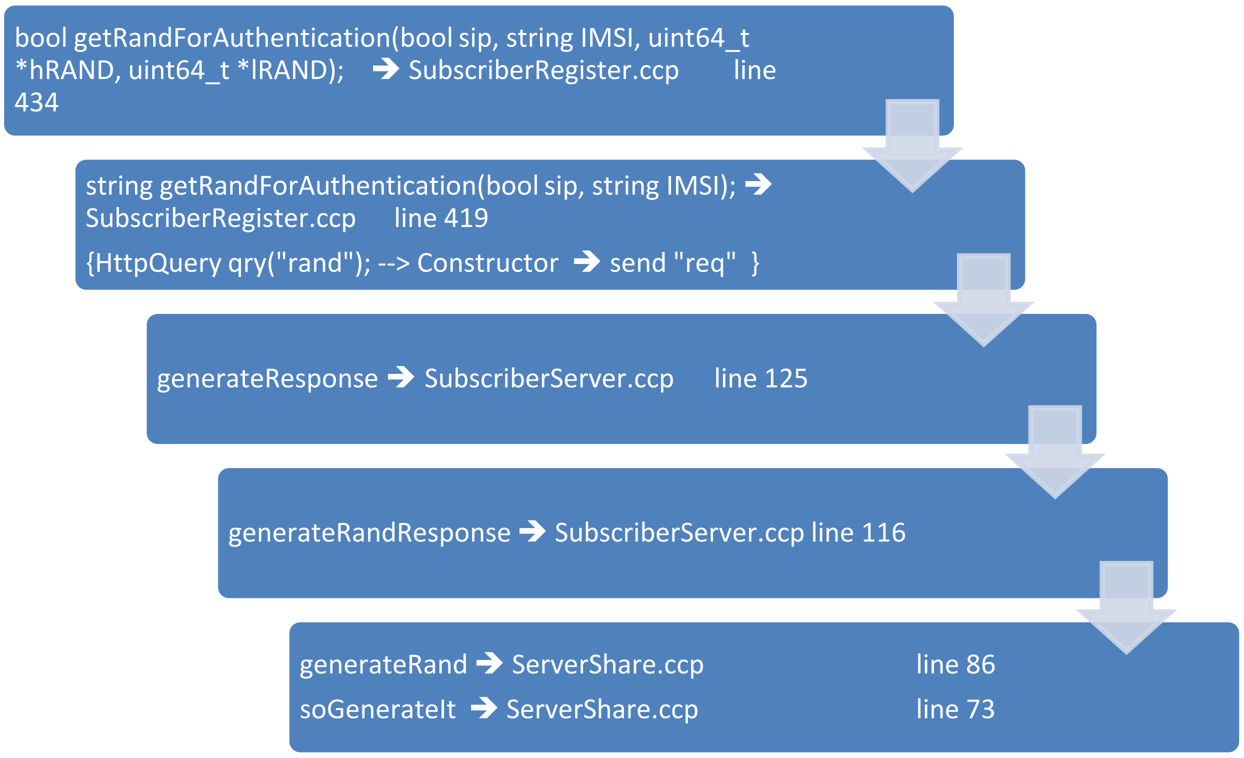
SubscriberRegister.ccp

- Contain 3 tables (RRLP [Radio Recourse Location Protocol] Table, DD [Dial Data] Table , SB [SIP_Buddies] Table)

- Contain 2 classes
  - SubscriberRegistry
  - HttpQuery ➔ uses to setup an http query, run it, and get the results.
    - Constructor ➔ send "req" [The type of http query (sql, rand, auth)]
    - Functions:
      - Send ➔ Specify a parameter to send in the http query.
      - Log ➔ Log the query.
      - http ➔ This runs the http query.
      - Receive ➔ Get result from the http query.

- Functions in class SubscriberRegistry :
  - ✓ getIMSI ➔ Resolve an ISDN (any numeric address) to an IMSI.
  - ✓ getCLIDLocal
  - ✓ getCLIDGlobal
  - ✓ getRegistrationIP ➔ Given an IMSI, return the IP address of the most recent registration.
  - ✓ addUser ➔ Add a new user to the SubscriberRegistry.
  - ✓ setRegTime ➔ Set the current time as the time of the most recent registration for an IMSI.
  - ✓ getRandForAuthentication



bool getRandForAuthentication(bool sip, string IMSI, uint64_t *hRAND, uint64_t *lRAND); ➔ SubscriberRegister.ccp line 434

string getRandForAuthentication(bool sip, string IMSI); ➔ SubscriberRegister.ccp line 419

{HttpQuery qry("rand"); --> Constructor ➔ send "req" }

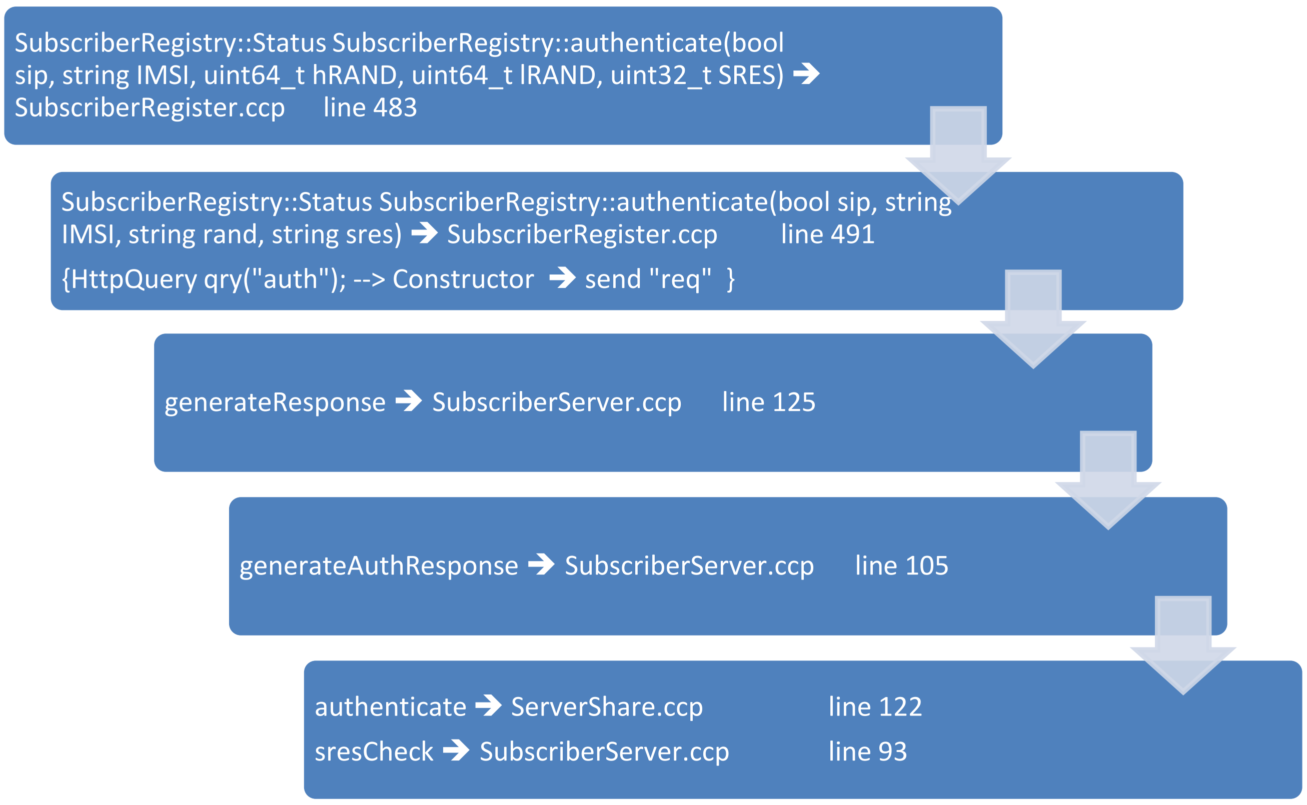
generateResponse ➔ SubscriberServer.ccp line 125

generateRandResponse ➔ SubscriberServer.ccp line 116

generateRand ➔ ServerShare.ccp line 86

soGenerateIt ➔ ServerShare.ccp line 73

- ✓ stringToUint (string strRAND, uint64_t *hRAND, uint64_t *lRAND)
- ✓ uintToString (uint64_t h, uint64_t l)
- ✓ uintToString (uint32_t x)
- ✓ Authenticate ➔ return ok or fail



SubscriberRegistry::Status SubscriberRegistry::authenticate(bool sip, string IMSI, uint64_t hRAND, uint64_t lRAND, uint32_t SRES) ➔ SubscriberRegister.ccp line 483

SubscriberRegistry::Status SubscriberRegistry::authenticate(bool sip, string IMSI, string rand, string sres) ➔ SubscriberRegister.ccp line 491

{HttpQuery qry("auth"); --> Constructor ➔ send "req" }

generateResponse ➔ SubscriberServer.ccp line 125

generateAuthResponse ➔ SubscriberServer.ccp line 105

authenticate ➔ ServerShare.ccp line 122

sresCheck ➔ SubscriberServer.ccp line 93

- ✓ sqlLocal ➔ Run sql statments locally.
- ✓ sqlHttp ➔ Run sql statments over http.
- ✓ sqlQuery ➔ Run an sql query (select unknownColumn from table where knownColumn = knownValue).
- ✓ sqlUpdate ➔ Run an sql update.

Subscriberserver.ccp

- int main()

```
{
        cout << "Content-Type: text\n\n";
        srand ( time(NULL) + (int)getpid() );
        decodeQuery(gArgs);
        logQuery();
        generateResponse();
        logResponse();
        respond();
}
```

- Function
  - ✓ getArg ➔ retrieve a query field from args
  - ✓ generateSqlResponse ➔ run an sql statement through sqlite3 to get a response
  - ✓ sresCheck
  - ✓ generateAuthResponse
  - ✓ generateRandResponse
  - ✓ generateResponse ➔ put the http response into the global array @response
  - ✓ logQuery ➔ write the query to the log file
  - ✓ logResponse ➔ write the http response from the global array @response to the log file
  - ✓ respond ➔ print the http response to stdout

Servershare.ccp
Functions

- ✓ imsiGet ➔ Given an IMSI, return value you want it if it in sip_buddies table.
- ✓ imsiSet ➔ Given an IMSI, name of pram. and its value to update sip_buddies table.
- ✓ soGenerateIt
- ✓ generateRand
- ✓ strEqual ➔ comparison function
- ✓ authenticate
- ✓ decodeQuery
- ✓ join
- ✓ split

MobiltyMangment.ccp

- Code authentication from line 419(it is began // Try to register the IMSI.)

- In this line 426 [success = engine.Register(SIPEngine::SIPRegister, &RAND);] will get rand, How??

- In this line 469 [success = engine.Register(SIPEngine::SIPRegister, &RAND, IMSI, SRESstr.c_str());] will verify sres , How??

engine is object from class *SIPEngine* ➔ SIPEngine engine

this class contain function *register*

```
Register(Method wMethod=SIPRegister, string *RAND = NULL,
const char *IMSI = NULL, const char *SRES = NULL);
```

Function register Call function *sip_register* from SIPMessage

In SIPMessage.ccp → function sip_register in its end will found

```
if (SRES) {

              // add authentication
              osip_authorization_t *auth;
              osip_authorization_init(&auth);
              osip_authorization_set_auth_type(auth, osip_strdup("Digest"));
              osip_authorization_set_nonce(auth, osip_strdup(RAND->c_str()));
              osip_authorization_set_uri(auth, osip_strdup(IMSI));
              osip_authorization_set_response(auth, osip_strdup(SRES));
              osip_list_add(&request->authorizations, auth, -1);
      }
```