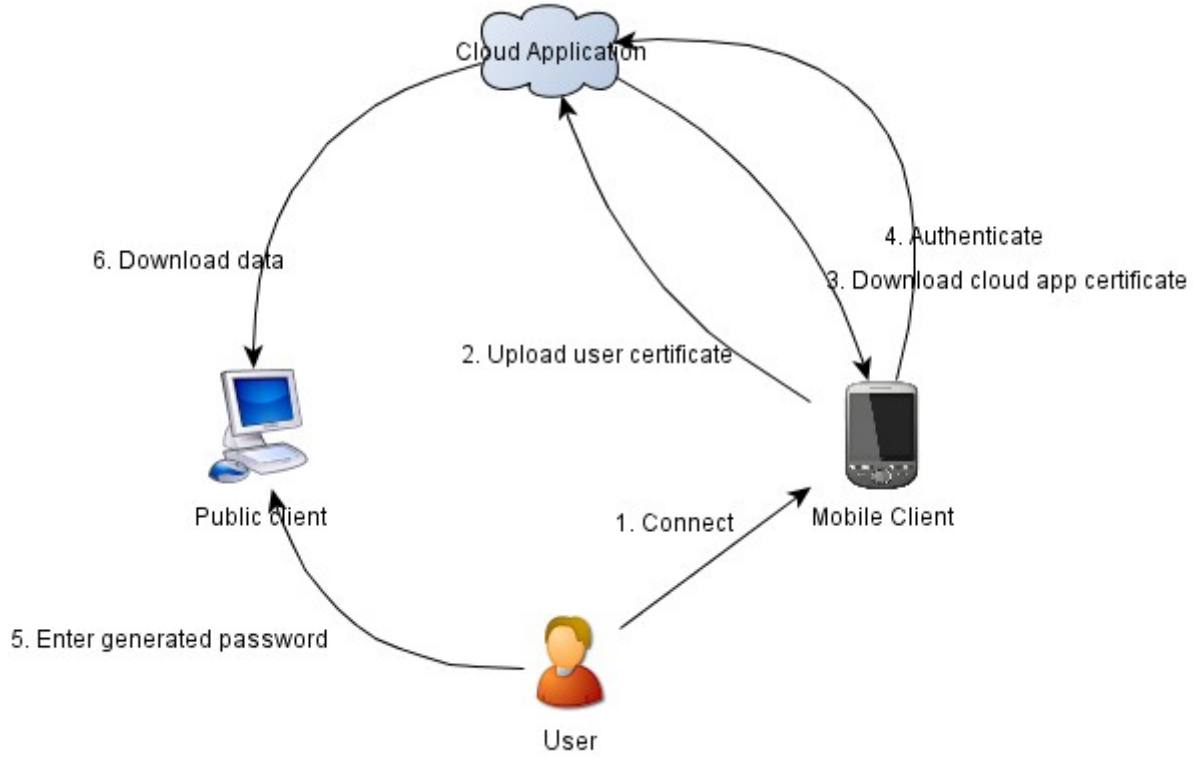
Smart Cloud Authentication

Functional specification

The goal of the project is the use of a smart phone for authentication of a user in a cloud application from an untrusted public machine.
The following systems are or will be developed:

1. **CloudApp** – a Google cloud based web application.
2. **MoblieApp** – an Android based authentication client. It's primary goal is to authenticate the public machine in the cloud.
2. **PublicApp** – a client on a public machine, accessing the CloudApp, authenticated previously by the MobileApp.

The diagram below presents the global view on the architecture.



Usage scenario

Prerequisites:

1. Cloud App and Mobile App have installed a common root certificate.

User has an accoun on Cloud App.

Offline the user generates a public/private key pair, sends to the CA a certificate signing request, receives a signed certificate put it on a SD card and inserts it in the phone. The user comes to a public internet machine and launches the Mobile App. On the first launch he enters the path to it's certificate and private key as well as the user id. After that he pushes a button to start authentication. The Mobile App generates a one time radom password, sends the signed and encrypted authentication request to the Cloud App and display the generated password on the screen, which

will be the secret shared between the three parties. The Cloud App authenticates the user request and store the session key derived from the password and user id. The user launches the Public App, which prompts it for user id and password. After the user enters these information, Public App also derives the session key. Subsequently the user can use Public App to perform encrypted and authenticated upload/download of files from the Cloud App.

Communication

1. Upload user certificate

Params: user id, user certificate

Result: empty/error

Sent in clear. No encryption, no integity protection. The Cloud App can verify certificate integrity by checking the it's signature. Substitute existing user certificate if any.

2. Download Cloud App certificate

Params: empty/error

Result: Cloud App certificate

Sent in clear. No encryption, no integity protection. The Mobile App can verify certificate integrity by checking the it's signature.

3. Authenticate

Params: user id, timestamp, nonce, generated password.

Result: empty/error

Signed with the user private key, password encypted with the Cloud App public key.

4. Download data

Params: user id, file name

Result: requested data/error

A hash is calculated on the message and both message and hash are encrypted to assure both confidentiality and integrity. The same for the response.

5. Upload data

Params: user id, file name

Result: empty/error

A hash is calculated on the message and both message and hash are encrypted to assure both confidentiality and integrity. The same for the response.

Detailed specification

Communication

1. Upload root certificate

Method: POST

URL: /addrootcert

Request: cert=<root CA certificate>

Authentication: user must be authenticated in the Google services, present in the datastore and have

ADMIN type

Response: error code

2. Add new user

Method: GET

URL: /adduser?user=<user name>&type=<NORMAL|ADMIN>

Request: empty

Authentication: user must be authenticated in the Google services, present in the datastore and have ADMIN type

3. Upload user certificate

    Method: POST

    URL: /addusercert?user=<user name>

    Request: cert=<user certificate in pem format>

    Authentication: none

    Response: error code

4. Download Cloud App certificate

Method: GET

URL: /getservercert

Request: empty

Authentication: none

Response: <Cloud App certificate in pem format>

5. Authenticate

Method: POST

URL: /auth

Request: user=<user name>

    timestamp=<expiration timestamp in milliseconds>

    nonce=<random number 0-some big number>

    secret=<Enc[Cloud App public key, password]>

    signature=<Sign[user private key, timestamp||nonce||secret]>

Authentication: using X.509 certificate

Encryption type: RSA

Signature type: SHA256withRSA

Response: error code

6. Download data

Method: GET

URL: /getdata?user=<user name>&dataid=<data entity id>

Request: empty

Authentication: encrypted hash

Encryption type: 3DES. Session key = SHA256[user_id||password][0:24]

Signature type: SHA256

Response: <data entity>

7. Upload data

Method: PUT

URL: /putdata?user=<user name>&dataid=<data entity id>

Request: <data entity>

Authentication: encrypted hash

Encryption type: 3DES. Session key = SHA256[user_id||password][0:24]

Signature type: SHA256

Response: error code

DB Schema

UserEntity

String *name*

Enum {NORMAL, ADMIN} *type*

CertificateEntity

String *alias*

byte[] *certificate*

SessionEntity

String *alias*

Long *session_start_time*

Long *authentication_exiration_time*

Int *nonce*

String *password*

byte[] *session_key*

Literature

“ Cryptography and network security principles and practices”. Esp. 11.2, 13.2 and 14.2.