Horus: Fine-Grained Encryption-Based Security for High Performance Petascale Storage

Ranjana Rajendran • Ethan L. Miller • Darrell D. E. Long
Storage Systems Research Center
University of California, Santa Cruz

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the researchers and do not necessarily reflect the views of the National Science Foundation or the Department of Energy.

Problem

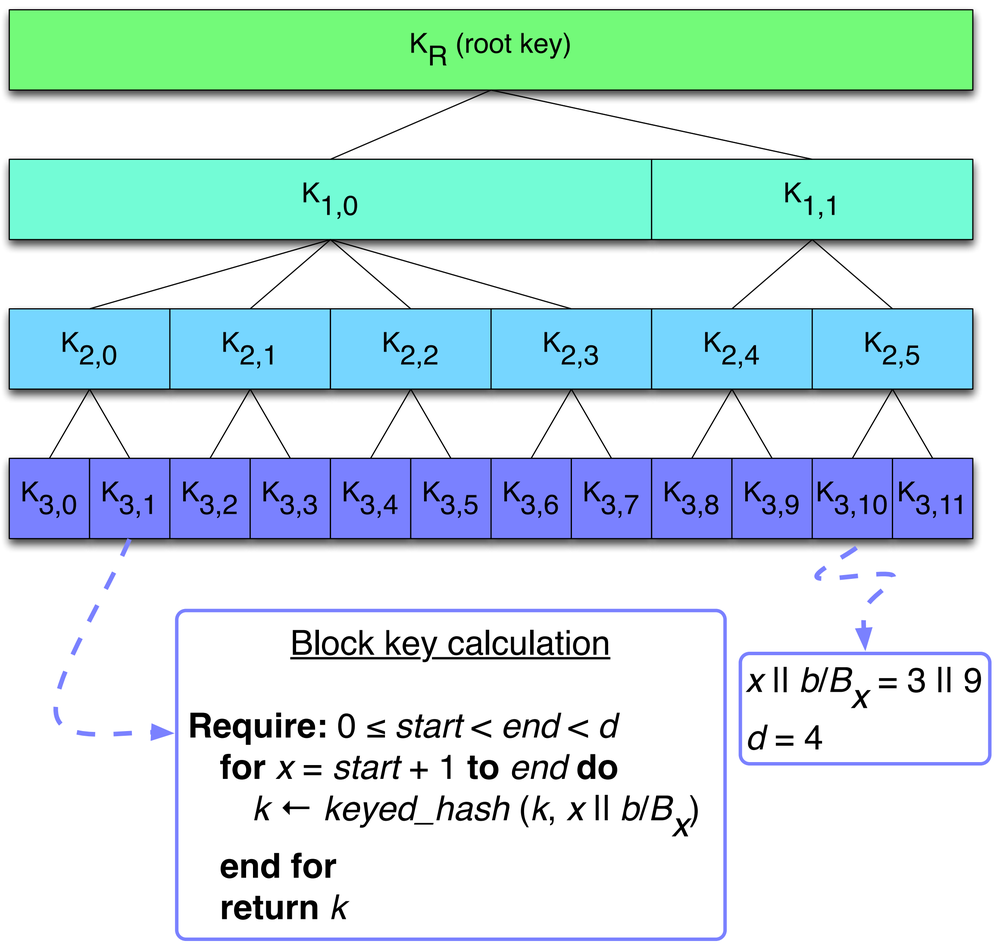
- Large files contain potentially sensitive data
- File data can be leaked by many HPC elements (disk, client, metadata server)
- ➨ Ensure data confidentiality in the face of physical and software attacks

Design Principles

- Prevent compromise by metadata server and storage nodes
  - Encrypt / decrypt all data at the client
- Restrict client leaks to only parts of the file to which the client has access
  - Most clients don't need access to the whole file
- Provide a small, stateless trusted computing base
  - Less vulnerable to compromise
  - Easier to erase between computations
- Work as a "filter" layer
  - Implement natively in the operating system
  - Implement as a client-level layer above existing file system calls

Hierarchical Keyed Hash Tree

- Single file root key can encrypt / decrypt the entire file
- Successively lower keys in the tree are based on a keyed hash depending on
  - Parent key
  - Level in the tree
  - Position in the level
- Deriving keys lower in the tree is fast and simple
- Deriving keys higher in the key or at the same level is "difficult"



Block key calculation

Require: $0 \le start < end < d$
  for $x = start + 1$ **to** end **do**
    $k \leftarrow keyed\_hash\ (k, x \,||\, b/B_x)$
  end for
  return k

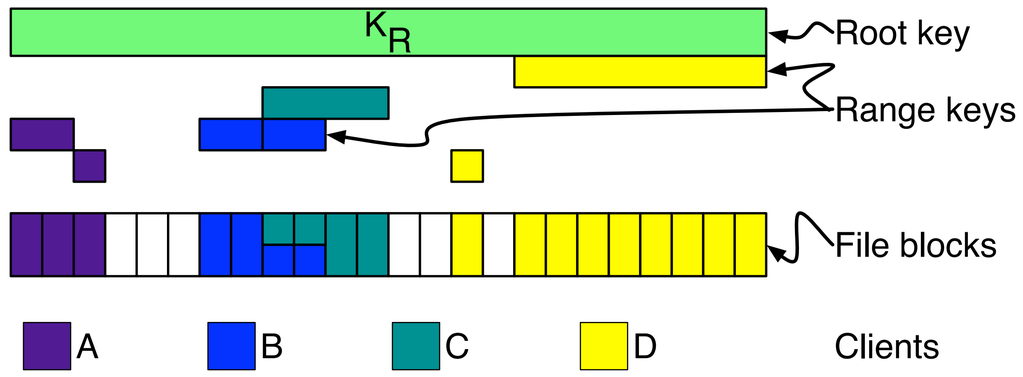
$x \,||\, b/B_x = 3 \,||\, 9$
$d = 4$

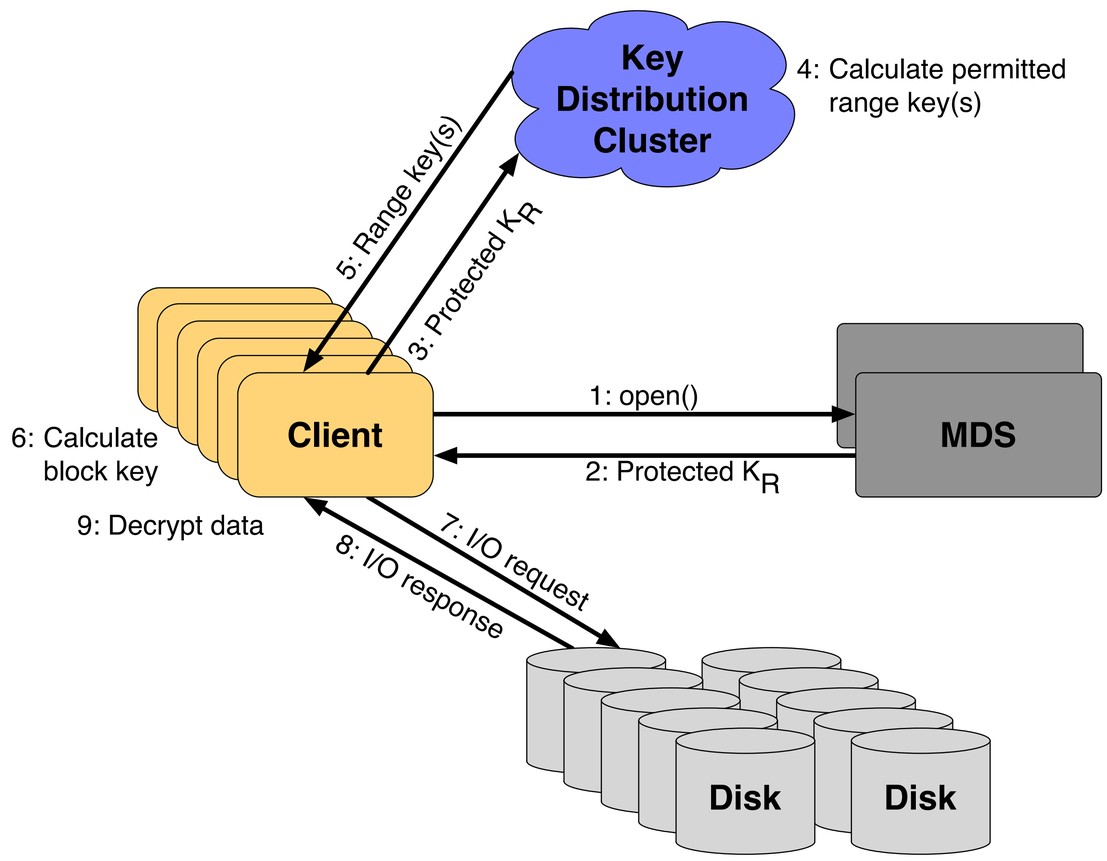
Storing the key on the metadata server

- Encrypt file root key with user's public key
- Store result on the MDS
  - Separate key file
  - Extended attribute
  - In-file metadata (*e. g.*, HDF5)

Key Distribution

- Client only receives range keys for blocks it's allowed to access
  - Client can derive a block key from any range key "above" it in the tree
- Different clients can receive the same (or different) keys for a given block
- Key distribution cluster can be run on MDS, on one or more clients, or separately
  - Can leverage application work distribution program logic to decide which clients access which ranges





Security Analysis

- Data only exists in the clear on a client and keys only in the clear on client and KDC
  - Compromise of a disk cannot reveal data
  - Compromise of a metadata server cannot reveal data
- Clients only receive range keys for blocks they need for the computation
  - Thousands of clients, each of which only needs to access a small fraction of the file
  - Individual compromised client can only reveal a small fraction of the file
- Range keys cannot be used to access data outside the range
  - Keyed hash is "one-way": cannot derive parent key from the child

Risks

- Fabricated data: encrypt cryptographic checksum along with data
- Access control for writing: use Maat
- Key revocation: use Plutus-like approach
- Access control for reading: no need (client can't read without key)

Ongoing Work

- Implementation of a user-level client library interposed above system calls
- Development of the protocol between the KDC and clients
- Performance testing
- Integration into Ceph?

Conclusions

- Security is an increasingly important problem for large-scale HPC storage
- Data can be protected against disclosure by disks and metadata servers
- A small number of compromised clients can only leak a small amount of data
- Horus can be implemented natively or as a client library
- ➨ **Horus is ideally suited to provide confidentiality for HPC data**