文件下载与删除

文件下载流程方案：

1. 用户登录到主页面后要下载文件，点击要下载文件左侧的勾选框，就会显示下载或删除的按钮，用户点击下载的按钮，即可进行文件下载。（用户还可以勾选多个文件一起下载。）
2. 前端获取用户要下载的文件的inode并将其发送给后端。
3. 后端接收前端发送的inode并在数据库中找到对应的加密的文件密钥、文件基本信息和数据块，将其发送给前端。
4. 前端接受后端发送的文件块信息，用登陆时储存在前端的主密钥将加密的文件密钥解密，并用文件密钥将加密的文件信息和数据块解密。
5. 根据文件块的顺序将文件块整合成一个完整的文件。

文件下载删除界面实现：

用户点击下载文件，  


文件下载部分代码详解：

1. 用户登录到主页面后要下载文件，点击要下载文件左侧的勾选框，就会显示下载或删除的按钮，用户点击下载的按钮，即可进行文件下载。（用户还可以勾选多个文件一起下载。）前端按钮：

<el-button  
 size="mini"  
 @click="download(scope.$index, scope.row)">下载  
</el-button>  
<el-button  
 size="mini"  
 type="danger"  
 @click="deleteFile(scope.$index, scope.row)">删除  
</el-button>

1. 前端获取用户要下载的文件的inode并将其发送给后端，将文件名，创建时间加密发送，然后发送后端加密文件名，mtime，用主密钥加密的密钥。

下载任务：

download(index, row) {  
 var userId = this.userId  
 download({  
 userId,  
 row  
 })

文件名，创建时间加密发送：

var folderKey = new Uint8Array(2 \*\* 4);  
  
let data1 = stringtoUint8Array(name)  
console.log("fileName:" + data1);  
var encryptedMasterKeyHashValue1 = await encryptKey(folderKey, clientRandomValue, data1)  
var encryptedData1 = new Uint8Array(encryptedMasterKeyHashValue1)  
console.log("encryptedData1:" + encryptedData1)  
  
var Mtime = new Date()  
var data2 = stringtoUint8Array(Mtime.toString())  
console.log("Mtime:" + data2);  
var encryptedMasterKeyHashValue2 = await encryptKey(folderKey, clientRandomValue, data2)  
var encryptedData2 = new Uint8Array(encryptedMasterKeyHashValue2)  
console.log("encryptedData2:" + encryptedData2)  
  
var encryptedMasterKeyHashValue = await encryptKey(masterKey, clientRandomValue, folderKey)  
var encryptedkey = new Uint8Array(encryptedMasterKeyHashValue)  
console.log("encryptedkey:" + encryptedkey)

发送后端加密文件名，mtime，用主密钥加密的密钥：

await request.post("/files/" + this.userId + "/" + this.curInode, JSON.stringify({  
 "filename": uint8ArrayToString(encryptedData1),  
 "size": 0,  
 "mtime": uint8ArrayToString(encryptedData2),  
 "fileKey": uint8ArrayToString(encryptedkey),  
 "type": "DIR",  
 "state": "UPLOADED",  
 })  
).then(function (res) {  
  
 alert("请求后端成功" + JSON.stringify(res))  
 if (res.code === 2000) {  
  
 console.log(JSON.stringify(res.data))  
 }  
})

1. 后端接收前端发送的inode并在数据库中找到对应的加密的文件密钥、文件基本信息和数据块，将其发送给前端。

当后端接受字符消息时响应：

@OnMessage  
public void onMessage(String message, @PathParam("userId") String userId) {  
 log.info("字符串消息" + message);  
 // 前端传进来一个字符串信息  
 FilePo filePo = JSONObject.parseObject(message, FilePo.class);  
 Set<FileBlockPo> allBlock = null;  
 if (filePo.getFileBlocks() == null) {  
 allBlock = blockService.findFileAllBlock(filePo);  
 } else {  
 allBlock = filePo.getFileBlocks();  
 }  
 List<URL> urlList = new ArrayList<>();  
 for (FileBlockPo fileBlockPo : allBlock) {  
 try {  
 URL url = blockService.downloadBlock(fileBlockPo);  
 urlList.add(url);  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 }  
 try {  
 this.sendMessage(JSONObject.toJSONString(urlList));  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
  
}

接受到字节流流时创建：

@OnMessage  
public void onMessage(byte[] message) {  
  
}

断开连接时触发：

@OnClose  
public void closeSession() {  
 subOnlineCount();  
 log.info("关闭socket通信:" + "当前连接数" + onlineCount);  
}

发生错误时触发：

@OnError  
public void sessionError(Session session, Throwable throwable) {  
 log.info("发生异常事件");  
}

在数据库去找到对应的加密的文件密钥、文件基本信息和数据块：

*/\*\*  
 \* 原子性的++操作  
 \*/* public static synchronized void addOnlineCount() {  
 BlockDownloadWebSocket.onlineCount++;  
 }  
  
 */\*\*  
 \* 原子性的--操作  
 \*/* public static synchronized void subOnlineCount() {  
 BlockDownloadWebSocket.onlineCount--;  
 }  
}

1. 前端接受后端发送的文件块信息，用登陆时储存在前端的主密钥将加密的文件密钥解密，并用文件密钥将加密的文件信息和数据块解密。

解密列表数据：

async function encryptlist(tdata) {  
 let clientRandomValue = stringtoUint8Array(localStorage.getItem('clientRandomValue'));  
 const masterKey = stringtoUint8Array(localStorage.getItem('masterKey'));  
 for (var i = 0; i < tdata.length; i++) {  
  
  
 if (tdata[i].type === "DIR") {  
  
 tdata[i].size = '-'  
 } else {  
 var encryptedfileKey = stringtoUint8Array(tdata[i].fileKey)  
 var fileKey = await dec(masterKey, clientRandomValue, encryptedfileKey)  
 var encryptedfilename = stringtoUint8Array(tdata[i].filename)  
 var encryptedmtime = stringtoUint8Array(tdata[i].mtime)  
 console.log(encryptedfilename)  
 console.log(encryptedmtime)  
 console.log(fileKey)  
 fileKey = new Uint8Array([166, 211, 186, 222, 50, 173, 124, 208, 231, 185, 89, 47, 99, 253, 157, 56]);  
 clientRandomValue = new Uint8Array([28, 99, 105, 140, 100, 252, 242, 31, 114, 250, 121, 220, 43, 185, 162, 151]);urn tdata  
}

解密文件名：

var filename = await dec(fileKey, clientRandomValue, encryptedfilename)  
 console.log("文件名：" + filename)  
 tdata[i].filename = uint8ArrayToString(filename)  
 // var mtime = await dec(fileKey, clientRandomValue, encryptedmtime)  
 // console.log("mtime：" + mtime)  
 // tdata[i].mtime=dateToString(uint8ArrayToString(mtime).toDate())  
 tdata[i].size = showfilesize(tdata[i].size)  
}

1. 根据文件块的顺序将文件块整合成一个完整的文件。

保存文件块到数据库，找到文件的所有块的信息：

*/\*\*  
 \* 保存文件块信息到数据库  
 \** ***@param*** *fileBlockPo  
 \** ***@return*** *\*/*FileBlockPo save(FileBlockPo fileBlockPo);  
  
*/\*\*  
 \* 找到文件的所有块  
 \** ***@param*** *filePo  
 \** ***@return*** *\*/*Set<FileBlockPo> findFileAllBlock(FilePo filePo);

public interface FileBlockRepository extends JpaRepository<FileBlockPo,Long> {  
 */\*\*  
 \* 找到当前文件的所有块  
 \** ***@param*** *filePo  
 \** ***@return*** *\*/* Set<FileBlockPo> findByParentFilePo(FilePo filePo);  
}

文件删除流程方案：

1. 用户登录到主页面后要删除文件，点击要删除文件左侧的勾选框，就会显示下载或删除的按钮，用户点击删除的按钮，即可进行文件删除。（用户还可以勾选多个文件一起删除。）
2. 前端获取要删除的文件的inode并将其发送给后端。
3. 后端接收前端发送的inode，并在数据库中找到改inode对应的文件信息并将其从数据库中删除，同时也要将Ceph系统储存的文件块一并找到并删除。
4. 后端删除成功后，前端将已经删除的文件从文件列表中删除。

文件下载删除界面实现：



文件删除部分代码详解：

1. 用户登录到主页面后要删除文件，点击要删除文件左侧的勾选框，就会显示下载或删除的按钮，用户点击删除的按钮，即可进行文件删除。（用户还可以勾选多个文件一起删除。）前端按钮：

<el-button  
 size="mini"  
 @click="download(scope.$index, scope.row)">下载  
</el-button>  
<el-button  
 size="mini"  
 type="danger"  
 @click="deleteFile(scope.$index, scope.row)">删除  
</el-button>

1. 前端获取要删除的文件的inode，发送请求给后端，请求成功后，并将其发送给后端，。

async clickFolder(row) {  
 if (row.type === "DIR") {  
 this.tableData = await encryptlist(row.childrenFiles)  
 this.breadlist.push({"name": row.filename, "inode": row.inode, "parent\_inode": this.curInode})  
 this.curInode = row.inode  
 }  
},

// 删除文件  
deleteFile(index, row) {  
 alert(index)  
 let \_this = this  
 request.delete("/files/" + this.userId + "/" + row.inode).then(function (res) {  
 if (res.code === 2000) {  
 alert("请求后端成功" + JSON.stringify(res))  
 \_this.tableData.splice(index, 1)  
 }  
 })  
},

1. 后端接收前端发送的inode，并在数据库中找到改inode对应的文件信息并将其从数据库中删除，同时也要将Ceph系统储存的文件块一并找到并删除。

*/\*\*  
 \* 获取指定用户，指定目录下的所有文件  
 \*  
 \** ***@param*** *userPo  
 \** ***@param*** *parentDir  
 \** ***@return*** *\*/*List<FilePo> findAllByUserAndParentDir(UserPo userPo, FilePo parentDir);  
  
*/\*\*  
 \* 删除指定用户的指定文件  
 \*  
 \** ***@param*** *inode 文件inode  
 \** ***@param*** *userPo 用户  
 \*/*void deleteByInodeAndUser(Long inode, UserPo userPo);  
  
FilePo findByInodeAndUser(Long inode, UserPo userPo);

*/\*\*  
 \* 删除文件  
 \** ***@param*** *inode 文件inode  
 \** ***@param*** *userId 用户id  
 \*/*void deleteFile(Long inode,Long userId);

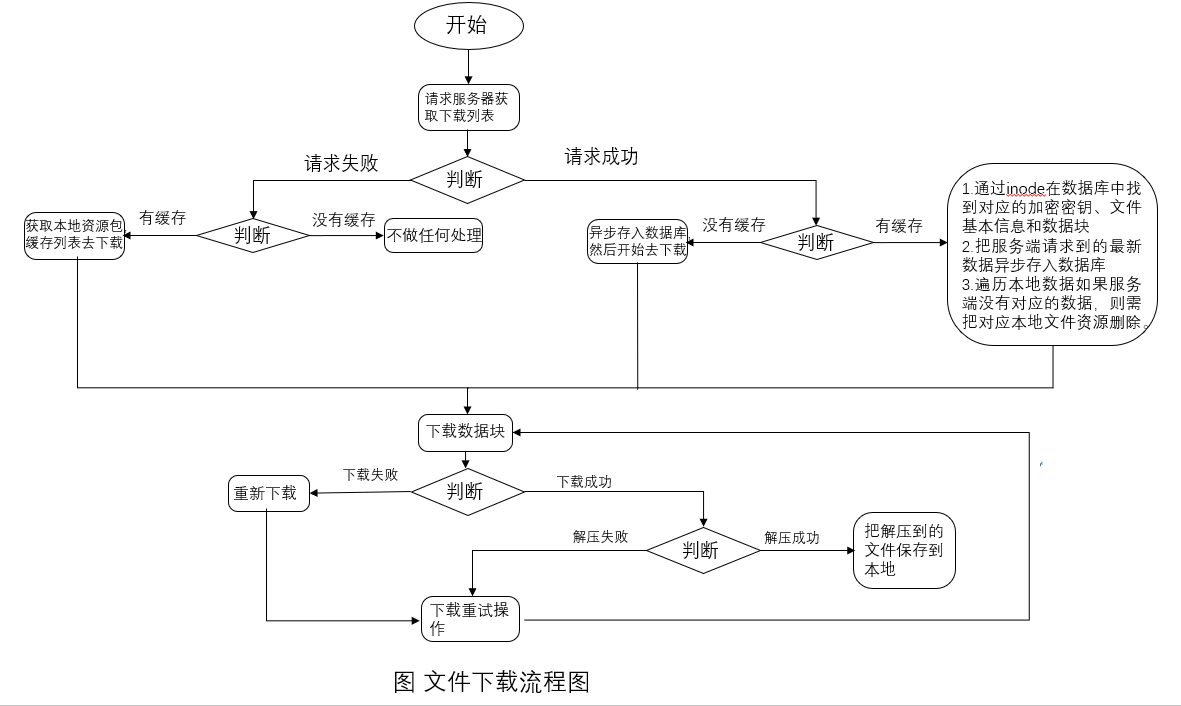
1. 后端删除成功后，前端将已经删除的文件从文件列表中删除。

//文件夹展示到当前目录  
var time = dateToString(Mtime)  
this.tableData.push({filename: name, mtime: time, size: '-', type: "DIR"})

async back() {  
 let b = this.fromData  
 for (var i = 0; i < this.breadlist.length - 1; i++) {  
 for (var j = 0; j < b.length; j++) {  
 if (this.breadlist[i].inode === b[j].inode) {  
 b = b[j].childrenFiles  
 }  
 }  
 }

// 删除文件  
deleteFile(index, row) {  
 alert(index)  
 let \_this = this  
 request.delete("/files/" + this.userId + "/" + row.inode).then(function (res)

附文件下载流程图：



若需源码，请访问https://github.com/darKmili/cloud