Liquid Unions Protocol

Data structure: 4 Associative arrays - 2, on the left, of 32bit numpy ordered set and 2, on the right, of python lists:

```
     [0-or-1-per-role-Pgid]       <= hash(Nick)          => [hash(symk),pubk,iiaom,link]
     [[W,I,O,R,A],[],[],[],[],[]] <= Pgid or hash(Pgid) => hash(Nick).
```

This data is decrypted on starting and encrypted on ending this app and the server-db is distributed encrypted on request of correspondences peers.
The app maintains a Role dependency by its corespondents for access and for feeds of notifications, while providing a P2p + client/server transactions of: S(+R)when sender adds reviver, S(-R) when sender remove reviver and otr+iiaom chat. This protocol uses the A (add) and R(remove) additionally to W(with)/O(outward)/I(inward) for making the transactions transparent between correspondences while involving both the peers and the server.

Protocol: [-I+W|+O] S(+R)--->server<---R [+A] or [-W+O|-I] S(-R)--->server<---R [+R]
                          \-----|-------/   \-----|-------/

```
Pgid = R<<24|id.
|R| Role           |  corespondents  |
|0|individual=i:           is,ig,il |
|1|group's admin:          gi,gc,gp |
|2|channel's admin=c:      cg,cp,cl |
|3|logistic admin= l:      lu,li,lc |
|4|project's admin=p:      pc,pg,pf |
|5|finance  admin=f:       fu,fp    |
|6|liquid-union admin=u:   uf,ul,us |
|7|population set admin=s: sp,si,su |
1--------------------------->m
^----------u
^---------/-\
^---------f---l
^--------/-----\
^--s<->p<-->c<->g<->i
m-----------------------------
```

H# of index of last element of each the next 5 arrays:
W# ordered array of pages linked to and from this page, where only owner can modify;
I# ordered array of pages linked from this page, where only owner can modify;
O# ordered array of pages linked to this page, where only owner can modify;
R# the number of is visible to any, where only owner can remove and any owner of page existing in [W][O][I]can add;
A# the number of is visible only to owner, where only owner can remove and if want would move to W or O and any owner of page can add.

Protocol:

The synchronization with the server is additional action to the sync between corespondent peers, as 2 peers meet via the server having the pub key of all users, such that when the reviver is available the sender gets the how to directly connect, otherwise the enc4recepaint(msg) is temporarily stored in server until being pulled and sent when reviver is available.
S(+R) or S(-R) as S wants to add page(S) to page(R)
Case1: when page(R) is not in outward(page(S)) insert,
       not outward.insert(pop(to(page(R)))) to.insert(page(R))
Case2: when S want to be added +outward-with or +to or removed in page of R, S update its outward(page) or with(page) and sync that page with the server updating: 1. outward/with(S) 2. add/remove(R) and a also sends directly to R add/remove(to), while R update that in its add/remove.

Functions:

```
CreatePage(pg ,role ,owner)
ReadPage(pg ,role ,user)
UpdatePage(pg ,role,user)
DeletePage(pg ,role,user)
AddToPage(pg,role,user,OtherPg,OtherRole)
SubtractFromPage(pg,role,user,OtherPg,OtherRole)
```