Algorithm Design of Graphic Password Authentication System

Peifeng Li

Xuzhou Technician College, Jiangsu Province, Xuzhou 221000, China

Keywords: authentication technology; graphical password; algorithm design

Abstract: With the development of computer application technology and the popularity of computer network applications, people are increasingly relying on computers and network systems to complete important tasks, which makes computer system security issues extremely important. As one of the core components of computer system security, identity authentication technology refers to the technical means used to confirm the operator identity process in computers and network systems. Currently, text-based password authentication is still the most widely used authentication technology because of its simplicity. Graphical password is a new type of password verification technology. It is an alternative solution to the shortcomings of text passwords that are difficult to remember and poorly resistant to cracking.

1. Introduction

According to the characteristics of some existing graphical password authentication systems, a simple graphical password authentication system is designed. At the same time, in order to enhance the security of the system and improve the system's ability to resist online cracking, a multi-factor authentication mechanism is proposed. That is, the combination of graphical password authentication and the currently popular additional code verification. The additional code technology is currently used more commonly. It is a security technology for dealing with passwords online. It is usually a sequence of characters and numbers randomly generated by the server and added with interferon. This will cause the characters to be distorted when displayed to the client. The effect is to make the computer difficult to recognize and easy to recognize.

2. Graphical password authentication based on memory and recognition

The so-called memory and recognition means that the user needs to select some specific pictures in advance for memory training; when performing authentication, the system randomly generates several sets of pictures from the pattern library, and then asks the user to select a preset picture from them, if Accurate identification can be authenticated. The most typical memory and recognition-based authentication schemes are RealUser's Passfaces solution and the Pass-Object solution developed by Sobrado and Birget. When setting the password, the system will first pop up a 3×3 grid interface, in which each grid randomly displays an avatar from a different person in the face database. The user can select and remember 3 images one by one. Password; In the authentication phase, nine new face images are randomly generated in the grid, including a password face and 8 interference faces. The user needs to pick out the face images in the originally set passwords in order. If the face image selected by the user multiple times is identical to the face image originally set, the authentication can be passed. Using the face as a password, the password space is large, and the human brain is good at remembering the facial features of the person, the recognition of the familiar face will be more rapid and accurate, and the authentication time will be greatly shortened. When setting the password, the Pass-Objects program also requires the user to pre-select some graphical objects as passwords. During the verification phase, the system displays an array containing password graphics and many interference patterns. In this graphics array, the user needs to be able to recognize the object in the original set password and move a fixed frame to cover all the objects in the password. This process can be repeated multiple times to prevent the possibility of random selection. If you can cover the password object accurately, you will pass the certification.

3. Encryption algorithm design

3.1 DES basic principles

DES is a symmetric encryption algorithm. The same algorithm and key are used for encryption and decryption, with only minor changes. The initial key is 64 bits, but each eighth bit of the key is discarded before the DES process begins, resulting in a 56-bit key. DES encrypts data by 64-bit block length, that is, 64-bit plaintext is used as the input of DES to generate 64-bit ciphertext output.

3.2 Implementation of the DES encryption algorithm

In order to ensure the security of data transmission, the data transmission between the client and the server in the system uses the symmetric encryption algorithm DES, which is implemented by VC programming [1].

(1) Handling of keys

After obtaining the 64-bit key from the user, first, the original key is transformed according to the sub key transposition table PC-1, and the number of bits of the key is changed from 64 bits to 56 bits, and the converted The key is divided into two parts, the first 28 bits are denoted as C0, and the last 28 bits are denoted as D0; then the sub-keys (16 total) are calculated by the loop structure for(i = 0; i < 16; i++), ie from Starting with i=1, C_{i-1} and D_{i-1} are respectively shifted to the left to generate Ci and Di (16 times in total); finally, the generated Ci and Di are connected in series to obtain a 56-digit number, and then the number is made. The sub key transposition table PC-2 is transformed to generate a 48-bit sub key Ki; 16 sub keys are calculated as described above。

(2) Processing of data blocks

DES is a block cipher method that encrypts data by 64-bit block length. Therefore, in the encryption and decryption, the data to be encrypted or decrypted is first divided into 64-bit data blocks, which are not enough for 64 bits to be filled in an appropriate manner; then the data block is transformed according to the initial transformed IP table, and the transformed data block is transformed. Divided into two parts, the first 32 bits are recorded as L0, and the last 32 bits are recorded as R0; respectively, the data is encrypted with 16 sub keys: 32 bits of R(i-1) data are expanded to 48 bits by extended permutation E. E(R(i-1)) and sub key K(i) are subjected to bitwise exclusive OR operation, and the obtained 48 digits are divided into 8 6-bit arrays, and 1-6 bits are Z1 and 7-12 bits are Z2. , ..., 43-48 bits are Z8, starting from j=1, replacing Zj with the value of S box, the value in S box is 4 digits, a total of 8 S boxes, output 32 digits; eight selection functions The output of Sj ($1\leq j\leq 8$) is spliced into 32-bit binary data, which is used as the input of the P-box permutation to obtain the output; the obtained result is XOR with L(i-1), and the result is assigned to R (i); assign the value of R(i-1) to L(i), complete 1 round of product transformation; execute 16 times as above, until K(16) is also used; finally, R(16) and L (16) The series is sequentially connected to obtain a 64-bit number, and the aforementioned IP conversion is performed on the number. Inverse transform IP-1. From the above introduction to the DES design, it can be seen that this encryption mechanism is quite complicated. To this end, the system implements the implementation code written by VC into the dynamic link library file DESDLL. In addition, the system also defines an encryption function for calling: CString jiami (CString fromStr, CString keystr, int mode=0); where, fromStr is the password to be encrypted or decrypted, keystr is the encrypted or decrypted password, and mode It can take a value of 0 or 1, when mode=0, it is encrypted, and when mode=1, it is decryption [2].

3.3 MD5 algorithm design

The MD5 Chinese name is the fifth version of the message digest algorithm, which is a hash function (HASH function) widely used in the field of computer security. A typical application of MD5 is to generate a message digest for a piece of information to prevent tampering. Specifically, MD5 can process an input piece of text (regardless of its size, format, and number) by an irreversible string transformation algorithm, ultimately resulting in a unique 128-bit message digest. If someone makes any changes to the information originally entered, the resulting message digest will be completely different. In this system, a function called md5 _password (char *old password, char *md5_passwd) that calls the MD5 algorithm is defined. Both the password entered by the user and the key encrypted by DES are processed by MD5 [3].

(1) On the server side, in order to improve security, the user in the user database user_data does not store the plaintext of the user password, but stores the hash value generated by the user password through the MD5 operation in a hexadecimal string format in the database. In this way, even the user himself cannot detect the user's initial password through the message digest; not only that, it is not feasible for the attacker to change the password of another account by copying the password hash value in the database, because the password hashes. There may be special characters in the value that cannot be copied.

(2) In addition to being designed to encrypt the server-side user password, the communication between the client and the server is also designed as two-way authentication. That is, after both communication parties generate the key ZAB, the system still uses the MD5 algorithm to extract the large number for use. The key is to the DES encryption algorithm. The key uses only the first 8 bytes, and the latter is discarded.

3.4 Use of dynamic link library DLL

DLL is the abbreviation of Dynamic Link Library. Chinese means "dynamic link library", which is a library containing code and data that can be used by multiple programs at the same time. Since the advent of the Windows operating system, DLL applications have become very popular. Many Windows applications have been partitioned into relatively independent dynamic link libraries, DLL files, which are placed in the system. When the user executes a program, the corresponding DLL file is called. Since the dynamic link library is not an executable file, it generally cannot run directly or receive messages. They are independent files that contain one or more executable programs that have been compiled, linked, or other DLL calls. Some work function. Therefore, it only works when a function in a DLL is called in another module. In order to improve the operating efficiency of the system and reduce the probability of error, the graphical password authentication system encapsulates the DES encryption algorithm and the MD5 hash algorithm into a DLL, so that the main program does not need to compile the source file during the execution process, and only needs to directly call The function can be.

3.5 Large prime generation

Large number decomposition is a classic difficult problem and the basis of the security of public key encryption algorithms. Experiments have shown that the "most difficult" decomposition numbers are those that appear to have only large prime factors, which means that how to efficiently generate large random prime numbers is a very practical problem in cryptography. At present, the solution to the problem of generating large prime numbers is generally as follows: randomly generate a large number, and use the "prime test" algorithm to determine whether the number is a prime number. If not, continue to generate another large number and judge; until a large prime is found.

(1) Distribution of prime numbers

In terms of quantity, Euclid has proved in the "Original Geometry" that there are infinite numbers of prime numbers, but how are the prime numbers distributed? In 1849, the mathematician Gaussian pointed out that for a sufficiently large number x, the prime degree theorem of "mean number distribution density" is an important conclusion in mathematics, which gives a fairly accurate proportion of prime numbers in integers of a specified length. The boundary [4]. According to this theorem, the probability of occurrence of prime numbers in random numbers of 128bit, 256bit, 512bit, and 1024bit is 0.022, 0.011, 0.0056, and 0.0028, respectively. It can be seen that the distribution density of prime numbers decreases linearly with the increase of the number. The larger the number, the more sparse is the distribution of prime numbers. According to this probability, even for a 1024-bit large number with a high security strength requirement, one large prime number can be found in about 300 random numbers.

(2) Prime test

The prime number theorem only gives the approximate proportion of the prime numbers in the specified number of large numbers. So far no one has found a formula that can directly calculate the prime numbers. Therefore, to generate a random prime number, only a random large number can be generated first. Then pass the prime test. There is a long history of effectively determining whether a given positive integer N is a prime number. The easiest way is to try the division method, that is, the number N is tried and divided by all the integers between 3 and #, if N cannot be divisible, then N is a prime number. This method is simple and straightforward, but at the current level of computer computing, the judgment of large prime numbers is powerless. In fact, it is very difficult to accurately prove that a large number is a prime number, and it is easy to judge a large number with a certain probability as a prime number. So mathematicians have developed a variety of probabilistic prime test methods. The fastest algorithm is the Miller-Rabin prime test. In order to improve the probability of Miller-Rabin test and the test speed, in practical applications, 300-500 small prime numbers can be selected first to test N; when generating random prime numbers, the random number selected is best to let r=0. This eliminates the need for test steps and further increases test speed. The function Get Prime(int bits) is designed in this system to generate random prime numbers.

(3) Generator generation

The generator element is a concept in number theory. The definition is: if each element of a group G is a power of a fixed element a of G, then G is called a cyclic group, or group G is a element a. The generated one is generally represented by the symbol G=(a). a is called a generator of G.2.6.

4. Test

(1) Large number of storage

Since most popular public key cryptosystems are based on 512-bit to 1024-bit large-number operations, the current computer word length is 32-bit or 64-bit, and its compiler can only support 64-bit integers. The operation is 0xFFFFFFFFFFFFFFFF. How to make the computer can carry out large number of storage and calculations, it is not difficult to solve this problem in practice. Since the problem is caused by insufficient digits, then we should think about how to use a smaller number of digits to represent a larger number. One way that is easier to think of is to use large numbers in humans' familiar decimal representations, treating numbers as strings, that is, using a sequence of characters consisting of ten numbers to represent large numbers, and then simulating the process of artificial "vertical calculations". However, since the binary is 1024-bit large number converted to decimal more than 300 bits, to complete any kind of operation, you need to do multiple loops in two array spaces with hundreds of elements, and also need a lot of extra The space is used to store the calculated advance and retreat markers and intermediate results. So this method is not a good method. Another idea is to take a linked list storage method to store large numbers. Since there are two cases from high to low calculation and low to high calculation in the calculation, we define the linked list as a doubly linked list, in which one unit is used to store data, one pointer points to the front data, and the other points to the latter data [5].

Considering that the number of numbers representing the same size is larger, the number of bits required is smaller. So why not use a large number system to represent large numbers? So we can improve the first method: the large number is represented as an nary array. For the current 32-bit computer system, n can take the highest value of 2 to the 32th power, that is, 0x100000000. Thus, converting a binary number of 1024 bits into 0x100000000 is only 32 bits, and the range of each bit is not binary 0-1 or decimal 0-9, but 0x100000000 0-0xffffffff, we can just use an unsigned long integer to represent this value. So a 1024-bit large number can be represented by an Unsigned Long array with 32 elements, and the loop size required for various operations on Unsigned Long arrays is at most 32 times. As for 0x100000000 and binary, it is almost the same thing for computers, and the number conversion is very easy.

For example, the large number of decimals is 18,846,443,737,955,516, which is represented by 0x100000000, which is FFFFFFFF, which is equivalent to the decimal number 99, there are two bits: each bit is FFFFFFFF; and the large number 18446744073709551616 can also be represented by 0x100000000 as 00000001 0000000000000000 , equivalent to the decimal number 100, there are three digits: the first digit is 1, the other two digits are 0, and so on. In practical applications, the order of the "number" arrays is in the lower order of the previous high order. Therefore, the large number A can conveniently represent its value in mathematical expressions: A=Sum[i=0 to n](A[i]*0x100000000^i), where Sum represents the sum, A[i] represents the ith element of the array used to record A, and ^ represents the power.

(2) Large number of inputs and outputs

The input and output of large numbers is done by strings. In this system, the following two functions are designed to implement the input of large numbers. The large number input function is as follows:

```
void CBigInt::Get(CString& str, unsigned int system)
{
      int len=str.GetLength(),k;
      Mov(0);
      for(int i=0;i<len;i++)
      {
          Mov(Mul(system));
          if((str[i]>='0')&&(str[i]<='9'))k=str[i]-48;
          else if((str[i]>='A')&&(str[i]<='F'))k=str[i]-55;
          else if((str[i]>='a')&&(str[i]<='f'))k=str[i]-87;
          else k=0;
          Mov(Add(k));
      }
}
```

The output function of large numbers:

```
void CBigInt::Put(CString& str, unsigned int system)
{
      if((m_nLength==1)&&(m_ulValue[0]==0)){str="0";return;}
  str="";
      CString t="0123456789ABCDEF";
      int a;
      char ch;
      CBigInt X;
      X.Mov(*this);
      while(X.m_ulValue[X.m_nLength-1]>0)
      {
           a=X.Mod(system);
           ch=t[a];
           str.Insert(0,ch);
           X.Mov(X.Div(system));
      }
}
```

(3) Large number of operations

When a large number is expressed in a large number system, its operation can be compared with ordinary numbers, as long as the integer participating in the operation is decomposed into an operation between numbers and numbers. For example, if a large number is represented by 0x100000000, its maximum "number" can reach 0xFFFFFFFF. It should be noted that when computing between its numbers and numbers, it is still possible to exceed the current 32-bit machine word. However, in VC++, there is a _int64 type that can handle 64-bit integers, so don't worry about this. If the machine's build system does not have a 64-bit data type, you need to use a smaller hexadecimal representation of the large number. For example, the WORD type (16 bits) can be represented by 0x10000.

Graphical password is a new type of password verification technology. It is an alternative solution to the shortcomings of text passwords that are difficult to remember and poorly resistant to cracking. In the graphical password authentication system, basic arithmetic functions such as addition, subtraction, multiplication, division, and modulo between large numbers and large numbers and between large numbers and ordinary numbers are designed.

Acknowledgments

This article covers the project:

1) Project of Jiangsu United Vocational and Technical College (B/2018/07/096)

2) Key Research Project of 2017 China Staff Education and Vocational Training Association (2017004)

References

[1] H. Jahankhani, K. Revett, and D. Palmer-Brown (Eds.). User Dynamics in Graphical Authentication Systems.[C].ICGeS 2008, CCIS 12, pp. 173–181, 2008.© Springer-Verlag Berlin Heidelberg,2008

[2] Schneier B. Application of cryptography - protocols, algorithms and C source programs. [M]. Beijing: Mechanical Industry Press, 2000

[3] Deng Lin. Research and Implementation of Accelerated Components for Configurable Cryptographic Algorithms. [D]. Changsha: National University of Defense Technology, 2005

[4] Zhang Jianwei.Research and Implementation of Identity Identification Technology Based on MD5 Algorithm[J]. Computer Engineering, 2003, 29(4)

[5] Shi Xi. Research and Implementation of Public Key Algorithm Based on Chaos. [D]. Chongqing: Chongqing University, 2005