On the Security Analysis of PBKDF2 in OpenOffice

Xiaochao Li [1], Cuicui Zhao[1], Kun Pan[1], Shuqiang Lin[1], Xiurong Chen[1], Benbin Chen[1], Deguang Le[1], Donghui Guo[1*]

[1] Xiamen University, Xiamen, 361005, P. R. China.

[*]Corresponding author. Email: dhguo@xmu.edu.cn
Manuscript submitted November 26, 2013; accepted September 20, 2014.

Abstract: Password-based KDF2 (PBKDF2) is widely used in file authentication mechanism and file encryption which could produce a derived key more than 160 bits long. In this paper, the security of PBKDF2 algorithm and its implementation in OpenOffice are analyzed in two modes: CSP-secure mode (Chosen Single Parameter) and CMP-secure mode (Chosen Multiple Parameters). The theoretical security of PBKDF2 is proved in CSP-secure mode by using Game-Playing technology to quantify the upper bound of adversary's advantage. However, a security flaw is explored in CMP-secure mode. This paper presents three proposals to address the security flaw. With the theoretical derivation, the actual safety of the OpenOffice encrypted file has been discussed under the latest developments for GPU-accelerated key recovery attack capability.

Key words: Key derivation functions, provable security, PBKDF2, adversary's advantage.

1. Introduction

During the communication, electronic transaction and data storage, authentication scheme [1], [2] is usually utilized to prevent an encrypted file from unauthorized access and faked message. Compared with other physical and biological characteristics of the authentication schemes, password-based message authentication scheme is widely employed [3]. Generally speaking, passwords chosen from a relatively small space (or have a low entropy) is vulnerable to exhaustive password-search attacks and dictionary attacks. Hence, password-based key derivation functions are used to resist these attacks which transform a non-uniformly distribution source of raw keying material to cryptographically strong secret keys. The techniques which enhance the security of password authentication scheme have been specified in industry standards such as PKCS#5 [4], PKCS#12 [5], IETF, etc. However, little work has been done on the security analysis of password-based KDFs. In 2005, even though Frances F. Yao and Yiqun L. Yin gave a security definition of PBKDF1 and derived the advantage of adversary in [6], the security analysis on PBKDF2 was not presented yet.

Password-based KDF2, specified by NIST [7] and IETF, which has been commonly used in OpenOffice, WAP2, TrueCrypt etc., was studied in this correspondence, as well as the illustration that the construction of KDF has insecure weakness against certain attack mode in spite that the passwords were chosen largely enough and the underlying hash function was reliable. With the assumptions that the underlying iterated hash function (HMAC) was random, the adversary's advantage between PBKDF2 and random function were quantized by using game-playing technique. In addition, combined with parallel computing capability of GPU, a series of passwords recovery tests were conducted.

2. Authentication Scheme and KDF

2.1. Key Derivation Function

Key derivation functions (KDFs) are deterministic algorithms that are used to derive cryptographic key from a secret value, such as a password, Diffie-Hellman shared secret or some non-uniformly random source material [8], [9]. Generally, a Password-Based KDF is defined as $K \leftarrow KDF(p, s, ctx, l)$,
where:

p is a private password. The space of all possible private passwords denoted by PSPACE and the probability distribution of password is assumed to be public;

s is a salt value, chosen from a set of possible salt values;

ctx is a public known context variable chosen from a context space CSPACE;

l denotes the length of derived key;

K is the derived cryptographic key in length of l bits.

Password-Based Cryptography Specification (PKCS#5) [4] specifies two standardized KDFs, namely, PBKDF1 and PBKDF2. With much more secure and complicated structure, PBKDF2 is widely used in information security industry. In PBKDF2, by applying a function PRF (in [7], [10] recommended HMAC with any approved hash function like HMAC-SHA1, HMAC-MD5, HMAC-RIPMED), an n-bit cryptographic key is derived from a password p, a random known value s (called salt) and iteration count c, represented by key=PBKDF$_{(PRF, c)}$ (p, s, n). The process is described in Table 1.

Table 1. The Algorithm of PBKDF2

Parameters and Symbol:	
hLen	Digest size of the hash function.
U_i	Intermediate variable
CEIL(x)	The ceiling of x is the smallest integer that is greater than or equal to x.
‖	Concatenation
⊕	Bit-wise exclusive -or.
T<0,1...,r-1>	The truncation of the binary string T that retains its first r bits.
Inputs:	
p	Password.
s	Salt.
n	Length of derived key in bits, at most (232-1)× hLen.
PRF	HMAC with an approved hash function.
c	Iteration count.
Output:	
key	derived key in n bits.
Steps:	
1.	if $(n > (2^{32} - 1) \times hLen)$
2.	Return an error indicator and stop
3.	$h = CEIL(n/hLen)$
4.	$r = n - (h - 1) *$ hLen
5.	for $i = 1$ to h Do
6.	$T_i = U_0 = PRF(p, salt\|\|i)$;
7.	for $j = 1$ to c Do
8.	$U_j = PRF(p, U_{j-1})$
9.	$T_i = T_i \oplus U_j$

Note that, salt s and iteration count c are publicly known, except for the secret password p. The salt s is

random and is used to create a large set of possible keys corresponding to a given private password.

Usually, the length of derived key n in a file authentication scheme is fixed by the file format. Thus, the form of KDF can be converted into $key = PBKDF^c(p,s)_n$. In the ODF [11] (open document file) of OpenOffice, the underlying hash function is HMAC-SHA-1, and key derivation process is to apply the underlying hash function for 1024 iterations with the input password p, the 16-byte salt s, then extract the first 128-bit as a key for Blowfish decryption algorithm.

2.2. The Security Definition of KDF

In the light of provable security theory, an adversary's indistinguishable advantage [12] is used to define algorithm discrepancy between the security of a KDF and an ideal random function, which assures that the cryptographic keys generated by the KDF are indistinguishable from truly random binary strings of the same length. In game-playing technique, the KDF is secure if the adversary who wins the game has a non-negligible probability advantage over the random function.

The construction of key derivation function was primarily focused and the underlying hash function was regarded as a black-box transformation in this paper. Bellare mentioned that HMAC can be seen as a PRF in [13] and gave a new proof under the sole assumption that the compression function is a PRF in [14], which motivates us to represent the underlying function HMAC as a random oracle H.

The attack model represents the capability of the active adversary. From the attack's point of view, a typical usage scenario of a password-based KDF in the encryption file format is that the salt s and the iteration count c are both known and the derived key may become known through the file format extraction. The attacker usually does not have control of s or c, but in certain scenarios they can be chosen.

Depending on attacker's capability [15], two strong security definitions, Chosen Single Parameter (CSP) and Chosen Multiple Parameters (CMP) were introduced. In CSP mode, adversary A can make queries with fixed parameters for adaptive attacks. While, in the CMP mode, A can make queries with the stronger attack capability of choosing any public parameters such as salt s, iteration count c, derived key length n etc.

Definition 1. CSP-secure/CMP-secure

In CSP-secure mode, adversary A can inquire the Oracle about a return value with an input x and can win the following distinguishing game with probability larger than $1/2+\varepsilon$ within t queries. At the same time, in CMP-secure mode, adversary A can query the Oracle on choosing any public inputs s, c, n.

1) Select a password $p \xleftarrow{R} PSPACE, s \xleftarrow{R} SSPACE$, $c \xleftarrow{R} CSPACE$, $n \xleftarrow{R} NSPACE$.
2) A bit $b \xleftarrow{R} \{0,1\}$ is chosen at random. If $b = 1$, attacker A is provided with the output of KDF, else A is given a random string of n bits.
3) for $i = 1, \ldots, t' \le t$:

if CSP-secure adversary A chooses x_s makes query to oracle H and gets the response $H(x_s)$;

if CMP-secure adversary A chooses s_i, c_i, n_i then makes query to the whole progress and gets the response $KDF^{c_i}(p_i, s_i)_{n_i}$;

4) Step 3 is repeated for up to t queries
5) A outputs a bit $b' \leftarrow \{0,1\}$. It wins if $b' = b$.

3. The Security Proof of PBKDF2

From PBKDF2 algorithm in Table 1, we can find the derived key is consist of a concatenation with iteration values T_i, and each T_i is independent from each other. The iteration values $(T_1, T_2, \ldots, T_h)$ computes in a counter mode [16] and the value of h is usually 1 or 2 in practice. Hence, we take a scenario $h = 1$ for simplicity and give a security proof of PBKDF2.

3.1. CSP Security Mode

According to the CSP-security definition discussed above, we design two attack experiments F_A and G_A specific to key $= \text{PBKDF}^c(p,s)_n$ and $\{0,1\}^* \longrightarrow \{0,1\}^n$ in Fig. 1. The adversary A obtains a $n-bits$ string y_0 which can be a key derived from PBKDF2 or a random string. To complete the game, A guesses whether y_0 is the key or a random string by making at most t queries to the oracle F. If A guesses that y_0 is a cryptographic key then A sends 1 as the outcome, otherwise, A sends 0.

Step1	$salt$, c, n are fixed
Step2	Choose $p_0 \leftarrow \lvert PSPACE \rvert$ randomly
Step3	if $b = 0$, $y_0 \leftarrow \{0,1\}^n$, else $y_0 \leftarrow PBKDF^c(p_0,s)_n$
Step4	$s \leftarrow 0$
Step5	**repeat**
Step6	A chooses x_s and is given $\text{H}(x_s)$
Step7	$s \leftarrow s+1$
Step8	**Until** s reaches the maximum number of query t
Step9	A outputs 1 if he believes that y_0 is a cryptographic key, else outputs 1.

Fig. 1. Attack experiments F_A and G_A.

The experiments F_A and G_A are the realization of KDF and a random function respectively. The advantage of adversary A in distinguishing experiments F_A and G_A is defined as:

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

We note oracle (or black-box) as a subroutine to which A has access. Adversary A has no control on how the answer is computed, nor can A see the inner workings of the subroutine. According to the attack experiments G_A and F_A, We defined two games R_0 and R_1 in Fig. 2. The set Y contains all distinct values of $H(x)$ for which x has been queried and the initialization values are $\{k_0, y_0\}$. The only difference between R_0 and R_1 is that game R_0 contains the assignment $y = y_0$, which is underlined in step 4.6. Step 4 simulates the adversary's queries on oracle . Two flags bad_1 and bad_2 are set when certain "bad" event occurs.

In Game R_1, the answers of any query to oracle H are generated randomly. From the view point of adversary, attack experiment G_A and Game R_1 are equivalent and $Pr[A^G = 1] = Pr_{R_1}[A = 1]$, where $Pr_{R_1}[A = 1]$ denotes the probability of adversary A's output to be 1 in Game R_1. In random oracle PBKDF is treated as PRF, there is no difference between $y_0 \leftarrow PBKDF^c(p_0,s)_n$ in attack experiment F_A and $y_0 \leftarrow \{0,1\}^n$ in Game R_0. In Game R_0's step 4.6, when query value x_s equals to U_{c-1} and the query index increases to c, the answer y to query oracle H will be $y_0 \leftarrow \text{PBKDF}^c(p_0,s)_n$, where y_0 is set at Step 2. For any other query values, oracle H will return a random value. Thus attack experiment F_A and Game R_0 are equivalent, and the success probability of A is the same. Therefore, the advantage of PBKDF2 can be defined as:

$$Adv_A^{prf}(t) = |Pr_{R_0}[A = 1] - Pr_{R_1}[A = 1]| \tag{2}$$

Step1	Set $salt$, c, n
Step2	Choose $p_0 \leftarrow \|PSPACE\|$ and $y_0 \leftarrow \{0,1\}^n, i = 0$
Step3	$k_0 =$ salt ‖ int(1), $Y \leftarrow \{k_0, y_0\}$
Step4	On the s^{th} query, A chooses x_s
4.1	$y_0 \leftarrow \{0,1\}^n$
4.2	if $y \notin Y$, then $Y \longleftarrow Y \cup \{y\}$
4.3	else $bad_1 = 1$
4.4	if $(i == 0 \ \&\& x_s == k_0)$ $\{U_i \leftarrow y,$ key $\leftarrow U_i, i = i + 1\}$
4.5	Else if $(0 < i < c \&\& x_s == U_{i-1})$ $\{U_i \longleftarrow y,$ key $\longleftarrow U_i \oplus key,\ i = i + 1\}$
4.6	Else if$(i == c \&\& x_s == U_{c-1})$ $bad_2 = 1$ if $b = 1$ $key \longleftarrow y_0$
4.7	$H_{p_0}(x_s) \longleftarrow y$, return key
Step5	Repeat step 4 for up to t queries.

Fig. 2. Game R_0 and R_1.

Let $Bad = bad_1 \cup bad_2$, if neither bad event occurs, return values are exactly the same for A in both games R_0 and R_1 according to the fundamental lemma of game playing in [17]. Furthermore, each bad event occurs with the same probability in the two games.

$$Pr_{R_0}\left[A = 1|\overline{Bad}\right] = Pr_{R_1}\left[A = 1|\overline{Bad}\right] \tag{3}$$

$$Pr_{R_0}[Bad] = Pr_{R_1}[Bad] \tag{4}$$

Based on a standard probability argument in [17], the two games R_0 and R_1 are satisfied with *identical-until-bad-is-set* condition. Therefore,

$$Adv_A^{prf}(t) \leq \Pr_{R_0}[Bad] = \Pr_{R_1}[Bad] \tag{5}$$

In the game R_1, when Bad is set, either step 4.3 or step 4.6 in Fig. 2 is set to 1, respectively. According to the proposition of "Union Bound" [2], we have

$$Pr_{R_1}[Bad] = Pr_{R_1}[bad_1 \cup bad_2] \leq Pr_{R_1}[bad_1] + Pr_{R_1}[bad_2] \tag{6}$$

The query process in step 4.3 is firstly choosing a $n - bits$ string y randomly and then testing whether it is contained in set *Y.* If not, it will be added into Y every time until t^{th}. If the random string y collides with the values in the set *Y*, bad_1 is set. Therefore, the probability of that bad_1 occurs can be divided into two parts. One part is similar to a birthday problem and the probability marked as P_0 is

$$P_0 = 1 - exp(-t(t-1)/2^{n+1}) < t^2/2^{n+1} \tag{7}$$

Another part is the collision probability of the t oracle values $(x_1, x_2, \cdots, x_t)$ with initial values $\{k_0, y_0\}$. And collision probability of any two elements is $Pr(col_{i,j}) = 1/2^n, i \neq j$. So,

$$Pr_{R_1}[bad_1] \leq (t^2/2 + 2t)/2^n \tag{8}$$

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

$$Pr_{R_1}[bad_1] \leq t^2/2^n \tag{9}$$

In order to calculate the boundary of $Pr_{R_1}[bad_2]$, the step 4.3 is deleted in Game R_1 to derive Game R_2, shown in Fig. 3, provided that event bad_1 doesn't occur. Then an equation could be obtained as follows,

$$Pr_{R_2}[bad] = Pr_{R_1}[bad_2] \tag{10}$$

Step1	Set $salt$, c, n
Step2	Choose $p_0 \leftarrow \|PSPACE\|$ and $y_0 \leftarrow \{0,1\}^n$,$i = 0$
Step3	$k_0 =$ salt ‖ int(1)
Step4	On the s^{th} query, A chooses x_s
4.1	$y_0 \leftarrow \{0,1\}^n$
4.4	if (i == 0 &&x_s == k_0) {$U_i \leftarrow y$, key $\leftarrow U_i$, $i = i + 1$}
4.5	Else if ($0 < i < c$&&$x_s == U_{i-1}$) {$U_i \leftarrow y$,$key \leftarrow U_i \oplus key$, $i = i + 1$}
4.6	Else if ($i == c$&&$x_s == U_{c-1}$) $bad = 1$
4.7	$H_{p_0}(x_s) \leftarrow y$, return key
Step5	Repeat step4 for up to t queries.

Fig. 3. Game R_2.

The Game R_2 is oblivious because it doesn't use anything about how the oracle H responses were made in order to computebad, except the input sequence $x_0, x_1, \cdots x_{t-1}$. So the random oracle in Step4 will be replaced by a *for-loop* within t queries using coin–fixing theorem in [17], shown in Fig. 4. According to coin-fixing lemma,

$$Pr_{R_2}[bad] \leq Pr_{R_3}[bad] \tag{11}$$

which assumes that the query sequence $x_0, x_1, \cdots x_{t-1}$ can reach the maximization of $Pr_{R_3}[bad]$.

Step1	Set $salt$, c, n
Step2	Choose $p_0 \leftarrow \|PSPACE\|$ and $y_0 \leftarrow \{0,1\}^n$,$i = 0$
Step3	$k_0 =$ salt ‖ int(1)
Step4	for $s = 0$ *to* $t - 1$ Do
4.4	if (i == 0 &&x_s == k_0) {$U_i \leftarrow \{0,1\}^n$ key $\leftarrow U_i$, $i = i + 1$}
4.5	Else if ($0 < i < c$&&$x_s == U_{i-1}$) {$U_i \leftarrow \{0,1\}^n$, $key \leftarrow U_i \oplus key$, $i = i + 1$}
4.6	Else if ($i == c$&&$x_s == U_{c-1}$) $bad = 1$
	end for

Fig. 4. Game R_3.

Step 4 in Fig. 4 indicates that adversary A can't cross any middle calculation to derive key or guess the intermediate state U_i directly. Thus the best method is exhaustive key search attack. Specifically, firstly a

password p is chosen from $|PSPACE|$at random. Then the initial value U_0 is generated with the password and salt. Finally each returned value from oracle is combined by xor operation until the end of t queries. As a result, Game R_3 can calculate the derived keys of $[t/c]$ passwords. Since the probability of $p = p_0$ is $1/|PSPACE|$, in the t times queries, the probability of x_s equals U_{c-1} is less than $[t/c]/|\text{PSPACE}|$, displayed in

$$Pr_{R_3}[bad] \leq [t/c]/|PSPACE| \tag{12}$$

Since $Pr_{R_2}[bad] \leq Pr_{R_3}[bad]$, $Pr_{R_2}[bad] = Pr_{R_1}[bad_2]$, so

$$\text{Pr}_{\text{R}_1}[\text{bad}_2] \leq [t/c]/|\text{PSPACE}| \tag{13}$$

Inserting (6), (9), (13) into (5) gives

$$\text{Adv}_{\text{A}}^{\text{prf}}(t) \leq [t/c]/|\text{PSPACE}| + t^2/2^n \tag{14}$$

3.2. CMP Security Mode

PBKDF2 was intended to provide more security for the exclusive-ors adds an extra layer of protection. In CMP security mode, attacker has full control of the salt and iteration count. Hence, the relations among keys seem slightly more complicated, but this relationship among keys opens the door to dictionary attacks.

Let s be any salt value selected by attacker from $SSPACE$ and let c_1, c_2, c_3 be three consecutive iteration counts. Then, we have:

$$y_1 = kdf(p, s, c_1) = U_0 \oplus U_1 \oplus \cdots \oplus U_{c_1} \tag{15}$$

$$y_2 = kdf(p, s, c_2) = U_0 \oplus \cdots \oplus U_{c_1} \oplus U_{c_2} \tag{16}$$

$$y_3 = kdf(p, s, c_3) = U_0 \oplus \cdots \oplus U_{c_1} \oplus U_{c_2} \oplus U_{c_3} \tag{17}$$

According to the xor operator, we have

$$y_1 \oplus y_2 = U_{c_2} \tag{18}$$

$$y_2 \oplus y_3 = U_{c_3} \tag{19}$$

This yields the following relation among the three keys: $y_2 \oplus y_3 = H_p(y_1 \oplus y_2))$. This relationship would become a fatal weakness in the practical application. The powerful attacker could establish a look up table for all possible passwords, and compute the corresponding sequence $U_0, U_1, \cdots, U_c$. When adversary queries the oracle, with the relationship, the intermediate variables U_i which should not be shown to anyone can be derived from the look up table. It is not hard for attacker to get the password p and any intermediate value U_i bypass the c times iteration. This lead to inefficiency of the iteration count c, therefore, the introduction of c does not improve the security of PBKDF2 in this mode.

4. New Proposals for Strong Security PBKDF2

In the preceding section, the analysis implies that PBKDF2 is secure as long as the adversary's computational resource is far less than $c|PSPACE|$ in CSP-secure mode. The security cannot be guaranteed

since adversary can influence multiple parameters in CMP attack mode.

Based on our analysis, in CMP-secure mode, the KDF should be constructed in a way that the values of $y = KDF(p, s, c)$, for different p, s, and c, are nearly independent of each other. Certainly, there are various ways to achieve this goal. Mei Zou introduces "i||" in [1], which we propose as following:

$$y = KDF(p, s, c) = U_0 \oplus U_1 \oplus \cdots \oplus U_c \tag{20}$$

where $U_1 = PRF(p, s \parallel i)$ and $U_i = PRF(p, i \parallel U_{i-1})$ for $i = 2, \ldots, c$.

Secondly, Yao propose a construction which includes iteration count explicitly as an input to the hash function H [6]. So we introduce the KDF as:

$$y = KDF^*(p, s, c) = H^c(p, U_i \parallel c) \tag{21}$$

The concatenation of iteration count c with intermediate U_i contributes to the independent values of KDF for different p, s, and c.

Thirdly, Hugo Krawczyk takes a more conservative approach via "feedback mode" in the HKDF design[8], each iteration is applied to the result of previous iteration, which not only minimizes correlation but also avoids low input variability and adds unpredictability to the PRF inputs, Thus, we propose the third KDF is:

$$y = XKDF(p, s, c, n) = T_1 \parallel T_2 \parallel \cdots \parallel T_h \tag{22}$$

where the value T_i is defined as follows:

$$T_i = U_0 \oplus U_1 \oplus \cdots \oplus U_c \tag{23}$$

When $i = 0, U_0 = PRF(p, salt \parallel i)$, when $i = 1, \ldots, h$, $U_0 = PRF(p, salt \parallel i \parallel T_{i-1})$, and

$$U_j = PRF(p, U_{j-1}) \tag{24}$$

XKDF achieves stronger security while preserving the same efficiency as existing KDFs.

5. Data and Analysis

In the ODF (OASIS Open Document Format) [11] of OpenOffice, the underlying hash function is HMCA-SHA-1. The derived key is a $128 - bits$ Blowfish cryptographic key. The iteration count value c is 1024 and the salt is a $128 - bits$ random string. We use l to denote the bit length of the password, so the password space is $|PSPACE| = 2^l$, then after A makes t times queries to the random oracle, from (14) the advantage of PBKDF2 in OpenOffice satisfies:

$$Adv_A^{prf}(t) \leq \frac{t}{2^{l+log_2^c}} + \frac{t^2}{2^{128}} \tag{25}$$

We can draw four conclusions as followed:

1) When $t \ll c * |PSPACE|$, the adversary A's advantage of PBKDF2 is negligible, the PBKDF is secure in OpenOffice authentication scheme.
2) The introduction of the iteration count c increases the workload of exhaustive password search nearly c times. In OpenOffice, the iteration count is 1024, so the password length stretches from l-bit to $l+10$-bit.

3) In brute force attack, two approaches could be used, one is exhaustive password space and the other is the cryptographic key space. When $l + log\ {}_2^c \geq n$, the time consuming with first approach is more than the second's. In OpenOffice authentication scheme, n is 128, when the length of the password x is larger than $15 - byte, l + log\ {}_2^c = 8x + log\ {}_2^{1024} > 128$. The exhaustive password space attack is superior over the latter.
4) Compared with the upper bound of the attack advantage in PBKDF1 [18], the security of PBKDF2 in OpenOffice doesn't improved.

The above result implies that, the security of the encrypted file in OpenOffice is mainly depends on the adversary A's computational power and users' password space.

Nowadays, GPU (Graphics Processing Unit) is widely used in cryptography due to its parallel computational grid structure. CUDA is one of the platform and programming model invented by NVIDIA [19]. The mainstream of general purpose computing card NVIDIA GeForce GTX580 and NVIDIA GeForce GTX280 can calculate SHA-1 670 M times/s, and 229 M times/s by experiments respectively [20]. We assume that users' commonly used passwords character set contains *a*~*z*, *A*~*Z*, 0~9, an blank space and a question mark which is 64 characters in total. In Table 2, the time cost of different passwords length on four different GPU cards are shown, which a series of exhaustive search attacks on the user's password space are conducted. According to the table, we recommend that a secure password should be chosen from a mixed up character set of numbers, letters, and special symbols, and the length of a password should be longer than 7.

Table 2. The Time Cost of Password Exhaustive Search Attack in OpenOffice

Password Length	The Set of Password	Time Cost of Attack on OpenOffice			
		GTX280	*GTX470*	*GTX480*	*GTX580*
6	0~9	1.77m	56.13s	42.656s	36.29s
6	a~z	8.55h	4.51h	3.43h	2.91h
7	0~9	17.82m	9.41m	7.15m	6.08m
7	a~z	9.27d	4.89d	3.72d	3.16d
7	0~9 a~z A~Z	11.14y	5.88y	4.47y	3.8y
8	0~9 a~z A~Z	690.81y	364.8y	277.21y	235.82y

6. Conclusion

In this paper, the security of PBKDF2 algorithm and its implementation in OpenOffice are analyzed in CSP secure mode and CMP secure mode. With game-playing technique we quantize adversary's advantage in CSP-secure mode. And we present three proposals to address PBKDF2 security flaw in CMP-secure mode. Based on our analysis and experimental results, we conclude that the security of PBKDF2 in OpenOffice doesn't improve. Regarding to computational power, the security of password authentication scheme in OpenOffice will sustain by using larger character set and password length at least 7.

Acknowledgment

This study is supported by Fujian Provincial Department of Science & Technology(No.2010H6026) , NSFC(No.61274133) and the Science and Technology Project of Quanzhou City (No.2012Z83).

References

[1] Zou, M., Wu, H. W., Zhou, J., & Li, X. C. (2012, April). Security analysis of key derivation function in file authentication scheme. *Journal of Computer Engineering*, *38(8)*, 101-104.

[2] Katz, J., & Lindell, Y. (2007). *Introduction to Modern Cryptography: Principles and Protocols*. CRC Press.
[3] Chen, J., Zhou, J., Pan, K., Lin, S. Q., Zhao, C. C., & Li, X. C. (2013). The security of key derivation functions in WINRAR. *Journal of Computers*, 8(10).
[4] RSA Laboratories. (2006). PKCS#5 v2.1: Password-based cryptography standard. Retrieved from http://ftp.rsasecurity.com/pub/pkcs/pkcs-5v2/pkcs5v2_1.pdf.
[5] RSA Laboratories. (2012). PKCS#12: Personal Information Exchange Syntax. Version 1.1.
[6] Frances, F., Yao, Y. Q., & Yin, L. (2005). Design and analysis of password-based key derivation functions. *Proceedings of Topics in Cryptology — CT-RSA 2005, Lecture Notes in Computer Science*, *3376* (pp. 245-261). San Francisco, CA, USA.
[7] Turan, M. S., *et al*. (2010). Recommendation for password-based key derivation. *NIST Special Publication*, *800*, 132.
[8] Krawczyk, H. (2010). Cryptographic extraction and key derivation: The HKDF scheme. *Proceedings of 30th Annual International Cryptology Conference* (pp. 631-648).
[9] Krawczyk, H. (2008). On extract-then-expand key derivation functions and an HMAC-based KDF. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.131.8254&rep=rep1&type=pdf
[10] Chen, L. (2008). Recommendation for key derivation using pseudorandom functions. *NIST Special Publication*, *800*, 108.
[11] Wheeler, D., Durusau, P., Rathke, E., & Weir, R. (2011). Open document format for office applications (OpenDocument) v1.2. *OASIS Standard, Oasis*, *10(1)*, 19-28.
[12] Wu, W. L., Feng, D. G., & Zhang, W. T. (2009). *Design and Analysis of Block Cipher* (2nd ed.). Beijing: Tsinghua Press.
[13] Bellare, M., & Goldwasser, S. (2008). *Lecture Notes on Cryptography*.
[14] Bellare, M. (2006). New proofs for NMAC and HMAC: Security without collision-resistance. *Advances in Cryptology-CRYPTO 2006* (pp. 602-619). Springer Berlin Heidelberg.
[15] Wen, C. C., *et al*. (2012). A framework for security analysis of key derivation functions. *Proceedings of 8th International Conference on Information Security Practice and Experience* (pp. 199-216). Hangzhou, China: Springer Verlag.
[16] Scarfone, K., & Souppaya, M. (2009). Guide to enterprise password management (draft). *NIST*, *800*, 118.
[17] Bellare, M., & Rogaway, P. (2004). The game-playing technique. *International Association for Cryptographic Research (IACR) e-Print Archive*, *331*.
[18] Zhou, J., Chen, J., Pan, K., *et al.* (2012). On the security of key derivation functions in office. *Proceedings of 2012 International Conference on Anti-Counterfeiting, Security and Identification* (pp. 1-5). Taipei.
[19] Cuda, C. (2013). Programming guide 5.0. *NVIDIA Corporation*.
[20] GPU speed estimations. Retrieved from http://golubev.com/gpuest.htm.

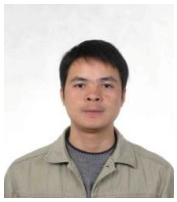
Xiaochao Li received his bachelor degree in electrical engineer from Beijing Institute of Technology in 1992, the master degree in mechanics and electrical engineer from Xiamen University in 1995 and the Ph.D. degree in physics from Xiamen University in 2004. From 2005-2008, he did his postdoctoral research in the School of Electronic Engineering, XiDian University. From 2010 to 2011, he was a visiting scholar at the Department of Electrical and Computer Engineering, North Carolina State University, USA. He is currently an associate professor in Xiamen University and his research areas include information security and mixed-signal IC design.

Cuicui Zhao was born in Xinjiang province, China. She received her B.E. degree from Xiamen University, Xiamen, China, in 2011. She is currently pursuing the M.E. degree with the major of electronic engineering in Xiamen University. Her research interests include information security and network security.

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

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

Xiurong Chen was born in Fujian province of China. She was graduated from the North University of China, Taiyuan, Shanxi, China, in 2012. So far, she is a master student in the College of Information Science and Technology, Xiamen University, Xiamen, Fujian, China.



Benbin Chen was born in Fujian, China. He received his B.E. degree on computer science from Changsha University of Science & Technology, China, in 2004. And he received his M.S. degree is software from Xiamen University, in 2008. He is currently working toward the Ph.D. degree in Xiamen University. His research interests include embedded system.

Deguang Le was born in Fujian, China. He received his Ph.D. degree (with honors) from Xiamen University, China in 2006. He now works as a teacher in the College of Engineering in HuaQiao University. His research interests lie in information security and next generation internet.

Donghui Guo was born in Fujian, China. He received his B.E. degree in radio physics in Physics Department, Xiamen University, China (1984-1988). And he received the M.S. degree and PhD degree in semiconductor, Physics Department, Xiamen University, China, in 1991 and 1994, respectively. Now he works as a professor in Information Science and Technology Institute in XMU and his research interests include IC design, computer network, neural network, mirco/nano device, BioMEMS.