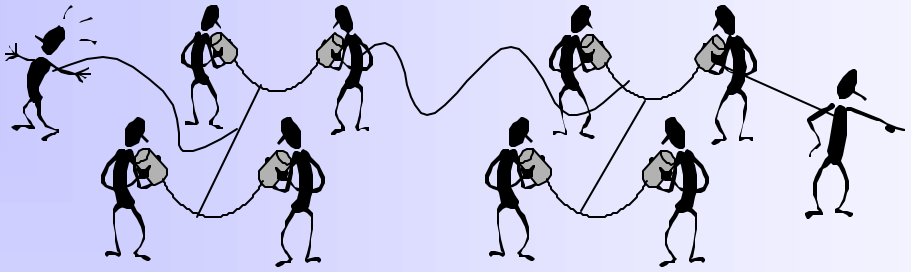
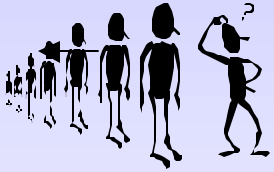
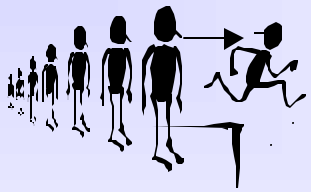
CLIQUES: Security for Dynamic Peer Groups



Formation



Member add



Member leave

Group fusion

Group fission

Problem: how to obtain security in peer groups with dynamic membership and decentralized control?

Targeted environment

- *Relatively small groups*
- *Dynamic membership*
- *No hierarchy*
- *Many-to-Many*

Services provided

- *Decentralized authenticated group key agreement with provable security based on group Diffie-Helman: each member contributes equally to group key*
- *Membership changes: single member, many members and sub-groups*
- *Membership authentication: based on knowledge of key-share*
- *Authenticated join/leave: requires long-term DH credentials*

Other pieces of the puzzle

- *Certification infrastructure*
- *Reliable group communication subsystem*
- *Membership Authorization / Access control*

Status

- *Initial Key Agreement*
- *Auxiliary Key Agreement (membership changes)*
- *Authenticated Key Agreement*
- *JAVA implementation*
- *C implementation (prototype) integrated with JHU's SPREAD package*
- *CLQ_API: coding completed end of 02/99.*
- *Currently testing and integrating with SPREAD*
- *Plan to obtain performance results very soon*
- *Integration with TOTEM on-going (LBL)*
- *Integration with AKENTI: near future*

CLQ_API prerequisites

Underlying group communication subsystem must provide reliable synchronized event notification for:

- *group joins*
- *group leaves*
- *partitions*
- *node failures or disconnects*
- *merges (heals)*

CLQ_API

```
/* called by a new group member who received a
 * NEW_MEMBER message from the current controller.
 */
int clq_join (CLQ_CONTEXT **ctx, CLQ_NAME *member_name,
              CLQ_NAME *group_name, CLQ_TOKEN *input,
              CLQ_TOKEN **output);


/* called by the current controller to hand over group
 * context to a new member (who will become the next controller).
 */
int clq_pass_ctx (CLQ_CONTEXT *ctx, CLQ_NAME *member_name,
                  CLQ_TOKEN **output);


/* called by every member upon reception of a
 * KEY_UPDATE_MESSAGE from the current group controller
 */
int clq_update_ctx (CLQ_CONTEXT *ctx, CLQ_TOKEN *input);
```

CLQ_API (contd)

```
/* clq_leave is called by every group member right after a member
 * leaves or a partition occurs; removes all valid members in
 * member_list from the group_member_list.
 */
int clq_leave (CLQ_CONTEXT *ctx, CLQ_NAME *member_list[],
               CLQ_TOKEN **output);


/* called by the controller only, when group_secret needs to be updated.
 */
int clq_refresh_key (CLQ_CONTEXT **ctx, CLQ_TOKEN **output) {
  return OK;
}
```