**static int process\_request\_key\_err(long err\_code)**

根据错误代码err\_code打印错误消息。

**static int process\_find\_global\_auth\_tok\_for\_sig\_err(int err\_code)**

**int ecryptfs\_parse\_packet\_length(unsigned char \*data, size\_t \*size,size\_t \*length\_size)**

data：包含包长度的地址；

size：保存解析以后的包长度（decoded size）；

length\_size：包长度所占的字节数（encoded size）；

成功返回0；

**int ecryptfs\_write\_packet\_length(char \*dest, size\_t size,size\_t \*packet\_size\_length)**

dest：根据size的大小是否大于192，将size转换后保存在dest[0]和dest[1]中；

size：要写的长度；

packet\_size\_length：长度的字节数，如果size小于192，则用一个字节保存，如果size在192和65536之间，则用两个字节表示；

**static int write\_tag\_64\_packet(char \*signature, struct ecryptfs\_session\_key \*session\_key,**

**char \*\*packet, size\_t \*packet\_len)**

signature：密钥标识；

session\_key：此函数会用到ecryptfs\_session\_key结构中的encrypted\_key\_size和encrypted\_key；

packet：\*packet保存该函数处理后的结果，也就是tag64包；

packet\_len：保存\*packet实际的长度字节数；

|  |  |
| --- | --- |
| Content Type | ECRYPTFS\_TAG\_64\_PACKET\_TYPE |
| Key Identifier Size | ECRYPTFS\_SIG\_SIZE\_HEX |
| Key Identifier | 传进来的signature |
| Encrypted File Encryption Key Size | session\_key->encrypted\_key\_size |
| Encrypted File Encryption Key | session\_key->encrypted\_key |

**static int parse\_tag\_65\_packet(struct ecryptfs\_session\_key \*session\_key, u8 \*cipher\_code,**

**struct ecryptfs\_message \*msg)**

|  |  |
| --- | --- |
| Content Type | ECRYPTFS\_TAG\_65\_PACKET\_TYPE |
| Status Indicator |  |
| File Encryption Key Size | session\_key->decrypted\_key\_size |
| File Encryption Key | session\_key->decrypted\_key从应用层传进来的decrypted key：1字节的cipher code+decrypted\_key+2字节的检验和 |

**static int write\_tag\_66\_packet(char \*signature, u8 cipher\_code,struct ecryptfs\_crypt\_stat \*crypt\_stat,**

**char \*\*packet,size\_t \*packet\_len)**

signature：密钥标识；

cipher\_code：

crypt\_stat：

packet：

packet\_len：

|  |  |
| --- | --- |
| Content Type | ECRYPTFS\_TAG\_66\_PACKET\_TYPE |
| Key Identifier Size | ECRYPTFS\_SIG\_SIZE\_HEX |
| Key Identifier | 传进来的signature |
| File Encryption Key Size | crypt\_stat->key\_size + 3 |
| File Encryption Key | crypt\_stat->key  =cipher\_code+crypt\_stat->key+checksum |

**static int parse\_tag\_67\_packet(struct ecryptfs\_key\_record \*key\_rec,struct ecryptfs\_message \*msg)**

|  |  |
| --- | --- |
| Content Type | ECRYPTFS\_TAG\_67\_PACKET\_TYPE |
| Status Indicator |  |
| Encrypted File Encryption Key Size | key\_rec->enc\_key\_size |
| Encrypted File Encryption Key | key\_rec->enc\_key |

**static int ecryptfs\_verify\_version(u16 version)**

**static int**

**ecryptfs\_verify\_auth\_tok\_from\_key(struct key \*auth\_tok\_key,struct ecryptfs\_auth\_tok \*\*auth\_tok)**

ecryptfs\_auth\_tok结构的token\_type必须为如下之一：

ECRYPTFS\_PASSWORD

ECRYPTFS\_PRIVATE\_KEY

如果是有效的auth token，返回0；

**static int**

**ecryptfs\_find\_global\_auth\_tok\_for\_sig(struct key \*\*auth\_tok\_key,struct ecryptfs\_auth\_tok \*\*auth\_tok,**

**struct ecryptfs\_mount\_crypt\_stat \*mount\_crypt\_stat, char \*sig)**

auth\_tok\_key：

auth\_tok：

mount\_crypt\_stat：

sig：

根据sig在mount\_crypt\_stat的global\_auth\_tok\_list链表中查找，该链表上的元素是ecryptfs\_global\_auth\_tok结构，找到并判断是正确的token\_type后，将ecryptfs\_global\_auth\_tok的key保存在auth\_tok\_key，而对应的auto token指针则等于(&((struct encrypted\_key\_payload \*)key->payload.data)->payload\_data)。

**static int**

**ecryptfs\_find\_auth\_tok\_for\_sig(struct key \*\*auth\_tok\_key,struct ecryptfs\_auth\_tok \*\*auth\_tok,**

**struct ecryptfs\_mount\_crypt\_stat \*mount\_crypt\_stat,char \*sig)**

直接调用ecryptfs\_find\_global\_auth\_tok\_for\_sig，从mount\_crypt\_stat的全局auth token链表中查找，这种方式要求在mount时就必须静态注册auth\_token；除此之外，如果没有静态注册，则通过ecyptfsd动态获得auth\_tok。

该函数通过两种方式：ecryptfs\_find\_global\_auth\_tok\_for\_sig 和ecryptfs\_keyring\_auth\_tok\_for\_sig来查找认证token。如果在mount的时候设置了ECRYPTFS\_GLOBAL\_MOUNT\_AUTH\_TOK\_ONLY标志，则只能通过ecryptfs\_find\_global\_auth\_tok\_for\_sig来查找认证token，否则在未找到全局token时，再通过ecryptfs\_keyring\_auth\_tok\_for\_sig从应用层获得认证token。

事实上，在ecryptfs\_mount时，ecryptfs\_parse\_options的最后，仍然会调用ecryptfs\_init\_global\_auth\_toks来初始化全局token，这个函数里同样调用了ecryptfs\_keyring\_auth\_tok\_for\_sig从应用层获得token。

**int ecryptfs\_write\_tag\_70\_packet(char \*dest, size\_t \*remaining\_bytes,size\_t \*packet\_size,**

**struct ecryptfs\_mount\_crypt\_stat \*mount\_crypt\_stat,**

**char \*filename, size\_t filename\_size)**

dest：保存函数处理后的tag70包，其中包含加密后的文件名，以及文件名前面的按块对齐的随机数；

remaining\_bytes：

packet\_size：tag70包的字节数长度；

mount\_crypt\_stat：

filename：

filename\_size：

该函数主要是将filename文件名进行加密处理，结果放在dest中。

|  |  |
| --- | --- |
| Tag 70 identifier | ECRYPTFS\_TAG\_70\_PACKET\_TYPE |
| Tag 70 packet size | ECRYPTFS\_SIG\_SIZE+1+ s->block\_aligned\_filename\_size |
| FNEK sig | mount\_crypt\_stat->global\_default\_fnek\_sig |
| Cipher identifier | mount\_crypt\_stat->global\_default\_fn\_cipher\_name |
| Block-aligned encrypted filename |  |

哈希标准处理：

|  |
| --- |
| void sg\_init\_one(struct scatterlist \*sg, const void \*buf, unsigned int buflen) |
| static inline struct crypto\_hash \*crypto\_alloc\_hash(const char \*alg\_name,u32 type, u32 mask) |
| static inline int crypto\_hash\_init(struct hash\_desc \*desc) |
| static inline int crypto\_hash\_update(struct hash\_desc \*desc,struct scatterlist \*sg,unsigned int nbytes) |
| static inline int crypto\_hash\_final(struct hash\_desc \*desc, u8 \*out) |

加密处理：

|  |
| --- |
| int virt\_to\_scatterlist(const void \*addr, int size, struct scatterlist \*sg,int sg\_size)针对源和目的 |
| static inline int crypto\_blkcipher\_setkey(struct crypto\_blkcipher \*tfm, const u8 \*key, unsigned int keylen) |
| static inline int crypto\_blkcipher\_encrypt\_iv(struct blkcipher\_desc \*desc,struct scatterlist \*dst,struct scatterlist \*src,unsigned int nbytes) |

**int ecryptfs\_parse\_tag\_70\_packet(char \*\*filename, size\_t \*filename\_size,size\_t \*packet\_size,**

**struct ecryptfs\_mount\_crypt\_stat \*mount\_crypt\_stat,char \*data, size\_t max\_packet\_size)**

与ecryptfs\_write\_tag\_70\_packet正好相反，将data中的加密的文件名进行解密放到filename中。

|  |  |
| --- | --- |
| ECRYPTFS\_TAG\_70\_PACKET\_TYPE  Tag 70的标识 | 1字节 |
| 包的字节数 | 1到2字节 |
| FNEK sig文件名加密密钥标识 | ECRYPTFS\_SIG\_SIZE 8字节 |
| Block aligned加密后的文件名 | 解析出的包的长度-ECRYPTFS\_SIG\_SIZE - 1 |
|  |  |

**static int ecryptfs\_get\_auth\_tok\_sig(char \*\*sig, struct ecryptfs\_auth\_tok \*auth\_tok)**

获得auth\_tok的sinature。

**static int decrypt\_pki\_encrypted\_session\_key(struct ecryptfs\_auth\_tok \*auth\_tok,**

**struct ecryptfs\_crypt\_stat \*crypt\_stat)**

|  |
| --- |
| ecryptfs\_get\_auth\_tok\_sig |
| write\_tag\_64\_packet |
| ecryptfs\_send\_message |
| ecryptfs\_wait\_for\_response |
| parse\_tag\_65\_packet |

根据sigature和session\_key调用write\_tag\_64\_packet组成tag64包，发送到应用层去解密会话密钥，再把应用层通过消息发送回来的解密后的会话密钥tag65包解析出来，赋值给crypt\_stat相应成员。

**static void wipe\_auth\_tok\_list(struct list\_head \*auth\_tok\_list\_head)**

**static int parse\_tag\_1\_packet(struct ecryptfs\_crypt\_stat \*crypt\_stat,**

**unsigned char \*data, struct list\_head \*auth\_tok\_list,**

**struct ecryptfs\_auth\_tok \*\*new\_auth\_tok, size\_t \*packet\_size, size\_t max\_packet\_size)**

crypt\_stat：

data：

auth\_tok\_list：

new\_auth\_tok：

packet\_size：

max\_packet\_size：

Public-Key Encrypted Session Key Packets（RFC 2440 Tag 1）

用公钥加密的会话密钥

|  |  |
| --- | --- |
| ECRYPTFS\_TAG\_1\_PACKET\_TYPE  Tag 1的标识 | 1字节 |
| 包的字节数 | 最多3字节，目前支撑最多2字节。 |
| 版本号 | 1字节，应为0x03 |
| Key identifier密钥标识 | ECRYPTFS\_SIG\_SIZE 8字节 |
| Cipher identifier公钥密码算法标识 | 1个字节，因为内核不需要知道公钥算法，解析时跳过此字节 |
| Encrypted key | ENCRYPTED\_KEY的字节数=  包的字节数-（ECRYPTFS\_SIG\_SIZE+2）  2表示版本号和公钥算法标识，共2字节 |
|  |  |

**static int parse\_tag\_3\_packet(struct ecryptfs\_crypt\_stat \*crypt\_stat,**

**unsigned char \*data, struct list\_head \*auth\_tok\_list,**

**struct ecryptfs\_auth\_tok \*\*new\_auth\_tok,size\_t \*packet\_size, size\_t max\_packet\_size)**

Symmetric-Key Encrypted Session-Key Packets（RFC 2440 Tag 3）

|  |  |
| --- | --- |
| ECRYPTFS\_TAG\_3\_PACKET\_TYPE  Tag 3的标识 | 1字节 |
| 包的字节数 | 最多3字节，目前支撑最多2字节。 |
| 版本号 | 1字节，应为0x04 |
| Cipher code算法对应的编号 | 1字节 |
| S2K specifier | 1字节，目前只支持0x03 |
| Hash identifier | 1字节，目前只支持0x01，即MD5算法 |
| Salt | 8字节，ECRYPTFS\_SALT\_SIZE |
| Hash iterations，哈希迭代次数 | 1字节 |
| Encrypted key | 字节数=包的字节数- (ECRYPTFS\_SALT\_SIZE + 5)  5表示版本号、Cipher code、S2K specifier、Hash identifier和Hash iterations |

**static int parse\_tag\_11\_packet(unsigned char \*data, unsigned char \*contents,**

**size\_t max\_contents\_bytes, size\_t \*tag\_11\_contents\_size,**

**size\_t \*packet\_size, size\_t max\_packet\_size)**

Literal Data Packet（RFC 2440 Tag 11）

|  |  |
| --- | --- |
| ECRYPTFS\_TAG\_11\_PACKET\_TYPE  Tag 11的标识 | 1字节 |
| 包的字节数 | 最多3字节，目前支撑最多2字节。 |
| Binary format specifier | 1字节，应为0x62 |
| Filename length | 1字节，文件名长度，应为0x08 |
| Filename ("\_CONSOLE") | 8个字节 |
| Modification date修改日期 | 4字节 |
| Literal data | 8字节ECRYPTFS\_SIG\_SIZE大小，实际保存的signature |

**int ecryptfs\_keyring\_auth\_tok\_for\_sig(struct key \*\*auth\_tok\_key,**

**struct ecryptfs\_auth\_tok \*\*auth\_tok,char \*sig)**

根据sigature，调用request\_key获得auth\_tok\_key，进一步获得auth\_tok。

**static int decrypt\_passphrase\_encrypted\_session\_key(struct ecryptfs\_auth\_tok \*auth\_tok,**

**struct ecryptfs\_crypt\_stat \*crypt\_stat)**

调用标准的内核密码接口对会话密钥进行解密，解密后auth\_tok\_key的session\_key有decrypted\_key，同时还会放在crypt\_stat的key中，此时内存中的会话密钥就是明文了。

struct ecryptfs\_session\_key {

…

u32 encrypted\_key\_size;

u32 decrypted\_key\_size;

u8 encrypted\_key[ECRYPTFS\_MAX\_ENCRYPTED\_KEY\_BYTES];

u8 decrypted\_key[ECRYPTFS\_MAX\_KEY\_BYTES];

};

**int ecryptfs\_parse\_packet\_set(struct ecryptfs\_crypt\_stat \*crypt\_stat,**

**unsigned char \*src,struct dentry \*ecryptfs\_dentry)**

1、parse\_tag\_1\_packet/3/11解析一个包时，都会构造一个auth\_tok\_list\_item及对应的ecryptfs\_auth\_tok结构，链接到以auth\_tok\_list为表头的双向链表中，直到把所有metadata解析完毕。解析的过程，最重要的就是根据密码算法类型，将token\_type 赋值为ECRYPTFS\_PASSWORD或者ECRYPTFS\_PRIVATE\_KEY，把从包中解析出来的密钥标识放到token中的signature，把从包中解析出的加过密的密钥放到session\_key中。

2、然后从auth\_tok\_list链表头开始循环处理，通过ecryptfs\_get\_auth\_tok\_sig一个一个的取出密钥标识，然后调用ecryptfs\_find\_auth\_tok\_for\_sig，根据密钥标识来获得auth\_tok。

3、找到匹配的认证token后，根据token\_type是ECRYPTFS\_PRIVATE\_KEY或ECRYPTFS\_PASSWORD来调用decrypt\_pki\_encrypted\_session\_key或者decrypt\_passphrase\_encrypted\_session\_key来解密会话密钥。

4、调用ecryptfs\_compute\_root\_iv计算出ecryptfs\_crypt\_stat中保存的解密后的密钥的哈希值，并保存在root\_iv中；

5、调用ecryptfs\_init\_crypt\_ctx，将ecryptfs\_crypt\_stat中的cipher也就是密码算法的字符串加上“cbc”后重新查找加上cbc的crypto\_blkcipher算法，赋值给ecryptfs\_crypt\_stat的tfm。

**static int pki\_encrypt\_session\_key(struct key \*auth\_tok\_key,struct ecryptfs\_auth\_tok \*auth\_tok,**

**struct ecryptfs\_crypt\_stat \*crypt\_stat,struct ecryptfs\_key\_record \*key\_rec)**

通过调用write\_tag\_66\_packet组织成tag66的包发送到应用层进行会话密钥的加密，应用层处理完后再送回来，由parse\_tag\_67\_packet来解析，放在key\_rec中。

|  |  |  |
| --- | --- | --- |
| pki\_encrypt\_session\_key  应用层加密会话密钥 | write\_tag\_66\_packet | 输入参数有signature，cipher\_code，crypt\_stat->key |
| ecryptfs\_send\_message |
| ecryptfs\_wait\_for\_response | 返回到key\_rec，保存加过密的密钥 |
| parse\_tag\_67\_packet |
| decrypt\_pki\_encrypted\_session\_key  应用层解密会话密钥 | write\_tag\_64\_packet | 输入参数auth\_toke中sigature和session\_key中的encrypted\_key |
| ecryptfs\_send\_message |
| ecryptfs\_wait\_for\_response | 返回到session\_key中的decrypted\_key，后面还要拷贝到crypt\_stat->key中 |
| parse\_tag\_65\_packet |

**static int write\_tag\_1\_packet(char \*dest, size\_t \*remaining\_bytes,**

**struct key \*auth\_tok\_key, struct ecryptfs\_auth\_tok \*auth\_tok,**

**struct ecryptfs\_crypt\_stat \*crypt\_stat,**

**struct ecryptfs\_key\_record \*key\_rec, size\_t \*packet\_size)**

会调用pki\_encrypt\_session\_key加密会话密钥放在key\_rec中，key\_rec->enc\_key加密后的会话密钥会放在Tag1包的最后。

**static int write\_tag\_11\_packet(char \*dest, size\_t \*remaining\_bytes, char \*contents,**

**size\_t contents\_length, size\_t \*packet\_length)**

**static int write\_tag\_3\_packet(char \*dest, size\_t \*remaining\_bytes,**

**struct ecryptfs\_auth\_tok \*auth\_tok,**

**struct ecryptfs\_crypt\_stat \*crypt\_stat,**

**struct ecryptfs\_key\_record \*key\_rec, size\_t \*packet\_size)**

用auth\_tok->token.password.session\_key\_encryption\_key会话密钥的加密密钥，对crypt\_stat->key中的会话密钥明文进行加密后，放到key\_rec->enc\_key中，放到Tag3包里。

**int ecryptfs\_generate\_key\_packet\_set(char \*dest\_base,struct ecryptfs\_crypt\_stat \*crypt\_stat,**

**struct dentry \*ecryptfs\_dentry, size\_t \*len,size\_t max)**

循环处理crypt\_stat->keysig\_list链表上的ecryptfs\_key\_sig。

|  |  |
| --- | --- |
| ECRYPTFS\_PASSWORD | write\_tag\_3\_packet |
| write\_tag\_11\_packet |
| ECRYPTFS\_PRIVATE\_KEY | write\_tag\_1\_packet |

**int ecryptfs\_add\_keysig(struct ecryptfs\_crypt\_stat \*crypt\_stat, char \*sig)**

list\_add(&new\_key\_sig->crypt\_stat\_list, &crypt\_stat->keysig\_list);

**int ecryptfs\_add\_global\_auth\_tok(struct ecryptfs\_mount\_crypt\_stat \*mount\_crypt\_stat,**

**char \*sig, u32 global\_auth\_tok\_flags)**

list\_add(&new\_auth\_tok->mount\_crypt\_stat\_list, &mount\_crypt\_stat->global\_auth\_tok\_list);