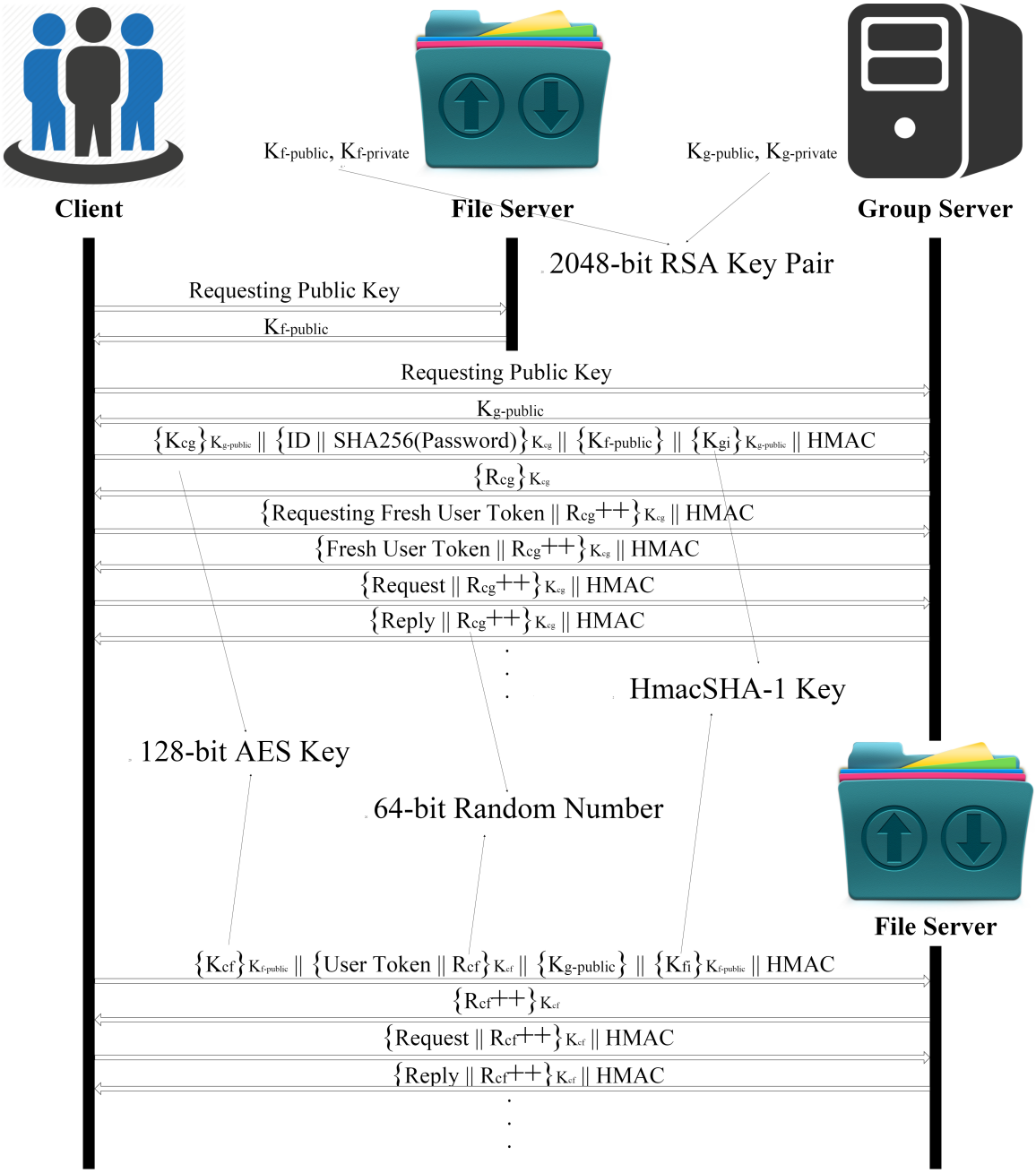
You Zhou kecen.zhou@gmail.com

Diagram



T5 Message Reorder, Replay, or Modification

- **Threat Description**

  In this phase of the project, attackers are assumed to be active as well, so besides confidentiality ensured in the last phase, integrity also becomes our concern. An active attacker may reorder, modify, or save the cipher texts being transmitted, so both client and server would receive a sequence of messages that are out of order, garbled messages, or a supposedly expired messages.

- **Mechanism**

The above diagram represents a general client-server scenario. The left icon represents clients, and the right icon can represent either the group server or the file server. The *key* used above might be either a public key or a symmetric key, but for the purpose of T5, we simply represent it as a constant.

R is a 64-bit random number chosen by whoever starts the communication, and it will be incremented after every transmission. Both parties will check the decrypted R against the old one so that any reordering attack will be detected immediately. It is also unlikely that any replay attack will succeed. Clearly, an attacker cannot use the intercepted message c' right away because it is computationally infeasible to increment the R within by manipulating the bits of c'. To have a successful replay attack, an attacker needs to wait until $R-1$ re-appears, so that c' containing R can be accepted. However, every time R reaches $2^{63}-1$, we will re-randomize R, so the chance for an attacker to notice $R-1$ will be either 0 (re-randomized to $R' > R-1$) or negligibly small. HMAC being used protects message from modification, and we choose SHA-family because the MD-family has been broken in 2008. An attacker may modify cipher text, HMAC, or both, but it is computationally infeasible for him/her to have a meaningful and matched < cipher_text, HMAC > pair.

T7 Token Theft

- **Threat Description**

  If an user token is stolen, then whoever steals it now gain all permissions this token grants – confidential files shared in the group would now be exposed to the theft, deleted without owner being aware of, or downloaded in secret.

- **Mechanism**

  A stolen token would be useless if it can only reach the theft file server containing no actual files. Since the IP address and the port number uniquely determine a file server, we add an extra filed for token to remember the file server it is communicating with. This new filed will store SHA256(IP $||$ Port#). When an user connects to a theft file server determined by IP$'$ and Port#$'$, its new field will store SHA256$\big($IP$'$ $||$ Port#$'\big)$. Theft trying to exploit the stolen token will fail if we force every file server to check the new filed before processing any request. As stated in T2, the group server signs the token before issuance, so the theft will not be able to modify the new filed of the stolen token.