

Numerical Method and Simulation Analysis

Final Presentation

Files Transition Platform with Encryption

Feng-Yu Wu, Pei-Hsuan Hsu

目錄

一、 系統架構	1
二、 執行流程	2
1. 產生私鑰及公鑰，並儲存成.PEM 檔（沛萱）	2
2. 加密檔案，並輸出.ZIP 檔（沛萱）	4
3. 傳送檔案	6
4. 解密檔案（豐育）	6

圖目錄

FIGURE 1 系統架構示意圖	1
FIGURE 2 產生公私鑰流程圖	2
FIGURE 3 加密流程示意圖	4
FIGURE 4 解壓縮流程圖(方法一)	6
FIGURE 5 解壓縮流程圖(方法二)	7

表目錄

TABLE 1 MANAGEMENT.GO 函數輸出輸入格式整理	3
TABLE 2 ENCRYPT.GO 函數輸出輸入格式整理	5
TABLE 3 DECRYPT.GO 函數輸出輸入格式整理	11

一、系統架構

(位置：\Files_Transition_Platform)

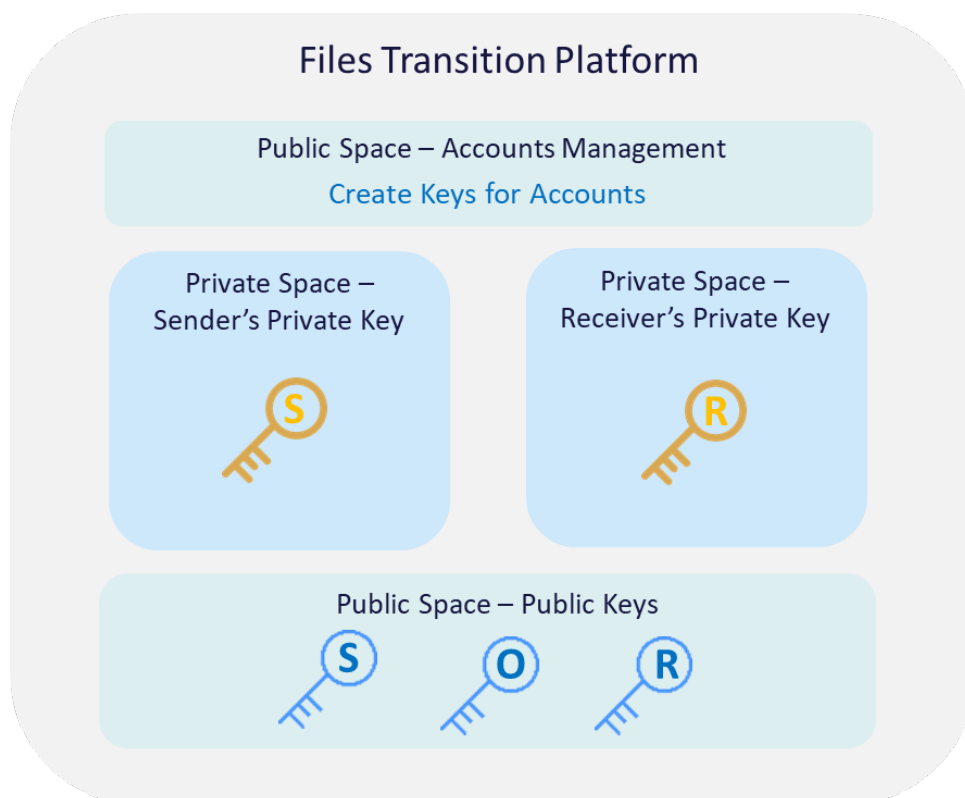


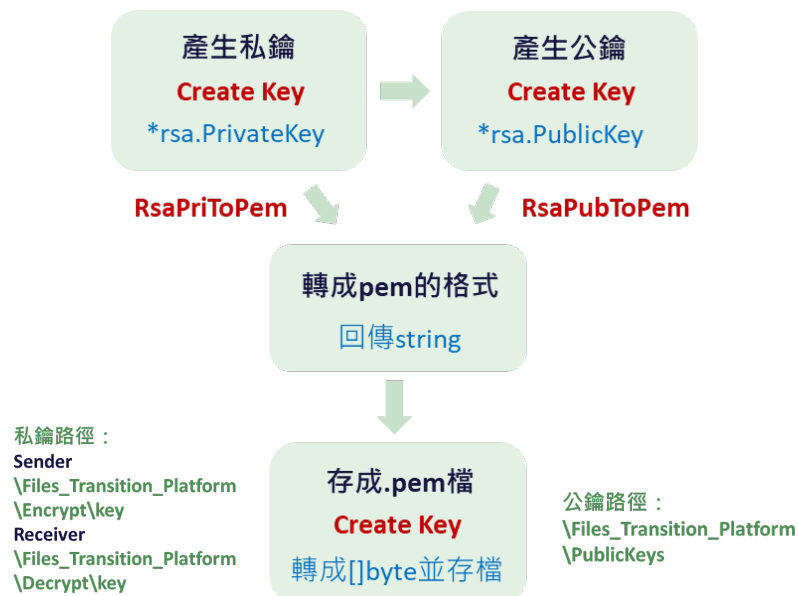
Figure 1 系統架構示意圖

 : 私鑰  : 公鑰  : 其他公鑰

二、執行流程

1. 產生私鑰及公鑰，並儲存成.pem 檔（沛萱）

(詳第三章 1. management.go 路徑：\Files_Transition_Platform\Account_Managment\management.go)



註：私鑰存至私人區域（\Encrypt\key or \Decrypt\key），公鑰存至公開區域(\PublicKeys)

Figure 2 產生公、私鑰流程圖

CreateKey：

```
func CreateKey(who, folder string, size int) {
    newPrivkey, err := rsa.GenerateKey(rand.Reader, size)
    Check(err)

    // create pem file
    PemRrikey := []byte(RsaPriToPem(newPrivkey))

    newPubkey := &newPrivkey.PublicKey
    PemPubkey := []byte(RsaPubToPem(newPubkey))

    err = ioutil.WriteFile("../"+folder+"/key/"+who+"pri_key.pem", PemRrikey, 0644)

    if err != nil {
        fmt.Printf("Error creating Key file!")
        os.Exit(0)
    } else {
        err = ioutil.WriteFile("../PublicKeys/"+who+"pub_key.pem", PemPubkey, 0644)
    }
}
```

RsaPriToPem :

```
func RsaPriToPem (rsaPrivkey *rsa.PrivateKey) string {
    privkey_bytes := x509.MarshalPKCS1PrivateKey(rsaPrivkey)

    privkey_pem := pem.EncodeToMemory(
        &pem.Block{
            Type: "RSA PRIVATE KEY",
            Bytes: privkey_bytes,
        },
    )
    return string(privkey_pem)
}
```

RsaPubToPem :

```
func RsaPubToPem (rsaPubkey *rsa.PublicKey) string {
    pubkey_bytes, err := x509.MarshalPKIXPublicKey(rsaPubkey)
    Check(err)

    pubkey_pem := pem.EncodeToMemory(
        &pem.Block{
            Type: "RSA PUBLIC KEY",
            Bytes: pubkey_bytes,
        },
    )

    return string(pubkey_pem)
}
```

以上轉換格式程式參考網址：

<https://stackoverflow.com/questions/13555085/save-and-load-crypto-rsa-privatekey-to-and-from-the-disk>

Table 1 management.go 函數輸出輸入格式整理

函數名稱	輸入格式	輸出格式/動作
RsaPriToPem	*rsa.PrivateKey	string
CreateKey	*rsa.PublicKey	string
CreateKey	string	存出.pem檔案

2. 加密檔案，並輸出.zip 檔（沛萱）

(詳第三章 2. encrypt.go 路徑：`\Files_Transition_Platform\Encrypt\encrypt.go`)

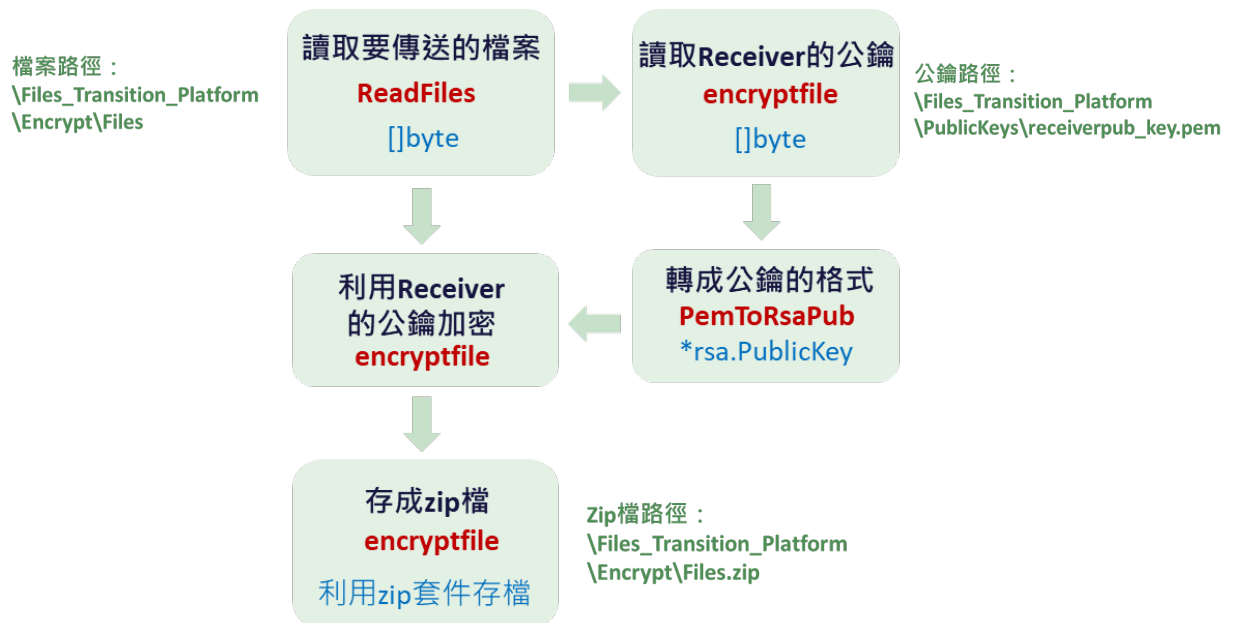


Figure 3 加密流程示意圖

encryptfile：

```
// Encrypt Files and Compress
func encryptfile(folder string) {

    // Read files
    FileData, FileName := ReadFiles(folder)

    // encrypt and zip files
    fzip, _ := os.Create(folder+"(encrypt).zip")
    w := zip.NewWriter(fzip)
    defer w.Close()
    for i, filedata := range FileData {
        fw, _ := w.Create(FileName[i])

        // encrypt file
        // hash_data := sha256.Sum256(filedata)
        // https://ithelp.ithome.com.tw/articles/10188698
        encryptedmsg, err := rsa.EncryptPKCS1v15(rand.Reader, ReceiverPubkey, filedata[:])
        Check(err)

        _, err = fw.Write(encryptedmsg)
        Check(err)
        // fmt.Println(n)
    }
}
```

ReadFiles :

```
// Read Files
func ReadFiles(folder string) ([][]byte, []string) {

    dir := folder+"/"
    files, err := ioutil.ReadDir(dir)
    fmt.Println(files, "\nlen=", len(files))

    Check(err)

    FileData := [][]byte{}
    FileName := []string{}
    for _, file := range files {
        filecontent, err := ioutil.ReadFile(dir + file.Name())
        Check(err)

        FileData = append(FileData, filecontent)
        FileName = append(FileName, "Crypt_" + file.Name())
    }
    return FileData, FileName
}
```

PemToRsaPub :

```
func PemToRsaPub (pubPEM string) (*rsa.PublicKey, error) {
    block, _ := pem.Decode([]byte(pubPEM))
    if block == nil {
        return nil, errors.New("failed to parse PEM block containing the key")
    }

    pub, err := x509.ParsePKIXPublicKey(block.Bytes)
    if err != nil {
        return nil, err
    }

    fmt.Println("pub=", pub)

    switch pub := pub.(type) {
    case *rsa.PublicKey:
        return pub, nil
    default:
        break // fall through
    }

    return nil, errors.New("Key type is not RSA")
}
```

Table 2 encrypt.go 函數輸出輸入格式整理

函數名稱	輸入格式	輸出格式/動作
PemToRsaPub	string (讀取後的.pem檔)	*rsa.PublicKey
rsa.EncryptPKCS1v15	*rsa.PublicKey	傳出加密檔案

3. 傳送檔案

利用 USB 將檔案(\Files_Transition_Platform\Encrypt\Files.zip)傳送至
\Files_Transition_Platform\Decrypt。

4. 解密檔案 (豐育)

(詳第三章 3. decrypt.go，檔案路徑：" \Files_Transition_Platform\Decrypt\decrypt.go")

方法一：

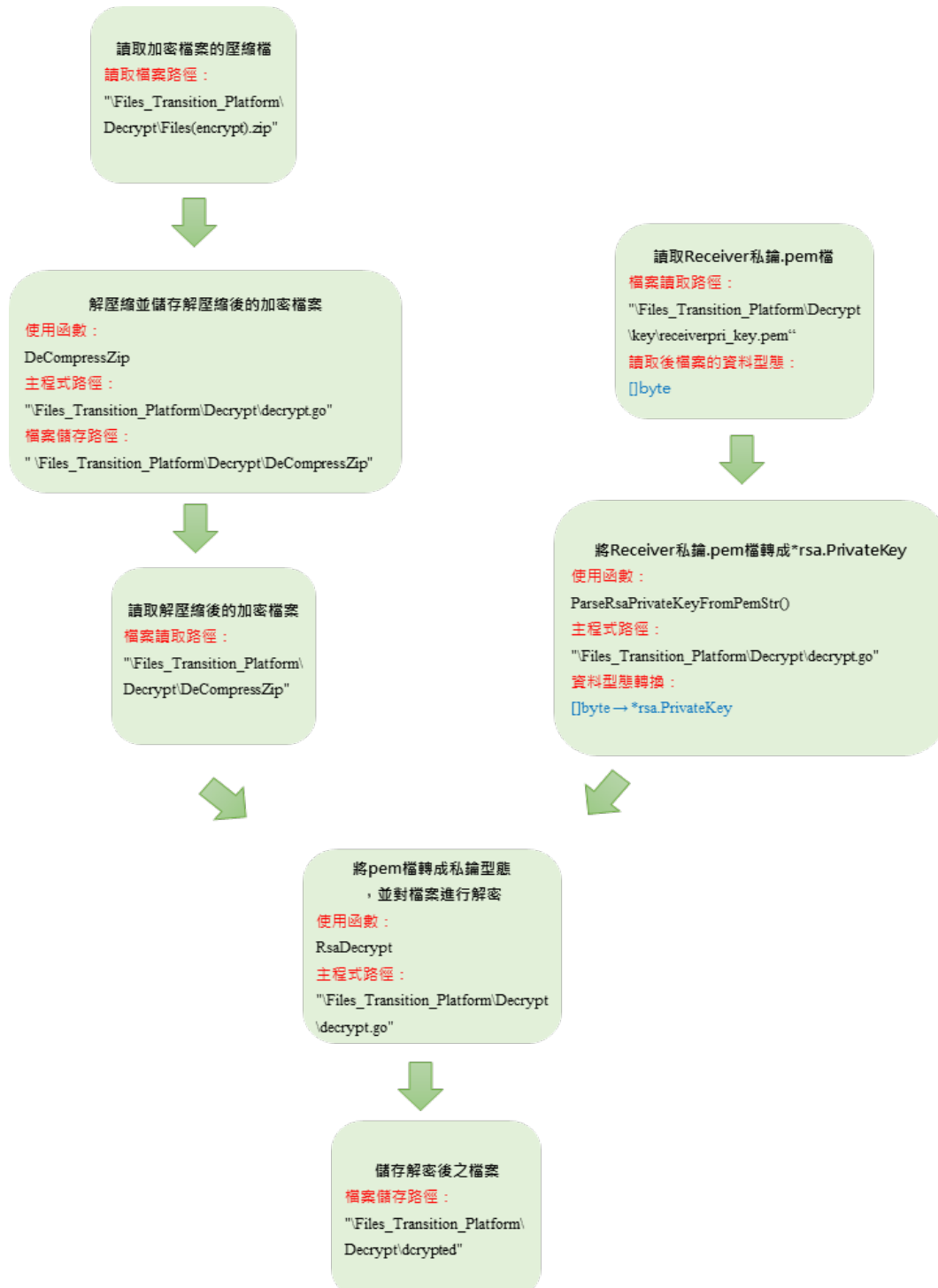


Figure 4 解壓縮流程圖(方法一)

方法二：



Figure 5 解壓縮流程圖(方法二)

a. 解壓縮

```
func DeCompressZip(File, dir string) {  
  
    os.Mkdir(dir, 0777) //建立一個目錄〈資料夾〉  
  
    cf, err := zip.OpenReader(File) //讀取zip檔案  
    if err != nil {  
        fmt.Println("Func DeCompressZip rader")  
        fmt.Println(err)  
    }  
    defer cf.Close()  
    for _, file := range cf.File {  
        rc, err := file.Open() //開檔  
        if err != nil {  
            fmt.Println("Func DeCompressZip open")  
            fmt.Println(err)  
        }  
  
        f, err := os.Create(dir + "/" + file.Name) //開檔案位置  
        if err != nil {  
            fmt.Println("Func DeCompressZip os.Create")  
            fmt.Println(err)  
        }  
        defer f.Close() //關檔  
        n, err := io.Copy(f, rc) //覆寫  
        if err != nil {  
            fmt.Println("Func DeCompressZip io.Copy")  
            fmt.Println(err)  
        }  
        fmt.Println(n)  
    }  
}
```

由網路上找到的解壓縮方法，可以將檔案解壓縮後儲存至指定的 dir 路徑，若無相對應資料夾，則透過 os.Mkdir() 函數會自動生成新資料夾。

參考資料：example:<https://www.itread01.com/content/1546726868.html>

b. 讀取解壓縮後檔案

```
list, err := ioutil.ReadDir(path) //ioutil.ReadDir讀資料夾、ioutil.ReadFile讀單一檔案；用list叫出資料夾的基本性質  
if err != nil {  
    //https://blog.csdn.net/yxys01/article/details/78136295  
    fmt.Println("Read DeCompressZip")  
    panic(err)  
}  
  
for _, files := range list {  
    fmt.Println(files.Name())  
    encrypted_file, err := ioutil.ReadFile(path + "/" + files.Name()) //確認讀入檔名 //用.Name()去抓出資料夾內的檔案的名稱並讀取  
    if err != nil {  
        fmt.Println("read dir error")  
        return  
    }  
    fmt.Printf("%T\n", encrypted_file) //確認讀入內容
```

ioutil.ReadDir 讀取資料夾、ioutil.ReadFile 讀取單一檔案，這裡使用 ioutil.ReadDir 並搭配.Name()函數，叫出所有檔案之檔名，並以檔案名稱呼叫資料。

c. 將解壓縮後檔案解密

- 方法一：讀取私鑰檔案之後，再使用 `rsa.DecryptPKCS1v15()` 函數去將檔案解密

```
func ParseRsaPrivateKeyFromPemStr(privPEM string) (*rsa.PrivateKey, error) {
    block, _ := pem.Decode([]byte(privPEM))
    if block == nil {
        fmt.Println("Func ParseRsaPrivateKeyFromPemStr Decode")
        return nil, errors.New("failed to parse PEM block containing the key")
    }

    priv, err := x509.ParsePKCS1PrivateKey(block.Bytes)
    if err != nil {
        fmt.Println("Func ParseRsaPrivateKeyFromPemStr x509.ParsePKCS1PrivateKey")
        return nil, err
    }

    return priv, nil
}
```

使用 `ParseRsaPrivateKeyFromPemStr()` 函數可將我們預存在電腦中的私鑰的 pem 檔讀取，並由 `[]byte` 轉換成為 `rsa.PrivateKey` 的資料型態，如此才可以利用 `rsa.DecryptPKCS1v15()` 函數去解密資料。

參考資料：

<https://stackoverflow.com/questions/13555085/save-and-load-crypto-rsa-privatekey-to-and-from-the-disk>

- 方法二：在讀取私鑰檔案之後，直接使用 `rsa.DecryptPKCS1v15()` 函數去將檔案解密，並將整個流程寫成函數。

```
func RsaDecrypt(ciphertext []byte) ([]byte, error) {
    block, _ := pem.Decode([]byte(privateKey))
    if block == nil {
        fmt.Println("Func RsaDecrypt Decode")
        return nil, errors.New("private key error!")
    }
    priv, err := x509.ParsePKCS1PrivateKey(block.Bytes)
    fmt.Println(priv)
    if err != nil {
        fmt.Println("Func RsaDecrypt x509.ParsePKCS1PrivateKey")
        return nil, err
    }
    return rsa.DecryptPKCS1v15(rand.Reader, priv, ciphertext)
}
```

參考資料：

<http://blog.studygolang.com/2013/01/go%E5%8A%A0%E5%AF%86%E8%A7%A3%E5%AF%86%E4%B9%8Brsa/>

d. 儲存解密後之檔案

```
fw, _ := os.Create(depath + "/" + "de" + files.Name()) //開檔案位置
n, err := fw.Write(TEStA)                               //將內容寫入檔案
if err != nil {
    fmt.Println("fw.Write error")
    fmt.Println(err)
}
defer fw.Close()
fmt.Println(n)
```

使用 `os.Create()`、`fw.Write()` 將資料寫入指定位置的檔案。

Table 3 decrypt.go 函數輸出輸入格式整理

函數名稱	輸入格式	輸出格式/動作
<code>ParseRsaPrivateKeyFromPemStr</code>	string (讀取後的.pem檔)	*rsa. PrivateKey
<code>rsa.DecryptPKCS1v15</code>	*rsa. PrivateKey	將檔案解密