

Secure Communications

Martin Kobetic
Cincom Smalltalk Development

CSTUC'06 Frankfurt
December 2006

Contents

Web Server

Certificates and Keys

Secure Web Server

Client Authentication

Performance Considerations

Simple Web Server

```
configuration :=  
  AdaptorConfiguration webServer  
    addExecutor: WebHello new;  
    addExecutor: WebEcho new;  
    transport: (  
      TransportConfiguration http  
        marshaler: MarshalerConfiguration web ).  
server := configuration newAtPort: 4242.  
server start.
```

```
url := 'http://localhost:4242/hello' asURI.  
client := HttpClient new.  
[ client get: url ] ensure: [ client close ].
```

```
server stop.
```

Certificate Authority

```
caKeys := DSAKeyGenerator keySize: 1024.  
caName := Name new  
    O: 'TrustMe'; L: 'Palermo'; C: 'Italy';  
    yourself.  
ca := Certificate new  
    serialNumber: 1;  
    issuer: caName;  
    subject: caName;  
    notBefore: Date today;  
    notAfter: (Date today addDays: 7);  
    publicKey: caKeys publicKey;  
    forCertificateSigning;  
    yourself.  
ca signUsing: caKeys privateKey.
```

Server Certificate

sKeys := RSAKeyGenerator keySize: 1024.

sName := Name new

O: 'Oogle Inc'; CN: 'localhost';
yourself.

sCert := Certificate new

serialNumber: 1000;

issuer: caName;

subject: sName;

notBefore: Date today;

notAfter: (Date today addDays: 7);

publicKey: sKeys publicKey;

forKeyExchange;

yourself.

sCert signUsing: caKeys privateKey.

sCert verifyIssuedBy: ca.

Key Storage

```
file := 'server.pk8' asFilename  
  readWriteStream  
  binary.  
pwd := 'sesame' asByteArray.
```

```
[ sKeys privateKey writeOn: file password: pwd.  
] ensure: [ file close ].
```

```
" openssl pkcs8 -inform DER -in server.pk8  
| openssl rsa -text"
```

```
file reset.  
[ PKCS8 readKeyFrom: file password: pwd.  
] ensure: [ file close ].
```

Certificate Storage

```
file := 'server.cer' asFilename  
  readWriteStream  
  binary.
```

```
[ sCert writeOn: file ] ensure: [ file close ].
```

```
"openssl x509 -inform DER -in server.cer -text"
```

```
file reset.
```

```
[ Certificate readFrom: file ] ensure: [ file close ].
```

Switching to HTTPS

```
sCtx := SSLContext newWithSecureCipherSuites.  
sCtx certificate: sCert key: sKeys privateKey.
```

```
transport := TransportConfiguration https.  
transport serverContext: sCtx;  
  marshaler: configuration transport marshaler.  
configuration transport: transport.  
configuration soReuseAddr: true.
```

```
server := configuration newAtPort: 4433.  
server start
```

```
'https://localhost:4433/hello' asURI get.
```

Certificate Registry

```
registry := X509Registry new.  
registry addTrusted: ca.
```

```
chain := Array with: sCert.  
registry validateCertificateChain: chain.
```

```
registry addRevoked: ca.  
registry validateCertificateChain: chain.  
registry removeRevoked: ca.
```

```
X509Registry default addTrusted: ca.  
'https://localhost:4433/hello' asURI get.  
CertificateFileReader  
readFromFile: '/etc/pki/tls/certs/ca-bundle.crt'.
```

HTTPS Client

```
cCtx := SSLContext  
  newWithSecureCipherSuitesUsing: registry.
```

```
url := 'https://localhost:4433/hello' asURI.
```

```
client := HttpClient new.
```

```
client sslContext: cCtx.
```

```
[ client get: url ] ensure: [ client close ].
```

```
file := 'ca.cer' asFilename readWriteStream binary.
```

```
[ ca writeOn: file ] ensure: [ file close ].
```

Client Certificate

cKeys := RSAKeyGenerator keySize: 1024.

cName := Name new

O: 'Oogle Inc'; N: 'John Doe';

E: 'john@oogle.com';

yourself.

cCert := Certificate new

serialNumber: 2001;

issuer: caName;

subject: cName;

notBefore: Date today;

notAfter: (Date today addDays: 7);

publicKey: cKeys publicKey;

forSigning;

yourself.

cCert signUsing: caKeys privateKey.

Client Authentication

transport clientValidator:

```
[ :nm | '*@oogle.com' match: nm email ].
```

echo := 'https://localhost:4433/echo' asURI.

```
[ client get: echo ] ensure: [ client close ].
```

cCtx certificate: cCert key: cKeys privateKey.

X509Registry default addTrusted: ca.

Client Authentication - Browser

```
"openssl pkcs8 -inform DER -in client.pk8  
-out clientk.pem"  
file := 'clientk.pem' asFilename writeStream.  
[ cKeys privateKey asX509Key writePEMOn: file  
] ensure: [ file close ].
```

```
"openssl x509 -inform DER -in client.cer  
-out client.pem"  
file := 'client.pem' asFilename writeStream.  
[ cCert writePEMOn: file ] ensure: [ file close ].
```

```
"openssl pkcs12 -export  
-in client.pem -inkey clientk.pem  
-out client.p12 -name 'John Doe'"
```

Performance

```
transport clientValidator: nil.  
[ [ client get: echo ] ensure: [ client close ]  
] repeatUntil: 5.
```

```
"Persistent connection"  
[ [ client get: echo ] repeatUntil: 5.  
] ensure: [ client close ].
```

Session Resumption

```
session := SSLSession newIn: cCtx.  
[ [ socket := SocketAccessor  
    newTCPclientToHost: 'localhost' port: 4433.  
  binary := socket readAppendStream binary.  
  connection := cCtx  
    connectionFor: binary using: session.  
  connection  
    connectSubject: [ :nm | nm CN = 'localhost' ].  
] ensure: [ connection close. socket close ]  
] repeatUntil: 10
```

```
sCtx allowResumableSessions.  
sCtx disallowResumableSessions.
```

Session Resumption - Data

```
[ [ socket := SocketAccessor
  newTCPclientToHost: 'localhost' port: 4433.
  binary := socket readAppendStream binary.
  con := cCtx connectionFor: binary using: session.
  con connectSubject: [ :nm | nm CN = 'localhost' ].

  stream := (con withEncoding: #iso8859_1)
    readAppendStream lineEndCRLF.
  stream nextPutAll: 'GET /echo HTTP 1.1'; cr;
  nextPutAll: 'Host: localhost:4433'; cr; cr; flush.
  stream upToAll: '\\' withCRs; upToAll: '\\' withCRs
] ensure: [ stream close. socket close ]
] repeatUntil: 10.
```

Performance - Data Transfer

```
[ [ client get: echo ] repeatUntil: 5.  
] ensure: [ client close ]
```

```
sCtx initializeSuites: (  
  Array  
    with: SSLCipherSuite  
      SSL_RSA_WITH_RC4_128_MD5);  
certificate: sCert key: sKeys privateKey.
```

```
SSL_RSA_WITH_RC4_128_SHA  
SSL_RSA_WITH_RC4_128_MD5  
SSL_RSA_WITH_DES_CBC_SHA  
SSL_RSA_WITH_3DES_EDE_CBC_SHA
```

Performance - DH Suites

pars := DHPParameterGenerator m: 256 l: 1024.

sCtx initializeSuites: (
 Array with: SSLCipherSuite
 SSL_DHE_RSA_WITH_DES_CBC_SHA);
certificate: cCert key: cKeys privateKey;
dhParameters: (Array with: pars p with: pars g)

client validationBlock: [:n | n name = 'John Doe'].
[[client get: echo] ensure: [client close]
] repeatUntil: 5.

SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA
SSL_DHE_DSS_WITH_DES_CBC_SHA
SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA

Books

Rescorla: SSL and TLS

Housley, Polk: Planning for PKI

Ferguson, Schneier: Practical Cryptography

Menezes, van Oorschot, Vanstone:
Handbook of Applied Cryptography

Schneier, B: Applied Cryptography

[http://www.cincomsmalltalk.com/publicRepository/
bundle Presentations-MK](http://www.cincomsmalltalk.com/publicRepository/bundle Presentations-MK)
class CSTUC06SecureCommunications

Simple Web Service

```
url := 'http://localhost:4242/timeserver' asURI.  
server := RequestBroker  
  newSoapHttpAtPort: 4242  
  bindingNamed: 'TimeServerSoapBinding'.  
server export: Time oid: url tail.  
server start.
```

```
(url copy query: 'wsdl') get contents.
```

```
client := WsdlClient url: (url copy query: 'wsdl').  
client executeSelector: #now.
```

```
client close.  
server stop.
```

Secure Web Service

```
url := 'https://localhost:4433/timeserver' asURI.  
server := BrokerConfiguration standard  
  adaptor: (AdaptorConfiguration connectionOriented  
    transport: (TransportConfiguration https  
      serverContext: sCtx;  
        marshaler: (MarshalerConfiguration soap  
          bindingNamed: 'TimeServerSoapBinding')))).  
server := server newAtPort: url port.  
server export: Time oid: url tail.  
server start.  
  
server stop.
```