This protocol is between metadata server (MDS), client (C), and key distribution server (KDS) for a file with root key K_R and user A. We assume that KDS is given private key KR_A, which is paired with public key KU_A.

The file in question is owned by the application A, and we want to allow the full access permission to the file only to the user A and the trusted KDS. In other words, both C and MDS are not allowed to access the entire file, since both C (i.e., a computation node in a supercomputer system) and MDS may be managed by a company that is different from the entity that is trying to run the application A on the supercomputer. Hence, the MDS should not have the file's master key K_R.

The block size bs, the depth of the key hash tree d, and the accessible block number (i, j) for a client C, which collectively determin the allowable range for C, are decided by either KDS or A. The range parameters (bs, d, i, j, C) for the file are either stored in MDS or in KDS.

MDS is trusted in a level that MDS is tamper-proof; if KDS or A store the range parameters (bs, d, i, j, C), they must provide the correct value to the correct entity. If MDS is compromised, it could alter the owner of the file, meaning that there is no point in worring about range alteration in MDS.

We assume That ID theft or spoofing is securely prohitited by the system. C cannot impersonate MDS or KDS in the system.

To counteract against an alteration of allowable range by C or a replay attack after range update in the system, KDS must not be provided the range parameters by C. This means that KDS must either 1) have a database storing allowable ranges for each clients C and for each files, or 2) directly talk with MDS to retrieve the range parameter for the file. This document assumes 2) for ease of implementation.

MDS provides the attributes of the file; i.e., the owner, permissions (modes), the key-hash parameter for the file (bs, d), permitted range (i, j) for a client C. The file attributes are generally open to public, meaning that MDS does not authenticate clients nor need to have clients' IDs. MDS does not need to distinguish KDS from C. MDS may provide storage service for the keys managed by KDS; MDS may hold in an extended attribute the master key K_R encrypted by KDS's public key KU_{KDS} or A's public key KU_A. In this way MDS cannot have K_R and hence cannot have the full access to the file. MDS can be implemented as the meta data server in Ceph or extended attributes (or "*forks*") in other file systems.

MDS	:	a meta data server.
C	:	a client or a computation node.
KDS	:	a key distribution server.
A	:	a user or an application.
KR_A	:	the private key for A.
KU_A	:	the public key for A.
(bs, d, i, j)	:	block size bs, depth d, range i, j.

Resulting key exchange protocol in Horus is as follows.

User A registers file attributes in extended attributes in filesystem or in MDS. A also registers KR_A in KDS.

$$\begin{aligned} A \rightarrow MDS &: \{path, bs, d, E_{KU_A}(K_R)\} & (1) \\ A \rightarrow MDS &: \{path, C_1, i_1, j_1\} & (2) \\ A \rightarrow MDS &: \{path, C_2, i_2, j_2\} & (3) \\ A \rightarrow KDS &: KR_A & (4) \end{aligned}$$

Computing node C_2 accesses the file data through the intervention by KDS.

$$\begin{aligned} C_2 \rightarrow KDS &: \text{open request: } \{path\} & (5) \\ KDS \rightarrow MDS &: \text{attr request: } \{path, C_2\} & (6) \\ MDS \rightarrow KDS &: \text{attr response: } \{path, bs, d, E_{KU_A}(K_R), C_2, i_2, j_2\} & (7) \\ KDS &: K_{i_2,j_2} \leftarrow E_{KU_A}(K_R) \times KR_A & (8) \\ KDS \rightarrow C_2 &: \{path, bs, d, i_2, j_2, K_{i_2,j_2}\} & (9) \\ C_2 \rightarrow OSD &: \text{data request} & (10) \\ OSD \rightarrow C_2 &: \text{data, then decrypted by } K_{i_2,j_2} & (11) \end{aligned}$$