项 目 编 号	2021092604
文 档 编 号	2
密 级	内部

【文件备份系统】
系统设计说明

[版本号 V1.1.0]

2021 年 9 月 8 日

组长：董文龙 **2018081309003**

组员：贾明昊 **2018081307011**

马宇成 **2018081307013**

1. 引言

1.1 编写目的

该文档的目的是描述《数据备份》项目的需求规格说明，其主要内容包括：

1．在实验一基础上完成系统的体系结构的建立和系统概要设计，并给出相应的软件体系架构。

2．用面向数据流的设计方法从需求分析的数据流图导出系统结构图，并进行优化，画出系统的软件结构图。

3．选择一个模块进行输入输出界面设计，输出设计主要指打印输出，设计输入设计主要指数据录入界面设计。

1.2 项目风险

具体说明本软件开发项目的全部风险承担者，以及各自在本阶段所需要承担的主要风险，首要风险承担者包括：

- 任务提出者；
- 软件开发者；
- 产品使用者。

1.3 预期读者和阅读建议

本文档的预期的读者是：

- 开发人员；
- 项目管理人员；
- 测试人员。

2. 词汇表

平面文件：一个文件夹中含有一个或多个文件，不含有子目录。

目录树：一个文件夹中含有文件和目录，并且子目录下可能还含有孙目录和文件依次下推。

打包：将平面文件或目录树合并成一个总的文件。

解包：将合并后的文件还原成原来的平面文件或目录树。

压缩：将打包后的文件通过算法压缩为压缩文件。

解压：将压缩文件通过算法解压为打包文件。

加密：使用密码将压缩文件加密为加密文件。

解密：使用密码将加密文件解密为压缩文件。

上传：将备份文件上传至云服务器。

下载：将云服务器上的备份文件下载到本地。

校验：对于定时的备份，可以将当前备份与磁盘上的原始文件进行对比，查看差异。

日志：用户操作记录

3. 设计概述

3.1 开发环境和工具

QT：模块开发和 GUI

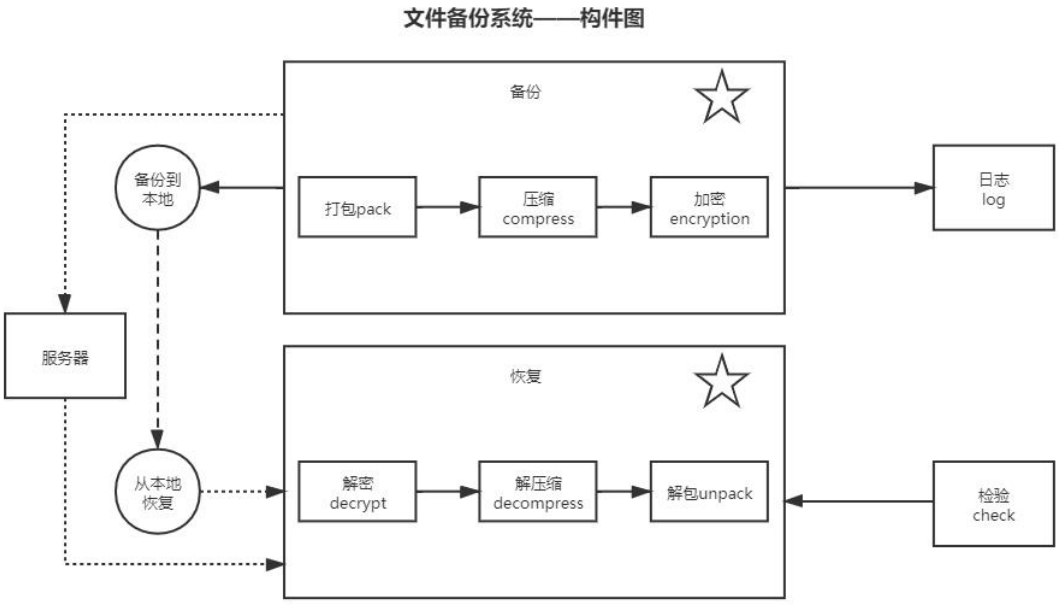
Git：项目版本控制

Star UML：UML 图绘制

Java：服务器开发

4. 详细设计

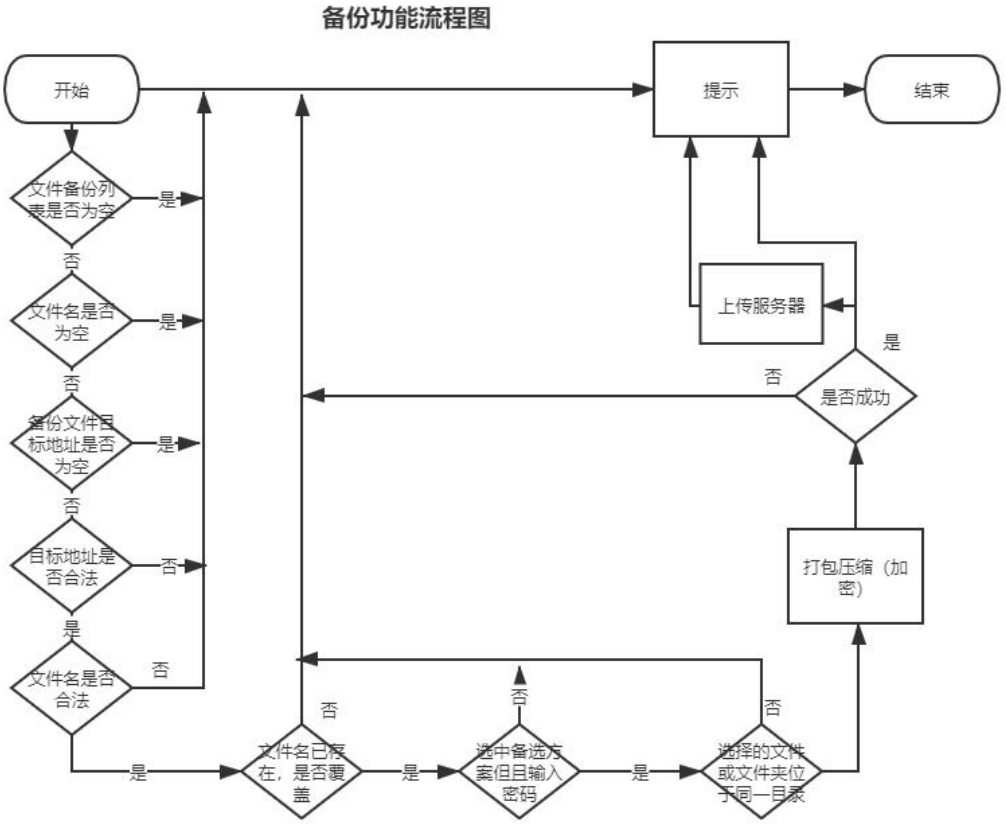
4.1 系统构件图



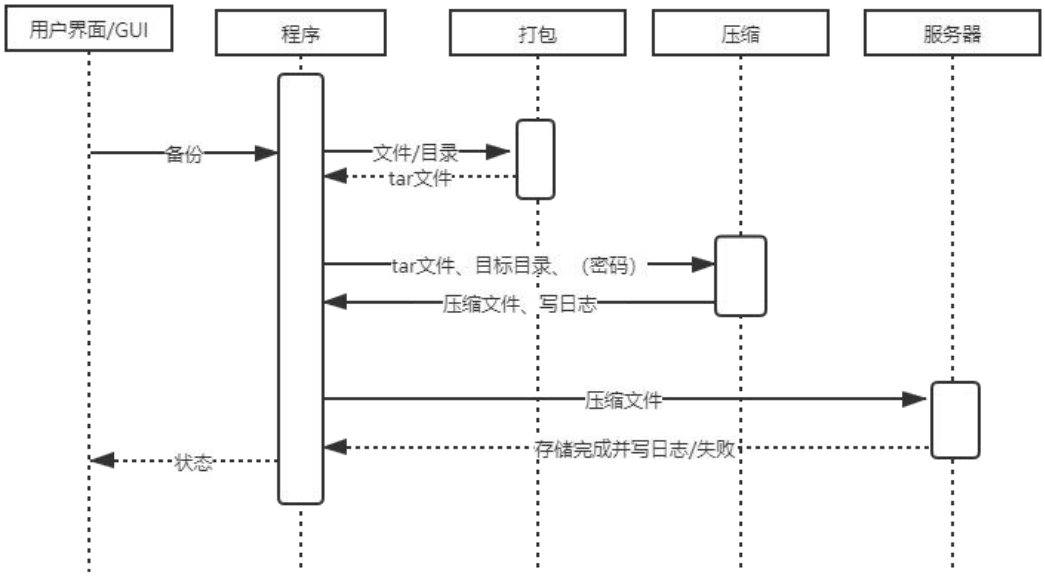
4.2 系统各界面流程描述

4.2.1 备份功能

流程图：

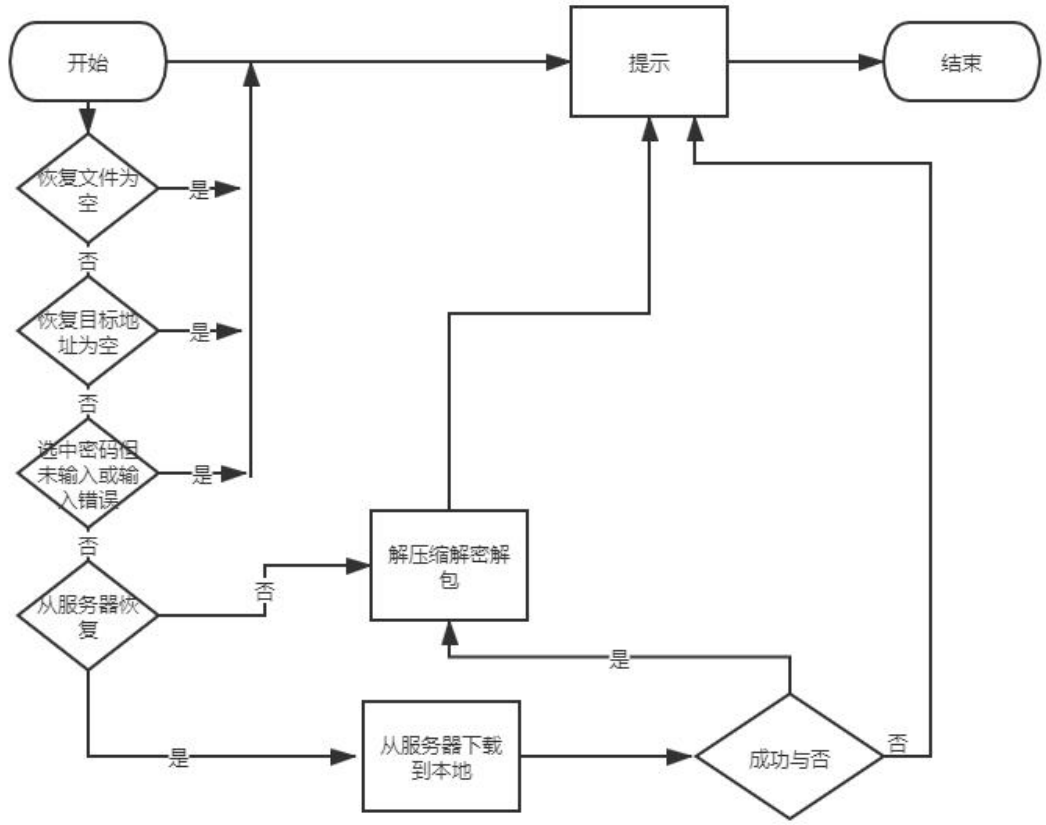


时序图：

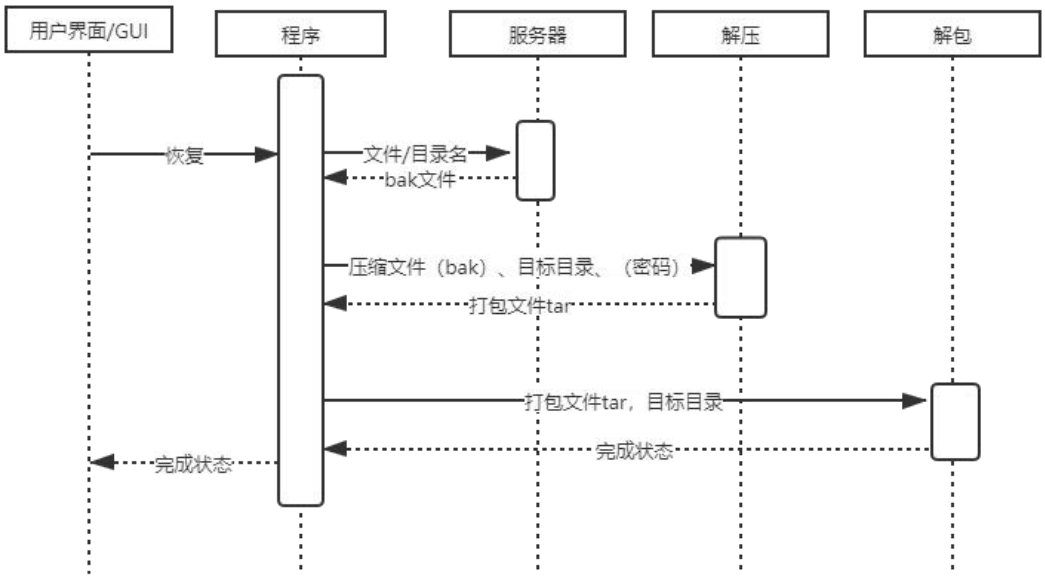


4.2.2 恢复功能

流程图：

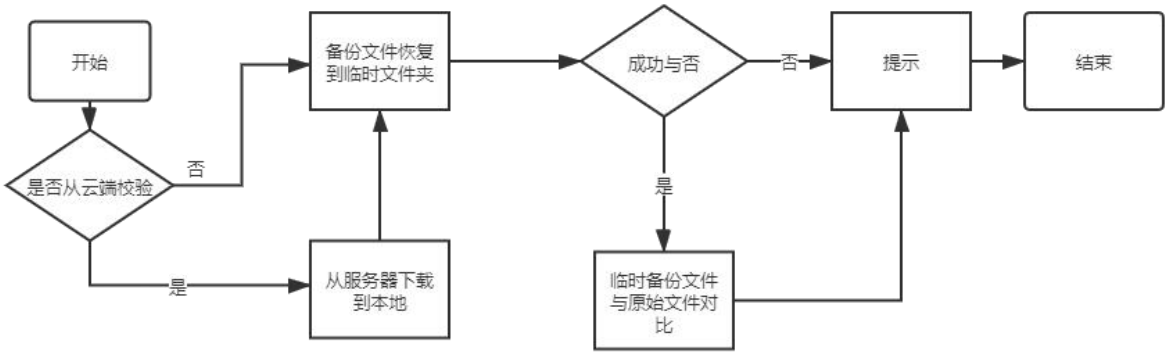


时序图：

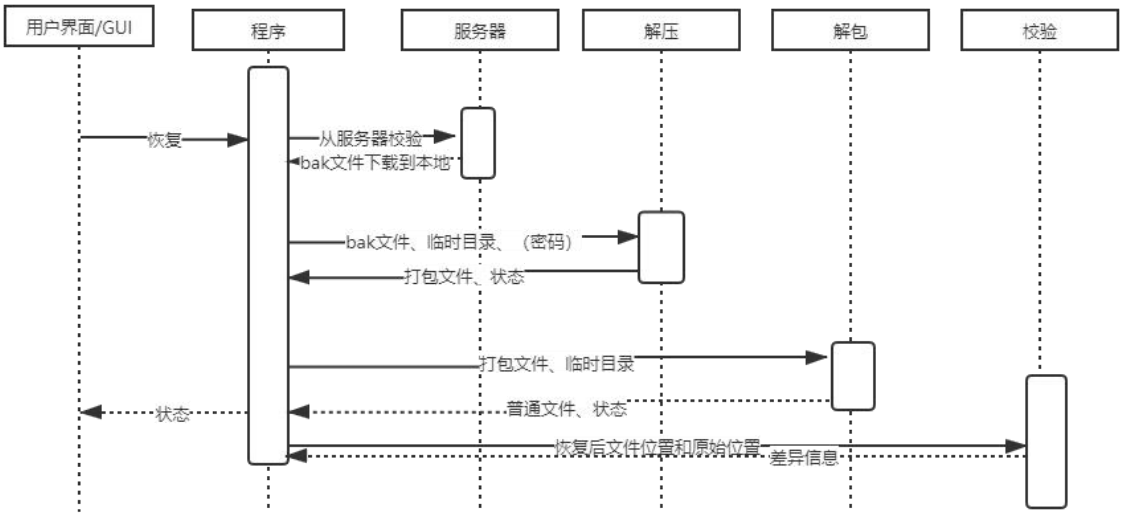


4.2.3 校验功能

流程图：



时序图：



4.3 系统功能模块说明

4.3.1 备份

功能：讲所选文件/目录打包压缩为一个压缩文件后进行备份，可以选择是否设置密码、是否备份到服务器
使用说明：同时备份多个文件时，所选文件/目录需要在同一目录下，且文件要求英文/数字作为名称

4.3.2 恢复

功能：将本地/服务器上的的备份文件解压、解包、解密（可能无）恢复为原来的文件和目录，各项属性保持不变
使用说明：服务器上的备份文件会先下载到本地后再恢复

4.3.3 服务器

功能：实现对接口的监听，支持查询服务器文件列表，增删查改备份文件

4.3.4 日志

功能：记录对文件进行的备份操作（服务器/本地）
使用说明

4.3.5 校验

功能：可以将备份的文件（服务器/本地）与原文件进行对比，查看文件是否变更。
使用说明：无

4.4 程序功能模块图

4.4.1package 类

将文件/目录打包为 tar 型的打包文件

4.4.2unpackage 类

将 tar 型的打包文件解包为文件/目录

4.4.3compressor 类

将 tar 类型的打包文件压缩为 bak 类型的压缩文件，使用 huffman 算法，可以使用 md5 算法进行加密

4.4.4decompressor 类

将 bak 类型的压缩文件使用 huffman 算法解压为 tar 类型的打包文件，如果是加密文件需要提供密码（md5 值）来解密（并非严格意义上的解密）

4.4.5md5 类

md5 加密算法，用来给压缩文件加密，md5 值填入压缩文件头部

4.4.6task 类

为 taskmanager 类做准备工作，构建相关数据结构和函数如 getFiles 和 getBackupFilename 等

4.4.7taskmanager 类

日志管理器，负责日志的初始化、增加、删除、清空

4.4.8check 类

负责对文件（本地文件及服务器端文件）进行校验

4.4.9backupscheme 类

二级窗口，备选方案，包括上传服务器和加密操作

4.4.10widget 类

主窗口，主要的 GUI 界面

4.4.11utils 类

工具类，转化文件时间格式，便于写入文件后读出

4.5 各个类的数据结构（主要）

4.5.1package 类

```
    int relativePathLength;//文件相对路径长度
    int fileLength;//文件长度
    bool isFileDictionary;//是否为文件
    FILETIME fileCreatedTime; //文件创建时间
    DWORD fileCreatedTime_dwLowDateTime;
    //存储文件创建时间 FILETIME，低 32bit
    DWORD fileCreatedTime_dwHighDateTime;
    //存储文件创建时间 FILETIME，高 32bit
    FILETIME fileAccessedTime; //文件访问时间
    DWORD fileAccessedTime_dwLowDateTime;
    //存储文件访问时间 FILETIME，低 32bit
    DWORD fileAccessedTime_dwHighDateTime;
    //存储文件访问时间 FILETIME，高 32bit
    FILETIME fileWrittenTime; //文件修改时间
    DWORD fileWrittenTime_dwLowDateTime;
    //存储文件修改时间 FILETIME，低 32bit
    DWORD fileWrittenTime_dwHighDateTime;
    //存储文件修改时间 FILETIME，高 32bit
```

4.5.2unpackage 类

```
    int relativePathLength;//文件相对路径长度
    int fileLength;//文件长度
    bool isFileDictionary;//是否为文件
    FILETIME fileCreatedTime; //文件创建时间
    DWORD fileCreatedTime_dwLowDateTime;
    //存储文件创建时间 FILETIME，低 32bit
    DWORD fileCreatedTime_dwHighDateTime;
    //存储文件创建时间 FILETIME，高 32bit
    FILETIME fileAccessedTime; //文件访问时间
    DWORD fileAccessedTime_dwLowDateTime;
    //存储文件访问时间 FILETIME，低 32bit
    DWORD fileAccessedTime_dwHighDateTime;
    //存储文件访问时间 FILETIME，高 32bit
    FILETIME fileWrittenTime; //文件修改时间
    DWORD fileWrittenTime_dwLowDateTime;
    //存储文件修改时间 FILETIME，低 32bit
    DWORD fileWrittenTime_dwHighDateTime;
    //存储文件修改时间 FILETIME，高 32bit
```

4.5.3compressor 类

```
    map<unsigned char, string> codeMap;//编码哈希桶
    struct haffNode{
        unsigned long long freq;//待编码字符频率
        unsigned char uchar;//待编码字符
        string code;//编码后的二进制串
        struct haffNode* left = 0;//哈夫曼节点左孩子
        struct haffNode* right = 0;//哈夫曼节店右孩子
    };
    struct Compare{
        bool operator ()(const haffNode* a, const haffNode* b) {
            return a->freq > b->freq;
        }};//仿函数实现 operator 来对公共代码进行复用
```

4.5.4decompressor 类

```
map<unsigned char, string> codeMap;//编码哈希桶
    struct haffNode{
        unsigned long long freq;//待编码字符频率
        unsigned char uchar;//待编码字符
        string code;//编码后的二进制串
        struct haffNode* left = 0;//哈夫曼节点左孩子
        struct haffNode* right = 0;//哈夫曼节店右孩子
    };
    struct Compare{
        bool operator ()(const haffNode* a, const haffNode* b) {
            return a->freq > b->freq;
        }};//仿函数实现 operator 来对公共代码进行复用
```

4.5.5task 类

```
    QList<QString> files; //文件列表
    QString backupFilename; //备份文件名
    QString password; //密码
    bool cloud; //是否云上传
    QDateTime currentTime; //当前时间
    bool operator == (const Task& t) const {
        return files == t.files &&
                backupFilename == t.backupFilename &&
                password == t.password&&
                cloud == t.cloud &&
                currentTime == t.currentTime;
    }//重载==运算符简化操作
```

4.5.6taskmanager 类

```
QList<Task> taskList; //任务列表
QJsonDocument config; //构建 QjsonDocument 对象来对 json 文件操作
```

4.5.7check 类

无

4.5.8backupscheme 类

```
namespace Ui {
class BackupScheme;
}二级窗口的 UI 界面命名空间
Q_OBJECT
Ui::BackupScheme *ui;
```

4.5.9widget 类

```
const QString api = "http://120.55.96.183:5000/";
服务器的 ip 地址的 5000 端口
namespace Ui { class Widget; }
QT_END_NAMESPACE
主窗口的 UI 界面，命名空间
Q_OBJECT
Ui::Widget *ui;//主窗口 ui
BackupScheme  *backupSchemeDialog;//二级窗口 ui
QPoint m_WindowsPos;//主窗口位置坐标类
QPoint m_MousePos; //鼠标坐标类
QPoint m_dPos;//鼠标点击坐标类
TaskManager *taskManager;//任务管理器
QMenu* popMenu;//弹出菜单
QAction* openFolder;//浏览文件的动作
QAction* removeCloudBackupFile;//移除云端备份文件的动作
QAction* check;//校验的动作
QString password;//密码
bool cloudCheckFlag;//是否云端备份的标志
bool passwordCheckFlag;//是否填写密码的标志
QTimer *timer;//定时器
```

4.6 各个类的接口

4.6.1package 类

```
int package(QStringList files, QString destination)
package 函数用来将传入的 files 中的文件或目录打包成 tar 文件，写道 destination 这个文件中，成功返回 0，失败返回对应的错误码
```

```
LPCWSTR MultiCharToUniChar(char* mbString);

DWORD ShowFileTime(PFILETIME lptime);
```
显示文件时间
```
DWORD ShowFileAttributes(LPSTR szPath);
```
显示文件属性

4.6.2unpackage 类

int **unPackage**(QString tarFilename, QString destination)
该函数用来将传入的 tarfilename 这个文件恢复成原来的文件和目录树，写道 destination 这个文件夹中，成功返回 0，失败返回对应的错误码
```
DWORD ShowFileAttributes(LPSTR szPath);
```
显示文件属性
```
DWORD ShowFileTime(PFILETIME lptime);
```
显示文件时间

4.6.3compressor 类

void **encode**(haffNode* pn, string code)
遍历 huffman 树，为 huffman 树生成 huffman 编码。
int **compress**(string sourcePath, string destinationPath, string pw)
将打包文件压缩为 destinationPath 下的 bak 文件，可以用 pw 进行加密，成功则返回 0，失败则返回对应的错误码

4.6.4decompressor 类

int **decompress**(string sourcePath, string destinationPath, string pw)
将传入的压缩文件通过 huffman 算法解压为打包文件，写到 destinationPath 下的文件夹中，用 pw 进行解密，成功则返回 0，失败则返回对应的错误码。

4.6.5task 类

```
Task(QList<QString> _files, QString _backupFilename,
QString_password, bool _cloud, QDateTime _currentTime);
```
初始化任务列表，含参数（文件、备份文件、密码、是否上传云端、当前时间）
```
    const QList<QString> &getFiles() const;
```
获取文件
```
    const QString &getBackupFilename() const;
```
获取备份文件名
```
    const QString &getPassword() const;
```
获取密码
```
    bool getCloud() const;
```
是否上传云端
```
    const QDateTime &getCurrentTime() const;
```
获取当前时间

4.6.6taskmanager 类

```
  void init();
初始化（不含参数）
  void addTask(Task task);
增加任务（日志）
  void removeTask(int index);
移除任务（日志）
  void clear();
清空任务（日志）
  const QList<Task>& getTaskList();
获取日志列表
  void writeJson();
写 json 文件（存储日志信息）
```

4.6.7check 类

```
static QByteArray getMD5ByFilename(QString filename)
通过文件名获取 md5 码
static QVector<QPair<QString, int>> check(QList<QString> files, QString directory)
实现检验文件功能
```

4.6.8backupscheme 类

```
explicit BackupScheme(QWidget *parent = nullptr);
显示备选方案（二级窗口）的 UI
void sendData(QString password,bool cloudCheckFlag,bool passwordCheckFlag);
信号函数，发送 DATA
void PasswordCheckBox_stateChanged();
槽函数，是否选中候选框
void GetData();
从 ui 处获取 DATA
```

4.6.9widget 类

```
Widget(QWidget *parent = nullptr);
显示主窗口
~Widget();
```

析构函数
```
void mousePressEvent(QMouseEvent *event) override;
```
鼠标点击事件
```
void mouseMoveEvent(QMouseEvent *event) override;
```
鼠标移动事件
```
void ShowImg(QLabel *pLabel, QString path);
```
显示图片
```
void SetBtnImage(QPushButton *pBtn, QString path);
```
设置按钮图片
```
void UpdateStackedWidget(int index);
```
更新 stackwidget
```
void UiInit();
```
UI 初始化
```
void MainConnect();
```
首页信号槽函数连接
```
void Page1Connet();
```
page1 信号槽函数连接
```
void Page2Connet();
```
page2 信号槽函数连接
```
void Page3Connet();
```
page2 信号槽函数连接
```
void updateTaskList();
```
更新任务列表
```
void updateCloudFileList()
```
更新云端文件列表

4.6.10utils 类

static char* **dwordToCharArray**(DWORD dwTime);
DWORD 类型转化为 char* 类型，写入文件时候，以 char*的方式写 DWORD 类型的时间。
static DWORD **charArrayToDword**(char* out);
char*类型转化为 DWORD 类型，以 char*读出文件后，转化成 DWORD 来给文件时间的 DWORD 类型进行赋值