Bulletin Board System

A Bulletin Board System (BBS) is a distributed service where users can read messages and add their own.

In BBS, every user is identified by means of a nickname that is established at registration time together with a password.

A message is a tuple composed of the following fields: *identifier*, *title*, *author*, and *body*. The identifier field uniquely identifies the message within the BBS. The author field specifies the nickname of the user who added the message to the BBS.

BBS provides users with the following operations:

- `List(int n)` which lists the latest *n* available messages in the BBS.
- `Get(int mid)` which downloads from the BBS the message specified by message identifier mid.
- `Add(String title, String author, String body)` which adds a message to the BBS.

Registered users may issue operations after successful login. Operations are executed over a secure channel. A user who logs out cannot perform operations until (s)he successfully logs in again.

BBS is implemented in a centralized way by a BBS server. The BBS server is attested at a well-known (`ip`, `port`) couple. Furthermore, the BBS server is equipped with a private-public key pair of which the public component $pubK_{bbs}$ is known to users.

Registration Phase

- A user securely connects to the BBS server and specifies an email address, a nickname and a password;
- the server sends a challenge to the email address specified by the user and waits for receiving the challenge back;
- if the user correctly returns the challenge to the server, the registration phase concludes successfully. Otherwise, it is aborted.

Login phase

- A registered user securely connects to the BBS server and logins by means of his/her nickname and password.
- The server lets the user log in if the submitted user's nickname and password are correctly verified.
- Upon successful login, a secure session is established and maintained until the user logs out.

Requirements

- Never store or transmit passwords in the clear.
- Fulfill confidentiality, integrity, no-replay, and non-malleability in communications.
- Guarantee perfect forward secrecy (PFS).
- Reduce code vulnerabilities as much as possible.
- Use C or C++ programming language and OpenSSL library but OpenSSL API TLS cannot be used.

Deliverables

- System specification and design.
- A running early prototype.