Smart Cloud Authentication

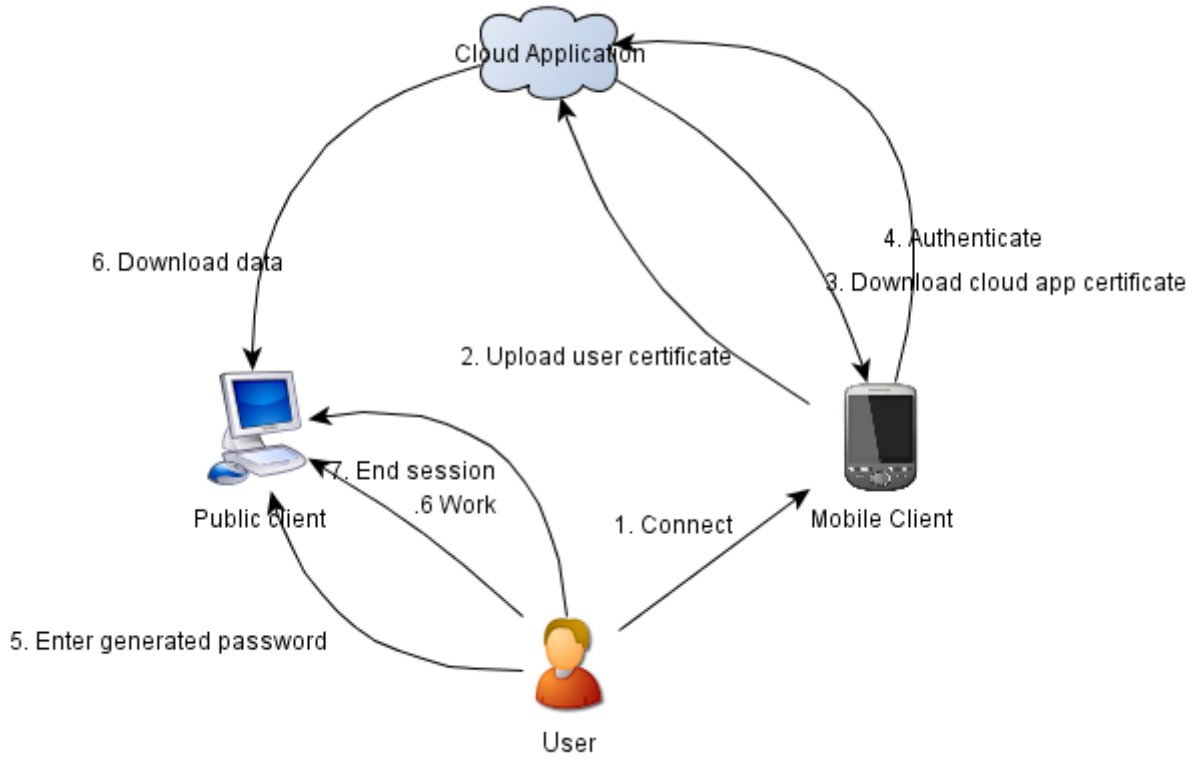
Functional specification

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. Its primary goal is to authenticate the public machine in the cloud.
3. **PublicApp** – a client on a public machine, accessing the Cloud App, authenticated previously by the Mobile App.

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.
2. User has an account on Cloud App.

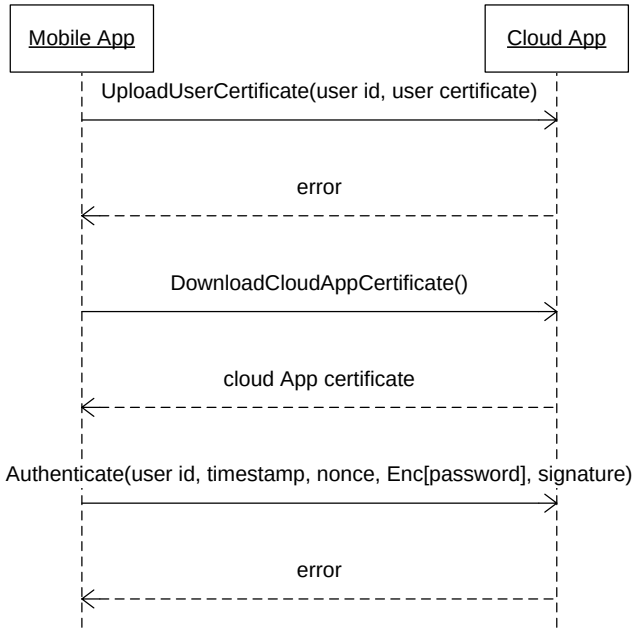
Offline the user generates a public/private key pair, sends to the CA a certificate signing request, receives a signed certificate put it on a SD card and inserts it in the phone. The user comes to a public

internet machine and launches the Mobile App. On the first launch he enters the path to its certificate and private key as well as the user id. After that he pushes a button to start authentication. The Mobile App generates a onetime random 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 this 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. After the session is over, the user sends the session termination request to the Cloud App through the Public App to purge the session from the Cloud App.

Communication

Mobile App – Cloud App

The diagram below illustrates the sample communication flow between Mobile App and Cloud App.



Below is the detailed list of requests types.

- Upload user certificate – upload the user certificate to the Cloud App certificate repository.

  Params: user id, user certificate.

  Result: empty/error

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

- Download Cloud App certificate – download the Cloud App certificate to the mobile phone and store it in the certificate repository of Mobile App.

  Params: empty/error

Result: Cloud App certificate

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

- Authenticate – authenticate the Mobile App on the Cloud App and share the secret, needed for securing the communication between the Public App and Cloud App.

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

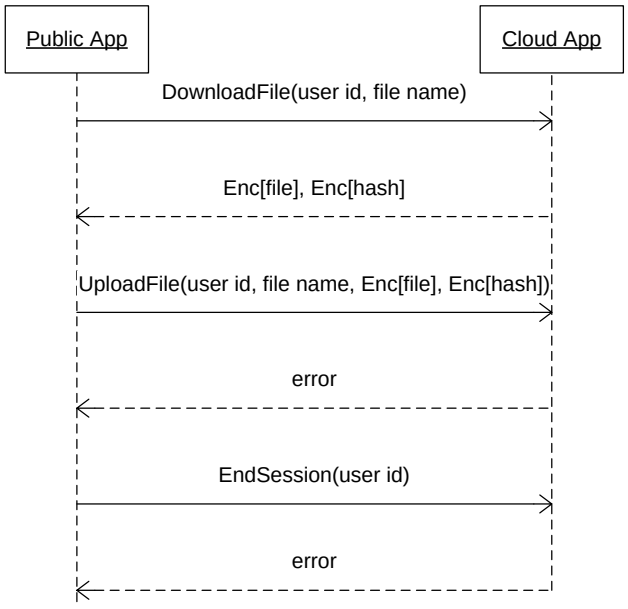
  Result: empty/error

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

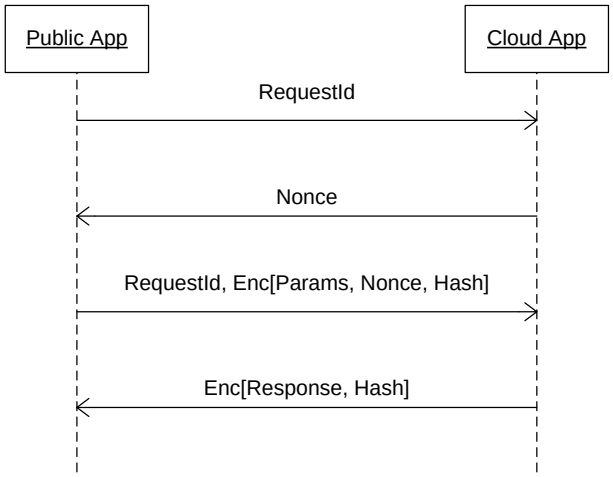
Public App – Cloud App

The diagram below illustrates the sample communication flow between the Public App and Cloud App.

N.B.: For simplicity, it doesn't challenge-response sequences, which are described below.



The diagram bellow illustrates the challenge-response mechanism used for each Public App-Cloud App request in order to protect against the replay attack.



All the communication between these entities is encrypted, using the session key and integrity protected using an encrypted hash

Below is the detailed list of requests types.

- Download file – download a file from the Cloud App.

  Params: user id, file name

  Result: file data/error

- Upload file – upload a file to the Cloud App.

  Params: user id, file name, file data

  Result: empty/error

- End session – delete the session data from both the Public App and Cloud App. After this request a new session must be opened to communicate with Cloud App from Public App.

  Params: user id

  Result: empty/error

Detailed specification

Communication

- Upload root certificate

  Method: POST
  URL: /addrootcert
  Request: cert=<root CA certificate>
  Authentication: user must be password-authenticated in the Google services, present in the data store and have ADMIN type
  Response: error code

- 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 data store and have ADMIN type

- Upload user certificate

  Method: POST
  URL: /addusercert?user=<user name>
  Request: cert=<user certificate in pem format>
  Authentication: none
  Response: error code

- Download Cloud App certificate

  Method: GET
  URL: /getservercert
  Request: empty
  Authentication: none
  Response: <Cloud App certificate in pem format>

- 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]>&
  iv=<initialization vector>
  Authentication: using X.509 certificate
  Encryption type: RSA/CTR
  Signature type: SHA256withRSA
  Response: error code

- Download data

  Method: GET
  URL: /getfile?user=<user name>&dataid=<file id>
  &iv=<initialization vector>
  [&challenge=<encrypted challenge>]
  Request: empty
  Authentication: challenge-response + encrypted hash
  Encryption type: 3DES/CTR. Session key = SHA256[user_id||password][0:24].
  Hash type: SHA256
  Response: file=<encrypted file data>&hash=<encrypted hash>

- Upload file

  Method: PUT
  URL: /putfile?user=<user name>
  &fileid=<file id>

&iv=<initialization vector>
[&challenge=<encrypted challenge>]
Request: file=<encrypted file data>&hash=<encrypted hash>
Authentication: challenge-response + encrypted hash
Encryption type: 3DES/CTR. Session key = SHA256[user_id||password][0:24].
Hash type: SHA256
Response: error code

- End session

Method: POST
URL: /endsession
Request: user=<user name>
&iv=<initialization vector>
[&challenge=<encrypted challenge>]
Authentication: challenge-response + encrypted hash
Encryption type: 3DES/CTR. Session key = SHA256[user_id||password][0:24].
Hash 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_expiration_time*
Int *nonce*
String *password*
byte[] *session_key*
String *challenge*

Literature

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