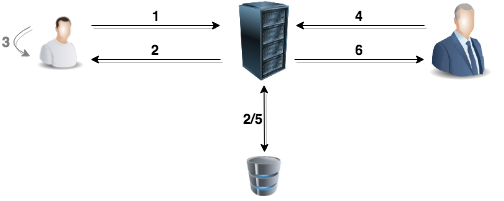
**System threats**

* Reverse engineering of the application, discovering the symmetric key which is locally stored and used in the communication between the user and the server.
* Reverse engineering of the application, skipping part of the code that deals with user authentication. This attack can be mitigated by storing a state flag in the database that indicates whether the user was authenticated or not, and the server will exchange information with the user only when the user is authenticated.
* The victim's cell phone can be stolen and the attacker can make a request to himself to follow the victim. This attack can be mostly prevented through a mandatory authentication before adding a follower.
* Certificate containing the server's public key can be changed locally, enabling a Man-In-The-Middle attack.
* Key loggers installed on users' devices can read their authentication credentials.

**Follower mechanism**

****

In this example, we call the user which is followed as the *Child* and the user that follows as the *Parent*.

1. Child intends to add Parent to his followers. To do this, he sends a message to the server with the mobile phone number of the Parent, used in the Parent's register.
2. Server generates an alphanumeric code from both the received number and a locally generated nonce. This verification code is sent to the Child and stored in the database in the form of a hash.
3. Verification code is made available in the application of the Child.
4. Parent, who is next to the Child, inserts the verification code in his application.
5. Server receives the verification code and compares its hash with the one stored in the database.
6. If they are the same, the Parent is given permission to follow the Child.