中山大学移动信息工程学院移动网络安全实验报告
(2016学年春季学期)

刘爽
班级：13M2
ID:13354222
专业：移动互联网

2016年4月8日

1 实验题目

A Password Manager

1.1 实验要求

设计一个Password Manager系统，实现如下功能：

- **用户登陆功能**

提示用户输入用户名和Master Password。具体做到：

I. 检查文档是否存在，如果不存在创建一个新的文档，并以输入的Master Password作为这个用户的Master Password。

II. 如果存在则检查文档中的Master Password（需要先进行解密）和用户输入的是否一致，如果不一致则报错，保留在登录界面。

III. 如果一致则进行完整性检查：将每一行的网站域名抽出来通过HMAC做哈希，将哈希以后的值和保存的对应哈希值最对比，如果有某一项不一致则报错，保留在登陆界面；如果一致就进入用户功能界面。

- **数据增加功能**

实现对于网站域名 + 哈希值 + 加密以后的密码的增加。具体做到：

I. 提示用户输入网站域名和密码。

II. 检查文档中是否有相同的域名，如果有则报错；没有则将网站域名 + 通过HMAC算出的哈希值 + 加密以后的密码增加到文件尾部，并输出成功信息。

- **数据删除功能**

实现对于网站域名 + 哈希值 + 加密以后的密码的删除。具体包括：

I. 提示用户输入网站域名。

II. 检查文档中是否有相同的域名，如果没有则报错；有则将对应的记录从该文档中删除。

- **数据查询功能**

实现对于输入网站域名后对于密码的查询。具体做到：

I. 提示用户输入网站域名。

II. 检查文档中是否有相同的域名，如果没有则报错；有则将对应加密后的密码解密后输出出来。

- **数据更改功能**

实现对于输入网站域名后对于密码的修改。具体做到：

I. 提示用户输入网站域名和密码。

II. 检查文档中是否有相同的域名，如果没有则报错；有则将输入的密码进行加密后替换掉文档中的对应项。

- **返回登陆界面功能**

1.2 实验目的

学习如何使用第三方开源库crypto++。
理解PBDKF2算法，并调用相应方法。
理解HMAC映射方法，并调用相应方法。
理解AES算法，并调用相应方法。

2 实验原理

总的来说，Password Manager会将用户的密码储存在硬盘上，由相应的Master Password保护。另外，当Password Manager被使用的时候，它会储存Domain Name的明文形式使得用户可以得到他想要的密码。Password Manager 以Key-Value(KVS)的形式将数据储存在硬盘上，如Fig.1所示。一般来说，将信息直接储存在硬盘上是不安全的，Password Manager用KVS进行储存的形式确保了数据的私密性和完整性。

Key	Value
www.google.com	password
www.example.com	123456
www.amazon.com	6U)qAlOBy%3SZX$o
www.ebay.com	guest

Figure 1: Key Value Pair

我们不想泄露有关用户存储的域名的任何信息，同时需要有能够查询到具体域名对应的密码的功能。因而，当一个KVS被存储时，KVS应该包含所存域名的对应的HMAC值。这样在查询相对应的域名的同时，我们将用户查询的域名与相应的HMAC进行比对便可以知道数据是否被进行篡改。

Password Manager自身被由Master Password产生的Key保护着。当有新用户使用Password Manager时，他会被要求提供Master Password。这个Master Password将被用来产生可用于HMAC和AES所使用的Key。一般来说，这个Key的生成需要使用PBKDF2方法来是Key达到要求的长度。如Fig.2所示。

由于可能会有多个用户(一个用户对应一个Master Password，因此会有多个Master Password)，这里采用的方法是每个用户一个txt文件，文件名为用户名，文件中第一行为加密以后的用户的Master Password，文件中一行一组数据，每组数据需要保存网站域名 + 哈希值 + 加密以后的密码。

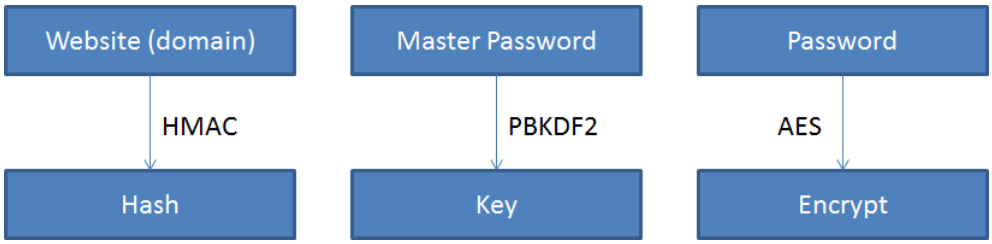

Figure 2: Framework of a Password Manager

3 实验内容

分析程序的整个实现过程，讲述每个函数的具体作用并附上核心代码部分。
整个工程包括三个文件，分别是keychain.cpp, keychain.h和main.cpp。

3.1 keychain.cpp和keychain.h

keychain.h定义了对数据的具体操作，如数据的插入，插入，查询和修改等，以及一些必要的辅助函数，如将字符串与十六进制数之间的相互转换等。而keychain.cpp实现了对数据的具体操作。

3.1.1 insert(string domain_name， string password)

-首先检查用户所输入的Domain Name是否存在；
-打开文件，用HMAC函数对Domain Name进行哈希，用AES对密码进行加密；
-将Domain Name，Hash后Domain Name的值以及加密后的Password以空格分离的方式添加到文件中。如Fig3所示。

```
ofstream f;
f.open(cur_user.c_str(),ofstream::app);
if(f.is_open()){
    string domName, hashDomName, decry_password;
    domName = domain_name;
    hashDomName = HMAC_Encryption(domain_name, decry_mas_pas);//map the domain name
    decry_password = AES_Encryption(password, decry_mas_pas); //encrypt the password
    f << domName << ' ' << hashDomName << ' ' << decry_password << endl;
    f.close();
    return true;
}
```

Figure 3: Core Codes of Insert Function

3.1.2 remove(string domain_name)

-首先检查用户所输入的Domain Name是否存在；
-打开用户文件，并创建一个临时文件temp；

-查找用户文件中的每一行，看是否为目标Domain Name，若不是则将此行输出至文件temp中。若找到目标Domain Name则不做操作;
-将用户文件删除，将temp文件改名为用户文件的名称。
因为C++中不允许直接对进行文件进行删除数据操作，只能用储存了删除过数据的文件来替换原文件达到删除数据的目的。如Fig.4 所示。

```
ifstream f(cur_user.c_str());
ofstream fn("temp");
std::string line;
while (std::getline(f, line)){
    std::istringstream iss(line);
    string domName, hashName, encryPas;
    if (iss >> domName >> hashName >> encryPas) {
        if(domName == domain_name){
            //do nothing to delete the data
        }
        else{
            fn << domName << ' ' << hashName << ' ' << encryPas << endl;
        }
    }
}
f.close();
fn.close();
//replace the old file with a new one without target domain name
remove(cur_user.c_str());
rename("temp", cur_user.c_str());
```

Figure 4: Core Codes of Remove Function

3.1.3 get(string domain_name, string& paswd)

-首先检查用户所输入的Domain Name是否存在；
-打开用户文件，查找目标Domain Name，若找到Domain Name，用AES_Decryption函数将对应的密码进行解密，并将解密后的结果赋值给paswd。如Fig.5所示。

```
fstream f(cur_user.c_str(), ofstream::in);
std::string line;
while (std::getline(f, line)){
    std::istringstream iss(line);
    string domName, hashName, encryPas;
    if (iss >> domName >> hashName >> encryPas) {
        if(domName == domain_name){
            //decrypt the encrypted password
            paswd = AES_Decryption(encryPas, decry_mas_pas);
            return true;
        }
    }
}
```

Figure 5: Core Codes of Get Function

3.1.4 update(string domain_name, string password)

-首先检查用户所输入的Domain Name是否存在；
-查找用户文件中的每一行，看是否为目标Domain Name，若不是则将此行直接输出至文件temp中。若找到目标Domain Name则将传进来的新的Password进行AES加密并替换原有的加密过后的password;
-将用户文件删除，将temp文件改名为用户文件的名称。
因为C++中不允许直接对进行文件进行修改数据操作，只能用储存了修改过数据的文件来替换原文件达到修改数据的目的。如Fig.6所示。

```
fstream f(cur_user.c_str(), ofstream::in);
ofstream fn("temp");
std::string line;
while (std::getline(f, line)){
    std::istringstream iss(line);
    string domName, hashName, encryPas;
    if (iss >> domName >> hashName >> encryPas) {
        if(domName == domain_name){
            fn << domName << ' ' << hashName << ' '
               << AES_Encryption(password, decry_mas_pas) << endl;
        }
        else{
            fn << domName << ' ' << hashName << ' '
               << encryPas << endl;
        }
    }
}
f.close();
fn.close();
```

Figure 6: Core Codes of Update Function

3.1.5 addUser(string username, string password)

-创建用户文件，名称为username；
-将Master Password用PBDKF2进行加密；
-将加密后的文件输入至文件中去。如Fig.7所示。

```
cur_user = username;
fstream f(username.c_str(), ofstream::out);
if(f.is_open()){
    f << "Master Password: ";
    //get the master password encrypted
    string result = PBFDK2_Encryption(master_password);
    decry_mas_pas = result;
    f << result << endl;
    f.close();
}
```

Figure 7: Core Codes of Update Function

3.1.6 check_master_password(string username, string master_password)

-打开用户文件，读取第一行数据；
-将传进来的password用PBDKF2进行加密并与原本储存在文件中的原密码加密后的结果进行对比，如果不完全相等则说明用户输入的密码错误，返回错误。否则就返回正确的值。如Fig.8所示。

```
cur_user = username;
fstream f(username.c_str(), ofstream::in);
std::string line;
string notation1, notation2, mas_paswrd;
if(f.is_open()){
    std::getline(f, line);
    std::istringstream iss(line);
    iss >> notation1 >> notation2 >> mas_paswrd;
    f.close();
}

string decy_pas = PBFDK2_Encryption(master_password);
//check whether the master password is correct or not
if(decy_pas == mas_paswrd){
    decry_mas_pas = decy_pas;
    return true;
}
else
    return false;
```

Figure 8: Core Codes of check_master_password Function

3.1.7 check_for_integrity()

-打开用户文件；
-对文件中的每一行的Domain Name用HMAC函数进行哈希，和原本存储在文件中的Hashed Domain Name进行对比；
-每一行的Domain Name对应的HMAC值都与存储的一样，则说明文件未经篡改。只要有一个不一样，则说明文件得到了篡改。 如Fig.9所示。

3.1.8 if_domain_exist(string domain_name)

-打开用户文件，搜寻每一行；
-如果搜到存储的Domain Name与domain_name相同，说明存在，返回正确的值。否则返回错误的值。

3.2 main.cpp

此文件定义了整个工程的框架，如登陆界面和数据修改界面的进入和跳转等。

```
while (std::getline(f, line))
{
    lineCnt++;
    std::istringstream iss(line);
    if(lineCnt == 1)  continue;
    else{
        string domName, hashName, encryPas;
        if (iss >> domName >> hashName >> encryPas) {
            //check whether the domain name is modified or not
            if(HMAC_Encryption(domName, decry_mas_pas) != hashName){
                f.close();
                return false;
            }
        }
    }
}
f.close();
```

Figure 9: Core Codes of check_for_integrity Function

3.2.1 Login_Signup()

-登陆界面，要求用户输入Username和Master Password；
-检查是否为新用户。若是新用户，则建立一个新的文件；
-若不是新用户，则检查文件的完整性。如Fig.10所示。

```
if(!KC.if_user_exist(username)){
    KC.addUser(username, master_password);
}
else{
    if(!KC.check_master_password(username, master_password)){
        cout << "The master_password is not correct." << endl;
        return false;
    }
    else{
        if(!KC.check_for_integrity()){
            cout << "Warning: The data has been modified!" << endl;
            return false;
        }
        return true;
    }
}
```

Figure 10: Core Codes of Login_Signup Function

3.2.2 data_oper(int choice)

-若传进来的choice为1，进行数据插入操作。要求用户输入要插入的域名和密码，调用keychian类变量中的insert函数；如Fig.11所示。
-若传进来的choice为2，进行数据删除操作。要求用户输入要删除的域名，调

```
if(choice == 1){//do the insertion
    string domain_name;
    string password;
    cout << "Please enter the domain name and the password
            (Enter Key for finishing)." << endl;
    cout << "Domain Name:" << endl;
    cin >> domain_name;
    cout << "Password:" << endl;
    cin >> password;
    if(KC.insert(domain_name, password)){
        cout << "Data Insertion Success." << endl;
    }
    else{
        cout << "Data Insertion Failure." << endl;
    }
}
```

Figure 11: Core Codes of Insert Operation

用keychian类变量中的delete函数；如Fig.12所示。
-若传进来的choice为3，进行数据查询操作。要求用户输入要查询的域名，调

```
string domain_name;
cout << "Please enter the domain name you intend to remove." << endl;
cout << "Domain Name:" << endl;
cin >> domain_name;
if(KC.remove(domain_name)){
    remove(curren_user.c_str());
    rename("temp", curren_user.c_str());
    cout << "Data Deletion Success." << endl;
}
else{
    cout << "Error: Domain Name don't exist!" << endl;
    cout << "Data Deletion Failure." << endl;
}
```

Figure 12: Core Codes of Remove Function

用keychain类变量中的get函数；如Fig.13所示。
-若传进来的choice为4，进行数据修改操作。要求用户输入要域名和替换用的密码，调用keychain类变量中的update函数。如Fig.14所示。

3.2.3 data_manage()

-要求用户输入想要进行的操作指令，1代表插入，2代表删除，3代表查询，4代表修改。并将值直接传入data_oper函数中。

```
string domain_name;
cout << "Please enter the domain name you look for." << endl;
cout << "Domain Name:" << endl;
cin >> domain_name;
string spec_password;
if(KC.get(domain_name, spec_password)){
    cout << "Data Acquition Success." << endl;
    cout << "The corresponding password is:" << spec_password << endl;
}
else{
    cout << "Data Acquition Failure." << endl;
}
```

Figure 13: Core Codes of Get Function

3.2.4 ask_for_login_again()

-询问用户是否原因重新登录。如果用户输入1则代表想重新登陆，返回正确的值。否则返回错误的值。

```
string domain_name;
string password;
cout << "Please enter the domain name and the
         new password you want to replace with." << endl;
cout << "Domain Name:" << endl;
cin >> domain_name;
cout << "Password:" << endl;
cin >> password;
if(KC.update(domain_name, password)){
    remove(curren_user.c_str());
    rename("temp", curren_user.c_str());
    cout << "Data Modification Success." << endl;
}
else{
    cout << "Error: Domain Name don't exist!" << endl;
    cout << "Data Modification Failure." << endl;
}
```

Figure 14: Core Codes of Update Function

3.3 与Cryptopp库相关的函数

在keychain类中有四个private类型的函数，分别是:PBFDK2_Encryption，HMAC_Encryption，AES_Encryption和AES_Decryption。这四个都用到了Cryptopp库中的相关变量类型和函数，来实现加密和解密的目的。

3.3.1 PBFDK2_Encryption(string master_password)函数

此函数用于将用户输入的Master Password转换为可用于HMAC和AES的Key。如Fig.15所示。

```
SecByteBlock derivedkey(AES::DEFAULT_KEYLENGTH);

PKCS5_PBKDF2_HMAC<SHA256> pbkdf;
pbkdf.DeriveKey(
    // buffer that holds the derived key
    derivedkey, derivedkey.size(),
    // purpose byte. unused by this PBKDF implementation.
    0x00,
    // password bytes. careful to be consistent with encoding...
    (byte *) master_password.data(), master_password.size(),
    // salt bytes
    pwsalt, pwsalt.size(),
    // iteration count. See SP 800-132 for details.
    //You want this as large as you can tolerate.
    // make sure to use the same iteration count on both sides...
    iterations
    );
```

Figure 15: Core Codes of PBFDK2 Function

3.3.2 HMAC_Encryption(string pt, string pas)函数

此函数用于将输入的pt以pas为Key进行哈希得到一个映射后的值。如Fig.16所示。

```
string mac, encoded;
encoded.clear();
HMAC< SHA256 > hmac(key, key.size());

StringSource(pt, true,
    new HashFilter(hmac,
        new StringSink(mac)
    ) // HashFilter
); // StringSource
```

Figure 16: Core Codes of HMAC Function

3.3.3 AES_Encryption(string pt, string mas_paswd)函数

此函数用于将输入的pt以pas为Key进行哈希得到一个加密后的值。如Fig.17所示。

```
// Cipher Text Sink
std::string CipherText;

// Encryptor
CryptoPP::ECB_Mode< CryptoPP::AES >::Encryption
//Encryptor( key, sizeof(key), iv );
Encryptor( derivedkey, derivedkey.size());

// Encryption
CryptoPP::StringSource( pt, true,
  new CryptoPP::StreamTransformationFilter( Encryptor,
    new CryptoPP::StringSink( CipherText )
  ) // StreamTransformationFilter
); // StringSource
```

Figure 17: Core Codes of AES Encryption Function

3.3.4 AES_Decryption(string ct, string mas_paswd)函数

此函数用于将输入的ct以pas为Key进行哈希得到一个解密后的值。如Fig.18所示。

```
// Recovered Text Sink
std::string RecoveredText;

// Decryptor
CryptoPP::ECB_Mode< CryptoPP::AES >::Decryption
// Decryptor( key, sizeof(key), iv );
Decryptor( derivedkey, derivedkey.size() );

// Decryption
CryptoPP::StringSource(cipher, true,
  new CryptoPP::StreamTransformationFilter( Decryptor,
    new CryptoPP::StringSink( RecoveredText )
  ) // StreamTransformationFilter
); // StringSource
return RecoveredText;
```

Figure 18: Core Codes of AES Decryption Function

4 实验结果与分析

4.1 登陆功能：

文本验证:

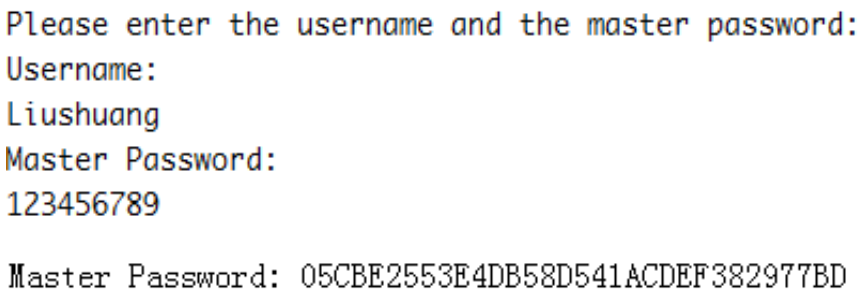

Figure 19: Validation of Login Function

如Fig.19所示，当用户输入一个新的Username时，产生文件，并在第一行保存相应的Master Password进行PBKDF2操作后的十六进制数。

4.2 登陆检查Master Password功能：

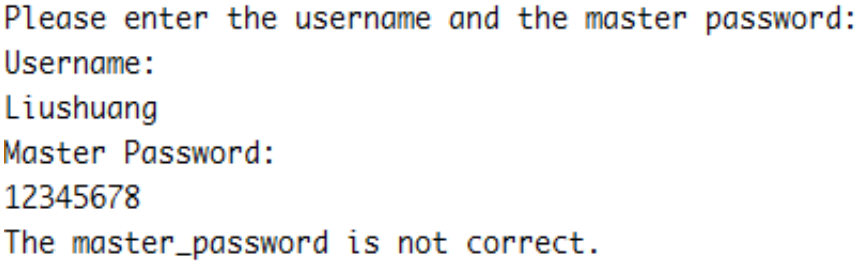

Figure 20: Validation of Check Function

如Fig.20所示，当用户的Master Password与原文件储存的不符时，程序报错。并返回登陆界面。

4.3 数据增加功能：

数据增加的运行结果如Fig.21所示。

文本验证：

如Fig.22所示，当用户输入需要增加的Domain Name和Password之后，文件中相应的增加了Domain Name，Hashed Domain Name和加密后的Password。

4.4 数据删除功能：

结果验证：

如Fig.23所示，当用户再次删除已删除的域名时，程序会报无法找到域名的错误，说明域名已经成功删除。

```
Please enter the domain name and the password(Enter Key for finishing).
Domain Name:
www.google.com
Password:
987654321
Data Insertion Success.
```

Figure 21: Result of Insert Function

```
www.google.com
341BB5021501F06218238FF5BA4EC6537068AF80AAC5522C8E7B5
737B1402992 6FBF3A2666E9EBF10E1A1A682277A5C9
```

Figure 22: Verification of Insert Function

```
Please enter the domain name you intend to remove.
Domain Name:
www.google.com
Data Deletion Success.
```

```
Please enter the domain name you intend to remove.
Domain Name:
www.google.com
Error: Domain Name don't exist!
Data Deletion Failure.
Do you want to alter the file? 1 for Yes, and 0 for No.
```

Figure 23: Verification of Remove Function

4.5 数据查询功能：

如Fig.24所示，当用户查询相应的之间已插入的域名时，程序会返回解密过的相对应域名的密码。

```
Please enter the domain name you look for.
Domain Name:
www.google.com
Data Acquition Success.
The corresponding password is:987654321
```

Figure 24: Verification of Get Function

4.6 数据修改功能：

结果如Fig.25所示。

```
Please enter the domain name and the new password you want to replace with.
Domain Name:
www.google.com
Password:
147258369
Data Modification Success.
```

Figure 25: Result of Update Function

结果验证：

```
Please enter the domain name you look for.
Domain Name:
www.google.com
Data Acquition Success.
The corresponding password is:147258369
```

Figure 26: Verification of Update Function

如Fig.26所示，再次查询同样的域名，返回的解密后的密码是修改过后的密码，说明修改成功。

4.7 域名存在查询功能：

如Fig.27所示，当使用插入功能时，如果域名已经存在，则会报域名重复的错误。

```
Domain Name:
www.google.com
Password:
1597534268
Error: Domain Name Duplicated!
Data Insertion Failure.
```

Figure 27: Verification of Domain Duplication Function

4.8 完整性验证的功能：

将域名由www.google.com改为www.google.com.hk，结果如Fig.28所示。

```
Please enter the username and the master password:
Username:
Liushuang
Master Password:
123456789
Warning: The data has been modified!
```

Figure 28: Verification of Integrity Check Function

5 实验感想

此次实验内容较多，并且较困难，用时较长，有以下感想和收获。

1. 在编写一个工程的时候，思路一定要清晰。在我们做一个类似于系统的工程时，一定要先想好它的功能有些，我们需要什么样的类或者文件，这样可以把一个抽象复杂的问题简化为一个个简单的块。我们接下来需要做的事就是补足每个块的功能而已。

2. 在做一个大的工程文件的时候，最好每写完一个函数或者一个功能就验证一下是否程序是否能够正常工作。这样的话如果出现逻辑上的问题或者编译上的问题可以立即发现。如果等整个程序写完之后再来Debug的话，有可能出问题的不只一个地方，这样效率会大大降低。

3. 若要使用PBKDF2,HMAC或者AES的算法，仅仅Crypto++库的相应的类是不够的，不存在直接传一个字符串就能出结果的情况。因为在大部分情况下，Crypto++库的变量是byte类型的，对于string类型的变量，我们要将其转化为byte类型的数组才能进行处理。另外，得到的结果很大程度上是乱码，因而我们还需要将结果转化为可以正常显示的十六进制数才行。

4. 事实上，Crypto++库比我们想象得更有用。Crypto++库有很多默认的常量或者类可以使我们的计算更加简单，代码更加简洁。比如，在使用PBDK-

F2的时候，结果的长度应该为能够给HMAC和AES做Key的标准长度。而Crypto++库中有一个DEFAULTKEYLENGTH的常量，定义了标准的Key的长度。这使得我们无需考虑究竟要结果长度要有多长，只要是DEFAULTKEYLENGTH那么长就可以了。

5. 在查询来源是国外的任何事物的相关信息时，一般来说用默认英文的搜索引擎所找到的内容会更精准更加符合我们的要求一些。Baidu 的结果经常与搜索的关键词不同，故而有时候搜索起来并没有效率。

6. C++语言本身是不允许直接对文件中的内容进行添加和删除操作的。因而如果我们要对文件中的内容进行修改的话，只能将需要保留的内容保留至一个临时文件中，将原文件删除后，并将临时文件的名称改成原文件的名称来完成操作。

7. Password Manager可以确保数据的Confidentiality和Integrity，但是依然无法抵挡所有的攻击，如DoS Attack。如果攻击者并没有意图得到密码，他的意图只是让用户不能够使用密码，那么他可以任意地修改加密后的内容，这样解密出来的内容就不是正确的密码，则达到了他的目的。

参考文献

[1] http://www.cryptopp.com/wiki/HMAC，Online.

[2] http:// stackoverflow.com/ questions/ 19301100/ how-use-custom-key-in-hmac-sha1-crypto-realisation，Online.

[3] http:// stackoverflow.com/ questions/ 27244449/ crypto-pbkdf2-output-is-different-than-rfc2898derivebytes-c-and-crypto-pbkd，Online.

[4] http://www.cplusplus.com/forum/beginner/60604/，Online.

[5] http:// www.codeproject.com/ Articles/ 21877/ Applied-Crypto-Block-Ciphers, Online.

[6] http://stackoverflow.com/questions/3381614/c-convert-string-to-hexadecimal-and-vice-versa, Online.