JavaOne℠

Metro Web Services Security Usage Scenarios

Harold Carr
Sun Microsystems
Metro Architect

Jiandong Guo
Sun Microsystems
Metro Security Architect

Metro Security Profiles

- ***Username Authentication with Symmetric Keys***
- ***Mutual Certificates Security***
- Symmetric Binding with Kerberos Tokens
- Transport Security (SSL)
- ***Message Authentication over SSL***
- SAML Authorization over SSL
- Endorsing Certificate
- SAML Sender Vouches with Certificates
- SAML Holder of Key
- ***STS Issued Token***
- STS Issued Token with Service Certificate
- STS Issued Endorsing Token
- STS Issued Supporting Token

Metro Security Profiles - Guarantees

	Peer Authentication	Message Origin Authentication	Message Integrity	Message Confidentiality	End-To-End Security
Username Authentication With Symmetric Key	✓	✓	✓	✓	✓
Mutual Certificates Security	✓	✓	✓	✓	✓
Symmetric Binding With Kerberos Tokens	✓	✓	✓	✓	✓
Transport Security (SSL)	✓		✓	✓	
Message Authentication Over SSL	✓		✓	✓	
SAML Authorization Over SSL	✓		✓	✓	
Endorsing Certificates	✓	✓	✓	✓	✓
SAML Sender Vouches With Certificates	✓		✓	✓	✓
SAML Holder Of Key	✓	✓	✓	✓	✓
STS Issued Token	✓	✓	✓	✓	✓
STS Issued Token With Service Certificates	✓	✓	✓	✓	✓
STS Issued Endorsing Token	✓	✓	✓	✓	✓
STS Issued Supporting Token	✓		✓	✓	✓

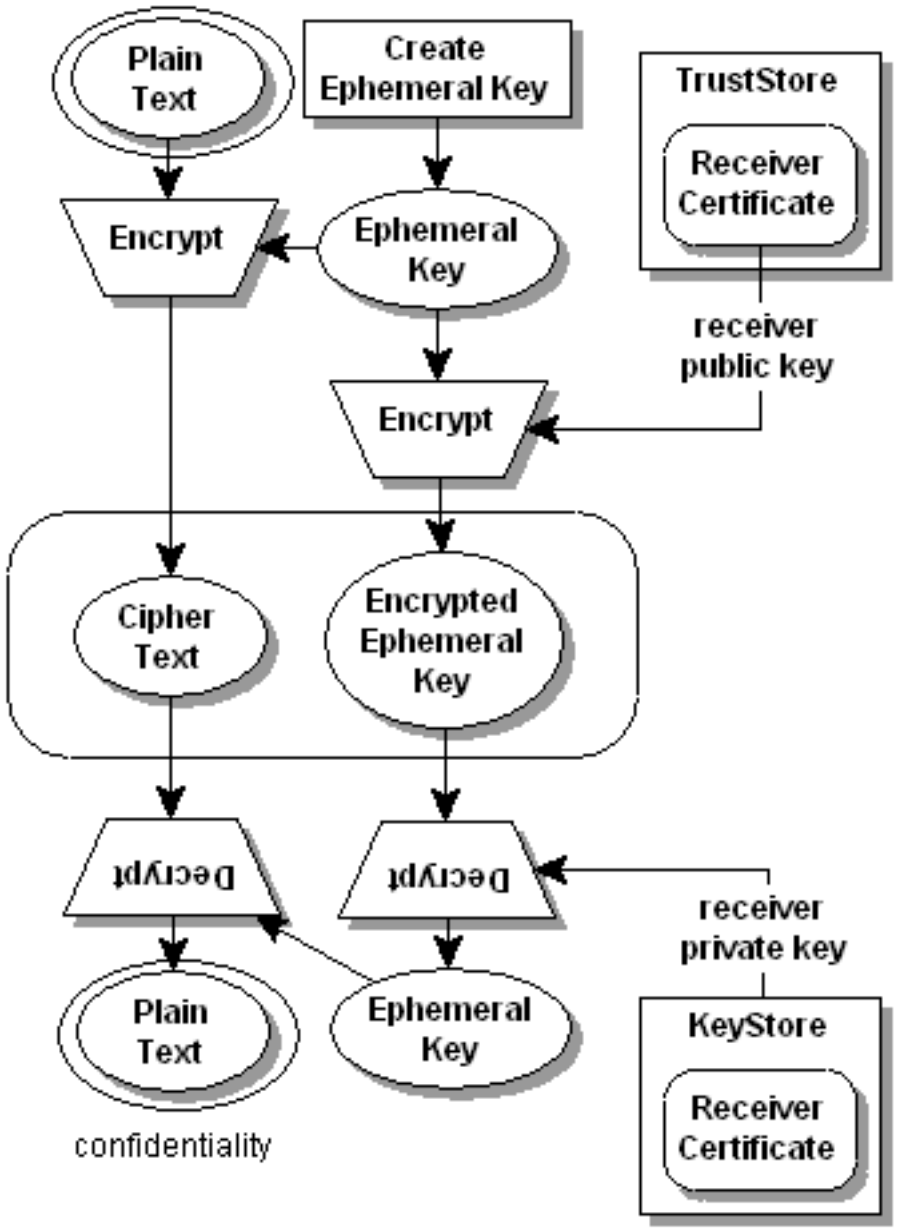
Selected Metro Security Profiles

- Message Authentication over SSL
  - Use: client has username/password or X.509 relationship with service
  - Point-to-point
- Username Authentication With Symmetric Keys
  - Use: client has username/password relationship with service
  - End-to-end

Selected Metro Security Profiles

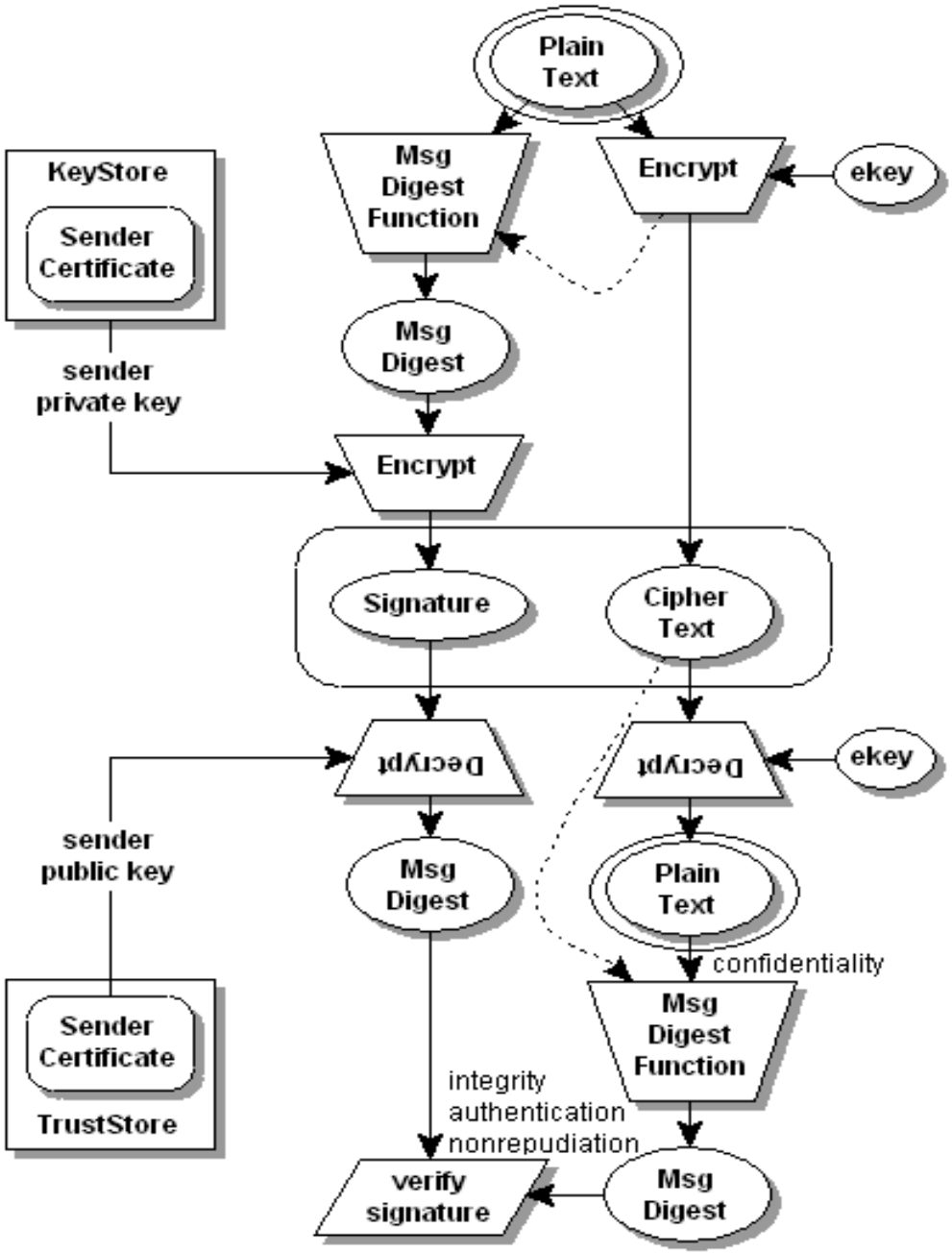
- Mutual Certificates
  - Use: interaction with known partners
  - Certificates exchanged out-of-band in advance
  - End-to-end
- STS Issued Token
  - Use: role-based access control
  - 3rd party handles authentication --- not service
  - End-to-end

Pattern
Confidentiality



Pattern

Integrity, Authentication, Nonrepudiation

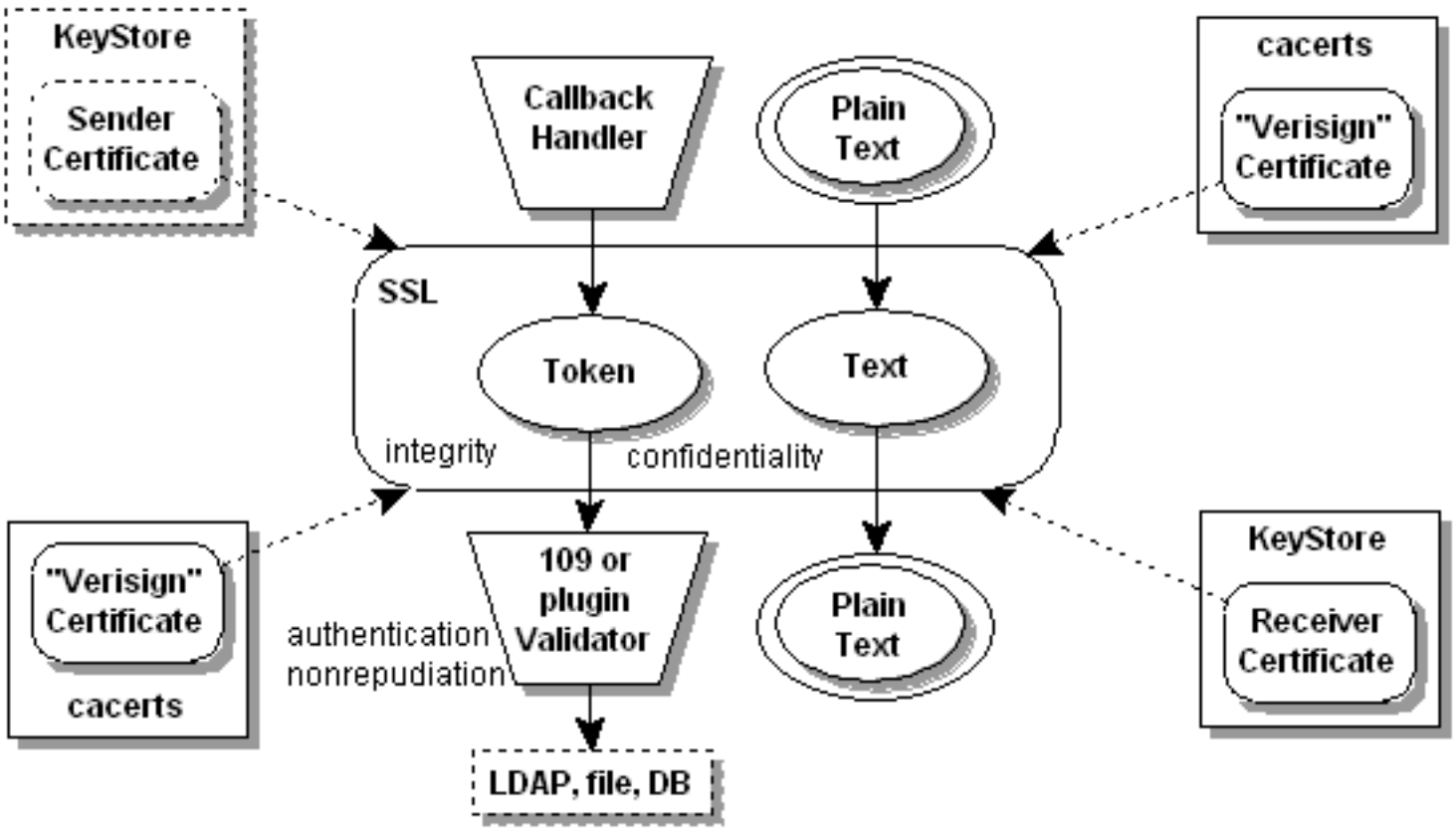


Message Authentication over SSL

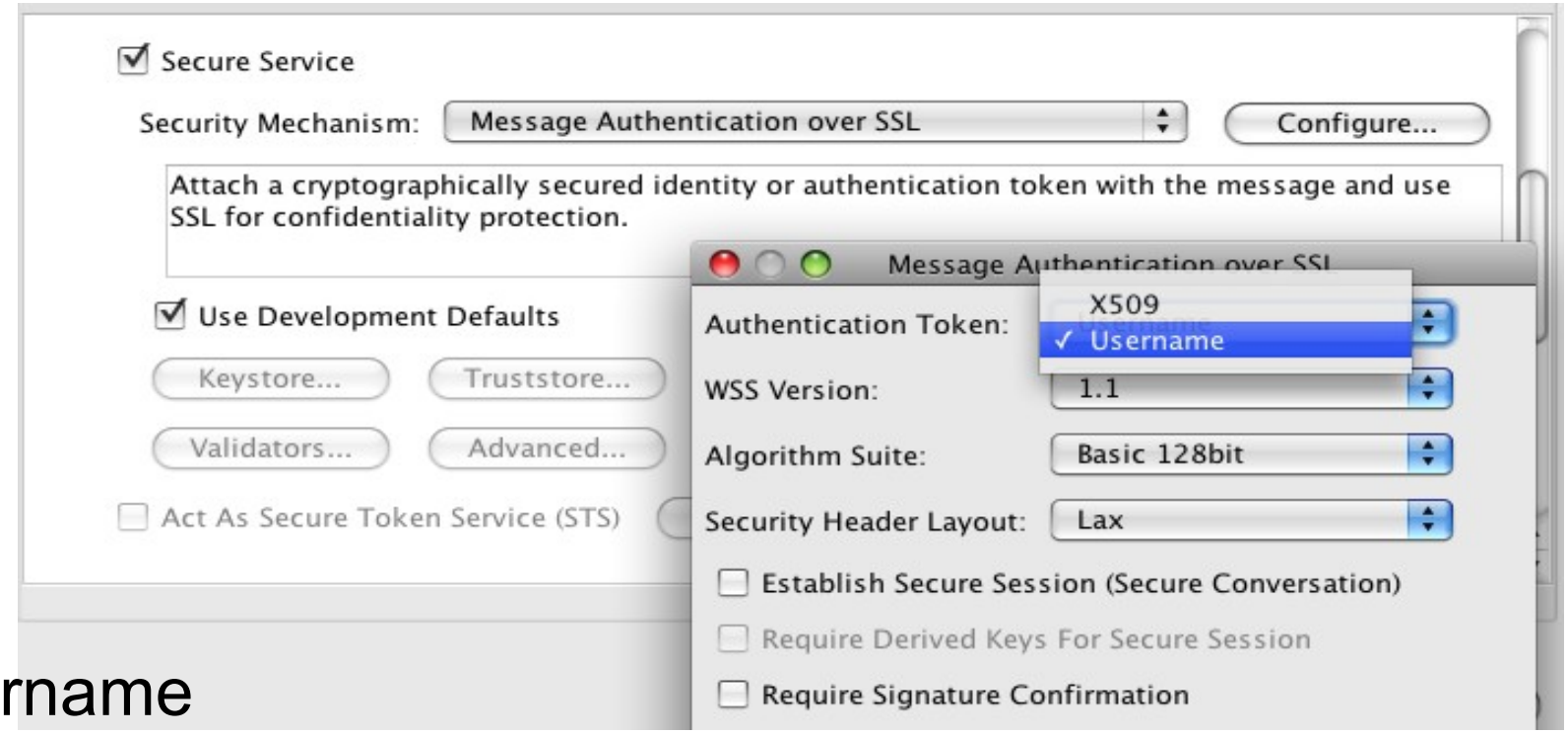
Use case: client/service ID/Auth token relationship

- Use token to plug into service's ID/Authentication infrastructure
- Options: Username/Password,X.509, (& SAML)
- Client dynamically obtain service's certificate; verifies with trusted CA
- Point-to-point; (SSL only use case: client with no relationship with service)

Message Authentication over SSL



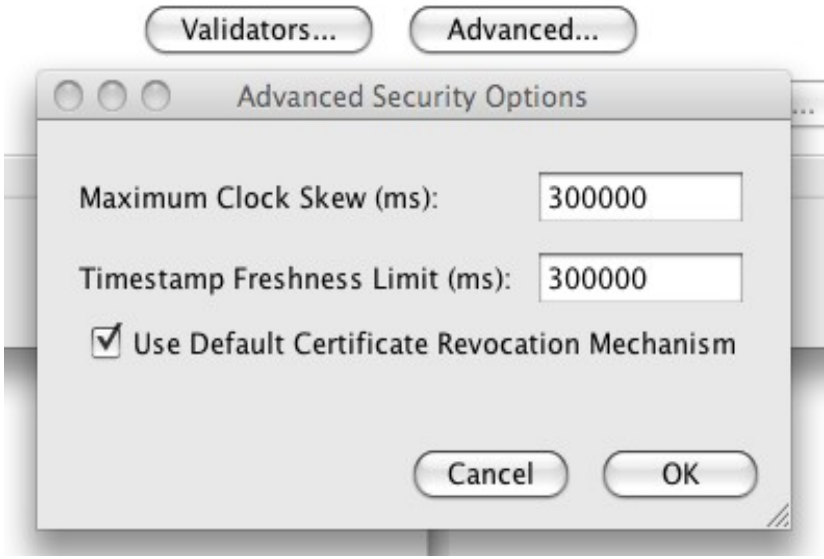
Message Authentication over SSL



- Token:
  - Username
  - X.509: better security than username
  - SAML: get user credentials from 3rd party; attributes
- WSS: 1.1 enables signature confirmation
- Sig confirmation: avoid man-in-the-middle attack

Message Authentication over SSL
Advanced

- Certificate Revocation:
  - Use: do not accept certs on reject list (CRL)
  - Click: use container mechanism
  - No click & no cert validator:
    - Always accept
  - No click & cert validator:
    - Custom revocation

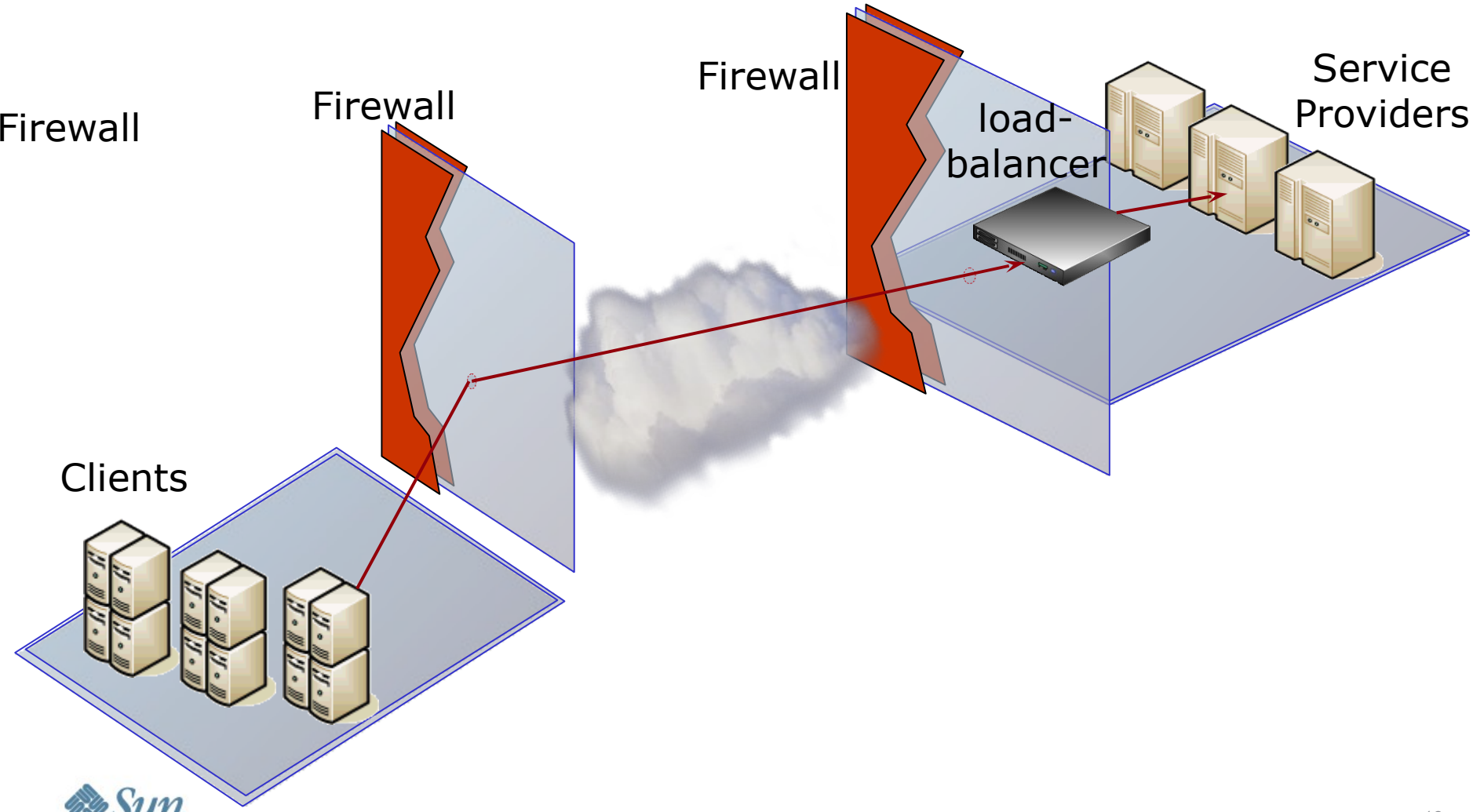


Message Authentication over SSL
Summary

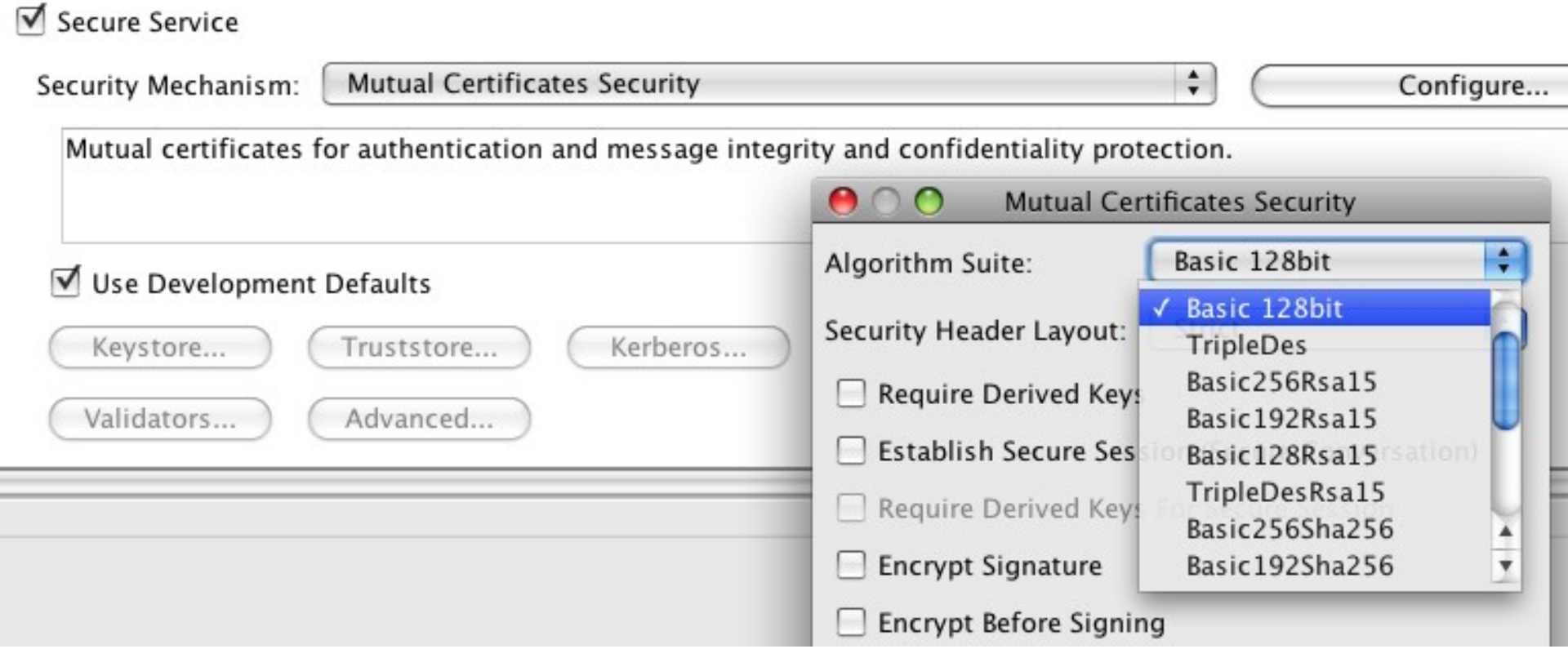
- Use:
  - Client authentication at message level
    - Auth can be propagated further after initial wire exchange
  - When you can't exchange certs in advance
  - Reuse existing web SSL infrastructure
- PROS:
  - Transport level Performance; little setup
  - Message level authentication
- CONS:
  - Point-to-point (confidentiality)

End-to-end Message Level Security

Options for message-level profiles



End-to-end security options



128 bit default: built into JDK; larger needs JCE adjustment

End-to-end security options

Algorithm Suite (applies to most security profiles)

- Algorithm and key lengths used for crypto ops
- Decisions:
  - Security versus performance
  - What is supported on both ends (e.g. Legacy)
- Best: AES + RSA OEAP:
  - Basic256SHA256, Basic192SHA256, Basic128SHA256, Basic256, Basic192, *Basic 128bit
- Legacy: TripleDES, Basic256RSA15, Basic192RSA15, Basic128RSA15, TripleDESRSA15, TripleDESSHA256, Basic256SHA256RSA15, Basic192SHA256RSA15, Basic128SHA256RSA15, TripleDESSHA256RSA15

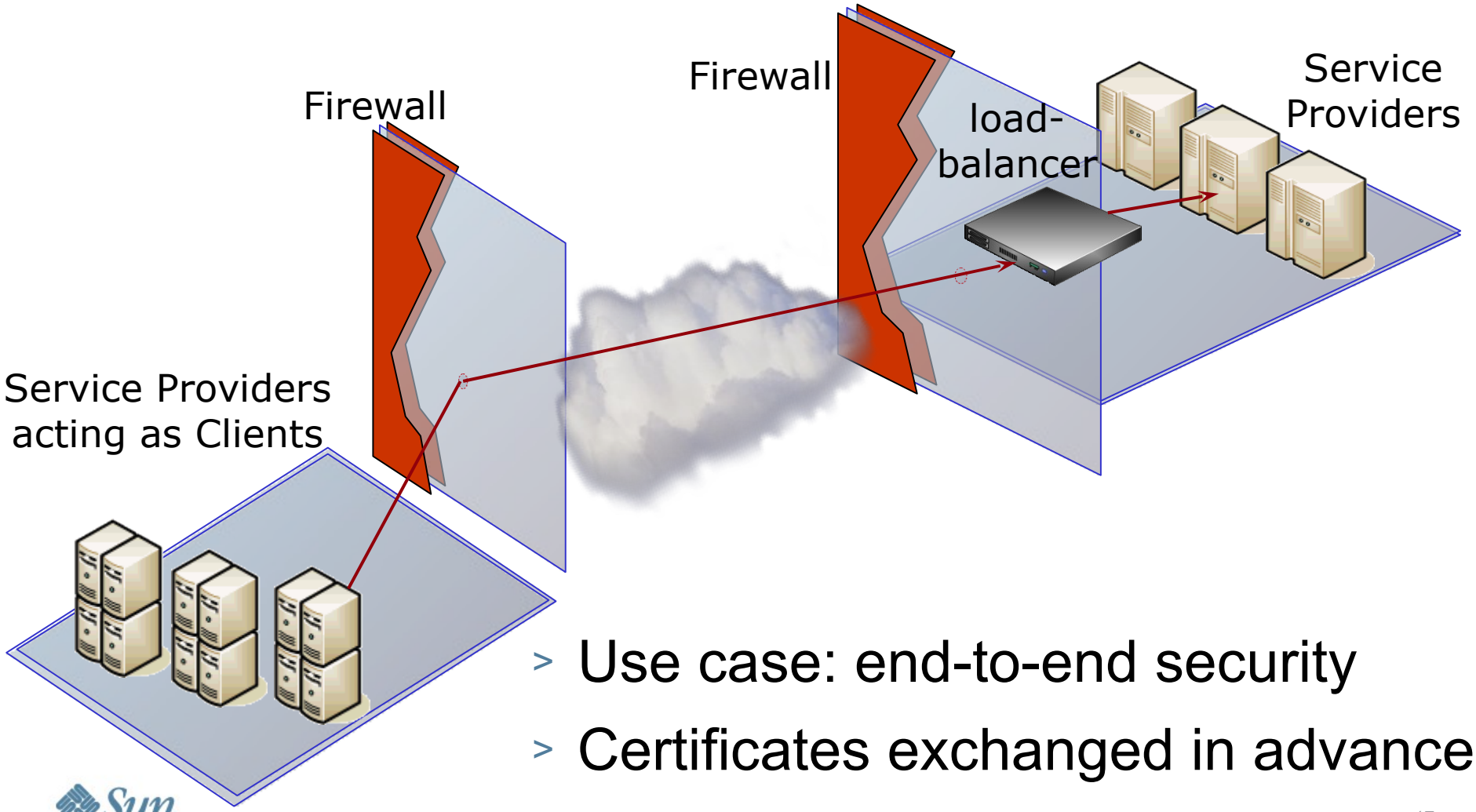
End-to-end security options

Security header layout (applies to all profiles)

- Rules for adding items to security header
- Recommended:
  - Strict (default): declare before use
- Legacy:
  - Lax: any order conforming to Oasis WSS
  - Lax (Timestamp First/Last):
    - Same as Lax but timestamp placed first/last

Mutual Certificates Security

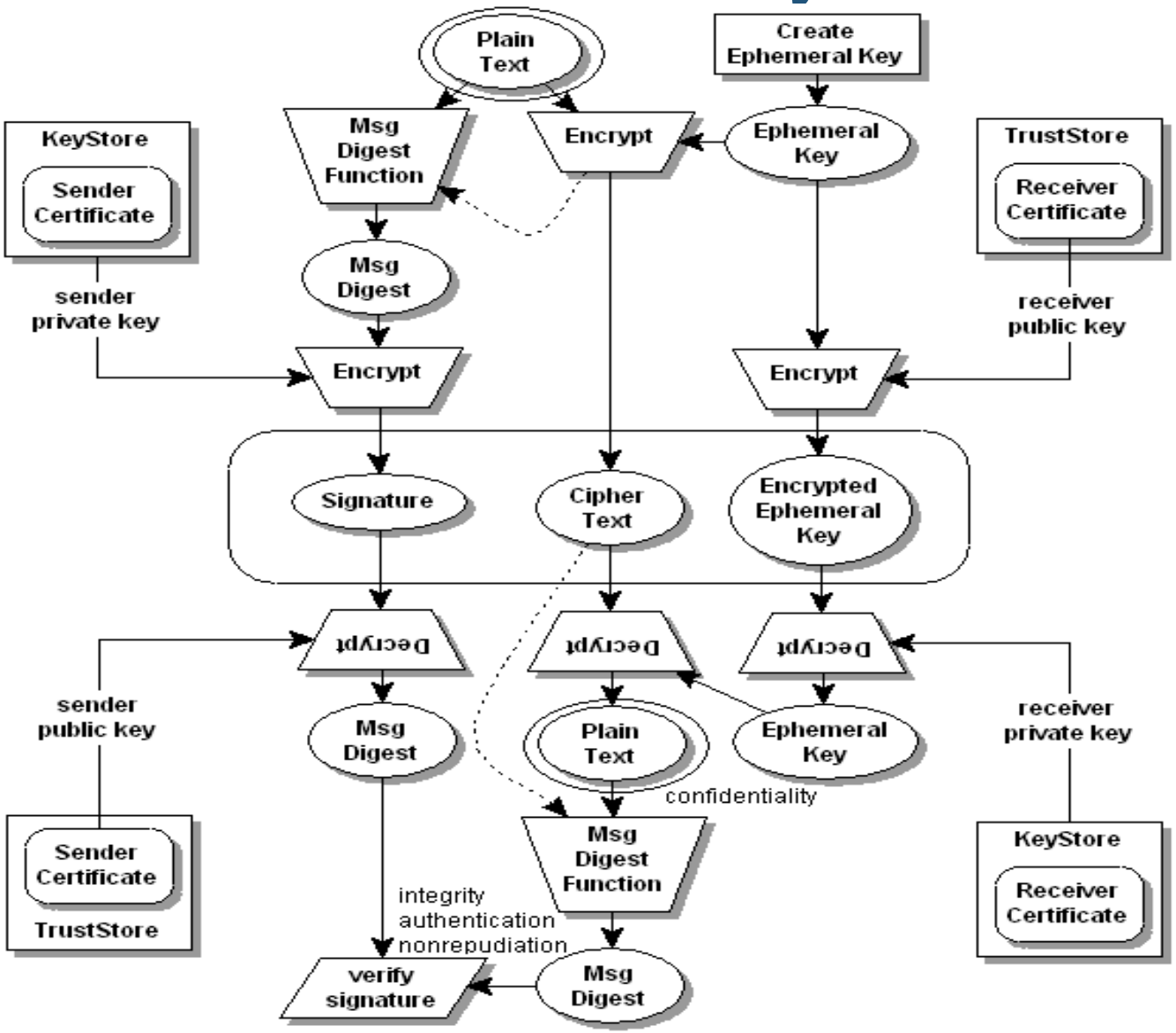
Use case: service to service (e.g., known partners)



- Use case: end-to-end security
- Certificates exchanged in advance

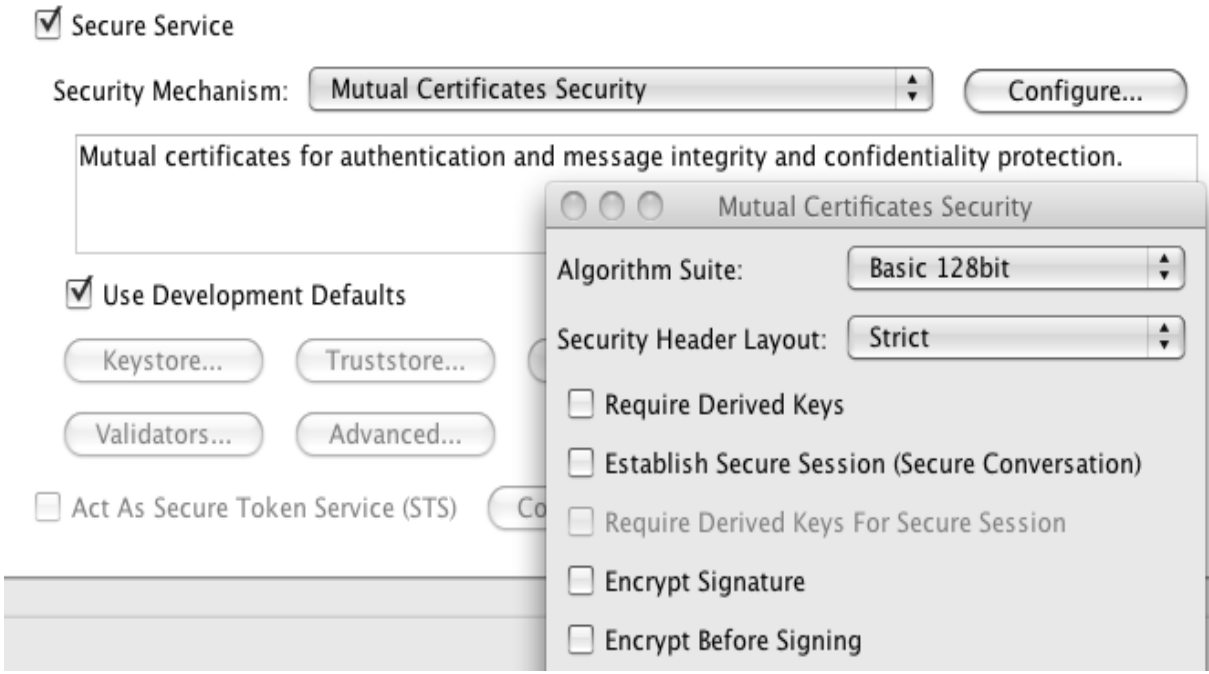
Mutual Certificates Security

Firewall



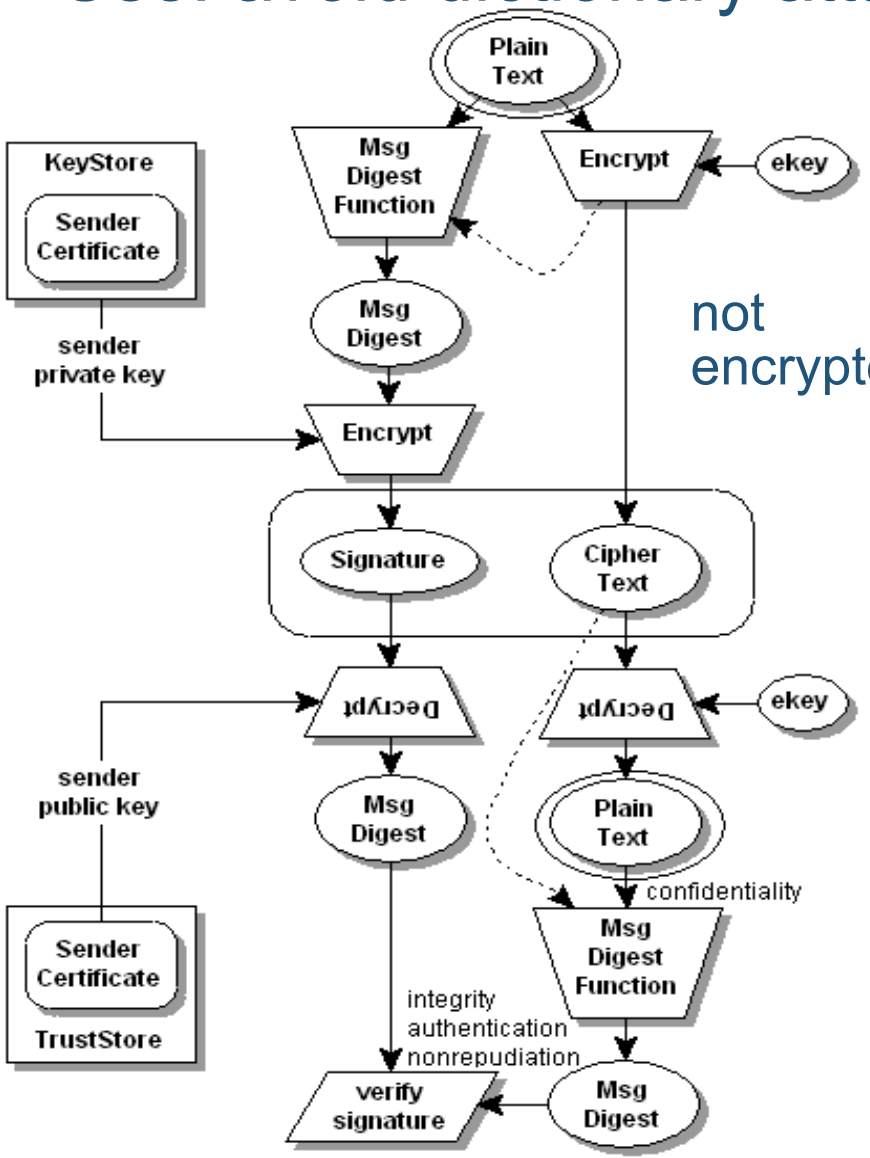
Mutual Certificates Security

- Derived Keys:
  - No need
  - Key only used once
- Encrypt Sig:
  - Avoid dictionary attack
  - CON: performance
- Encrypt before signing:
  - Reject msg with bad sig before decrypting (potentially large) msg
  - Better performance (needs measurement)

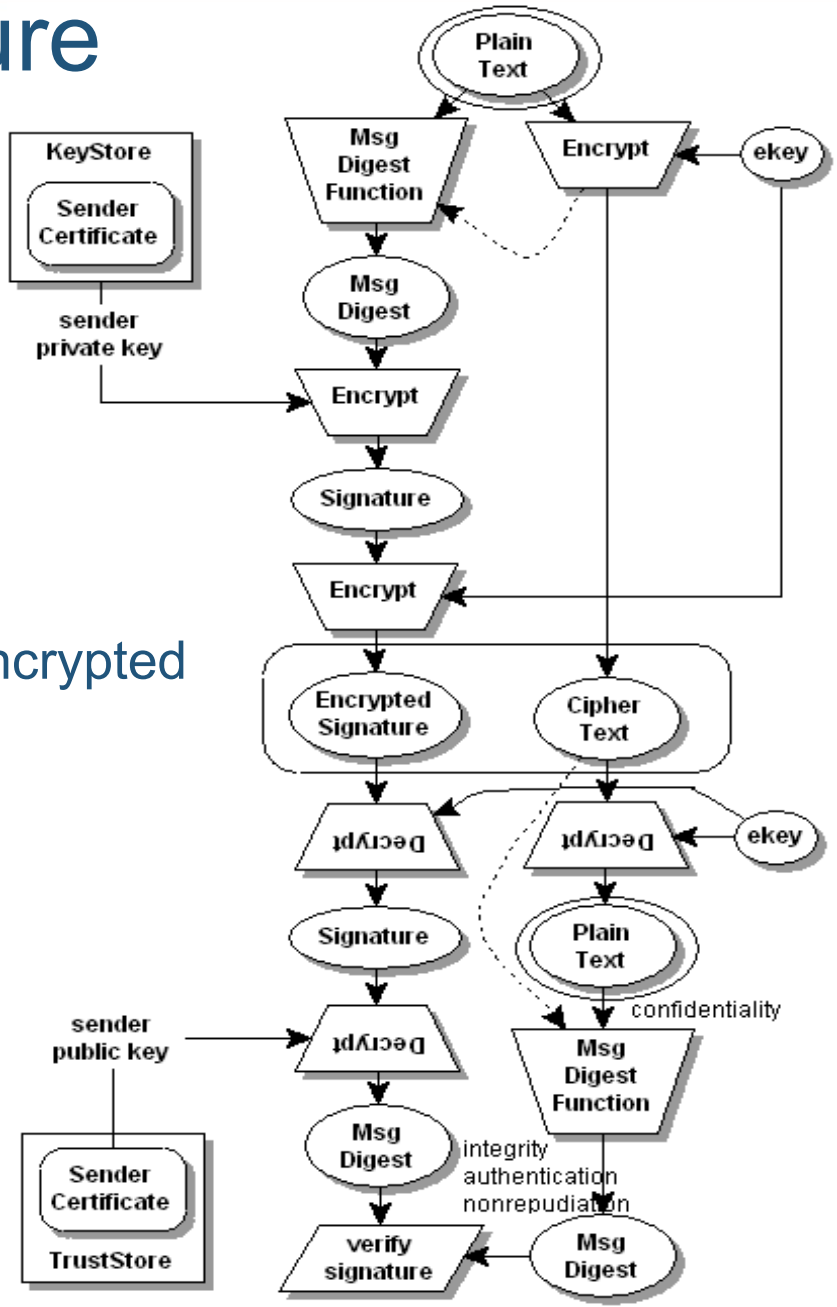


Option: Encrypt Signature

Use: avoid dictionary attack



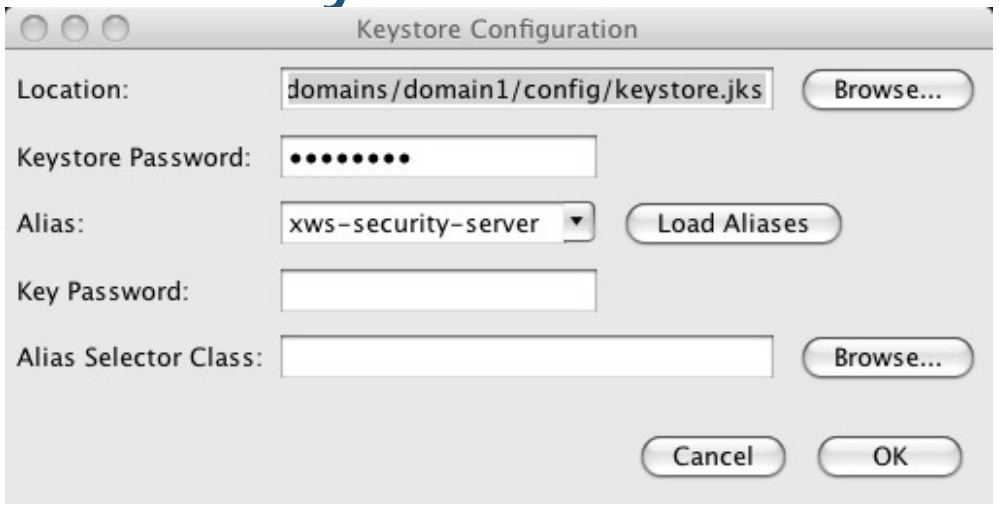
not encrypted

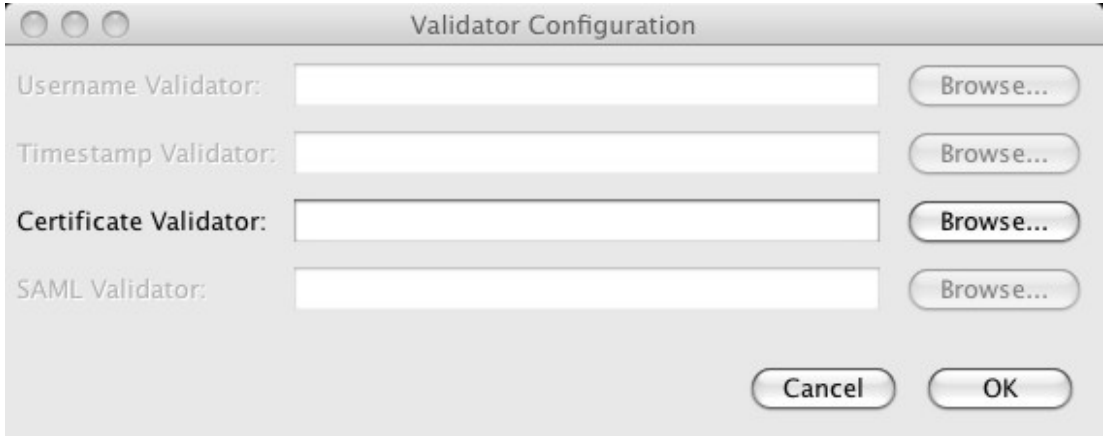


encrypted

Mutual Certificates Security

- Keystore:
  - Alias: static
  - Selector class:
    - Runtime
- Certificate validator:
  - Signature, identity, CA, revoked, …
  - Plug into other frameworks
    - Ex: OCSP
    - Company specific CAs



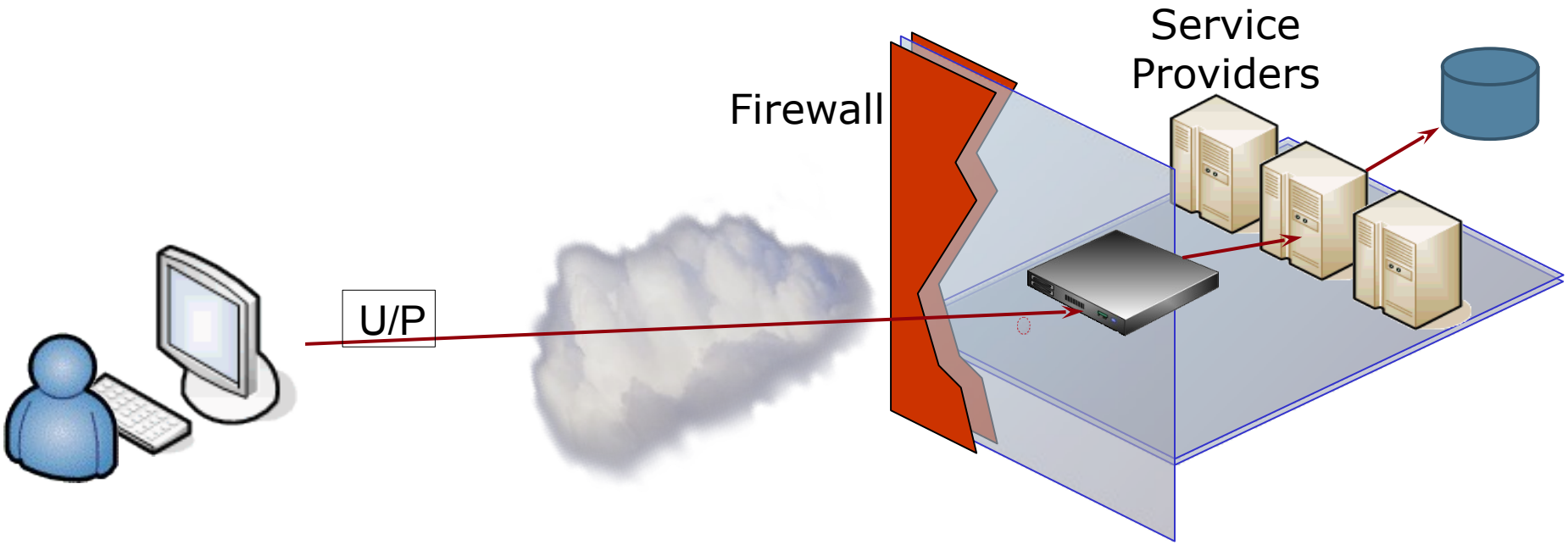


Mutual Certificates Security
Summary

- Use: server to server (i.e., partner to partner)
- Certificates exchanged OOB in advance
- PROS:
  - End-to-end security (e.g., through load-balancers)
- CONS:
  - Secure response does same as request
    - Generate and encrypt a NEW ephemeral key
    - Use that NEW key to encrypt and sign the response
  - See “Endorsing Certificate” profile for optimization
    - (note: endorsing is for 3rd party endorsing signature)

Username Auth with Symmetric Keys

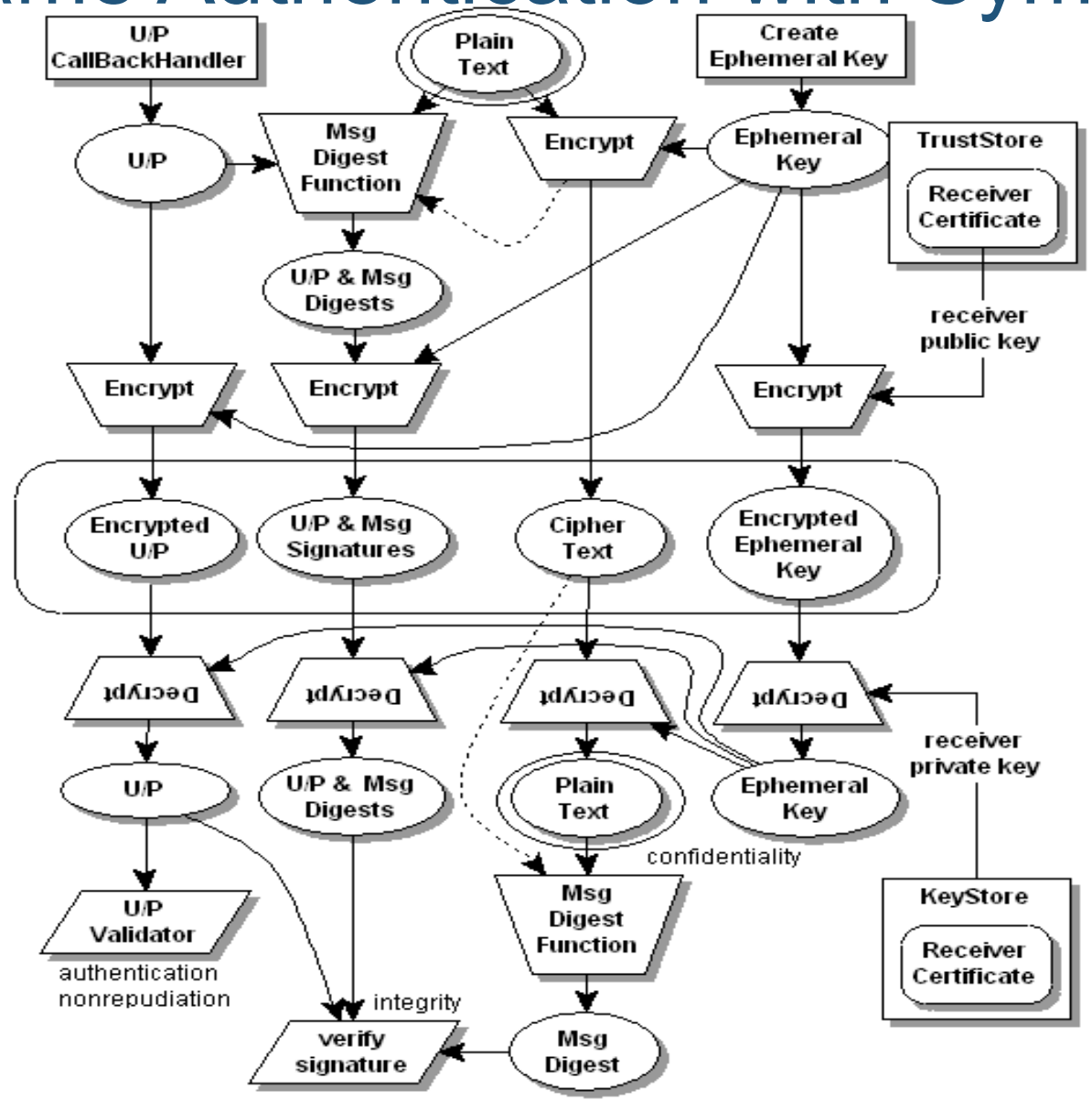
Use case: client/service ID/Auth token relationship



- Use token to plug into service's ID/Authentication infrastructure; no client certificate required

Username Authentication with Sym Keys

Firewall



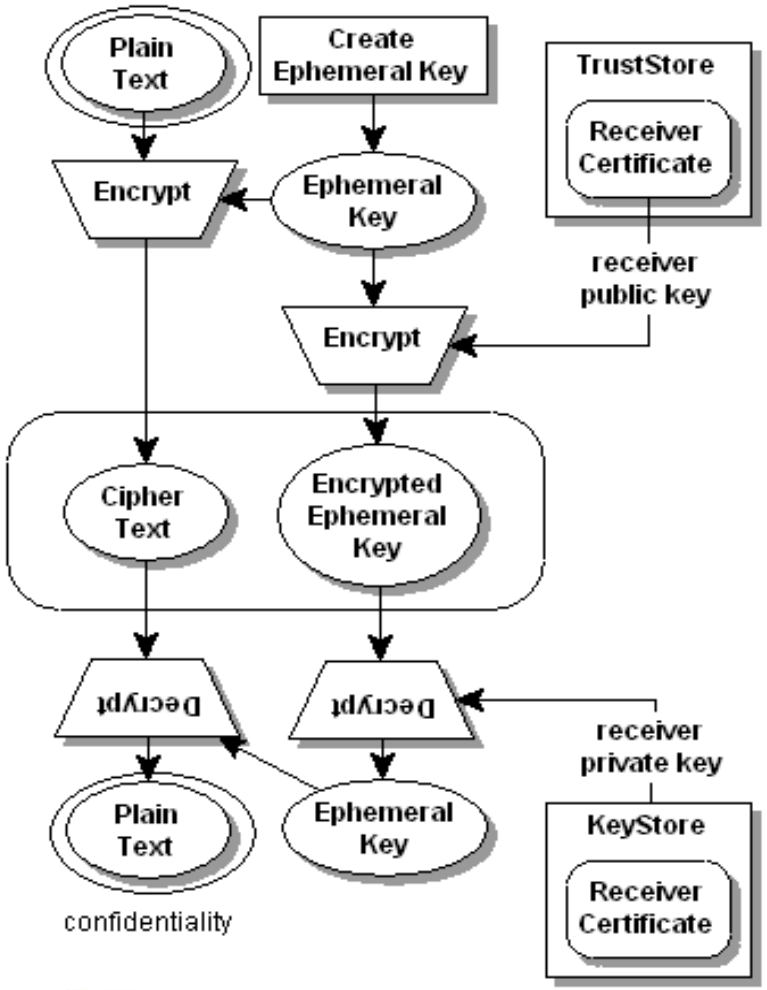
Username Auth with Symmetric Keys

Options: Derived Keys and Secure Conversation

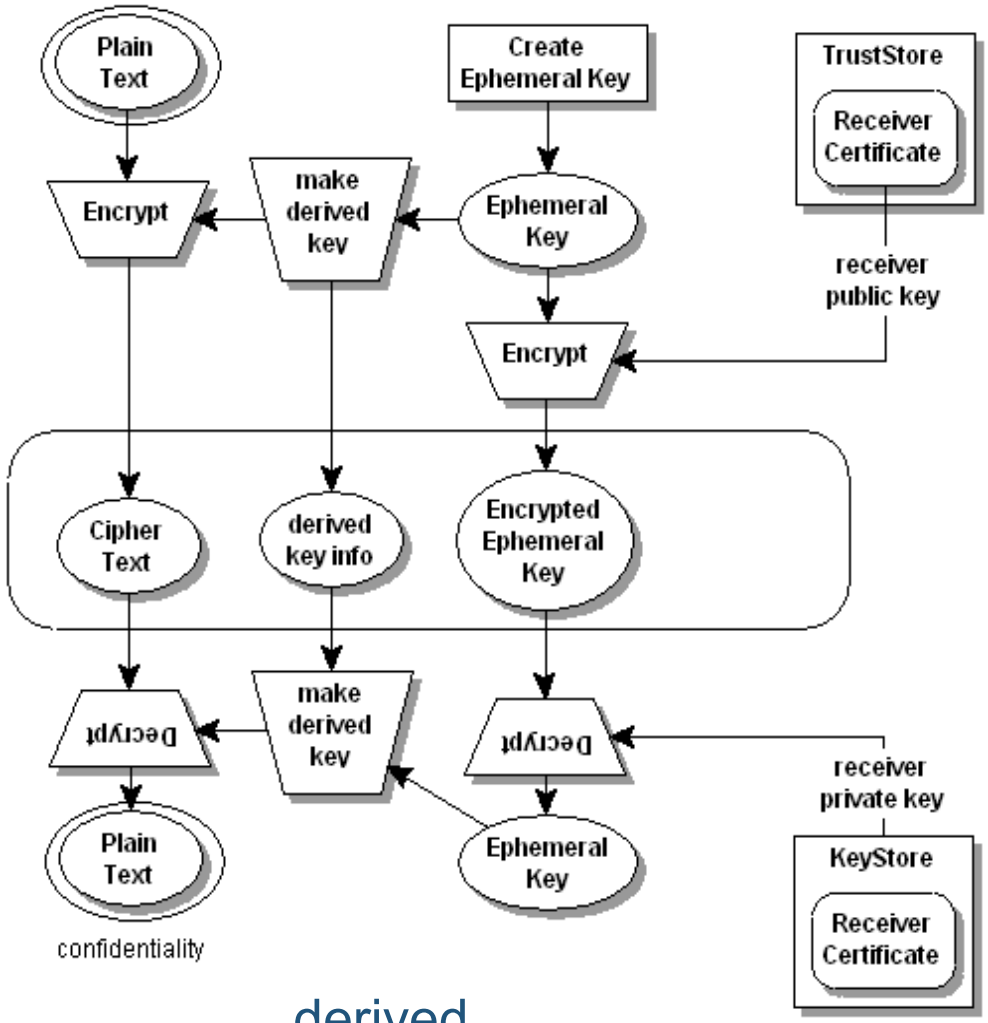
- Derived Keys
  - Key reused in this profile: therefore recommend using derived keys for increased security
- Secure Conversation
  - Better performance for multiple message exchange

Option: Derived Keys

Use case: increase security when reusing keys



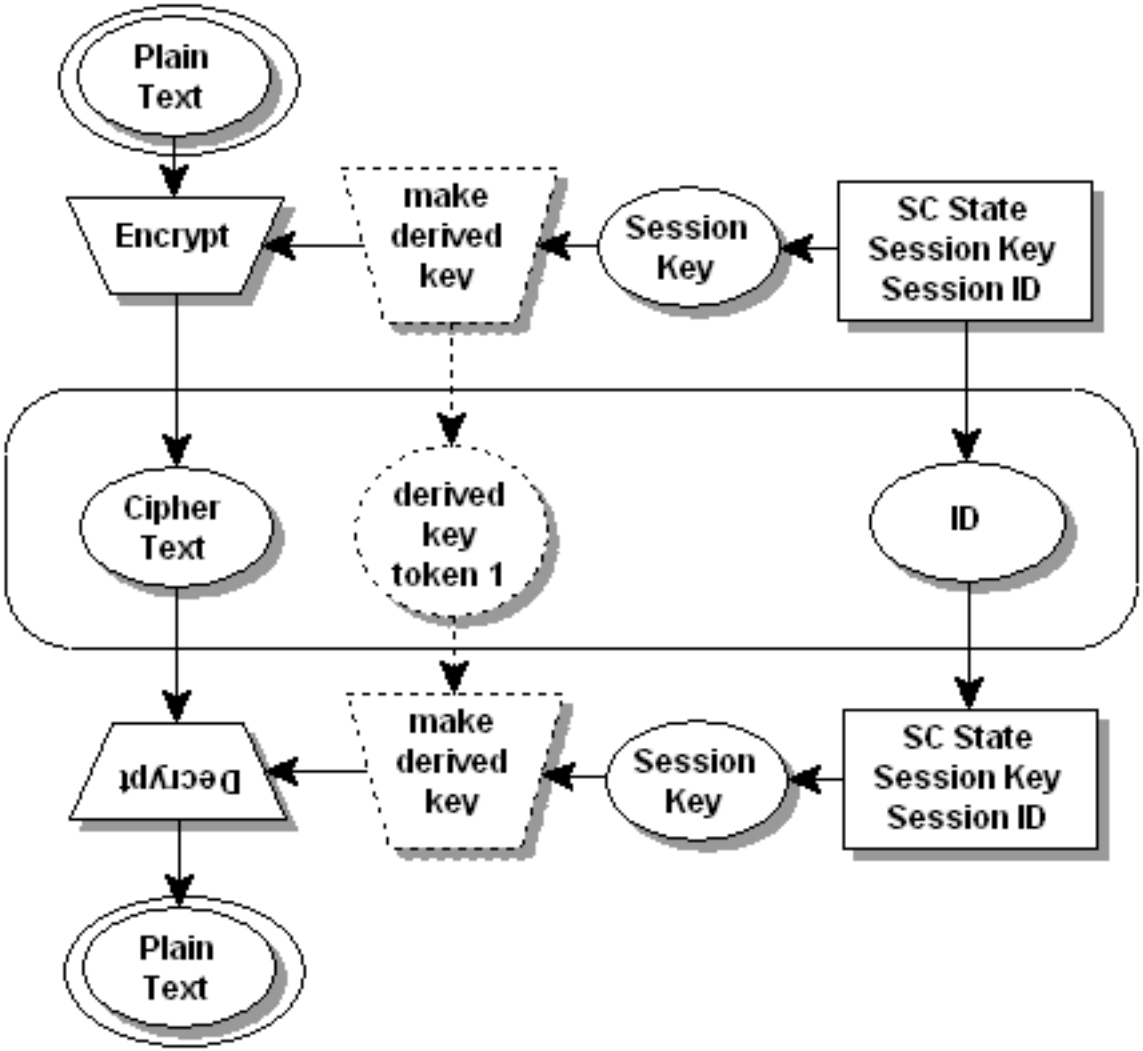
non derived



derived

Option: Secure Conversation

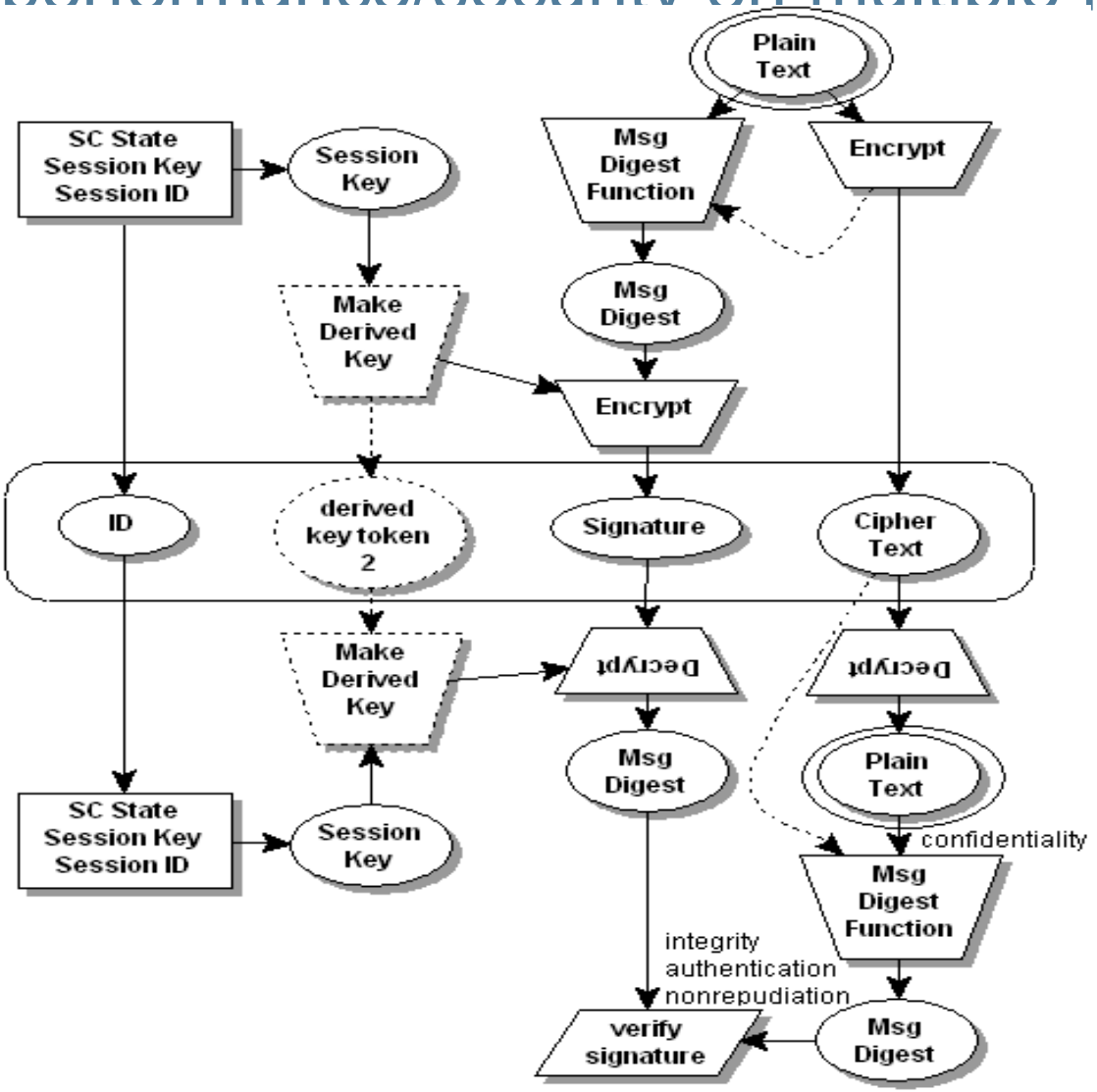
Use: increase performance/security on multiple msgs



confidentiality

Option: Secure Conversation

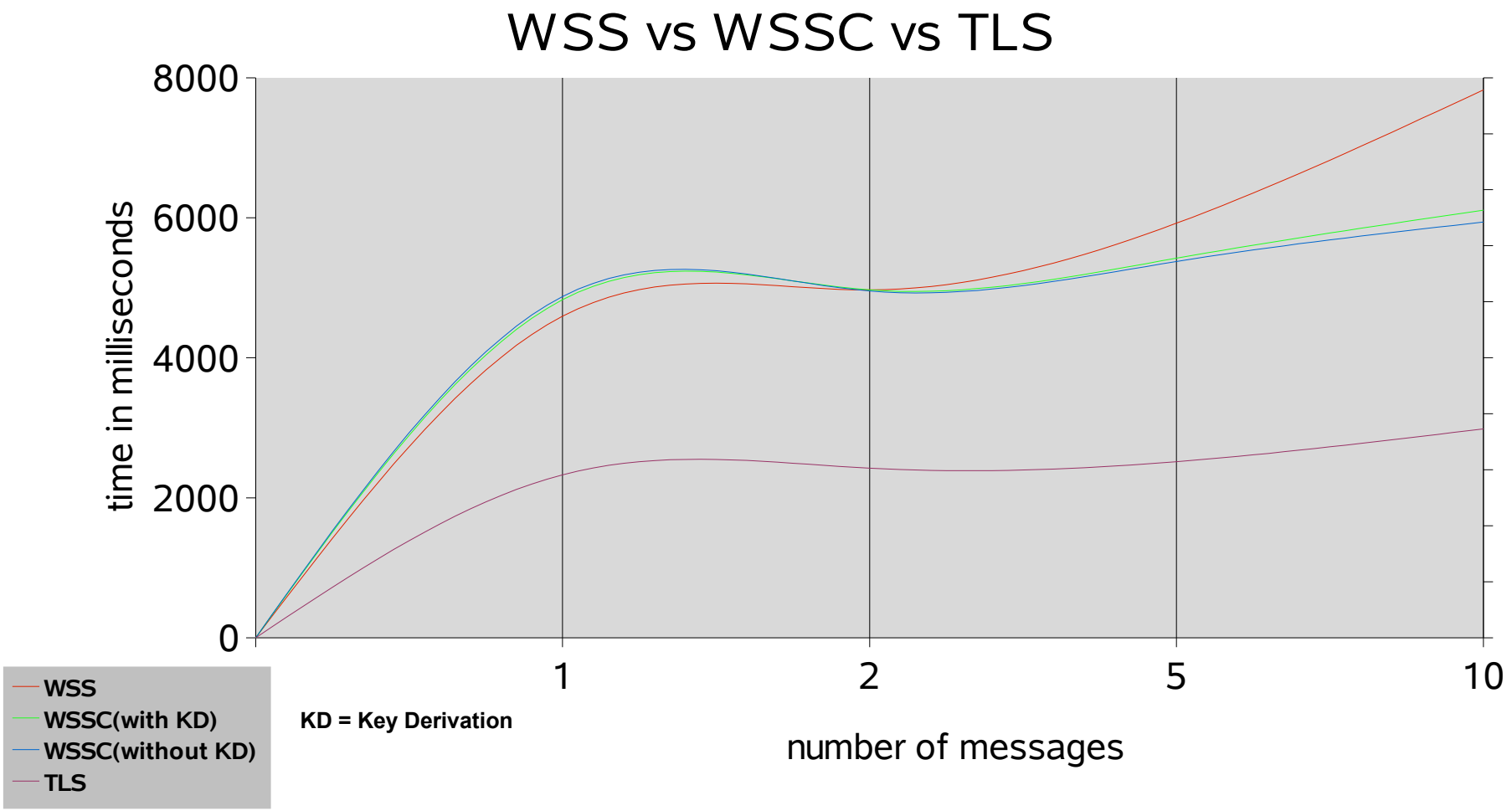
Use: increase performance/security on multiple msgs



Note: 2 derived keys (one for encryption, one for signing) for Increased security

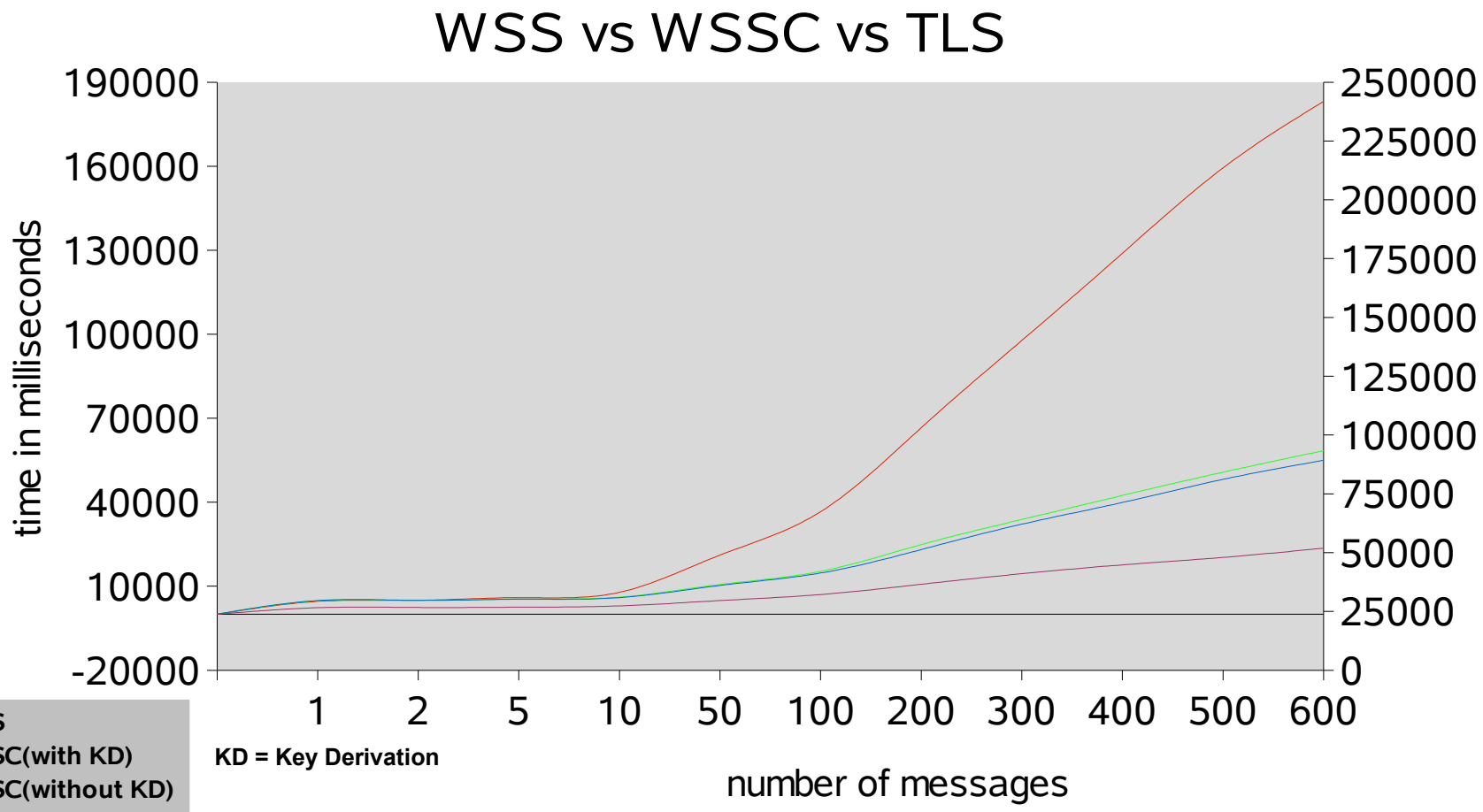
Option: Secure Conversation

Use: increase performance/security on multiple msgs



Option: Secure Conversation

Use: increase performance/security on multiple msgs

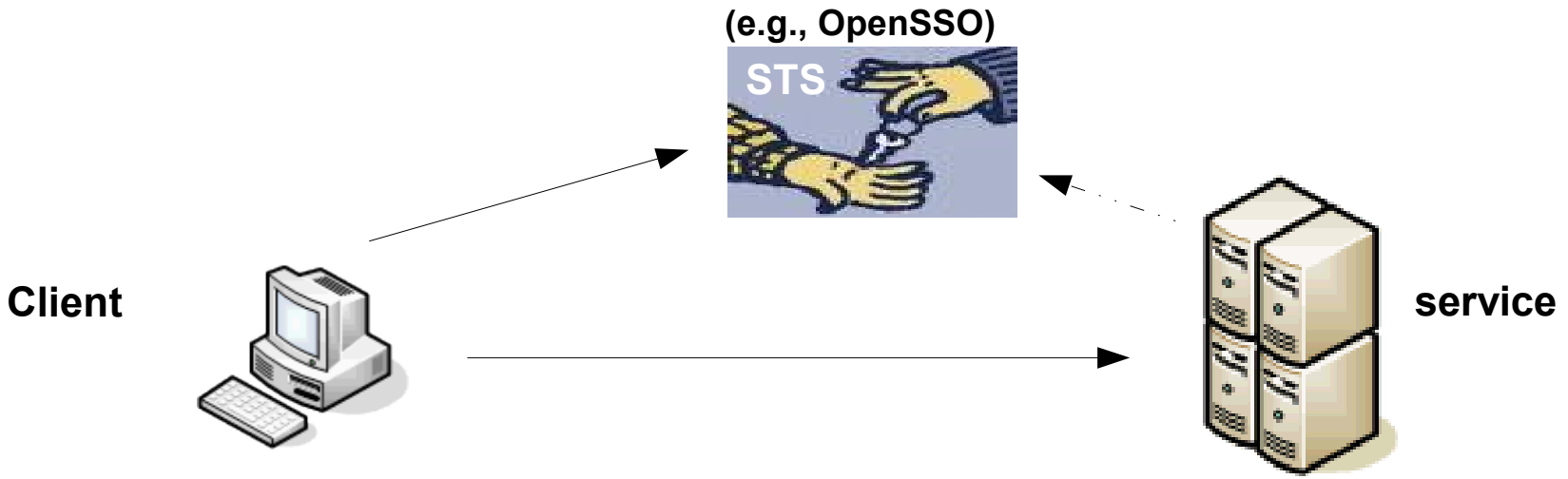


(note: baseline is plain text)

STS Issued Token

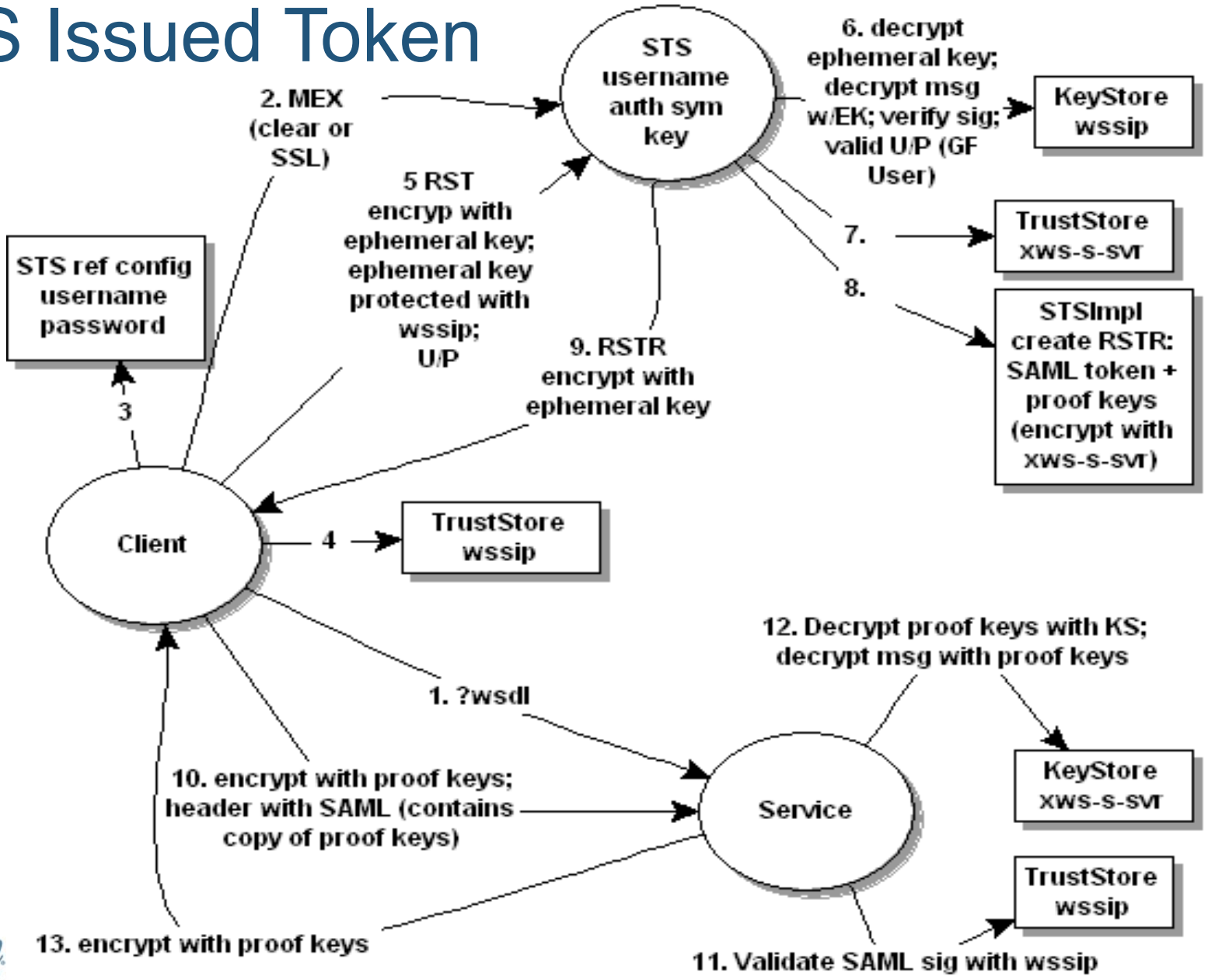
Use: Client gets auth/protection token from 3rd party

Firewall

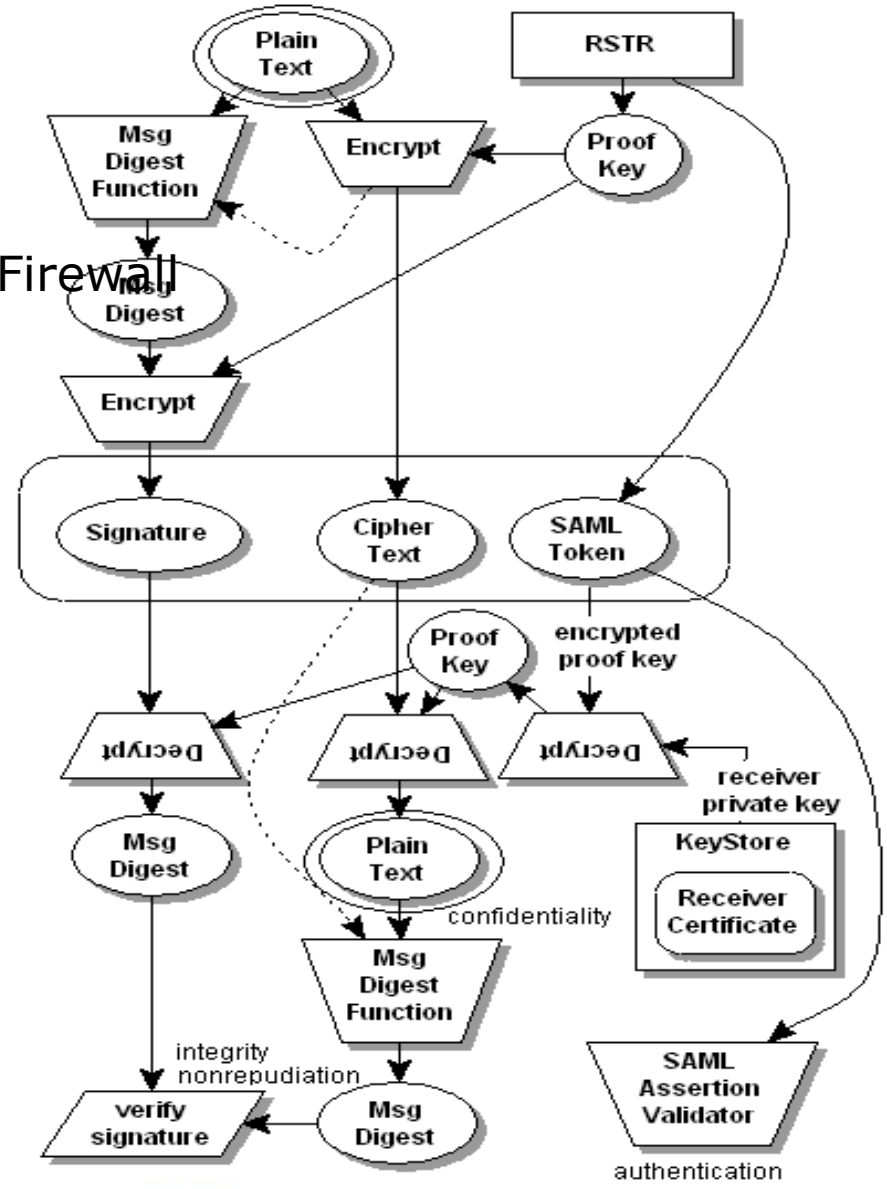


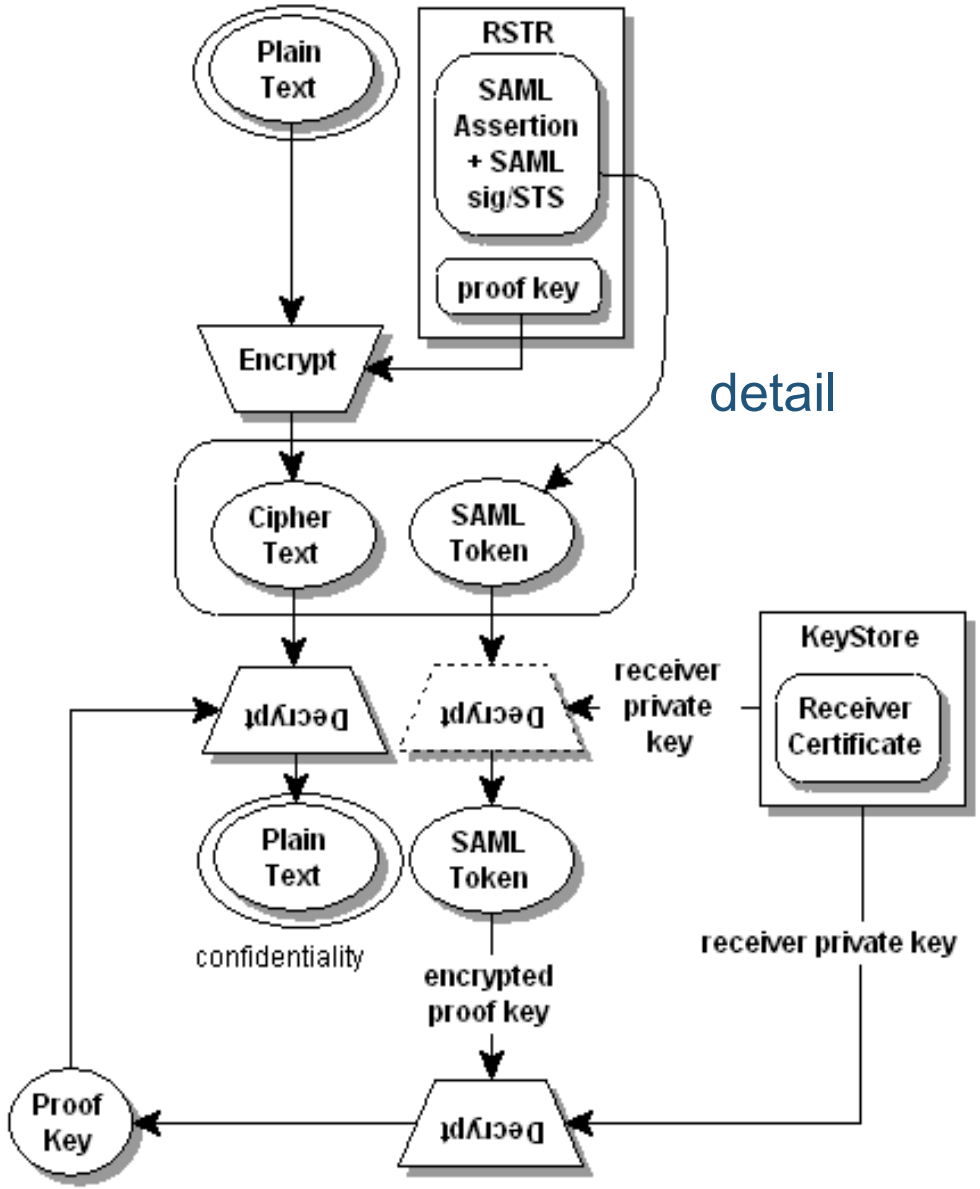
STS Issued Token

Firewall



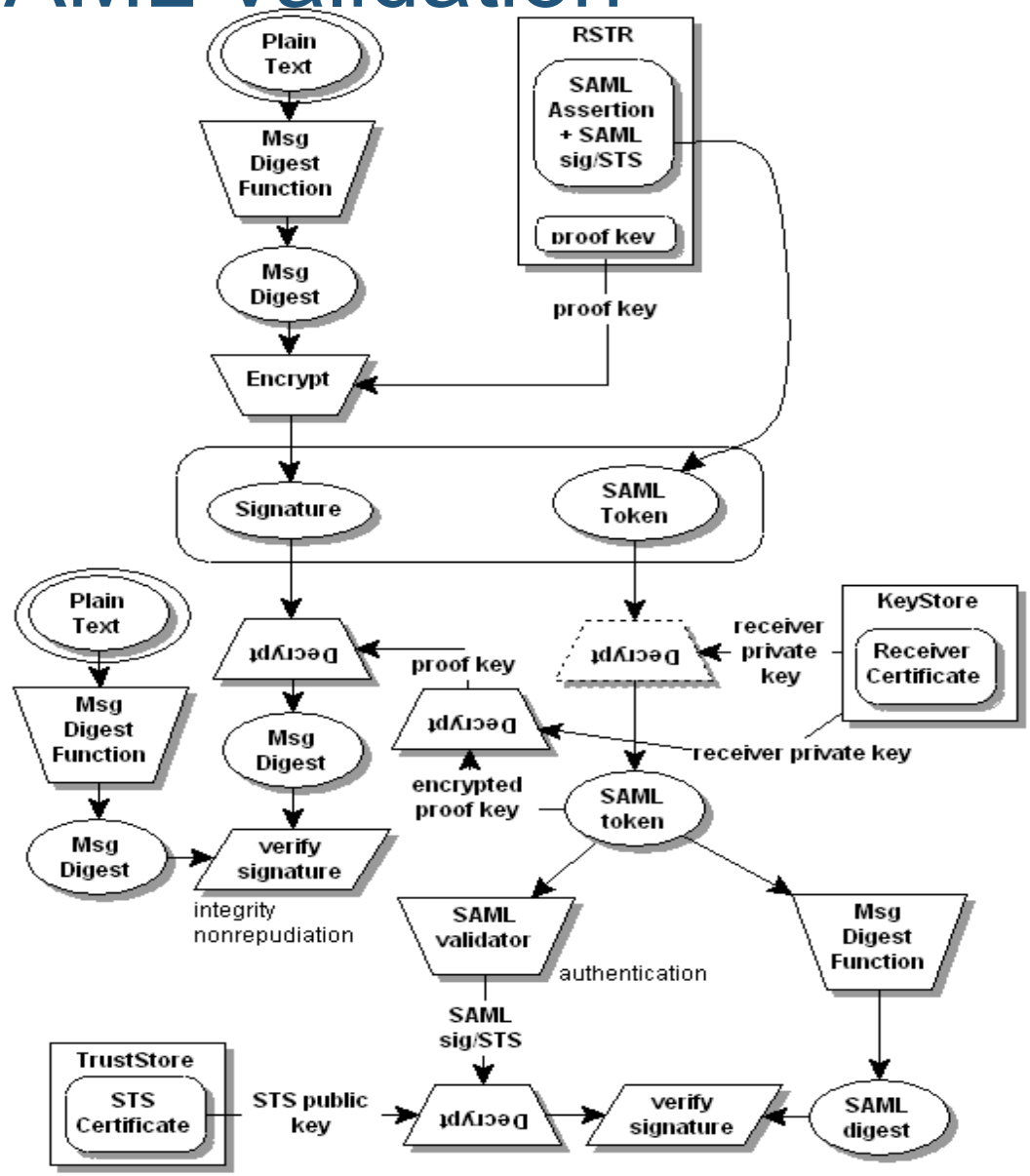
STS Issued Token : C → S





STS : SAML Validation

Firewall



STS Issued Token – Options

- Token Type: 1.0, 1.1*, 2.0
  - 1.1 : most widely adopted
- Key Type: Symmetric*, Public
  - Symmetric: best perf; but STS masquerade as client
  - Public: no masq., problem
- Size: 256*, 192, 128
  - Security viz performance
- Derived X509/Issued:
  - Security viz performance

Firewall

STS Issued Token

Summary

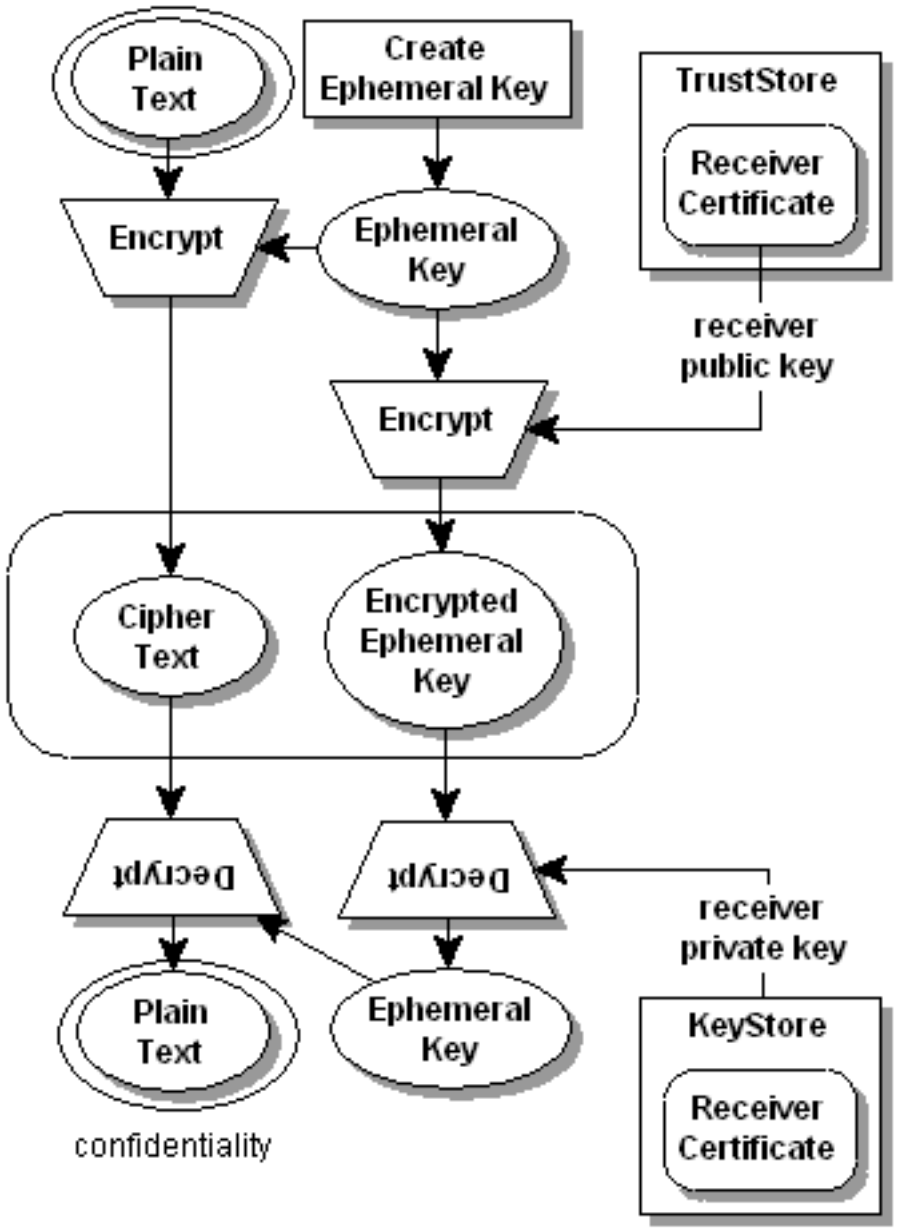
- No service certificate required at client
- Service directs client to 3rd party token issuers (STS)
- Issued token for authentication & protection
- Can use symmetric (better performance) or public (better user privacy) proof key

Firewall

Recap
Patterns, profiles, use cases

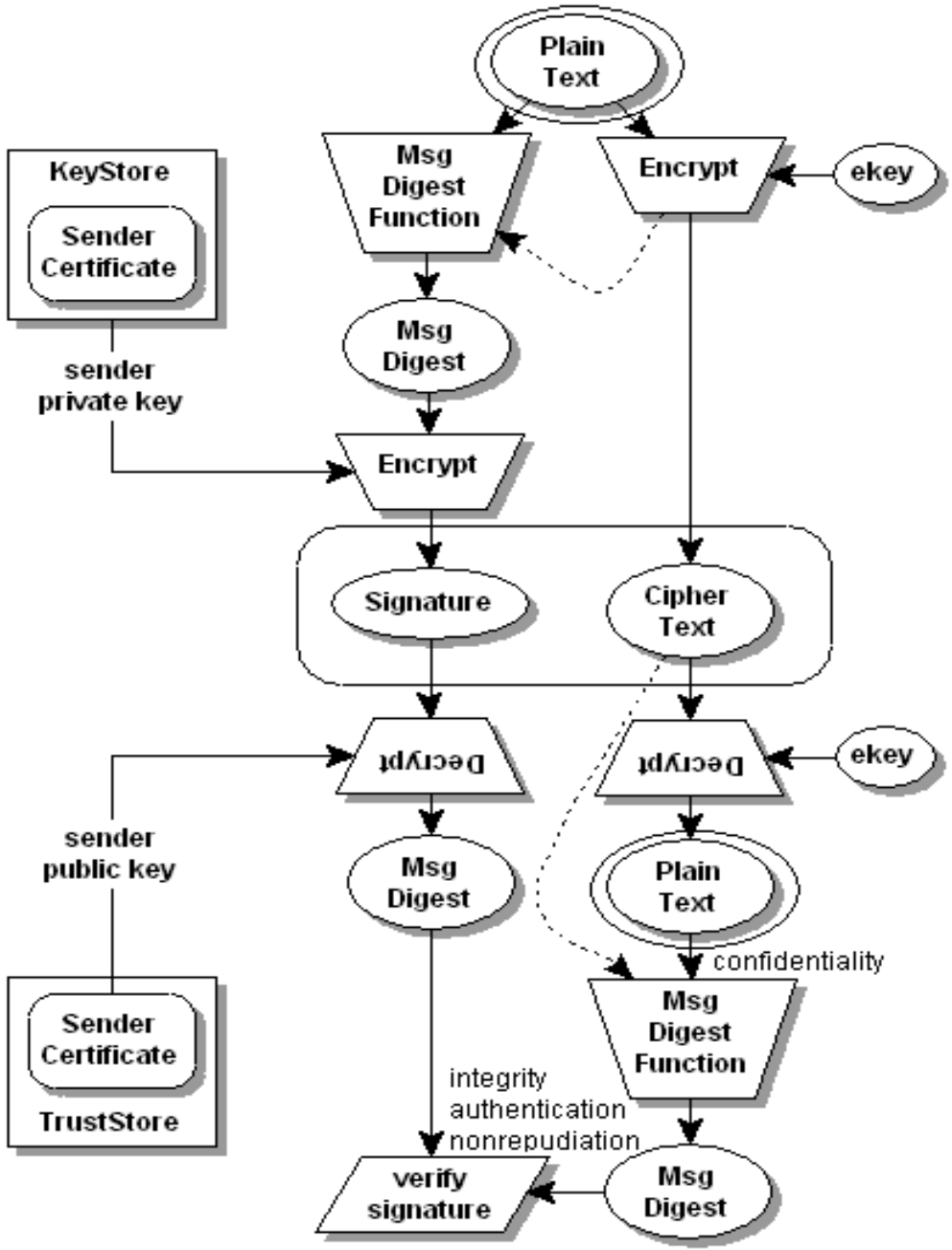
Firewall

Pattern
Confidentiality



Pattern

Integrity, Authentication, Nonrepudiation



Selected Metro Security Profiles

- Message Authentication over SSL
  - Use: client has username/password or X.509 relationship with service
  - Point-to-point
- Username Authentication With Symmetric Keys
  - Use: client has username/password or X.509 relationship with service
  - End-to-end

Selected Metro Security Profiles

- Mutual Certificates
  - Use: interaction with known partners
  - Certificates exchanged out-of-band in advance
  - End-to-end
- SAML Holder of Key
  - Use: role-based access control
  - Plug into existing SAML infrastructure; e2e
- STS Issued Token
  - Use: role-based access control
  - 3rd party handles authentication --- not service; e2e

Upcoming Security Features in Metro

- Convenience/control of use of issued tokens
  - Token cache & sharing among services
  - Token renewal & cancellation
  - Access different services with same credential
- Manage/exchange credentials at run-time
  - SSL/TLS Handshake for Web Services
  - SPNego
  - WS-Addressing Endpoint References & Identity

Upcoming Security Features in Metro

- Password derived keys
  - No certificates required
  - But not as strong as PKI or symmetric
- Support for @RolesAllowed
  - Servlet Web Services
  - (109/EJB Web Services already supported)

Metro and GlassFish

Open Source and Enterprise Ready

- **GlassFish v3 Preview Available now!**
  - Java EE 6 reference implementation
  - Modular OSGi architecture – easy to develop & deploy
  - Runs in-process and easy to extend
  - Support for Ruby-on-Rails, Groovy and Grails, Python and Django
- **GlassFish v2 – Production Ready**
  - Best price/performance open source App server with Clustering, High Availability, Load Balancing
  - Secure, Reliable, Transactional, .NET-interop Web svcs
  - Support for Ajax and Comet
- **GlassFish ESB**
  - SOA and Business Integration platform
- **GlassFish Communications App Server**
  - SIP servlet technology for converged services

glassfish.org

- **24x7 Enterprise and Mission Critical Support**
  - sun.com/appserver
- **Tools Integration**
  - NetBeans and Eclipse
- **Pavilion booth numbers: 550, 566, 567**
- **Meet Java EE spec leads and experts at Ancillary Event & Booth**

metro.dev.sun.net

Thank You

Harold Carr
weblogs.java.net/blog/haroldcarr

Jiandong Guo
blogs.sun.com/trustjdg

metro.dev.java.net

EXTRA SLIDES

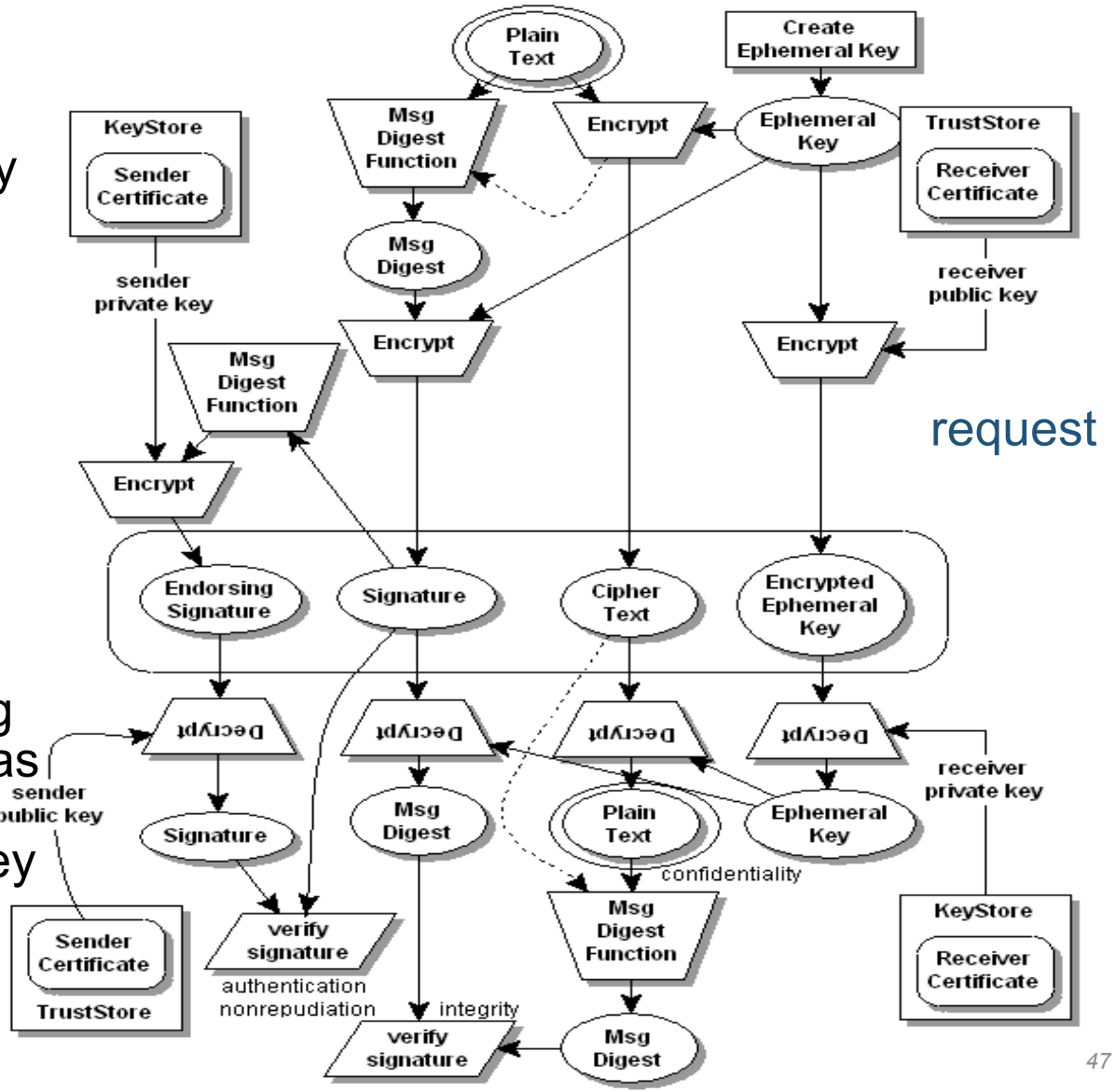
- Please do NOT remove

Endorsing Certificate optimization of mutual certificates

- Mutual: generates/encrypts another ephemeral key for response

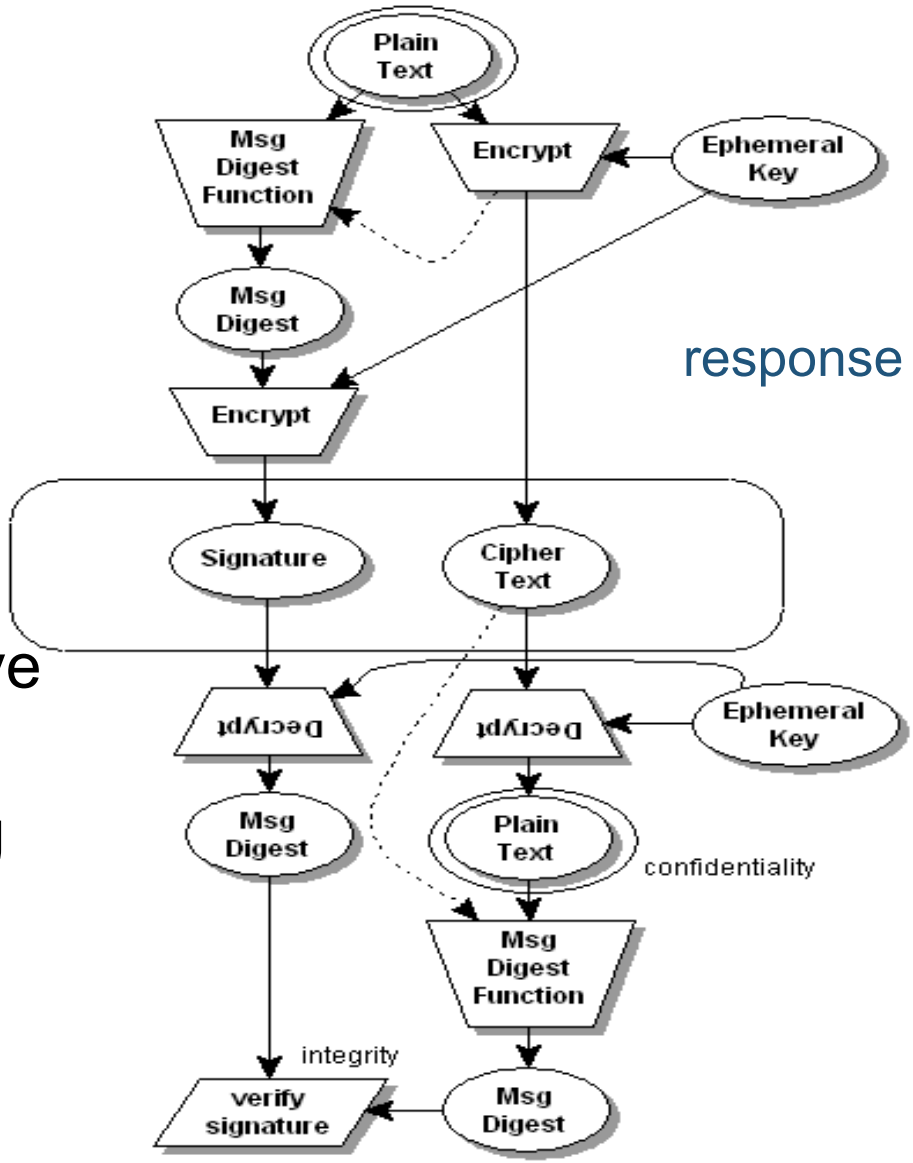
Firewall

- Endorsing reuses key
- (Use derived key since reusing key)
- Endorsing Signature to prove sender has private key
- Response: no endorsing sig because service has decrypted ephemeral key using its private key



Endorsing Certificate optimization of mutual certificates

- Mutual: generates/encrypts another ephemeral key for response
- Endorsing reuses key
- (Use derived key since reusing key)
- Endorsing Signature to prove sender has private key
- Response: no endorsing sig because service has decrypted ephemeral key using its private key



Slide Heading: 36pt
Subhead: 28pt

- All text is **Arial**
- Level One bullet point: 28pt
  - Level Two bullet: 26pt
    - Level Three: 22pt
      - Level Four and subsequent: 18pt
- Text block is aligned to the left