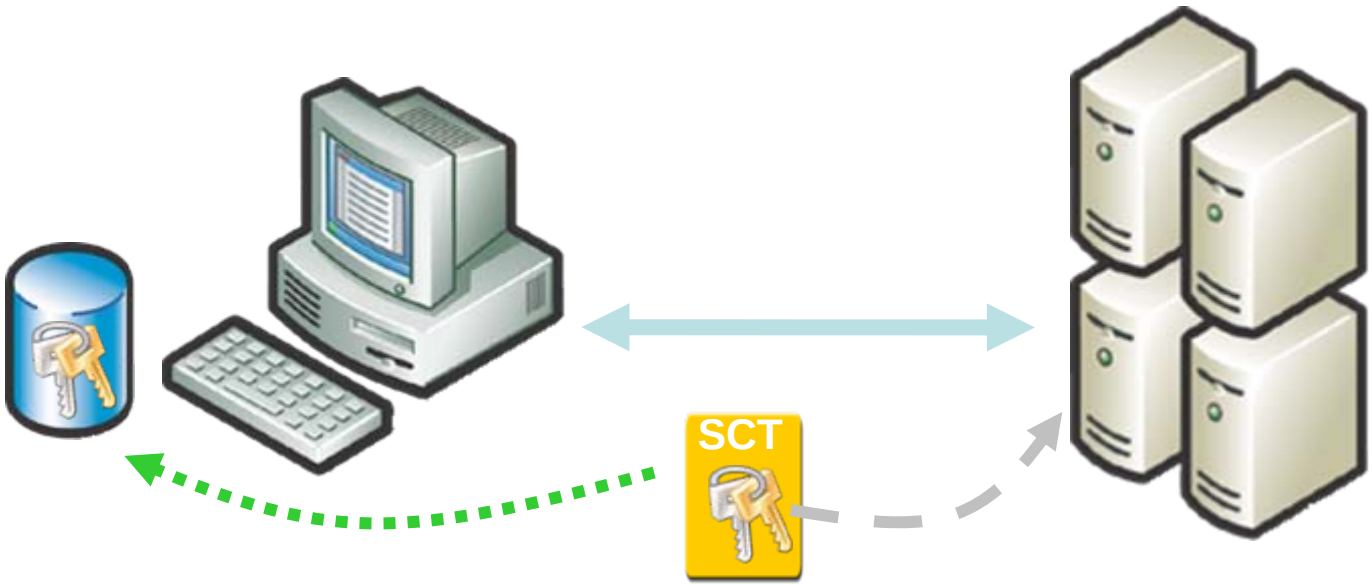
Using SCT Example

```
<soap:Envelope>
 <soap:Header>
  <wsse:Security>
   <wsc:SecurityContextToken>
    <wsc:Identifier>
    uuid:ed5ef37f-5822-4436-ad37-2e7c23d19b4c
    </wsc:Identifier>
   </wsc:SecurityContextToken>
  </wsse:Security>
 </soap:Header>
 <soap:Body>
  <xenc:EncryptedData>
  ...
   <wsse:SecurityTokenReference>
    <wsse:Reference URI='uuid:ed5ef37f-5822-4436-ad37-2e7c23d19b4c' />
   </wsse:SecurityTokenReference>
  ...
  </xenc:EncryptedData>
 </soap:Body>
</soap:Envelope>
```

Persisted Context



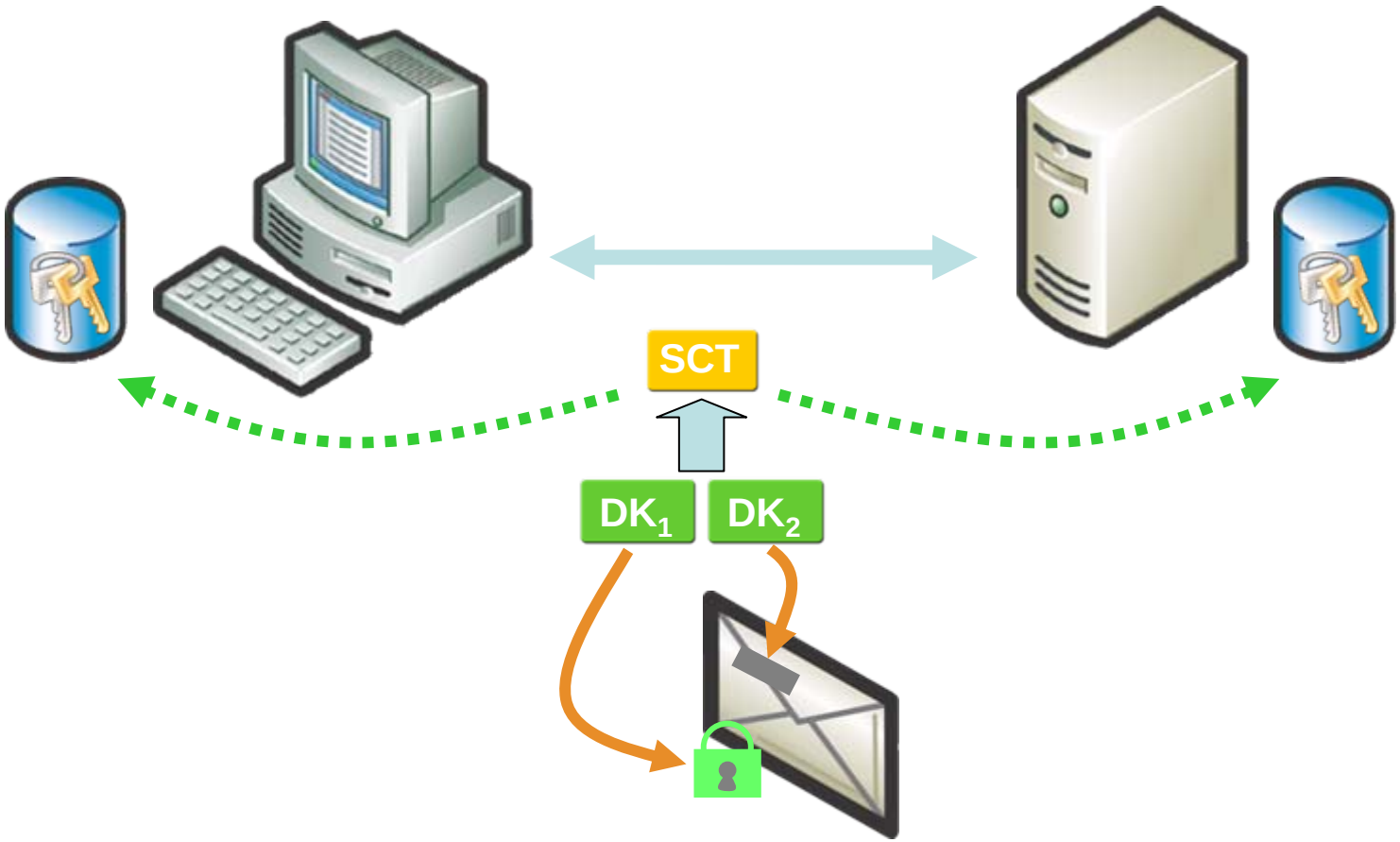
Farm Context



Derived Keys

- Exchanging keys and re-using them has security vulnerabilities
  - More secure to exchange a secret and derive keys from it
- Spec defines derived key usage
- Derived key tokens reference secret associated with some other token
  - Not restricted to referring to SCT

Derived Keys



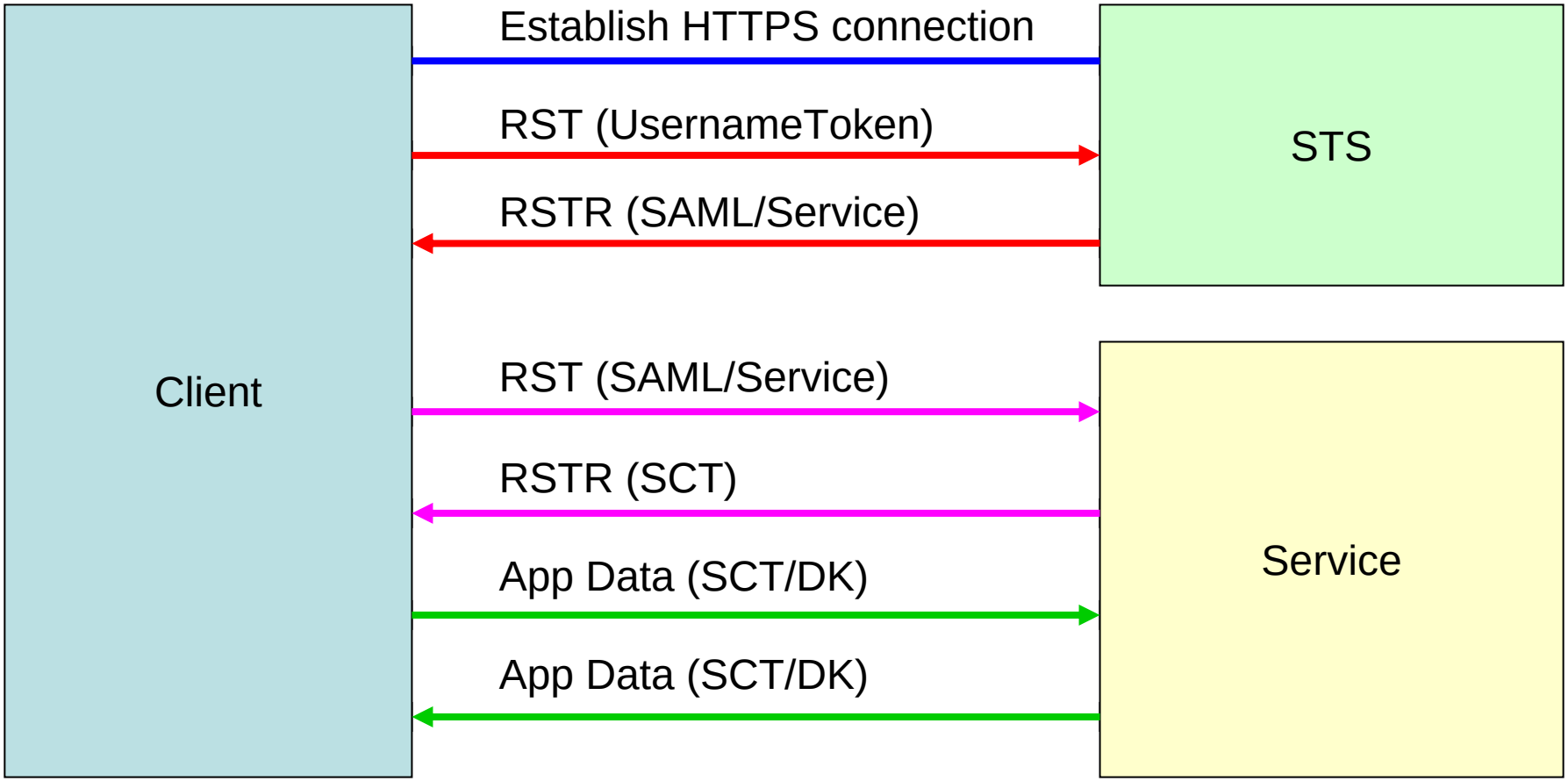
DerivedKeyToken Example

```
<wsc:DerivedKeyToken wsu:Id=‘dk1’ >
 <wsse:SecurityTokenReference>...</wsse:SecurityTokenReference>
 <wsc:Nonce>zIjS+kFH9p6iRF+W342wlw==</wsc:Nonce>
</wsc:DerivedKeyToken>
```

Trust and SC Interop Scenario

- Obtain token from STS using WS-Trust
  - Username->SAML
- Establish secure session with target service
  - SAML->SCT
- Send secure application dialogue
- Terminate dialogue

Interop Scenario



Scenario Variations

- Different initial token types used:
  - Username Token
  - X509 Certificate
  - Binary token (using SPnego blobs)
- With or without secure transport connection (SSL/TLS)
- Piggybacking (RSTR is included with the first application message)
- 3-way testing (Different companies provide client, STS and target service)

Agenda

- WS-SecurityPolicy
  - Introduction
  - Assertion types
  - Token assertions
  - Binding assertions
  - Protocol assertions

WS-Policy

- General framework for endpoints to express requirements
- Provides various operators
  - wsp:All, wsp:ExactlyOne
- Defines intersection
  - Based on matching of assertion names
- Domain Assertions are just XML elements

WS-Policy Example

```
<wsp:Policy>
 <wsp:ExactlyOne>
  <wsp:All>
   <A/>
   <B/>
  </wsp:All>
  <wsp:All>
   <A/>
   <C/>
  </wsp:All>
 </wsp:ExactlyOne>
</wsp:Policy>
```

WS-SecurityPolicy

- Builds on WS-Policy
  - Uses nested policy to provide scope
- Defines various groups of policy assertions
- Expressed in WSDL per WS-PolicyAttachment

WS-SecurityPolicy

- Designed for expressing security requirements
  - What needs to be protected
  - What tokens to use
  - Algorithms, reference types, etc.
- Constrains content and layout of wsse:Security header

WS-SecurityPolicy

- Uses assertions to define exchange pattern in use
- A given pattern has fixed and variable aspects
- Variable aspects modelled as properties
  - Policy assertions populate properties

Assertion types

- Protection assertions
- Token assertions
- Binding assertions
- Supporting Token assertions
- Protocol assertions

Protection Assertions

- Specify what needs to be protected
  - Integrity protection
  - Confidentiality
- Part and element based assertions defined

Protection Assertion Examples

```
<sp:SignedParts>
 <sp:Body />
 <sp:Header
  Namespace='http://schemas.xmlsoap.org/ws/2004/09/addressing' />
</sp:SignedParts>
```

```
<sp:SignedElements>
 <sp:XPath>/soap:Envelope/soap:Body</sp:XPath>
 <sp:XPath>
 /soap:Envelope/soap:Header/*[namespace-uri()=
'http://schemas.xmlsoap.org/ws/2004/09/addressing']</sp:XPath>
</sp:SignedElements>
```

Token Assertions

- Specify the type of token to be used
- Take the form of token type and nested version assertion
  - Other nested assertions also allowed
- Carry an inclusion attribute
  - Specifies which messages token appears in

Token Assertion Examples

```
<sp:X509Token sp:IncludeToken='.../IncludeToken/AlwaysToRecipient' >
 <wsp:Policy>
  <sp:WssX509V3Token10 />
 </wsp:Policy>
</sp:X509Token>
```

```
<sp:SamlToken
  sp:IncludeToken='.../IncludeToken/Always' >
 <wsp:Policy>
  <sp:WssSamlV11Token11 />
 </wsp:Policy>
</sp:SamlToken>
```

Security Bindings

- Collections of properties
  - Tokens
  - Algorithms
  - Processing order et.al.
- Properties populated by assertions
  - Some have default values
- Spec defines three broad types

Security Binding Properties

- [AlgorithmSuite]
  - Populated by sp:AlgorithmSuite and nested assertions
- [Timestamp]
  - Defaults to false
  - sp:IncludeTimestamp sets property to true
- [Protection Order]
  - Defaults to SignBeforeEncrypting
  - sp:EncryptBeforeSigning sets property to EncryptBeforeSigning

Security Binding Properties

- [Signature Protection]
  - Defaults to false
  - sp:EncryptSignature sets property to true
- [Token Protection]
  - Defaults to false
  - sp:ProtectTokens sets property to true
- [Entire Header and Body Signatures]
  - Defaults to false
  - sp:OnlySignEntireHeadersAndBody sets property to true.

Security Binding Properties

- [Security Header Layout]
  - Populated by sp:Layout assertion and nested assertions
  - Defaults to ‘Lax’

Transport Binding

- Indicates that the transport layer is used to satisfy the security requirements
- Allows specification of such things as
  - Security header layout
  - Timestamp presence
  - Supporting tokens

Transport Binding Example

```
<sp:TransportBinding>
 <wsp:Policy>
  <sp:TransportToken>
   <wsp:Policy>
    <sp:HttpsToken />
   </wsp:Policy>
  </sp:TransportToken>
  <sp:AlgorithmSuite><sp:Basic256Rsa15 /></sp:AlgorithmSuite>
  <sp:IncludeTimestamp />
 </wsp:Policy>
</sp:TransportBinding>
```

Symmetric Binding

- Indicates that the message layer is used to satisfy the security requirements
- Defines [Encryption Token] and [Signature Token] properties
- Where multiple messages are exchanged the tokens perform the same functions for all messages

Symmetric Binding Example

```
<sp:SymmetricBinding>
 <wsp:Policy>
  <sp:ProtectionToken>
   <wsp:Policy>
    <wsp:KerberosToken sp:IncludeToken='.../IncludeToken/Once' />
   </wsp:Policy>
  </sp:ProtectionToken>
  <sp:AlgorithmSuite><sp:Basic128Rsa15/></sp:AlgorithmSuite>
  <sp:EncryptBeforeSigning />
 </wsp:Policy>
</sp:SymmetricBinding>
```

Asymmetric Binding

- Indicates that the message layer is used to satisfy the security requirements
- Defines [Initiator Token] and [Recipient Token] properties
- Where multiple messages are exchanged the tokens perform different functions

Asymmetric Binding Example

```
<sp:AsymmetricBinding>
 <wsp:Policy>
  <sp:InitiatorToken>
   <wsp:Policy>
    <wsp:X509Token
  sp:IncludeToken='.../IncludeToken/AlwaysToRecipient' />
   </wsp:Policy>
  </sp:InitiatorToken>
  <sp:RecipientToken>
   <wsp:Policy>
    <wsp:X509Token
  sp:IncludeToken='.../IncludeToken/Never' />
   </wsp:Policy>
  </sp:RecipientToken>
  <sp:AlgorithmSuite><sp:Basic128Rsa15/></sp:AlgorithmSuite>
  <sp:EncryptBeforeSigning />
 </wsp:Policy>
</sp:AsymmetricBinding>
```

Supporting Tokens

- Services may require multiple sets of claims to be presented
- Corresponds to additional tokens in a message

Supporting Token types

Type	**Sign main signature?**	**Signed by main token?**
Supporting	No	No
Endorsing	Yes	No
Signed	No	Yes
Signed Endorsing	Yes	Yes

Supporting Tokens Example

```
<sp:TransportBinding>
 <wsp:Policy>
  <sp:TransportToken>
   <wsp:Policy>
    <sp:HttpsToken />
   </wsp:Policy>
  </sp:TransportToken>
  <sp:AlgorithmSuite><sp:Basic256Rsa15 /></sp:AlgorithmSuite>
  <sp:IncludeTimestamp />
  <sp:SupportingTokens>
   <wsp:Policy>
    <sp:UsernameToken sp:IncludeToken='.../IncludeToken/Once' />
   </wsp:Policy>
  </sp:SupportingTokens>
 </wsp:Policy>
</sp:TransportBinding>
```

WSS Assertions

- Specify supported version of WSS
  - sp:Wss10
  - sp:Wss11
- Specify supported token reference mechanisms via boolean properties
- Specify Signature Confirmation requirements for WSS 1.1

WSS10 Properties

Property Name	Default Value	Assertion
[Direct References]	True	None
[Key Identifier References]	False	sp:MustSupportKeyIdentifierReferences
[Issuer Serial References]	False	sp:MustSupportIssuerSerialReferences
[External URI References]	False	sp:MustSupportExternalURIReferences
[Embedded Token References]	False	sp:MustSupportEmbeddedTokenReferences

WSS11 Properties

Property Name	Default Value	Assertion
[Thumbprint References]	False	sp:MustSupportThumbprintReferences
[Encrypted Key References]	False	sp:MustSupportEncryptedKeyReferences
[Signature Confirmation]	False	sp:MustSupportExternalURIReferences
[Embedded Token References]	False	sp:RequireSignatureConfirmation

WSS Assertion Examples

```
<sp:Wss10>
 <wsp:Policy>
  <sp:MustSupportRefKeyIdentifier />
  <sp:MustSupportRefExternalURI />
 </wsp:Policy>
</sp:Wss10>
```

```
<sp:Wss11>
 <wsp:Policy>
  <sp:MustSupportRefExternalURI />
  <sp:MustSupportRefThumbprint />
  <sp:RequireSignatureConfirmation />
 </wsp:Policy>
</sp:Wss11>
```

Trust Assertions

- Specify supported version of WS-Trust and associated properties
  - sp:Trust10

Trust Properties

Property Name	Default Value	Assertion
[Client Challenge]	False	sp:MustSupportClientChallenge
[Server Challenge]	False	sp:MustSupportServerChallenge
[Client Entropy]	False	sp:RequireClientEntropy
[Server Entropy]	False	sp:RequireServerEntropy
[Issued Tokens]	False	sp:MustSupportIssuedTokens

Trust Assertion Example

```
<sp:Trust10>
 <wsp:Policy>
  <sp:RequireClientEntropy />
  <sp:RequireServerEntropy />
 </wsp:Policy>
</sp:Trust10>
```

Where are we?

- WS-Trust provides flexible framework for building token processing protocols
- WS-SecureConversation provides secure sessions
- WS-SecurityPolicy describes security configuration
  - WSS, WS-Trust, WS-SecureConvesation

Backup

Token Assertions

- UsernameToken
  - WssUsernameToken10
  - WssUsernameToken11
- IssuedToken

Token Assertions

- X509Token
  - WssX509V3Token10
  - WssX509Pkcs7Token10
  - WssX509PkiPathV1Token10
  - WssX509V1Token11
  - WssX509V3Token11
  - WssX509Pkcs7Token11
  - WssX509PkiPathV1Token11

Token Assertions

- KerberosToken
  - WssKerberosV5ApReqToken11
  - WssGssKerberosV5ApReqToken11
- SpnegoContextToken
- SecurityContextToken
  - SC200502SecurityContextToken
- SecureConversationToken
  - SC200502SecurityContextToken

Token Assertions

- SamlToken
  - WssSamlv11Token10
  - WssSamlv11Token11
  - WssSamlv20Token11
- RelToken
  - WssRel10Token10
  - WssRel20Token10
  - WssRel10Token11
  - WssRel20Token11
- HttpsToken

Transport Binding Properties

- [Transport Token]

Symmetric Binding Properties

- [Encryption Token]
- [Signature Token]
- [Protection Order]
- [Signature Protection]
- [Token Protection]
- [Entire Header and Body Signatures]

Asymmetric Binding Properties

- [Initiator Token]
- [Recipient Token]
- [Protection Order]
- [Signature Protection]
- [Token Protection]
- [Entire Header and Body Signatures]

*Binding Properties

- [Algorithm Suite]
  - Has sub-properties
- [Security Header Layout]
- [Timestamp]
- [Supporting Tokens]
- [Signed Supporting Tokens]
- [Endorsing Supporting Tokens]
- [Signed Endorsing Supporting Tokens]