A Python Interface to SSLeay

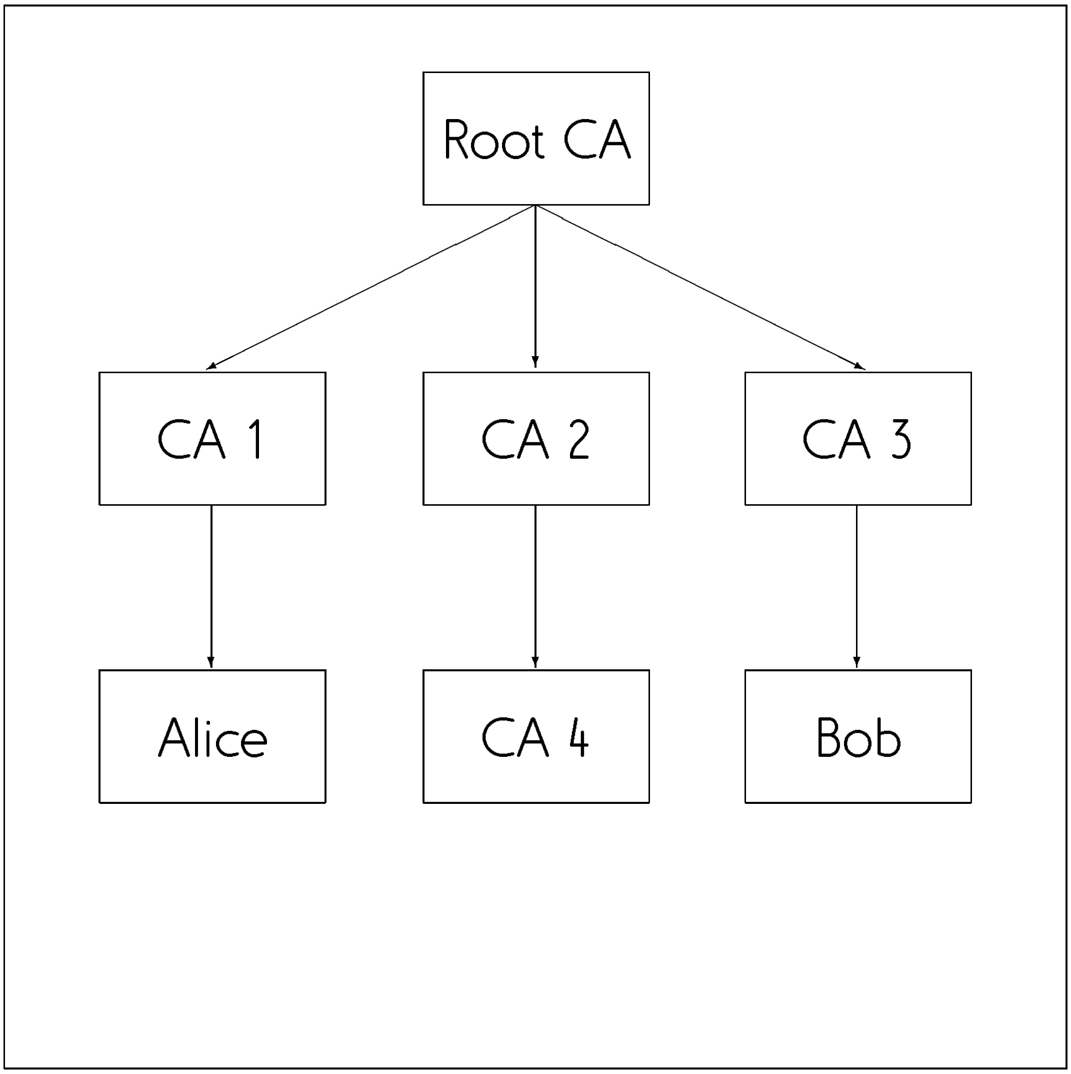
A.M. Kuchling

December 4, 1995

Secure Socket Layer (SSL)

- Was developed by Netscape Corp.
- Client/server negotiate algorithm + session key.
- Optionally, clients and servers can authenticate each other.
- Provides easy security; just run protocols unchanged over an encrypted connection.
- Insecure protocols are still insecure
- Uses X.509 framework, w/ added assumptions

X.509 framework



CAs certify both other CAs and users. Verifying a user may require searching for a common CA in the hierarchy.

4 steps to SSLeay-capable code:

1. Server only: generate an RSA key & keep it.

2. Call `SSLeay.fromfd()` to create an SSLeay object attached to the socket.

3. Server only : set a public key and certificate with appropriate functions.

4. Call the `connect` or `accept` method.

Simple client & server

Server:

```
PORT=4433 ; LENGTH=60
s=socket(AF_INET, SOCK_STREAM)
s.bind('', PORT) ; s.listen(1)
conn, addr = s.accept()
query=conn.recv(LENGTH)
conn.send(string.ljust(str(eval(query)),
          LENGTH))
conn.close()
s.close()
```

Client:

```
PORT=4433 ; LENGTH=60
s=socket(AF_INET, SOCK_STREAM)
s.connect('', PORT)
query="2+2"
s.send(string.ljust(query, LENGTH))
answer=s.recv(LENGTH)
print query, ':', answer
s.close()
```

End-to-end encryption

Client changes:

```
  s.connect('', PORT)
+ ssl=SSLeay.fromfd(s.fileno())
+ ssl.connect()
```

Server changes:

```
  conn, addr = s.accept()
+ ssl=SSLeay.fromfd(conn.fileno())
+ ssl.use_certificate_file("server.PEM",
+    SSLeay.X509_FILETYPE_PEM)
+ ssl.use_RSAkey_file("server.PEM",
+    SSLeay.X509_FILETYPE_PEM)
  ssl.accept()
```

A test run

```
Cipher alg. used: CBC-IDEA-MD5
Peer name:
 '/C=AU/SP=QLD/O=Mincom Pty. Ltd./OU=CS/
  CN=SSLeay demo server'
2+2 : 4
```

Server authentication

Added to client.py:

```
def vercall(ok, xs, xi, depth, err):
    if depth>0:
        return 1 # We won't verify too deeply
    if err==VERIFY_ERR_UNABLE_TO_GET_ISSUER:
        return 0 # Can't get the issuer
    return 1     # Otherwise, it's OK

ssl=SSLeay.fromfd(s.fileno())
ssl.set_verify(SSLeay.VERIFY_PEER)
SSLeay.set_verify_callback(vercall)
SSLeay.X509_add_cert_dir('./certs',
                  SSLeay.X509_FILETYPE_PEM)
```

A test run

Client-side view with X509_add_cert_dir:

```
Parameters of verify callback:
  Depth= 0
  Subject name:
    '/C=AU/SP=QLD/CN=SSLeay demo server'
  Issuer name :
    '/C=AU/SP=QLD/CN=SSLeay/rsa test CA'
Parameters of verify callback:
  Depth= 1
  Subject name: '/C=AU/SP=QLD/CN=SSLeay/rsa test CA'
  Issuer name : ''
```

Without X509_add_cert_dir:

```
Parameters of verify callback:
  Depth= 0
  Subject name:
    '/C=AU/SP=QLD/CN=SSLeay demo server'
  Issuer name : ''
Can't find issuer certificate
Traceback (innermost last):
  File "client.py", line 25, in ?
    ssl.connect()
SSLeay.error: SSL error:error:15018105
   lib(21) func(24) reason(X509 lib)
```

Summary

- SSL makes it easy to gain some security.
- Python makes it easy to use SSLeay.
- SSLeay is still in an alpha state.
- Interfaces to other SSL implementations could be written.