DOI: 10.24996/ ijs.2017.58.4A.16

ISSN: 0067-2904

Efficient Plain Password Cryptanalysis Techniques

Sajaa G. Mohammed*, Abdulrahman H. Majeed
Department of Mathematics, College of Science, University of Baghdad, Baghdad, Iraq,.

Abstract

In this research work, some low complexity and efficient cryptanalysis approaches are proposed to decrypt password (encryption keys). Passwords are still one of the most common means of securing computer systems. Most organizations rely on password authentication systems, and therefore, it is very important for them to enforce their users to have strong passwords. They usually ignore the importance of usability of the password for the users. The more complex they are the more they frustrate users and they end up with some coping strategies such as adding “123” at the end of their passwords or repeating a word to make their passwords longer, which reduces the security of the password, and more importantly there is no scientific basis for these password creation policies to make sure that passwords that are created based on these rules are resistance against real attacks. The current research work describes different password creation policies and password checkers that try to help users create strong passwords and addresses their issues. Metrics for password strength are explored in this research and efficient approaches to calculate these metrics for password distributions are introduced. Furthermore, efficient technique to estimate password strength based on its likelihood of being cracked by an attacker is described. In addition, a tool called PAM has been developed and explained in details in this paper to help users have strong passwords using these metrics; PAM is a password analyzer and modifier.

Keywords: Cryptanalysis, ciphering, password, cracking, Cryptology, Cryptography.

تقنيات كفوءة لتحليل تشفير كلمة المرور الصريحة

سجا غازي محمد * ، عبد الرحمن حميد مجيد
قسم الرياضيات، كلية العلوم، جامعة بغداد، بغداد، العراق.

الخلاصة

في هذا البحث، تم اقتراح طرق تشفيركفوءة لفك تشفير كلمة المرور (مفاتيح التشفير) ذات تعقيد منخفض. كلمات المرور لا تزال واحدة من أكثر الوسائل شيوعا لتأمين أنظمة الكمبيوتر. تعتمد معظم المنظمات على أنظمة موثقة بكلمة المرور، وبالتالي كان من المهم جدا بالنسبة لهم أن يفرضوا على المستخدمين الحصول على كلمات مرور قوية، وعادة ما يحاولون فرض الأمن من خلال تكليف المستخدمين

*Email: saj85_gh@yahoo.com

باتباع سياسات إنشاء كلمة المرور. فهي تجبر المستخدمين على اتباع بعض القواعد مثل الحد الأدنى للطول، أو استخدام الرموز والأرقام، بيد أن هذه السياسات لا تتفق مع بعضها البعض؛ على سبيل المثال، طول كلمة المرور الجيدة يختلف في كل سياسة. وعادة ما يتجاهلون أهمية قابلية استخدام كلمة المرور للمستخدمين. وكلما ازداد تعقيدها كلما أحبطت المستخدمين وانتهت بهم بعض هذه السياسات مثل إضافة "123" في نهاية كلمات المرور أو تكرار كلمة لجعل كلمات مرورهم أطول، مما قلل من أمان كلمة المرور، ويتضح من ذلك عدم وجود أساس علمي لهذه السياسات (إنشاء كلمة المرور) للتأكد من أن كلمات المرور التي يتم إنشاؤها على أساس هذه القواعد هي مقاومة للهجمات الحقيقية.

يصف عملنا هذا سياسات إنشاء كلمة مرور مختلفة وفحص كلمة المرور التي تحاول مساعدة المستخدمين على إنشاء كلمات مرور قوية ومعالجة بعض من مشكلاتهم. بحيث يتم استكشاف مقاييس لقوة كلمة المرور ، كما يتم تطبيق طرق فعالة لحساب هذه المقاييس لتوزيعات كلمة المرور. وعلاوة على ذلك، يتم وصف تقنية فعالة لتقدير قوة كلمة المرور على أساس احتمال أن يكون متصدع من قبل المهاجم. وبالإضافة إلى ذلك، تم تطوير أداة تسمى PAM وشرحها بالتفصيل في عملنا هذا لمساعدة المستخدمين على كلمات مرور قوية باستخدام هذه المقاييس.

1. Introduction

In the past decade the large amount of Cryptanalysis applications has been developed in order to enhance the cryptographic techniques. Whole variety of techniques was developed using various brute force algorithms that crack password. In our work we focus to the brute force by using byte recursive permutation algorithm. The analytical processes used by a cryptanalyst require a number of techniques: some mathematical, some linguistic, some of an engineering character, and even some not readily describable such as luck, flair, sixth sense, etc.

Passwords play a significant role in the life of the average user. As the primary form of gaining access to accounts and services the function of the password faces heavy scrutiny from the IT community. While arguably more secure systems exist, such as biometrics or one-time passwords, user-defined passwords continue to have a nigh-universal role in IT infrastructure. It is not possible for a person to manage, either personally or professionally, without the use of user-defined passwords [1].

Cryptography study the mathematical techniques related to information security aspects such as confidentiality, data integrity, entity authentication, and data authentication. Ciphering is a method for data encryption. Cryptanalysis is a method for breaking ciphers and to attack the cipher text [2]. Cryptography is not the only means of providing information security, but rather one set of techniques [3]. Cryptology has played a role in political and military matters from medieval times through the 20th century. Perhaps most famous is the cryptologic effort of Great Britain and the United States during World War II [4]. Cryptology can be subdivided into cryptography (the art and science of making secret codes) and cryptanalysis (the breaking of secret codes) [5].

Passwords, along with the associated account names, are a way to indicate who we are online. They are used to control social media, email, banking and a plethora of other services. As such, user accounts can be seen as our personal presence on the internet. As with any method of personal identification it is vitally important to keep our user accounts safe. A malicious agent obtaining a username and password is akin to a fraudster gaining access to a person's banking details and home address. A great deal of damage can be done with even the smallest of information.

A great deal of password alternatives exists. Hardware verification, such as security tokens are often used in corporate situations, particularly for delicate and/or restricted network access. The cost of such systems prevent wide spread distribution, however, given that an authority capable of the maintenance and issuing of such tokens is required. Biometric evaluation, such as finger print scanning, provides another more secure form of access but in several situations such measures are impractical due to expense of implementation.

This paper sets out to discuss the idea that despite the prevalence of the user/password identification system, people rarely use passwords of any reliable strength or complexity, instead sticking rigidly to out-dated guidelines which make passwords difficult for people to remember but easy for computers to break.

2. Literature Survey

There are much work in Cryptanalysis .The related work below is an important review:
In 2016 Amin R. [6] Introduced Cryptanalysis of the proposed scheme shows that the authentication system is more authentic, secure and efficient than related schemes published earlier.

In 2016 Caragata D. and et al. [7] introduced a study for analyzing Teng et al.'s fragile watermarking algorithm. This study shows that it is not secure against two different cryptographic attacks.

In 2016 Sun B. and et al. [8] investigated the security of structures against impossible differential and zero correlation linear cryptanalysis.

3. Problem Statement and Motivations

The main problems could be summarized in the next few points:
1. This paper sets out to discuss the idea that despite the prevalence of the user/password identification system, people rarely use passwords of any reliable strength or complexity, instead sticking rigidly to out-dated guidelines which make passwords difficult for people to remember but easy for computers to break.

4. Proposed Image In-painting System (IIS)

The suggested schemes consist of some coding modules. The first coding modules is indicated to perform recursive byte array permutation (i.e., generate all possible keys), their suitability was investigated experimentally. The second cryptanalysis method step is the application of the password strength tester (checker), this system performance have been tested. The message digest hashing function MD5 algorithm is exploited to regenerate password, and used to give more secure on password OS file system. The regenerated passwords have been created using traditional MD5. The third cryptanalysis method is XOR cracking with password brute-force attacking.

The full cryptanalysis system have been designed and implemented and their results were analysed. Standard cracked password data sets were used as test materials to investigate the performance of the suggested cryptanalysis scheme; the results indicate that the efficiency of proposed scheme is encouraging when it is compared with state of the art other cryptanalysis scheme..

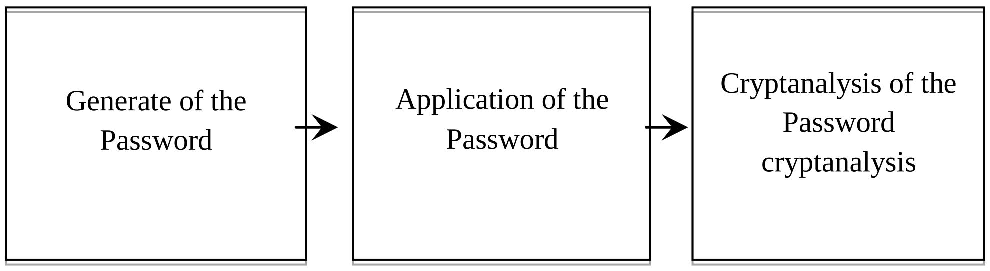
Generate of the Password
Application of the Password
Cryptanalysis of the Password cryptanalysis

Figure 1- The proposed system Cryptanalysis

The layout of proposed cryptanalysis system consists of two modules (XOR cracking with password brute-force attacking). Each module has its own method. Figure-2 shows the system layout.

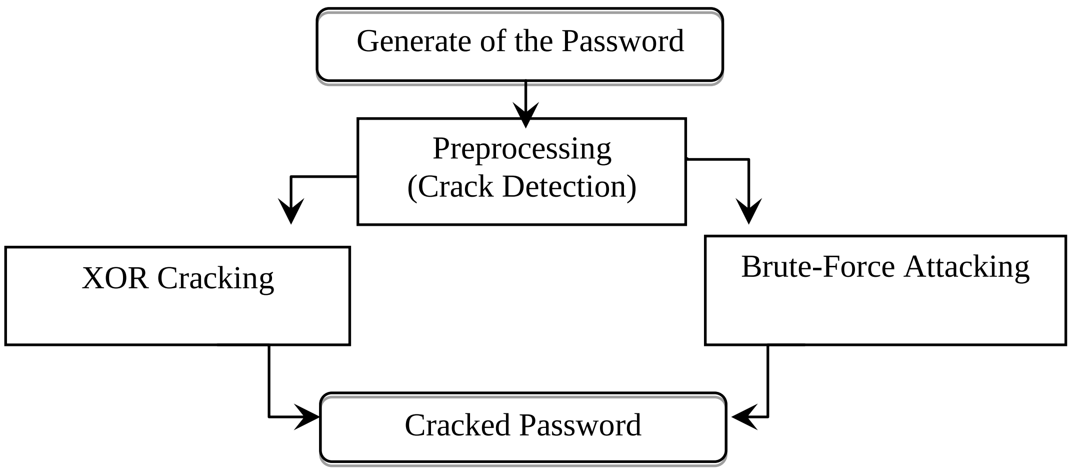

Figure 2- The block diagram of proposed system layout

4.1. Simple Recursive Permutation

This section is related to the password Generator Possibilities method. Takes in a string and splits out all possible permutations of the inputted characters using a simple recursive routine.

Table 1- The block diagram of Simple Recursive Permutation.

Algorithm (1):Generate All Possible Combinations of a Given List of Numbers

Goal: Takes in a string and spits out all possible permutations of the inputted characters using a simple recursive routine.

Input: Letters[] // String of characters

Output: Permut // List of all Possible Permutations of Letters characters

```
Algorithm Steps:
Step 1:  If (Length (Letters) =1) Then
    Print Built & Letters
    Go to Step 3
Step 2: For all i Do {where  0 < i > Length (Letters) }
    st  ← Letters[i]
    stmo ← Letters[i – 1]
    stpo ← Letters[i + 1]
    Letters ← stmo & stpo
    Built ← Built & st
    Go to Step 1
Step 3:  Go to Step 2
```

4.2 XOR Password Cracker

The binary operation XOR (stands for exclusive OR) is a binary operand (as are AND, OR, etc) from Boole algebra. This operand will compare two bits and will produce one bit in return. That bit will be equal to 1 if the two compared bits were different, 0 if they were equal. Xor encryption is commonly used in several symmetric ciphers (especially AES). A symetric cipher is simply a cipher in which the key is used for encryption and decryption process. The XOR operand is so applied to each bit between the text you want to encrypt and the key you'll choose. Examples are better than words , let's take the word "xor".

4.3 Offline Attack Phase (Brute Force Attack)

This method permutated a byte array of a given size using a given byte set. Note "repetitions" means the same character can be repeated in the permutation not that there are repeats of the permutation. Compile before testing it will be slow in the IDE.

The attack this time is much faster. Running at over 16000 attempts per second, a much larger password file (in excess of 300 megabytes) was used and in a far shorter time. Additionally, the GPU can be used instead of the CPU to dramatically increase cracking speed.

5. Experimental Results

The implementation of cryptanalysis steps (Brute Force Simulation) is explained in Figure-3 see appendix (A.1).

1. This method permutated a byte array of a given size using a given byte set. Note "repetitions" means the same character can be repeated in the permutation not that there are repeats of the permutation. Compile before testing it will be slow in the IDE. The attack this time is much faster.

2. Running at over 16000 attempts per second (as demonstrated in Figure- 3), a much larger password file (in excess of 300 megabytes) was used and in a far shorter time. Additionally, the GPU can be used instead of the CPU to dramatically increase cracking speed. It is this result that lends credence to the idea that password-cracking requirements have stayed stagnant while cracking techniques have advanced.

3. Even on a standard processor, a password with 8 characters is surprisingly weak. Sixteen thousand attempts a second would take several years to break into an 8 character password. However, when running on GPU the dramatic increase in speeds renders an 8-letter password mostly useless. At this point in the experiment, brute force options were considered. However there were several drawbacks that prevented us from running them. Primary among them was the available hardware.

4. While GPUs do drastically increase the speed of cracking passwords, the devices available in our testing environment would not have provided the required power. The cost of obtaining such devices was also unfeasible for testing purposes, as for a reliable result, high end graphics cards would be required. A second drawback was in the speed of the devices. Running with substandard hardware, if we could attain 200,000 guesses a second, an 8 letter password containing only lowercase letters would still take 12 days to run

5. . While this is not an unreasonable runtime in real world situations, the number of passwords we needed to test combined with the complexity made testing brute force methods impossible. For the purposes of this thesis, however, we can consider theoretical values. Brute force attacks can be considered more effective but far slower than dictionary attacks. Both attacks will be subject to similar network traffic when used online and thus our online results only need consideration when it comes to dictionary attacks.

6. Brute force attacks excel when it comes to offline password cracking on dedicated hardware. As such, we can presume that any system designed for these attacks will provide far better results than our offline dictionary attack.

7. A loop counting in a given base is the mechanism for this permutation with repetitions algorithm. It’s easy to figure out if you think of a mile-o-meter (the small group of reels found on a speed-o-graph) that records a vehicles mileage.

As you know the reels are numbered from 0 – 9 (base 10) and when a reel rotates and reaches 0 again the next reel to the left is incremented by 1. Each element of the count array (m_bCountArr) is a virtual reel but numbered in the base (integer representation) according to the chosen character set. So, if the chosen character set was “Lower Case” the base would be 26 and each reel would be numbered 0 - 25. Using each current reel value as an index to the character set (m_bByteSet) it is then possible to permute the password array (m_bPwrdArr) in the correct sequence.

The actual code stays from this concept slightly as each element of the count array can have because only hold one value, it is merely incremented or reset according to the last value. Also the reel that stores the units (the reel on the right) is not used.

At the first tested this algorithm, it must admit didn’t expect it to be so fast, having said that don’t be disappointed if the input a long password with all the characters selected and the program just keeps churning away. Brute forcing a password could take years in some cases.

(A.1) Brute Force Simulation

Description:

A password cracker simulator using a permutation of a byte array algorithm. Hold on to your hat this class is fast!

Algorithm

A loop counting in a given base is the mechanism for this permutation with repetitions algorithm. It's easy to figure out if you think of a mile-o-meter (the small group of reels found on a speed-o-graph) that records a vehicles mileage.

As you know the reels are numbered from 0 – 9 (base 10) and when a reel rotates and reaches 0 again the next reel to the left is incremented by 1. Each element of the count array (m_bCountArr) is a virtual reel but numbered in the base (integer representation) according to the chosen character set. So, if the chosen character set was "Lower Case" the base would be 26 and each reel would be numbered 0 - 25. Using each current reel value as an index to the character set (m_bByteSet) it is then possible to permute the password array (m_bPwrdArr) in the correct sequence.

The actual code stays from this concept slightly as each element of the count array can have because only hold one value, it is merely incremented or reset according to the last value. Also the reel that stores the units (the reel on the right) is not used. At the first tested this algorithm, it must admit didn't expect it to be so fast, having said that don't be disappointed if the input a long password with all the characters selected and the program just keeps churning away. Brute forcing a password could take years in some cases.

Table (A.1)- *This class permutated a byte array Permutation with repetitions algorithm.*

This class permutated a byte array of a given size using a given byte set. Note "repetitions" means the same character can be repeated in the permutation ' not that there are repeats of the permutation. Compile before testing it will be slow in the IDE. Use VB5 to compile if possible, VB6 is slightly slower.

```
Option Explicit
'******* For faster DoEvents
Private Declare Function GetQueueStatus Lib "user32" (ByVal fuFlags As Long) As Long
Private Const QS_MOUSEBUTTON As Long = &H4
Private Const QS_PAINT As Long = &H20
Private Const QS_TIMER As Long = &H10
'******* For faster DoEvents
Private Declare Sub CopyMemory Lib "kernel32" Alias _
"RtlMoveMemory" (dest As Any, source As Any, _
ByVal numBytes As Long)
Private m_bSoughtArr() As Byte     ' known password
Private m_iSoughtLen As Integer    ' known length
Private m_bByteSet() As Byte
Private m_bCountArr() As Byte
Private m_bPwrdArr() As Byte
Private m_iPwrdLen As Integer
Private m_iBase As Integer
Public fActive As Boolean
```

```
Public Function PermByte() As Boolean
```
Had to write this just to cope with a password length of one! I didn't want to include error trapping/If statement in the main loop. Align length to ubound.
```
   iCurLen = m_iPwrdLen - 1
    For lPos = 0 To m_iBase - 1
```
Change last byte on every pass. The only byte in this case.
```
       m_bPwrdArr(iCurLen) = m_bByteSet(lPos)
       '****** Test for password match. ******
       iFound = m_iSoughtLen
```

```
        If m_bPwrdArr(iCurLen) = m_bSoughtArr(iCurLen) Then iFound = iFound - 1
        If iFound = 0 Then GoTo PasswordFound
        '****************************************
    Next
PasswordNotFound:
    Exit Function
PasswordFound:
    PermByte = True
End Function
```

```
Friend Function PermByteArr() As Boolean
    ' Warning password length most be > 1 to use this function.
    Dim lPos As Long
    Dim lTemp As Long   ' Temp var gets reused several times.
    Dim iCurLen As Integer
    Dim iFound As Integer
    ' Align length to ubound.
    iCurLen = m_iPwrdLen - 1
    'start the loop
    Do
        For lPos = 0 To m_iBase - 1
            ' Change last byte on every pass.
            m_bPwrdArr(iCurLen) = m_bByteSet(lPos)
            '****** Test for password match.  ******
            ' In most brute force examples the test for password match usually
            ' consists of "if StringGenerated = StringPassword then" this uses a
            ' loop to compare each byte.  It is important to process all the bytes
            ' for a true simulation.  We could of cause exit the loop as soon as
            ' the "byteGenerated <> bytePassword" but that would be cheating!
            ' It may appear to some that using the known length (m_iSoughtLen) of
            ' the sought password is cheating.  This is not the case, no advantage
            ' is gained using this value it is used to maintain the integrity of
            ' the simulation.
            ' Bare in mind changes here only effect the simulation speed!  As such
            ' should not be seen as a way of improving the algorithm.
            iFound = m_iSoughtLen
            For lTemp = 0 To iCurLen
                If m_bPwrdArr(lTemp) = m_bSoughtArr(lTemp) Then iFound = iFound - 1
            Next
            If iFound = 0 Then GoTo PasswordFound
            '************************************
        Next
        ' Base counter loop.  Change other bytes?  Will be at least one to change.
        For lPos = iCurLen - 1 To 0 Step -1
            lTemp = m_bCountArr(lPos) + 1
            If lTemp = m_iBase Then  'carry
                lTemp = 0    ' On the last pass this value is used to stop the main loop.
                m_bCountArr(lPos) = lTemp   'reset
                m_bPwrdArr(lPos) = m_bByteSet(lTemp)
                If GetQueueStatus(QS_MOUSEBUTTON Or QS_PAINT Or QS_TIMER) Then
                    DoEvents
                    If Not fActive Then GoTo PasswordNotFound   ' Cancel?
                End If
            Else
            m_bCountArr(lPos) = lTemp    'increment
            m_bPwrdArr(lPos) = m_bByteSet(lTemp)
            Exit For    ' nothing to carry so bail out.
         End If
      Next
   Loop Until lTemp = 0
PasswordNotFound:
   Exit Function
PasswordFound:
   PermByteArr = True
End Function
```

```
Public Property Let InitByteSet(ByVal NewVal As String)
   ' initialise m_bByteSet with charset
  m_iBase = Len(NewVal)
   ReDim m_bByteSet(0 To m_iBase - 1)
   CopyMemory m_bByteSet(0), ByVal NewVal, m_iBase
End Property
Public Property Let InitSoughtArr(ByVal NewVal As String)
   ' Initialise m_bSoughtArr with the password to find.
   m_iSoughtLen = Len(NewVal)
   ReDim m_bSoughtArr(0 To m_iSoughtLen - 1)
   CopyMemory m_bSoughtArr(0), ByVal NewVal, m_iSoughtLen
End Property
```

```
Public Property Get PermCount() As Double
   ' Compute count from m_bCountArr by converting the base count back to base 10.
   ' Accurate to within the givin base value as the count array does not
   ' store the units.
   Dim i As Long
   Dim iPow As Integer
   On Error GoTo NoCount    ' Necessary for password length of one.
   iPow = UBound(m_bCountArr) + 1
   For i = 0 To UBound(m_bCountArr)
      PermCount = PermCount + m_bCountArr(i) * (m_iBase ^ iPow)
      iPow = iPow - 1
   Next
   Exit Property
NoCount:
   PermCount = 0
End Property
```

```
Public Property Let Password(ByVal NewVal As String)
Initialise password array with the first password.
   m_iPwrdLen = Len(NewVal)
   ReDim m_bPwrdArr(0 To m_iPwrdLen - 1)
   CopyMemory m_bPwrdArr(0), ByVal NewVal, m_iPwrdLen
Dimension count array while we are here.  Always one less than m_bPwrdArr.
   ReDim m_bCountArr(0 To m_iPwrdLen - 2)
End Property
```

```
Public Property Get Password() As String
Convert m_bPwrdArr to a string.
   Password = Space$(m_iPwrdLen)
   CopyMemory ByVal Password, m_bPwrdArr(0), m_iPwrdLen
End Property

Private Sub Class_Terminate()
   'not sure if this has got to be done but it does no harm.
   Erase m_bByteSet, m_bCountArr, m_bPwrdArr, m_bSoughtArr
End Sub
```

Figure 3- Snapshot of password cracking using Brute force method.

6. Conclusions

As mentioned previously in the current research work, with further studies on usability of the passwords, it could improve the distance function to capture everything the user needs. By getting some feedback from users it could learn which passwords are more usable for them and give those operations more weight.

References

1. Richardson D. **2015**. *Information Security An investigation into password habits*. Turku University Of Applied Sciences, thesis Bachelor's.
2. Elizabeth, D., Denning, R. **1982.** *Cryptography and Data Security*. Addison-Wesley, City.
3. Menzes, A., Van Oorschot, P., and Vanstone, S. **1996.** *Hand Book of Applied Cryptography*. CRC Press.
4. Durfee G. **2002**. *Cryptanalysts of RSA Using Algebraic and Lattice Methods*. Stanford university, thesis Phd ,
5. Stamp M. and Low R.M. **2007**. *Applied Cryptanalysis Breaking Ciphers in the Real World*. John Wiley & Sons, Inc.
6. Amin R. **2016.** Cryptanalysis and Efficient Dynamic ID Based Remote User Authentication Scheme in Multiserver Environment Using Smart Card. *International Journal of Network Security*, **18**(1): 172-181.
7. Caragata D., Mucarquer J.A., Koscina M., El Assad, S. **2016**. Cryptanalysis of an improved fragile watermarking scheme. *Int. J. Electron. Commun*. (AE-), **70**: 777–785.
8. Sun B., Liu M., Guo J., Rijmen V., and Li R. **2016.** Provable Security Evaluation of Structures Against Impossible Differential and Zero Correlation Linear Cryptanalysis. *International Association for Cryptologic Research*.