The Security of Key Derivation Functions in WINRAR

Jie Chen[1], Jun Zhou[2], Kun Pan[3], Shuqiang Lin[4], Cuicui Zhao[5], Xiaochao Li[1*]
[1, 2, 3, 4, 5, 1*]Department of Electronic Engineering, Xiamen University, Xiamen, China
Email: chenjfrank@gmail.com, hellozj@gmail.com, pankun1988@gmail.com, linshuqiang10@gmail.com, cuicuizhao@foxmail.com, leexcjeffrey@xmu.edu.cn[*]

***Abstract*—In various versions of WINRAR, the file security is mainly protected by user authentication and files encryption. Password based key derivation function (PBKDF) is the core of the WINRAR security mechanism. In this paper, the security of PBKDF algorithm and the encrypted file in WINRAR are analyzed by the Game-Playing approach. We show the upper boundary of the Adversary's Advantage over the random function. With the theoretical derivation, the actual safety of the WINRAR encrypted files has been discussed. According to the latest developments for GPU-based exhaustive password search attacks, we do some experiments and draw a conclusion that if the length of password is longer than 6, the WINRAR and later versions are secure.**

***Index Terms*—Message Authentication Code, Key Derivation Functions, Provable Security, Adversary's Advantage, Random Oracle Model**

I. Introduction

Aiming at preventing unauthorized access and modification during the file transmission or the process of storage, message integrity is the essential requirement in the information security. We, usually, use message authentication scheme [1] to ensure symmetric cryptographic message integrity. Compared with other physical and biological characteristics of the authentication mode, password based message authentication scheme is widely used and it's a major identity authentication mechanism. However, inputs to an application are typically raw key materials for passwords, which are not in the form to be used as keys. Therefore, a key derivation function is often a necessary component in all security applications. Generally, passwords are chosen from a relatively small space, so it is hard to prevent exhaustive password search attacks and dictionary attacks. Message authentication scheme security based on password is defined on the premise of the distribution of encrypted keys randomly, and uneven distributed key will reduce the security of the system. As a result, the security of password authentication in the encrypted file mainly depends on key derivation functions, in other words, the random of the derivation key.

As mentioned above, passwords, in particularly those chosen by a user, are often short. Therefore, we add iteration count and a specific string called salt in password-based key derivation functions (KDFs) to increase the workload of exhaustive password search attacks and dictionary attacks. These techniques have been specified in widely adopted industry standards such as PKCS [2], but we can't find the work done on analyzing the security of password-based KDFs respectively. In 2005, Frances F. Yao and Yiqun Lisa Yin define the security of PBKDF1 in [3], and give the boundary probability that attackers can distinguish derived key from random string successfully.

We analyze the process of WINRAR password based authentication scheme and conclude that, it is insecure under the exhaustive password search attack if the WINRAR's cryptographic key is only 40-bit long. Even though the cryptographic key is upgraded to 128-bit long in WINRAR, the lack of iteration in PBKDF makes the speed of password search attack very fast anyway. In WINRAR, the cryptographic key length is 128-bit and iteration count is 65,536 at least, the PBKDF is given in the form of $key = H^{(c)}(Byte8\_Padding \| p \| salt)$.

In the provable theory of cryptography, there is an unified proof method game-playing [5] along with the random oracle model [4] to testify the system security. This technique is first proposed by Rogaway, and then widely used in various security proofs [6][7]. This paper bases on security definition of KDFs [3], and uses game-playing technique to prove the security of PBKDF in WINRAR password-based authentication scheme. With the assumptions that the underlying hash function (*H*) is pseudorandom permutation, we quantize the adversary's advantage between KDF and random function and do a series of tests on GPU.

II. Authentication Scheme and KDF

A. Message Authentication Scheme

The password-based authentication scheme is made up of a *MAC* generation algorithm, a verification algorithm and a key generation algorithm, namely *MAC=(K,T,V)* where *K* is key derivation algorithm ,the derivation key is used to generate *MAC* generating algorithm, *T* is *MAC* generating algorithm, whose inputs are the key *K* and the message *M* and output is message authentication code (σ), defined as, $\sigma \leftarrow T_{Key}(M)$, *V* is verification algorithm, whose inputs are key, *M* and σ

and output is a bit verification message, defined as $d \leftarrow V_{key}(M, \sigma)$.

Figure 1 shows the process of password-based message authentication. We define $V_{key}(M, T_{key}(M)) = 1$ to verify whether the access file is legal. It means we need to recalculate the authentication code and verify it with the code in the file to judge whether the user's password is correct or not. There are two ways of *MAC* generation algorithm, one is to use block cipher, and the other one *HASH* function which is always used in password-based authentication scheme in WINRAR.

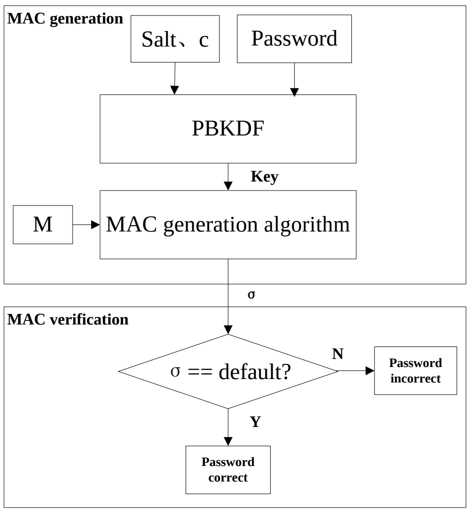
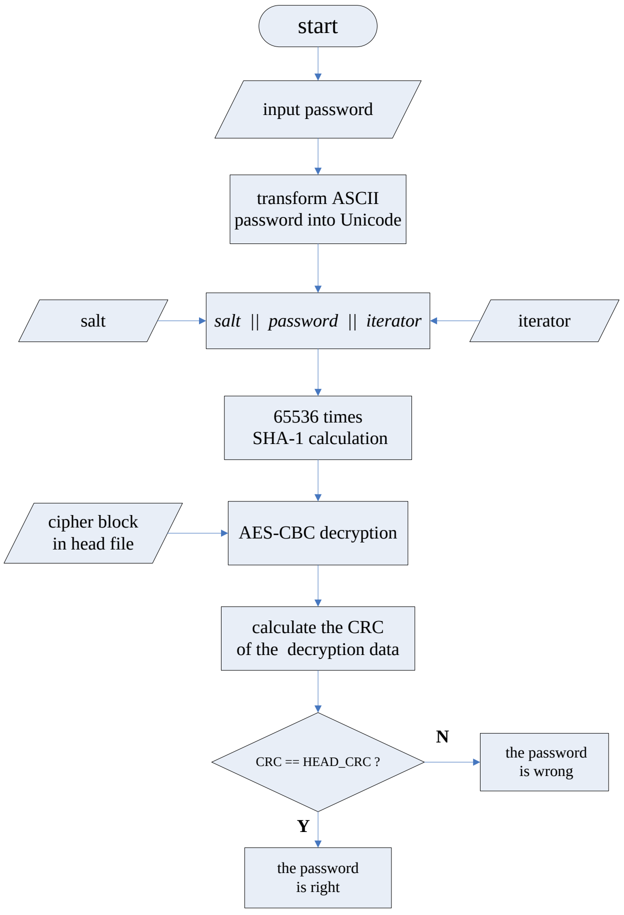


Figure 1. The Description of Password-based Authentication Scheme.

Due to the importance and wide-spread usage, the security of WINRAR mainly depends on file authentication scheme. WINRAR provides two different ways of password based encryption [8]. One is to encrypt the content and file name. As shown in Figure 2, the password authentication process is as blow:

1. We store the user's input password in Unicode form, and then concatenate it with 8 bytes salt and low three significant bytes of iteration count to construct a new string.
2. We put the new string as the input to SHA1 function, with iteration count-65536, and derive the decrypted key and initial vector ultimately.
3. The decrypted key and initial vector are applied to AES-CBC, and then, we decrypt the data block in head file and obtain the decrypted data and HEAD_CRC.
4. We compare the HEAD_CRC above with the one computed from the decrypted data and find that if they are completely equal, then the password is right, otherwise it's wrong.



start
input password
transform ASCII password into Unicode
salt
salt || password || iterator
iterator
65536 times SHA-1 calculation
cipher block in head file
AES-CBC decryption
calculate the CRC of the decryption data
CRC == HEAD_CRC ?
N
the password is wrong
Y
the password is right

Figure 2. The Description of Password-based Authentication in WINRAR (encrypted file name).

In the other encryption, only the content of the file is encrypted, but the data block in head file is stored in plain text. As shown in Figure 3, the corresponding password authentication process is as follows:

1. It's the same to step 1 and step 2 above.
2. The decrypted key and initial vector are applied to AES-CBC. Then, we decrypt the data in the content and obtain the decrypted data and CRC.
3. Comparing the CRC stored in plain text with the one computed from the decrypted data, we find that if they are completely equal, then the password is right, otherwise it's wrong.

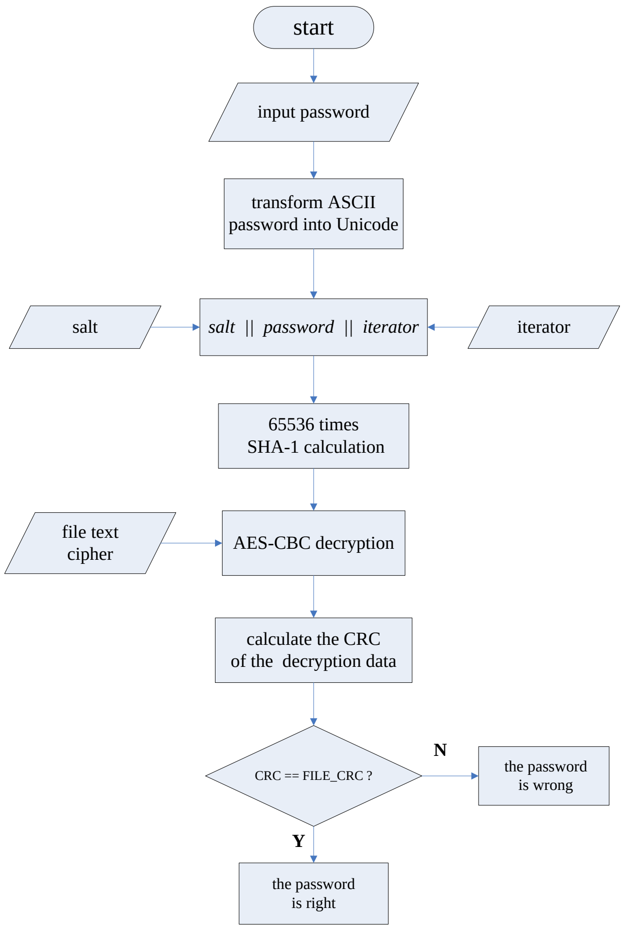

start
input password
transform ASCII password into Unicode
salt
salt || password || iterator
iterator
65536 times SHA-1 calculation
file text cipher
AES-CBC decryption
calculate the CRC of the decryption data
CRC == FILE_CRC ?
N
the password is wrong
Y
the password is right

Figure 3. The Description of Password-based Authentication in WINRAR (unencrypted file name).

B. Key Derivation Function

WINRAR's key derivation algorithm is $key = H^{(c)}(Byte8\_Padding \square p \square salt)$, where key is the derived cryptographic key of length *n* bits, *c* is iteration count, *p* is a private seed, *s* is a public random string called *salt* and *H* is a function can be MD2, MD5 or SHA-1, *Byte8_Padding* is a length of 8 bytes array and the value initial to be {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00}. Put the low three bytes of iterator *i* into the first three bytes of array *Byte8_Padding* and then we can get the vale of *Byte8_Padding*. The processes are to hash for *c* iterations to the concatenation of password *p*, the 16 bytes for the salt *s* and 8 bytes for array *Byte8_Padding*, and then derive the first 128 bits as the key of AES decryption algorithm. The pseudo-code for key derivation algorithm of file encryption in WINRAR is:

```
$U_0=p||s$
For i=0 to c-1 Do
    $U_{i+1}=H(Byte8\_Padding||U_i)$
End For
$key=U_c<0,...,15>$
```

C. The Security Definition of KDF

In the theory of provable security, an adversary's advantage [4] is used to define the security of a KDF in terms of indistinguishability between a newly designed algorithm and a perfect algorithm. A cryptographic algorithm is secure if the probability of the adversary winning the game has a non-negligible advantage over the random function. We treat the underlying hash function as a black-box transformation, and replace it with random oracle. Therefore, the key security is mainly decided by the construction of key derivation function.

In our attack model, adversary A obtains a *n* bits string y_0, the value of this y_0 can be the output of a key derivation function *F* or the output of a random string, then the attacker A queries the oracle *H* and obtains an answer to judge whether y_0 is a key derived from PBKDF or random string. Finally, if adversary A judges y_0 as a derived key, output 1, otherwise output 0. According to the attacking process of adversary A, we designed the following Figure 4 which shows experiments F_A and G_A.

```
1. salt,c are fixed and known
2. randomly choose $p_0$ and is given $U_0=p_0||salt$
3. $y_0 \leftarrow H^{(c)}(Byte8\_Padding \,||\, u_0)$
4. s=0
5. repeat
6.   A chooses $x_i$ and is given $H(x_i)$
7.   s=s+1
8. until s reaches the maximum number of queries t
9. A output either 0 or 1
```

F_A

```
1. salt,c are fixed and known
2. randomly choose $p_0$ and is given $U_0=p_0||salt$
3. $y_0 \xleftarrow{R} \{0,1\}^n$
4. s=0
5. repeat
6.   A chooses $x_i$ and is given $H(x_i)$
7.   s=s+1
8. until s reaches the maximum number of queries t
9. A output either 0 or 1
```

G_A

Figure 4. Attack Experiment.

In the experiment, y_0 can be an output of the key derivation function *F*: $key = H^{(c)}(Byte8\_Padding \,||\, p \,||\, s)$ or a random map function $G: \{0,1\}^n \to \{0,1\}^n$. *t* is the number of adversary A queries. For certainty, the attack's advantage is the difference between PBKDF *F* and random function *G*. The advantage of adversary A is defined as:

$$Adv_{F,G}^{prf}(t) = |\Pr[F_A = 1] - \Pr[G_A = 1]| \qquad (1)$$

In which $\Pr[F_A = 1]$ and $\Pr[G_A = 1]$ denotes the probability of adversary A queries *F*, *G* to output 1 respectively. Therefore, the formula (1) is the probability of distinguishing PBKDF from random function successfully after *t* times query oracle. For certain query times *t*, if the attack advantage A obtains can be ignored, then, we can say password-based key derivation function *F* is secure, namely, we cannot distinguish PBKDF from random function. Hereafter, we will use game-playing technique [3] to prove the security of PBKDF used in WINRAR file format.

III. The Security Proof of KDF

According to attack experiment F_A and G_A, we define two games *R0* and *R1* respectively. In Figure 5, we give a detailed description of the game *R0* and game *R1* is defined as same as game *R0* without $y=y_0$ in step 4.5. *Y* is the set of the initial values u_0, y_0 and $H(x_s)$ is the answer of adversary A on query x_s.

```
1. Set salt,c
2. Choose: $p_0 \xleftarrow{r} PW$ , $y_0 \xleftarrow{r} \{0,1\}^n$ , $i$ =0
3. $Y \leftarrow \{u_0, y_0\}$ , $u_0 \leftarrow p_0 \parallel salt$
4. On the s^th query $H(x_s)$ :
   4.1 $y \xleftarrow{r} \{0,1\}^n$
   4.2 If $y \notin Y$ , then $Y \leftarrow Y \cup \{y\}$
   4.3 else { bad =1}
   4.4 If($i$<$c$-1&& $x_s == Byte8\_Padding \square u_i$) {$i$=$i$+1, $u_i$ =$y$}
   4.5 elseif($i$==$c$-1&& $x_s == Byte8\_Padding \square u_i$ ) {bad=1, $y = y_0$}
4.6 $H(x_s) \leftarrow y$ , return $y$
```

Figure 5. Game *R0*.

```
1. Set salt,c
2. Choose: $p_0 \xleftarrow{r} PW$ , $y_0 \xleftarrow{r} \{0,1\}^n$ , $i$ =0
3. $Y \leftarrow \{u_0, y_0\}$ , $u_0 \leftarrow p_0 \parallel salt$
4. On the s^th query $H(x_s)$ :
   4.1 $y \xleftarrow{r} \{0,1\}^n$
   4.2 If $y \notin Y$ , then $Y \leftarrow Y \cup \{y\}$
   4.3 else { bad =1}
   4.4 If($i$<$c$-1&& $x_s == Byte8\_Padding \square u_i$) {$i$=$i$+1, $u_i$ =$y$}
   4.5 elseif($i$==$c$-1&& $x_s == Byte8\_Padding \square u_i$ ) {bad=1}
4.6 $H(x_s) \leftarrow y$ , return $y$
```

Figure 6. Game *R1*.

Comparing Figure 5 with Figure 6, we can see that both F_A and game *R0* choose y_0. At the start of experiment F_A, value y_0 is set to be $y_0 = H^{(c)}(i \parallel u_0)$. In game *R0*, y_0 is set to be $y_0 \xleftarrow{r} \{0,1\}^n$ at the beginning. Since *H* is a random oracle, and u_0 is randomly selected, there is no difference in the adversary's view point. In game *R0*'s step 4.5, when query value x_s is (c-1 $\parallel$ u_{c-1}), the answer *y* to query oracle *H* will be modified to y_0. In other words, $y_0 = H(c-1 \parallel u_{c-1}) = H^{(c)}(i \parallel u_0)$. For any other query values, oracle *H* will return a random value so that *H* is equivalent to the random oracle in experiment F_A. Thus experiment F_A and game *R0* are equivalent, and the probability of A's success is the same, namely, $\Pr[F_A = 1] = \Pr_{R0}[A = 1]$. Where $\Pr_{R0}[A = 1]$ denotes the probability that adversary A's output is 1 in game *R0*.

Experiment F_A and game *R1* are equivalent from the adversary's point of view. In game *R1*, the answers of any query oracle *H* are at random, and the probability of A's success is the same, namely $\Pr[G_A = 1] = \Pr_{R1}[A = 1]$, where $\Pr_{R1}[A = 1]$ denotes the probability that adversary A's output is 1 in game *R1*. Therefore, the advantage of WINRAR password-based KDF can be defined as:

$$Adv_{winrar}^{prf}(t) = |\Pr_{R0}[A=1] - \Pr_{R1}[A=1]| \tag{2}$$

In game *R0/1*, if the collision occurs in step 4.3 or step 4.5, the flag *bad* will be set 1, where step 4.3 for detecting the collision of Hash Function inner and step 4.5 for $H^{(c)}(i \parallel u_0)$. The only difference between game *R0* and game *R1* is the operations after *bad* to be set, which meets identical-until-bad-is-set condition. Therefore, according to lemma 5 [5] $Adv_{winrar}^{prf}(t)$ satisfies:

$$Adv_{winrar}^{prf}(t) \le \Pr_{R0}[BAD] = \Pr_{R1}[BAD] \tag{3}$$

Where $\Pr_{R0}[BAD]$ and $\Pr_{R1}[BAD]$ denote the probability of the flag *bad* to be set in game *R0* and game *R1*, respectively. In the game *R1*, when *BAD* was 1, either step 4.3 or step 4.5 in Figure 6 is set to 1, respectively named BAD_1 and BAD_2, namely BAD=BAD_1 | BAD_2, where "|" denotes the union of two bad events. According to the proposition of Union Bound [1], we have:

$$\Pr_{R1}[BAD] = \Pr_{R1}[BAD_1 \mid BAD_2] \le \Pr_{R1}[BAD_1] + \Pr_{R1}[BAD_2] \tag{4}$$

When BAD_1 to be set occurred in step 4.3, it equals to choose a *n* bits long string *y* randomly and then test whether it is in set *Y*. If not, add *y* to the set of *Y*, go around the process until t^{th}. Thus, the probability of BAD_1 occurs can be divided into two parts:

A. The first part is choosing *t* elements from a 2^n set randomly, when collision happened, mark it as P_0. This problem is similar to birthday problem [9]

$$P_0 = 1 - e^{-(t(t-1))/(2\wedge(n+1))} \le \frac{t(t-1)}{2^{n+1}} \le \frac{t^2}{2^{n+1}} \tag{5}$$

B. The second part is the collision probability of *t* input elements (x_1, x_2, ..., x_t) with initial elements {u_0, y_0}, we mark it as P_1. Any two elements' collision probability is $\Pr(col_{i,j}) = 1/2^n$, $i \ne j$, so we have:

$$P_1 = C_t^1 C_2^1 \frac{1}{2^n} = \frac{2t}{2^n} \tag{6}$$

If $t \geq 4$, $(t^2/2+2t)/2^n \leq t^2/2^n$,

$$\Pr_{R1}[BAD_1] \leq t^2/2^n \tag{7}$$

We delete step 4.3 in game *R1* to derive game *R2*. The only difference between game *R1* and game *R2* is step 4.3, so we can obtain the equation $\Pr_{R2}[BAD] = \Pr_{R1}[BAD_2]$, and we have:

```
1. Set salt, c
2. Choose: $p_0 \xleftarrow{r} PW$, $y_0 \xleftarrow{r} \{0,1\}^n$, $i$ =0
3. $Y \leftarrow \{u_0, y_0\}$, $u_0 \leftarrow p_0 \| salt$
4. On the $s^{th}$ query $H(x_i)$ :
   4.1 $y \xleftarrow{r} \{0,1\}^n$
   4.4 If($i$<$c$-1&& $x_s == Byte8\_Padding □ u_i$ ){$i$=$i$+1, $u_i = y$ }
   4.5 elseif($i$==$c$-1&& $x_s == Byte8\_Padding □ u_i$ ) {$bad$=1}
4.6 $H(x_s) \leftarrow y$, return y
```

Figure 7. Game *R2*.

In game *R2*, there is a single oracle. The return value *y* of oracle in game *R2* is chosen randomly, namely the *bad* variable and *y* are independent. Now, we say that game *R2* is oblivious that it doesn't use anything about how the *y* was made in order to compute the value of *bad*: no variable influences a *y* and influences *bad* at the same time, informally the *y* does not affect the operation of setting *bad*. According to theorem-Coin fixing [5], we can form a new game *R3*, shown in Figure 8. Game *R3* is similar to game *R2* except that it has no oracle, the random oracle mode is simulated by a for-loop. Each use of a variable *y* is replaced by an arbitrary constant of the correct type. In the game *R3*, we assume the query sequence $x_0, x_1, \ldots, x_{t-1}$ is the maximization of $\Pr_{R3}[BAD]$, namely $\Pr_{R2}[BAD] \leq \Pr_{R3}[BAD]$.

```
1. Set salt,c
2. Choose: $p_0 \xleftarrow{r} PW$, $y_0 \xleftarrow{r} \{0,1\}^n$, $i$ =0
3. $u_0 \leftarrow p_0 \| salt$
4. For $s$ =0 to $t$ -1Do
   4.4 If($i$<$c$-1&& $x_s == Byte8\_Padding □ u_i$ ) {$i$=$i$+1, $u_i \xleftarrow{r} \{0,1\}^n$ }
   4.5 elseif($i$==$c$-1&& $x_s == Byte8\_Padding □ u_i$ ) { $bad$ =1}
End For
```

Figure 8. Game *R3*.

From 4.4 in figure 8 we can find that $u_1, u_2, \ldots, u_s, \ldots, u_{c-1}$ are generated randomly in the process of the game, namely adversary A can't cross the middle *c-1* times *H* calculation to derive key or guess the intermediate state *i* directly. Therefore, the best adversary method is to choose a password *p* from password space at random, generate the message *0* || *p* || *salt* and calculate cycle iteration *c* times to get the derived key. And then, we choose another password until *t* times calculation. As a result, game *R3* can calculate $\lfloor t/c \rfloor$ passwords *p*'s derived key, and the probability that any *p* equals p_0 is $1/|PW|$, so among the *t* queries the probability that x_s equals *c-1* || u_{c-1} is $\lfloor t/c \rfloor / |PW|$ at most. Finally, we can derive:

$$\Pr_{R3}[BAD] \leq \lfloor t/c \rfloor / |PW| \tag{8}$$

Above all, we can derive the security theorem of PBKDF in WINRAR file format:

$$Adv_{winrar}^{prf}(t) < \lfloor t/c \rfloor / |PW| + t^2/2^n \tag{9}$$

When $n \geq 128$, the second term upper bound is negligible.

IV. Data and Analysis

In WINRAR the hash function is SHA-1, the cryptographic key length is 128-bit and iteration count is 65,536. We use *l* to denote the bit length of password, so the password space is $|PW| = 2^l$ and then after *t* times (*t* is an integral multiple of *c*) queries to the random oracle, an adversary A gets the advantage, which satisfies:

$$Adv_{winrar}^{prf}(t) < t/2^{l+\log_2 c} + t^2/2^{128} \tag{10}$$

We can draw these four conclusions as followed:

1. When query time t □ $c|PW|$, the advantage that an adversary A gets from the password-based key derivation in the WINRAR can be ignored so that we can get that password-based key derivation algorithms is safe in the WINRAR authentication scheme. In WINRAR KDF configuration, *c* = 65536, therefore, the password space place a critical role in exhaustive password search attacks. Attack time *t*, as you can see from the Table 1. The first attack experiment, the passwords character set only contains 0~9 and the length of password is 6, there are 10^6 kinds of possibility in the $|PW|$. As the passwords character set becomes 0~9, a~z, A~Z, and the length becomes 8, the possibility of the $|PW|$ turns out to be 62^8. Obviously, from the attack result, the time cost of the attack increases thousands of times. It will take us a plenty of time more than we can calculate when $|PW|$ is more complicated.

2. Due to the iteration count *c*, the workload of exhaustive password search that attacks the WINRAR authentication scheme has increased nearly *c* times. It is equivalent that we expand the length of the password from l-bit to ($l+log_2c$)-bit.

Based on the above discussion, there are two ways to attack password authentication scheme, the first way is exhaustive password search attacks for the password space and the second way is the attacks for the cryptographic key space. When ($l+log_2c$) > *n*, the cost of the first way is more than the second way. n = 128, if length of the password satisfies $l \geq 112$ bit, we should choose the second way. The introduction of the "*Byte8_Padding* ||" does not make the security of the PBKDF better in the WINRAR. Its adversary's advantage is the same as the PBKDF1's.

From the above two conclusions, the security of the WINRAR and the later versions encrypt file depends on the attackers' computational power and users' password space. There is no shortcuts to compute the based hash function expect for calculating cycles iteration c times for the attacker, so the most effective method to decrypt the WINRAR encrypt file is the brute force attack and dictionary attack.

GPU (Graphics Processing Unit) is widely used in cryptography [10] because of strong parallel computational power. We do a series of password recovery tests of WINRAR on different processor platforms, and obtain the data shown in Figure 9.

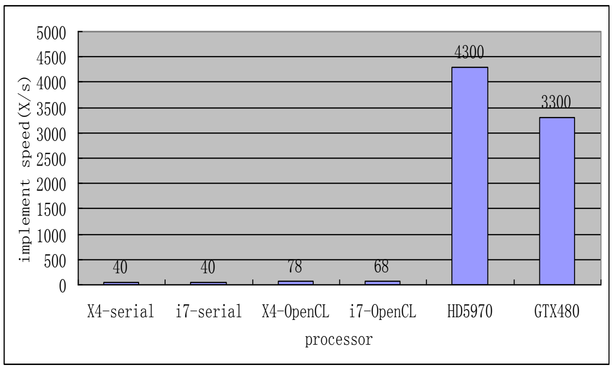


Figure 9. Contrast Chart of Recovery Speeds of WINRAR Password on Different Processors.

From the experiments we obtain that the mainstream of general purpose computing card GPU-ATI HD5970 can calculate 2300M times/s, where "$1M=2^{20}$", namely it costs $2^{-31.2}$s/time to compute SHA-1 respectively. We assume that users' commonly used passwords character set contains 0~9, a~z, A~Z, a blank space and a question mark which are 64 characters in total, so the set space of the password length of 4 is $64^4=2^{24}$. According to the third conclusion, we conclude that when the length of password is shorter than 18 we should attack the space of user's password. The time costs of exhaustive search attack to different passwords length with WINRAR are shown in Table I.

TABLE I. THE TIME COST OF PASSWORD EXHAUSTIVE SEARCH ATTACK IN WINRAR

Character	Password Length	Time Cost of Attack WINRAR
(0~9)	6	3.9s
(0~9,a~z)	6	5.8h
(0~9,a~z,A~Z)	6	152d
(0~9,a~z,A~Z)	7	26.6y
(0~9,a~z,A~Z)	8	1610.1y

From the data in Table I, we notice that there is potential insecurity in password authentication scheme of WINRAR encrypt file when the character of password is in simple set. When the length of users' password character is in complicated set and longer than 6, it works ineffectively with GPU's parallel password exhaustive search because of tremendous time cost. As a result, the password authentication scheme of WINRAR encrypt file is secure.

V. CONCLUSION

This paper uses Game-playing technique to prove that the adversary's advantage between KDFs in WINRAR format and random function is indistinguishable. Thus we prove the security of WINRAR KDFs. According to security theorem, we can conclude that the security of password authentication scheme in WINRAR and later versions depend on attackers' computational power and users' password space. Meanwhile, according to the latest developments for GPU-based exhaustive password search attacks, we do some experiments and conclude that if the length of password is longer than 6, the WINRAR and later versions are secure.

ACKNOWLEDGEMENTS

This study is supported by Fujian Provincial Department of Science & Technology, Project No.2010H6026 and NSFC No.61274133.

REFERENCES

[1] Jonathan Katz, Yehuda Lindell, Introduction to Modern Cryptography: Principles and Protocols. Chapman & Hall / CRC Press, 2007.

[2] RSA Laboratories. (2006). PKCS#5 v2.1: password-based cryptography standard [Online]. Available FTP: ftp.rsasecurity.com/pub/pkcs/pkcs-5v2/pkcs5v2_1.pdf.

[3] Frances F. Yao, Yiqun Lisa Yin, "Design and analysis of password-based key derivation functions," Topics in Cryptology – CT-RSA 2005, San Francisco, CA, USA, Lecture Notes in Computer Science, 2005, Vol.3376, pp. 245-261.

[4] Wen-ling Wu, Deng-guo Feng, Wen-tao Zhang, Design and Analysis of Block Cipher, 2nd ed. Bei Jing: Tsinghua Press, 2009.

[5] M. Bellare, P.Rogaway, "The game-playing technique," Cryptology ePrint Archive, Report 2004/331, 2004, http://eprint.iacr.org.

[6] XU Jin, WEN Qiao-yan, WANG Da-yin, "A new one-pass authenticated encryption model," Acta Electronica Sinica, Vol.37, pp. 2186-2192, Oct 2009.

[7] Mihir Bellare, Phillip Rogaway, "Code-based game-playing proofs and the security of triple encryption," Advances in Cryptology–Eurocrypt 2006, St. Petersburg, Russia, Springer, Vol.4004, 2006, pp. 409-426.

[8] RARlab: WinRAR—at a Glance. (2005) Available at http://www.win-RAR.com/rarproducts.html

[9] Wikipedia. Birthday attack [EB/OL]. http://en.wikipedia.org/wiki/Birthday_attack.

[10] Changxin Li, Xiaochao Li, "Research and implementation of encryption and authentication in storage encryption system," http://210.34.4.13:8080/lunwen/detail.asp?serial=28300.

[11] Mei Zou, Hongwei Wu, Jun Zhou, Xiaochao Li, "Security Analysis of Key Derivation Function in File

Authentication Scheme," Journal of Computer Engineering, Vol.38 (8), pp101-104, April 2012.

Jie Chen received his B.E. degree from Xiamen University, Xiamen, China, in 2010. He is currently pursuing the M.E. in Xiamen University with the major of electronic engineering. His research interests include information security and network security.

Jun Zhou was born in Zhejiang province, China. He received his B.E. degree from Zhejiang Science and Technology University, Hangzhou, China, in 2010. He is currently pursuing the Master Degree of Engineering in Xiamen University with the major of electronic engineering. His research interests include embedded system and information security.

Kun Pan received his B.E. degree from Xiamen University, Xiamen, China, in 2011. Currently he is a master student in College of Information Science and Technology, Xiamen University, Xiamen, China. His primary research area focused on information security.

Shuqiang Lin received his B.E. degree from Xiamen University, Xiamen, China, in 2011. Currently he is a master student in College of Information Science and Technology, Xiamen University, Xiamen, China.

Cuicui Zhao received her B.E. degree from Xiamen University, Xiamen, China, in 2011. Currently she is a master student in College of Information Science and Technology, Xiamen University, Xiamen, China.

Xiaochao Li received the B.E. degrees in wireless from Beijing Institute of Technology (BIT), Beijing, China, in 1992. And he received the M.E. and PhD. degrees from Xiamen University (XMU), Fujian, China, in 1995 and 2004, respectively. From 2007 until now he works as an associate professor in Information Science and Technology Institute in XMU. He has authored/coauthored more than 20 scientific papers. His research interests include embedded system and applications, integrated circuit (IC), information security and software.