INTERNATIONAL JOURNAL OF COMMUNICATION SYSTEMS
Int. J. Commun. Syst. (2013)
Published online in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/dac.2699

Distributed method for cracking WPA/WPA2-PSK on multi-core CPU and GPU architecture

Liu Yong-lei[1,2] and Jin Zhi-gang[3,*,†]

[1]*School of Computer Science and Technology, Tianjin University, Tianjin, China*
[2]*School of Computer and Information Engineering, Tianjin ChengJian University, Tianjin, China*
[3]*School of Electronic and Information Engineering, Tianjin University, Tianjin, China*

SUMMARY

To overcome the limitations of the existing brute force cracking method of Wi-Fi Protected Access/Wi-Fi Protected Access II (WPA/WPA2)-pre-shared key (PSK) based on single core CPU or one core of a multi-core CPU, a new distributed multi-core CPU and GPU parallel cracking method (DMCG) was first proposed. Colored Petri nets was used to validate the four-way handshake protocol and proved that DMCG could successfully crack WPA/WPA2-PSK. In DMCG, the PSK list was distributed to each PC reasonably using distributed technology. Multiple computing cores were made up of multi-core CPU and GPU on single PC to crack in parallel. GPU contributed to the cracking speed improvement due to the strong computing power for intensive parallel tasks. Experimental results showed that DMCG improved the cracking speed by two orders of magnitude and would exhibit more notable advantages with high-performance distributed system as the cracking speed improved by three or four orders of magnitude, compared with the computing power of one CPU core. An improved Amdahl's law was first proposed, by which the upper bound of the cracking speedup was analyzed. Aiming to the DMCG expansion of cloud computing based on GPU, a lightweight framework called Dandelion computing model was first proposed. Moreover, the analysis of the influences of the graphics card parameters on the cracking speed was processed, and accordingly, the decision support for choosing graphics card in DMCG based on analytic hierarchy process was provided. Finally, the performance optimization of DMCG was processed. Copyright © 2013 John Wiley & Sons, Ltd.

Received 12 February 2013; Revised 14 October 2013; Accepted 20 October 2013

KEY WORDS: wi-fi protected access (WPA); pre-shared key (PSK); graphics processing unit (GPU); multi-core CPU; colored Petri nets (CPN); Dandelion computing; Amdahl's law

1. INTRODUCTION

With the rapid development of wireless local area network (WLAN), more and more related researches have been carried out [1–3]. The new secure scheme of Wi-Fi Protected Access/Wi-Fi Protected Access II (WPA/WPA2) [4] was made to fix WEP, which was known to be insecure. Nevertheless, in the practical application, some vulnerabilities of WPA/WPA2 still existed [5].

Martin Beck *et al*. [6] pointed out that it was possible to mount chopchop-like attack in networks (802.11e/WMM of Wi-Fi Alliance) with QoS enabled. Attackers could acquire RC4 key stream and MIC key. This attack is improved in reference [7]. Further, reference [8] has found the vulnerabilities of Michael message integrity code in temporal key integrity protocol (TKIP) and could decrypt all the traffic from access point (AP) to station (STA).

Vebjørn Moen *et al*. [9] discovered the vulnerabilities of key mixing function in the TKIP and indicated the possibility to launch a TK recovery attack on WPA with complexity $O(2^{105})$ compared

*Correspondence to: Jin Zhi-gang, School of Electronic and Information Engineering, Tianjin University, Tianjin 300072, China.
†E-mail: zgjin@tju.edu.cn

with a brute force attack with complexity $O(2^{128})$. Rui A. C. Ferreira [10] further studied the probability problem of this TKIP attack.

M. Junaid *et al*. [11] pointed out that both the initial counter value used in the CCMP of WPA/WPA2 Wireless LANs and its update could be predicted so that time-memory trade-off (TMTO) attack was possible. On the basis of this attack, the advanced encryption standard (AES) counter mode key size of WPA/WPA2 was decreased from 128 bit to 85 bit.

Flood attacks of denial of service (DoS) attacks were against IEEE 802.11 control frames, the attack and defense details could be found out in Reference [12–14]. WPA/WPA2 was also threatened by the attacks against authentication layer such as 802.1X message spoofing [15, 16] and PHY and MAC layer such as RF jammers [17, 18].

The chopchop-like attacks [6–8] against TKIP only recovered the MIC key from AP to STA and could perform illegal injection attacks. The TKIP key mixing function attack [9] required a few (less than 10) RC4 keys computed under the same IV32, and the fact whether this attack conditions could be satisfied was questionable. The 802.1X message spoofing attack only undermined the authentication process. Jammers and flood attacks mainly destroyed the availability of WLAN (DoS attack). To destroy WPA/WPA2 security completely, acquiring PSK or master session key (MSK) was the ultimate goal, because with PSK or MSK, all traffic of both sides would be fully exposed to the attacker. Many WLANs adopted PSK mode, and only the brute force attack against WPA/WPA2-PSK could acquire PSK. Therefore, we only discuss the improvements of brute force attack against WPA/WPA2-PSK in this paper.

Brute forcers [19] against WPA/WPA2-PSK attacked the four-way handshake protocol and acquired PSK consequently. However, traditional brute forcers adopted single-core CPU or one core of multi-core CPU in single PC, and the cracking speed was limited. Combined with the key mechanism of regular automatic updates, the attack was always failed as not completing the search of the entire key space in the PSK update cycle. Therefore, the known improvement researches were increasing the cracking speed.

Russian software company ElcomSoft introduced the single-PC software wireless security auditor (EWSA) [20], which used GPU to improve the cracking speed, and as a commercial software, the implementation details were not disclosed.

Open source software Pyrit [21], written by Python, supported multi-core CPU and GPU. It introduced the conception of network core and made the collaborative PCs on the network as extra computing cores. But in the distributed cracking process, real time PSK list transmission was required, and the network overhead was large.

ZerOne security team processed the attempt of the distributed cracking WPA/WPA2-PSK [22] based on rainbow table [5]. This project was internal and partly opened the project schedule [22]. It was still experimental and without giving the approach framework and process. Moreover, the cracking process wound fail if the SSID was wrong in the rainbow table built beforehand, and the real time PSK list transmission was also required and was processed by the server.

As relative large key space, brute forcer was not notably effective in the condition that password was set correctly. For instance, letter, number, and symbol were mixed, and enough length such as 16 characters was adopted. Therefore, the improvement research of brute forcer did not attract attentions of researchers. However, the researchers neglected the facts that users always were tired of remembering long and complicated passwords. In the common use, people always chose passwords that were easy to remember, such as 8 to 10 characters. And due to the weak network security awareness, even though the users adopted more complicated passwords, the users did not change the passwords in long time. From these perspectives, the brute forcer could be improved to crack WPA/WPA-PSK.

As we knew, this paper was the fourth research work in WPA/WPA2 brute forcer improvement field. In this paper, the approach framework, process, and implementation details of distributed multi-core CPU and GPU parallel cracking method (DMCG) were first proposed. Experimental results showed that DMCG improved the cracking speed by two orders of magnitude and would exhibit more notable advantages with high-performance distributed system as the cracking speed improved by three or four orders of magnitude, compared with the computing power of one CPU core. And the disadvantages of crackers [20–22] were overcome. Moreover, the theoretical basis

DOI: 10.1002/dac

that DMCG could crack WPA/WPA2 successfully was given. Through analyzing the upper bound of the cracking speedup by the improved Amdahl's Law that was first proposed, the effectiveness of DMCG was verified. The factors affecting the cracking speed was analyzed, and the decision support for choosing graphics card in DMCG was given accordingly. The performance optimization of DMCG was processed, and a lightweight cloud computing model based on GPU was first put forward.

The rest of this paper was organized as follows. In Section 2, the four-way handshake protocol was verified. In Sections 3 and 4, the framework and design details of DMCG were discussed. In Section 5, the upper bound of the cracking speedup was analyzed. In Section 6, the concepts of Dandelion computing model (DCM) was presented. In Section 7, experimental results were shown, and related analysis was processed. And finally, conclusions were given.

2. VERIFICATION OF FOUR-WAY HANDSHAKE

2.1. Handshake authentication protocol

Wi-Fi Protected Access/WPA2-PSK adopted IEEE 802.1X EAPOL-Key frames, called the four-way handshake to achieve the authentication. It is shown as follows:

(1) Before exchanging four-way handshake messages, AP and STA used PSK to compute PMK by function PBKDF2 [4]. PMK = PBKDF2(PSK, SSID, SSID length,4096,256).
(2) Message 1 (M1): AP randomly chose ANonce and sent it to STA in plaintext.
(3) Message 2 (M2): STA randomly chose SNonce and computed PTK by function PRF-X [4]. PTK = PRF-X(PMK,'pairwise key expansion', M), M = Min(AA,SPA)||Max(AA,SPA)||Min(ANonce, SNonce)||Max(ANonce,SNonce), AA and SPA were the MAC address of AP and STA, respectively. PTK was mapped into KCK, EAPOL-key encryption key (KEK), and TK. Finally, STA sent SNonce to AP in plaintext and attached MIC of M2 using MD5 or HMAC-SHA1-128 according to key descriptor version field of EAPOL-Key frames.
(4) Message 3 (M3): AP computed PTK and verified the MIC of M2. If the verification was correct, AP will tell the STA that the PTK was installed.
(5) Message 4 (M4): STA sent the acknowledgment message to AP.

2.2. Modeling and verification by colored Petri nets (CPN)

Using CPN [23, 24], the handshake authentication protocol of WPA/WPA2-PSK was modeled and analyzed.

First, the CPN model of the handshake authentication protocol was established. M3 and M4 were just used for the acknowledgment, PTK was already formed through M1 and M2, and the CPN model of the protocol (shown in Figure 1) was simplified.

The cracking WPA/WPA2-PSK steps were as follows. First, attackers passively monitored M1–M4 and AP's SSID or used de-authentication attack to accelerate this process. Second, once the first step succeeded, attackers performed the off-line attack. Every possible PSK was attempted and computed into PMK using an SSID. Third, M1 and M2 were analyzed to acquire ANonce, SNonce, AA, and SPA and then, the PMK was calculated into the first 128 bit of PTK (KCK). Finally, MIC was computed using M2 and KCK. If the computed MIC equaled the MIC in the captured M2, the guessed PSK was correct, otherwise, the next PSK would be attempted. The intruder model is shown in Figure 2, not listing AP and STA parts which were the same as the aforementioned CPN model of the original protocol.

Therefore, the insecure state was that AP and STA shared the PTK. The intruder would decrypt all the traffic between AP and STA if it also acquired the same PTK. The matrix analysis was used, and the simplified state Equation (1) was adopted [23], where each transition was allowed to fire only once (as the handshake authentication protocol).

$$\boldsymbol{M_n} = \boldsymbol{M_0} + \boldsymbol{A}\boldsymbol{\sigma}^t \tag{1}$$

DOI: 10.1002/dac

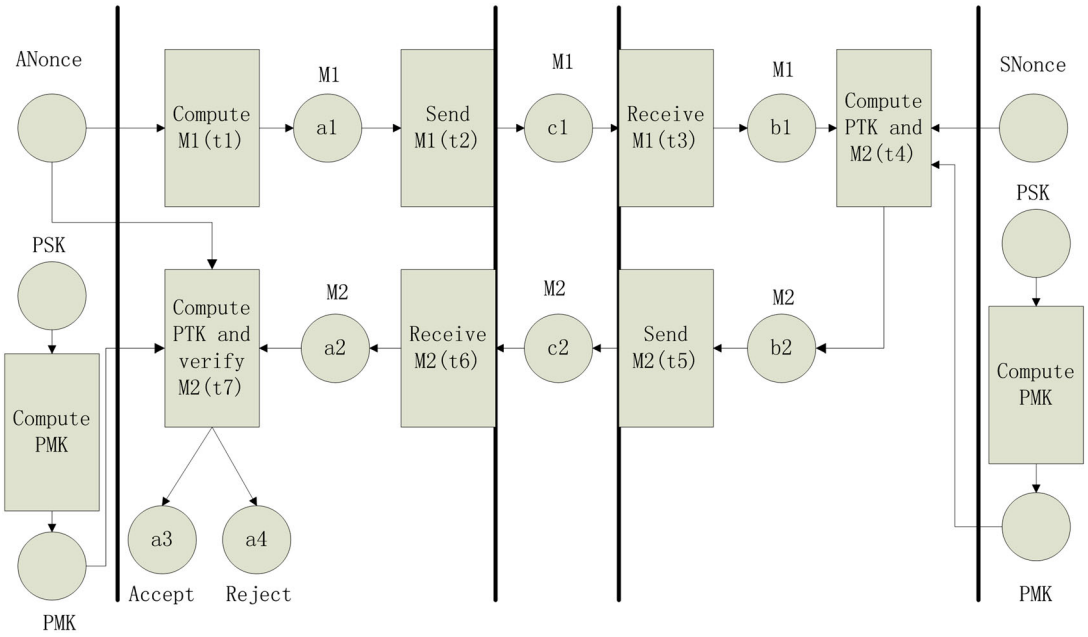

Figure 1. CPN model of handshake authentication protocol.

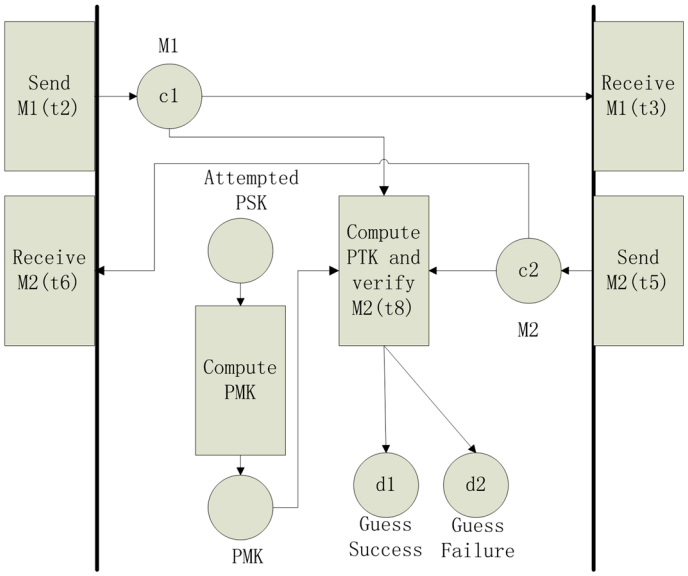

Figure 2. The intruder model.

ANonce and SNonce were chosen by AP and STA randomly and were not the results of the transition. Meanwhile, the computation of PMK in AP and STA and the intruder could be carried out beforehand. Therefore, only 10 states were included in the intruder model. The insecure state M_n and initial state M_0 were defined as the 10-dimensional column vectors. $M_n^{\mathrm{T}} = [0,0,\mathrm{AC},0,0,0,0,0,\mathrm{GS},0]$. $M_0^{\mathrm{T}} = [0,0,0,0,0,0,0,0,0,0]$. σ^t was the transpose vector that determined the firing states and was the eight-dimensional column vector. The incidence matrix A was the 10×8 matrix and is shown in Table I, whereas AC meant accepting without MIC error. RE meant rejecting with MIC error, GS meant guessing successfully that attempted that PSK was equal to PSK, and GF meant guessing unsuccessfully as inequality.

DOI: 10.1002/dac

Table I. The incidence matrix A.

	t1	t2	t3	t4	t5	t6	t7	t8
a1	M1	−M1	0	0	0	0	0	0
a2	0	0	0	0	0	M2	−M2	0
a3	0	0	0	0	0	0	AC	0
a4	0	0	0	0	0	0	RE	0
b1	0	0	M1	−M1	0	0	0	0
b2	0	0	0	M2	−M2	0	0	0
c1	0	M1	−M1	0	0	0	0	−M1
c2	0	0	0	0	M2	−M2	0	−M2
d1	0	0	0	0	0	0	0	GS
d2	0	0	0	0	0	0	0	GF

$M_0{}^{\mathrm{T}}$, $M_n{}^{\mathrm{T}}$, and A were substituted into Equation (1). Under the conditions of no transmission error and modification and correct guess of PSK, according to the discriminant method of solutions of equations, the conclusion was reached thatσ^t was solvable, which meant that the insecure state M_n was reachable from the initial state M_0. Therefore, the four-way handshake protocol had serious weakness, and brute force attack could be launched accordingly.

However, because of the limited cracking speed and short key update cycle between 60 s and 180 s, the search of the entire key space was not completed in the PSK update cycle. The condition that M_n was reachable could not be satisfied. In the following sections, the fact that DMCG solved this problem could crack most regular passwords, and some irregular passwords such as eight-number password could be seen.

3. FRAMEWORK OF DMCG

In order to overcome the disadvantages of the existing crackers, the characteristics of DMCG mainly included that the approach framework and process were proposed, the multi-core CPUs, GPUs, and distributed cracking were supported, different types of supercomputer centers were supported by DCM model, and the network overhead was low in the distributed cracking process by transmission of PSK list beforehand or automatic traversal of PSK key space.

The framework of DMCG is shown in Figure 3 and was first proposed. First, the pre-processor removed the unexpected PSKs in the dictionary file, which did not satisfy the demand [4] that the PSK was a sequence of between 8 and 63 ASCII-encoded characters with the range of 32–126 (decimal). This step was originally performed by PBKDF2 function, so the PBKDF2 function was simplified a little.

Second, before the start of cracking, the PSK list and the captured messages were distributed to each PC by the distributed coordinator (DC) through a pipeline. After the start of cracking, the DC updated the PSK list right of use of each PC at regular intervals according to the current cracking speed of each PC. If one PC acquired the correct PSK, the DC notified the rest of PCs to stop cracking. If errors occurred in one PC, the DC revoked the PSK list right of use.

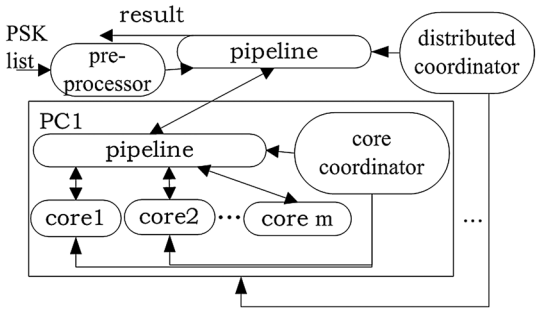

Figure 3. The framework of DMCG.

DOI: 10.1002/dac

Third, the core coordinator (CC) of each PC fetched the PSK list before the start of cracking. The CC managed the collaboration of computing cores, including the assignation of computing work in each computing core, the collection of computing result, and the revocation of the PSK list. The CC coordinately worked with the DC. Because the cracking tasks assigned to each core were independent, the workload of the CC was low. Therefore, the CC was optional and could be taken over by one core of a multi-core CPU.

Fourth, the computing process of the computing core is shown in Figure 4, where P is the preprocessing module, R-H is the repeated hash module, and M-A is the message analysis module. M-C-C is the MIC of message 2 computing and comparing module. If the computing core was one core of multi-core CPU, this core completed all the work; if the computing core was a collaboration of one core of multi-core CPU and one GPU, the GPU completed the repeated hash module and the rest of the modules were completed by one core of multi-core CPU. The M-A module could also be preprocessed by the CC and would be outside the range of core i in Figure 4.

4. DESIGN DETAILS OF DMCG

4.1. Distributed technology adopted by DMCG

In DMCG, before the start of cracking, the PSKs were preprocessed into the PSK list[r], and the PSK list[r] and captured messages were distributed to PC_i ($i = 1,2,\ldots n$) through a pipeline to reduce the communication cost. The DC acquired the initial cracking speed-C_i (the amount of attempted keys per second—k/s) of PC_i, C_i was an estimated value according to the hardware or historical data of the cracking speed of PC_i. The update cycle of DC was t seconds, and a parameter was a. The DC allowed PC_i to attempt j_i keys in the PSK list[r] and notified the PC_i of the index range, where $j_i = a \times t \times C_i$, and the assignment of the index range was sequential.

After the start of cracking, in each t seconds, the DC acquired the current cracking speed—CC_i, the amount of unattempted keys—k_i, and the result of PC_i. If $k_i < CC_i \times t \times a$, the DC granted the right of use of the next $CC_i \times t \times a$ keys to PC_i and notified PC_i of the index range of this part of the PSK list[r], otherwise, the right of use of the PSK list[r] of PC_i was not updated. If network and other errors occurred, the DC revoked the right of use of the PSK list[r] of PC_i and could grant the right of use of this part of the PSK list[r] to other PCs in the future. If one PC acquired the correct PSK, the DC notified the rest of the PCs to stop cracking. To avoid reducing the cracking speed as idle in PC_i, $a = 5$ (the amount of unattempted keys was five times as high as the current cracking speed).

4.2. Multi-core CPU technology

Because of the difficulty of increasing operating frequency and instruction level parallelism of single core CPU, CPU manufacturers had to integrate more processor cores within a single chip to improve CPU performance, and multi-core CPU came in.

Multi-core CPU provided the fine-grained parallelism of computing task with necessary hardware foundation. To make full use of multi-core CPU, computing task must be performed with multiple threads. It was efficiently distributed to cores and balanced among them as possible.

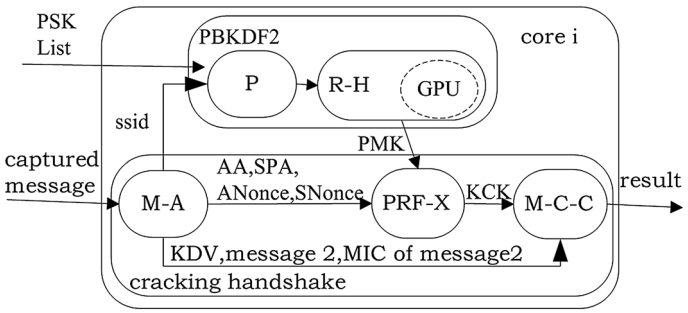

Figure 4. The computing process of core i.

DOI: 10.1002/dac

In DMCG, one thread was launched for each computing core, and each thread cracked one part of the PSK list in parallel. The computing core was one core of multi-core CPU or a collaboration of one core of multi-core CPU and one GPU. Each computing core sets the minimum buffer size (mib), the maximum buffer size (mab), the current cracking speed of computing core (s), and the current buffer size (cb). The aforementioned buffer sizes were measured in number of PSKs. The load balancing method was that each computer core fetched PSKs into its buffer and dynamically adjusted the buffer size. Exploiting the computational power of multi-core and GPU through ATI-Stream, Nvidia CUDA, and OpenCL, Pyrit is a currently relative mature attack tool against WPA/WPA2-PSK. Moreover, Pyrit was an open source tool; the program optimization was well, and Pyrit had been applied in key cracking of real environment, key strength assessment and other WLAN security fields. Therefore, we adopted the load balancing algorithm of Pyrit [21] according to Equation (2).

$$cb = \max(mib, \min(mab, 2/3*cb + s)) \tag{2}$$

4.3. Parallel computing of GPU

Although it is not as powerful as a CPU, a GPU is composed of many transistors as data parallel arithmetic units had a strong parallel computing power. Floating-point computing power could be achieved 10 times or even higher than the contemporary CPU, and external memory bandwidth could be achieved five times or even more than the contemporary CPU [25]. Meanwhile, with Nvidia's CUDA, which made the programmability of GPU better, GPU-based general-purpose computing (GPGPU) was gradually active and had a wide range of application [26, 27].

Graphics processing unit-based general-purpose computing generally adopted heterogeneous parallel model of CPU and GPU. CPU was responsible for the complex logic processing and transaction management, which were not suitable for data parallel computing. GPU was responsible for computation of intensive large scale data parallel computing.

A CUDA program is comprised of parallel steps of a series of GPU (device) kernel functions and serial steps of CPU (host) processing. These steps were executed in turn according to the order of statements in the program and satisfied sequential consistency. Meanwhile, there were two levels of parallelism in one kernel function, block parallelism in grid and thread parallelism in block. Wherein, the threads in the same block could synchronize and share shared memory on GPU, and the threads from the different blocks could only communicate with each other through global memory on GPU. The memory structure of CUDA was presented in detail in the CUDA programming guide.

In DMCG, the processing steps of the CPU (host) side were as follows. First, PSKs were fetched into buffer, and P and M-A modules were executed to generate the intermediate results. Second, the intermediate results were copied to the global memory on the GPU, and the kernel function was launched. Third, as the kernel function finishes, the results of the R-H module PMKs were fetched from the global memory on the GPU to the CPU buffer, and the rest of the steps of the cracking handshake continued.

The processing steps of the kernel function of the GPU (device) side were as follows. Each thread calculated the 4095×2 times [4] of hash operations of one key. Each block is comprised of threads per block (TPB) threads, and reference [28] pointed out that TPB should be a multiple of 64 and between 64 and 256, and the amount of blocks should also be large as possible. In DMCG, the amount of blocks-n was given by Equation (3), where the size was the amount of keys to be computed in GPU each time (that was the amount of keys fetched into CPU buffer each time).

$$n = size/TPB + (size\%TPB == 0?0 : 1) \tag{3}$$

5. UPPER BOUND OF CRACKING SPEEDUP

5.1. Improved Amdahl's law

Cracking WPA/WPA2-PSK belonged to fixed load problem. The Amdahl's law gave the fixed-load speedup model [29].

DOI: 10.1002/dac

Speedup was defined as the original execution time divided by the enhanced execution time. A fraction f was a program's execution time that was indefinitely parallelizable. The speedup S that could be achieved by parallelizing the fraction f across n processors is shown in Equation (4), and the remaining fraction $(1-f)$ was executed sequentially.

$$S = \frac{1}{(1-f)+\frac{f}{n}} \tag{4}$$

The significance of Amdahl's law for parallel computing was that it showed that the speedup was bound by a program's sequential part (shown in Equation (5)).

$$\lim_{n\to\infty} S = \frac{1}{(1-f)} \tag{5}$$

The Amdahl's law presumed that the computing power of all n processors was the same. However, in GPGPU, the computing power of CPU and GPU was different. Therefore, the Amdahl's law was improved. In the collaboration of one core of multi-core CPU and one GPU, the speedup S that could be achieved by parallelizing the fraction f across n streaming multiprocessors (SMs) of GPU is shown in Equation (6), and the remaining fraction $(1-f)$ was executed sequentially by one core of multi-core CPU, where r is the computing power ratio of one SM and one core of multi-core CPU. The upper bound of S is still $1/(1-f)$.

$$S = \frac{1}{(1-f)+\frac{f}{nr}} \tag{6}$$

5.2. *Upper bound of cracking speedup in DMCG*

In the collaboration of one core of multi-core CPU and one GPU, the parallelizable part was 4095×2 times of hash operations in function PBKDF2 of k PSKs to be attempted. The pre-computation of PMK is comprised of two times of hash operations. The computation of KCK is comprised of one time of hash operation. The comparison of MIC in message 2 is comprised of one time hash operation. Because the hash operation was far more time-consuming than other operations in DMCG, the fraction f was computed by Equation (7).

$$f = \frac{k\cdot(4095\cdot 2)}{k\cdot(4095\cdot 2+2+1+1)} \approx 0.9995 \tag{7}$$

The upper bound of speedup was 2000. Because the MIC of message 2 was possibly computed through HMAC-MD5, the rest of the hash operations were processed through HMAC-SHA1-128. The upper bound of speedup was between 2000 and 2500.

In DMCG, each computing core independently cracked one part of PSKs to be attempted and so did each collaborative PC. Therefore, the fraction f was equal to 1, and the upper bound of speedup of DMCG was positive infinity.

6. EXPANSION OF CLOUD COMPUTING ON GPU

Recently, the cloud computing [30] based on GPU was rising, such as Nvidia Reality Server 3.0 and HPC cloud computing with GPUs of Penguin Computing. Especially, in the supercomputer center, each computational node was equipped with an amount of CPUs and GPUs. The computational nodes were generally classified into three types as follows: (Type 1) node with massive CPUs, (Type 2) node with massive CPUs and a spot of GPUs, and (Type 3) node with massive GPUs and a spot of CPUs.

DOI: 10.1002/dac

Expecting DMCG to expand to cloud computing based on GPU, a lightweight framework DCM was proposed. The inspiration was from the plant of dandelion. When the dandelion was mature, small flowers were collected together into a composite flower head. As winds blew, the flower head dispersed, and each flower carried one seed. Once the suitable growing environment was encountered, the seed would sprout. In DCM, the computing task was divided into subtasks and packaged into dandelion seeds, which spread among the GPU server cloud. Once suitable computing resource was satisfied, the task in the seed would be computed. The framework of the DCM is shown in Figure 5.

The dandelion computing client communicated with the dandelion seeds generator through a secure protocol about computing task, payment, and so on (shown in Figure 6). The application type field meant the application the GPU server cloud supported. The unsupported application could also be installed through the application code field. The payment information field was used to negotiate the price of renting computing resource. Other fields were the related data of specified applicaton-WPA/WPA2-PSK. It was assumed that the WPA/WPA2-PSK cracking task on the GPU server cloud was so complicated that PSK was out of the range of dictionaries; therefore, we had to traverse the entire key spaces of several specified key lengths. To reduce the communication overhead, only an amount of the keys field was demanded, as the GPU server could jump to the next key according to the PSK requirements automatically.

Once two parties reached to an agreement with the price, the application code, if business data had, would be installed among the GPU servers beforehand. The dandelion seeds generator started to distribute dandelion seeds (shown in Figure 7).

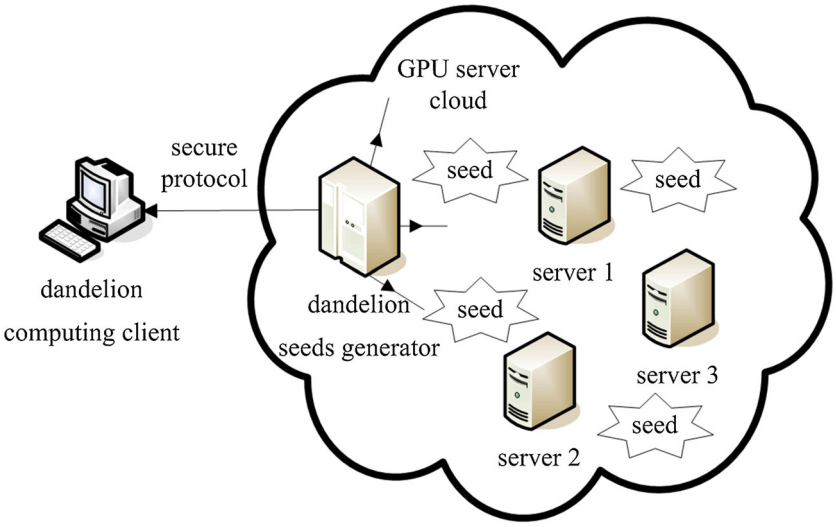

Figure 5. The framework of DCM.

application type	payment information
application code	
start of the keys	
amount of the keys	
ssid	
handshake messages	
result	

Figure 6. The business data format.

DOI: 10.1002/dac

seed ID	application type
start of the keys	
amount of the keys	
ssid	
handshake messages	
result	
seed number	seed survival time

Figure 7. The Dandelion seed format.

It was assumed that the GPU servers could constitute a connected graph through network connections. Taking the node failure into account, the seed number field was set. Avoiding seed flooding, the seed survival time field was set. The process of handling the seed of GPU server is shown in Figure 8.

Where the decision condition C was whether the GPU server had available computing resource and no being computed seed or the seed forwarding was to forward to only one adjacent GPU server randomly. When the seed was computed completely, the result was written, seed number field was set to -1, and the seed was sent back to the dandelion seeds generator.

Particularly in the supercomputer center, if the computational node is coming from type 1, it is simple to averagely assign the cracking tasks. However, if the computational node is coming from type 2, DMCG (collaboration of one core of multi-core CPU and one GPU as a computing core and the remainders of CPU cores as other computing cores) will be used to assign tasks, and the shared cracking was processed in whole computing cores according to DMCG; if the computational node belonged to type 3, the unpaired GPUs existed. All the cracking work should be completed by GPU, and DMCG should be modified in code level.

7. EXPERIMENT AND DISCUSSION

PC1 equipped with Intel Core i5-760 Quad processor and Nvidia GeForce GTX480 graphics card and PC2 equipped with Mobile AMD Athlon 64 single core processor were used for measuring the cracking speed. Two wireless cards of Linksys® WUSB54GC were used, one for packet capturing and another for launching the four-way handshake to AP. The AP was netis® NW705 PLUS and set to WPA/WPA2-PSK mode. The PSK was set to 'qazwsxedc'.

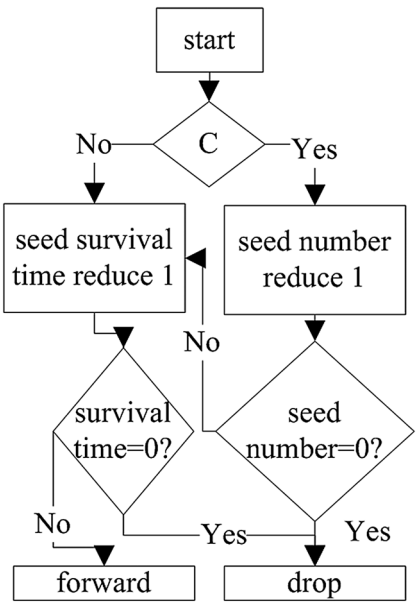

Figure 8. The process of handling seed.

DOI: 10.1002/dac

According to the discussion of Section 3, the IEEE 802.11 standard [4] demanded that the PSK was a sequence of between 8 and 63 ASCII-encoded characters with the range of 32–126 (decimal). Therefore, the WPA/WPA2-PSK key space was huge according to the key form. The size of key space is shown in Equation (8).

$$KeySpace = \sum_{i=0}^{55} 95^{8+i} \approx 10^{124} \tag{8}$$

Using dictionary files was a commonly used method in key cracking. To illustrate the cracking effect of DMCG, some dictionary files were shown as follows but not restricted to.

(1) Birthday dictionary. The birthday dictionary contained 100 years of birthdays and about 10 sorts of variants such as '1983-06-04' and '1983/06/04'. The average key space was about 180,000.
(2) English words dictionary. About 90% of the concepts in 25,000 words Oxford Pocket English Dictionary could be achieved with 850 words, which was called Basic English and was popularized in China [31]. Users, especially Chinese users, were used to use two or three simple words. So the average key spaces of English words dictionary were about 360,000 (two Basic English words) and 300 million (three Basic English words).
(3) Chinese name dictionary. In a password, a Chinese name was presented as Pinyin. The amount of Pinyin syllable was about 400, such as 'liu' and 'yan'. Most Chinese names contained two or three Chinese characters (that meant two or three syllables), and reference [32] pointed out that people of 129 family names were 87% of the total population. So the average key space of the Chinese name dictionary was about eight million.

On PC2, the cracking speed of the traditional method on a single core CPU was measured, and on PC1, the cracking speed of DMCG on a single PC was measured. The experiment process was as follows.

First, we used one wireless card to monitor the wireless channel for capturing the four-way handshake packages by Aircrack-ng [19], and the network protocol analyzer Wireshark [33] could also be adopted. Second, we used another wireless card to access the AP. When the former wireless card captured the four-way handshake packages successfully, we used PC1 and PC2 for cracking by DMCG (in another experiment, the wireless card had accessed the AP, we used de-authentication attack by Aircrack-ng [19] and also acquired four-way handshake packages). In the cracking, we adopted a birthday dictionary with 'qazwsxedc' in the end. Finally, DMCG cracked the correct PSK 'qazwsxedc' successfully. The cracking speed results are shown in Table II. And DMCG was also effective using other dictionary files, such as the social security number of USA.

From the results, it was found out that adopting non-distributed DMCG, the cracking speed was improved by two orders of magnitude compared with Mobile AMD Athlon 64 single core processor. Because of the collaboration of each computing cores, the total cracking speed was slightly lower than the sum of cracking speeds of all the computing cores. We used the mainstream CPU (such as Intel Core i5-760 Quad processor used in the experiment) as the reference of the computing power of one CPU core. Therefore, the cracking speed of one CPU core was about 600 (k/s) in this paper.

Table II. Experimental results.

PC	Computing core	Cracking speed (k/s)
PC1	CPU core 1 + GPU	31,186.68
	CPU core 2	691.9
	CPU core 3	658.65
	CPU core 4	678.7
	Average of CPU cores	676.42
	Sum of cracking speeds	33,215.93
	Total cracking speed	32921
PC2	Single core CPU	275
Mainstream CPU	One CPU core	600
Speedup	32,921/275 ≈ 119.7	

DOI: 10.1002/dac

7.1. Cracking speed estimation of DMCG

In DMCG, by the beforehand distribution of the PSK list or automatic PSK key space traversal in DCM, the network overhead was low during the cracking. The information that DC and PC_i exchanged is shown in Figure 9.

If the PSK list is large, such as traversing entire key spaces of several specified key lengths, the PSK list index range field could be expanded or the fields of start of the keys and amount of the keys as presented in DCM could be used. Moreover, according to the current hardware level, the cracking speed did not reach one million, the fields of CC_i and K_i were 4 octets long. Even if errors occurred frequently on each PC_i, the field of the PSK list index range or more would not be large. The data length would not be beyond 1 KB. In 100 M or faster Ethernet environment, the time of each data exchanging between DC and PC_i would not be beyond 1 KB/100Mbps × 8–0.08 ms. Moreover, the update cycle of DC − t was much larger than 0.08 ms such as several seconds or even more. Therefore, the communication cost between DC and PC_i was so low that it could be concluded that the cracking speed of DMCG was more or less equal to the sum of the cracking speeds of all PCs.

In the limitations of laboratory hardware conditions, the test on high-performance single PC was not processed. Obviously, adopting multi-core CPU with more cores or more powerful GPU, the cracking speed could be improved. The testing data of Pyrit and EWSA both indicated that the cracking speed of high-performance GPU could reach 10,000 or even 100,000 (k/s). For instance, adopting Nvidia GTX295 or GTX480, the cracking speed could reach above 10,000 [20, 21]. And adopting higher performance GPUs such as Nvidia GTX Titan or AMD Radeon HD 7970, the cracking speed could reach about 100,000 [20]. Moreover, with the addition of distributed technology, the total cracking speed of DMCG could reach one million. So as general performance DMCG (GDMCG) with about eight mainstream PCs, the cracking speed was 200,000 (k/s); in high-performance DMCG (HDMCG) with more than 10 high performance PCs, the cracking speed was one million (k/s). Taking security into consideration, using a short rekeying time (120 s), relative secure WLAN (SWLAN) had a key update cycle between 60 s and 180 s.

In SWLAN, with regard to regular passwords, adopting dictionary files, the GDMCG and HDMCG could crack the birthday, Chinese name, and two Basic English words dictionary successfully. The average cracking time of three Basic English words dictionary was, respectively, 1500 s and 300 s. In SWLAN, with regard to irregular passwords, such as the 8-number password, 10-number password, and 8-English letters password, the HDMCG could crack the eight-number password successfully, and the GMGCG needed 250 s. The rest of the passwords, even using DMCG, needed several hours or even much longer time. But In SWLAN, traditional cracking method using the computing power of one CPU core on single PC could not crack passwords of any forms successfully. The details of the cracking effect of DMCG are shown in Table III.

7.2. Factors analysis based graphics card selection decision

In DMCG, GPU mainly contributed to the cracking speed. Therefore, it is important to analyze the relationship between the graphics card parameters and the cracking speed. The main parameters of the graphics card included the GPU frequency, streaming processor (SP) frequency, SM number, memory frequency, and memory size.

result-64 octets
PSK list index range (min)-4~8 octets
PSK list index range (max)-4~8 octets
PSK list index range or more
CC_i-4 octets
K_i-4 octets

Figure 9. Exchanging data format of DC and PC_i.

DOI: 10.1002/dac

Table III. The cracking effect of DMCG.

Cracking speed (k/s)		Regular passwords				Irregular passwords			
		Birthday	Two basic English words	Chinese name	Three basic English words	8-number password	10-number password	8-English letters password (case insensitive)	8-English letters password (case sensitive)
600	Average key space	1.8×10^5	3.6×10^5	8×10^6	3×10^8	5×10^7	5×10^9	10^{11}	2.7×10^{13}
	Cracking time-one CPU core (seconds)	300	600	13333	5×10^5	8.3×10^4	8.3×10^6	1.7×10^8	4.5×10^{10}
2×10^5	Cracking time-GDMCG (seconds)	0.9	1.8	40	1500	250	2.5×10^4	5×10^5	1.35×10^8
10^6	Cracking time-HDMCG (seconds)	0.18	0.36	8	300	50	5×10^3	10^5	2.7×10^7

DMCG, distributed multi-core CPU and GPU parallel cracking method; GDMCG; general performance DMCG; HDMCG, high-performance DMCG.

DOI: 10.1002/dac

7.2.1. GPU frequency and SP frequency. The GPU frequency was the operation frequency of the most units on GPU including texture pipelines and units in the SM except the SP. The SP frequency was the operation frequency of the SP. The experimental result is shown in Figure 10. The default GPU frequency of GeForce GTX480 of PC1 was 700 MHz. As shown in Figure 10, the cracking speed was linear with the GPU frequency. As the limitation of frequency adjusting tool in Linux, the test of the SP frequency was not processed. It was concluded that the cracking speed was also linear with the SP frequency, as proof, reference [34] acquired the same conclusion in different application.

7.2.2. Memory frequency. Memory frequency was the operation frequency of the memory control unit on GPU. In DMCG, the work load of memory access was far below other operations of cracking WPA/WPA2-PSK. Therefore, the influence of the memory frequency was low. The experimental result is shown in Figure 11, where the default memory frequency of GeForce GTX480 of PC1 is 1.8 GHz.

7.2.3. Memory size. In WPA/WPA2-PSK, adopting HMAC-SHA-1, the inputs in the memory of each key including two outputs of HMAC-SHA-1 and pre-computed f(IV,(K $\oplus$ ipad)) and f(IV, (K $\oplus$ opad)) were 80 bytes. The outputs in the memory of each key were 40 bytes. The cracking speed of the current single GPU was not beyond one million. Therefore, the max memory bandwidth needed was $120 \times 1{,}000{,}000 \approx 115$ MB/s. The memory bandwidth of the mainstream GPU was tens of or hundreds of gigabit per second, and the memory size was more than 1 GB. So, adopting the mainstream GPU, the influence of the memory size was none.

7.2.4. SM number. In the complete processor core of GPU, the number of SM was directly related to the performance of cracking in parallel that was presented in speedup Equation (6). According to our past experimental result of GeForce GT220 and the testing data (GeForce GTX 260,280,295) given by Pyrit, it was concluded that the cracking speed was linear with the SM number when the SMs had the same amounts of SPs (shown in Figure 12). One SM of the GeForce GT220, GTX 260, 280, and 295 had eight SPs in the GPU with compute capability 1.x. In the GPU with compute capability 2.x, one SM had 32 or 48 SPs. The relationship of the cracking speed and the SM number when the SM had different amounts of SPs was the next step of the work.

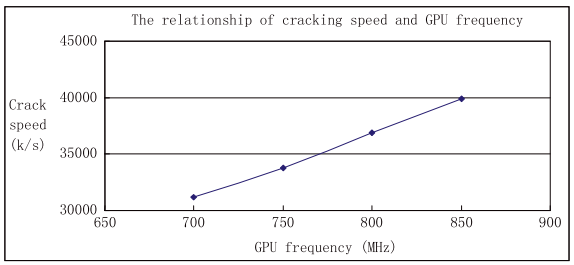

Figure 10. Experimental result of GPU frequency.

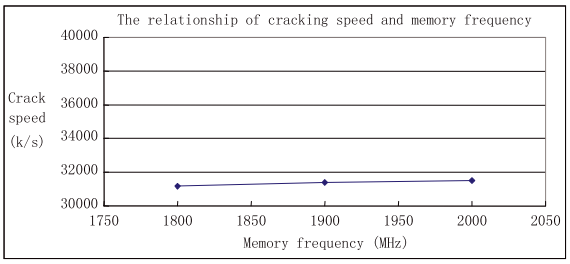

Figure 11. Experimental result of memory frequency.

DOI: 10.1002/dac

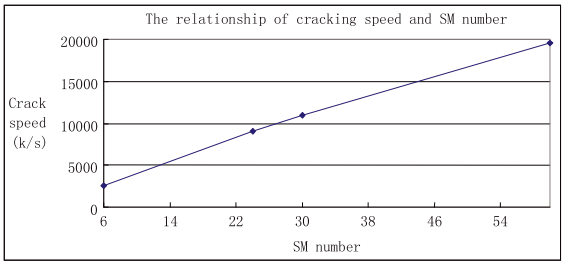

Figure 12. Experimental result of SM number.

7.2.5. Graphics card selection. From the aforementioned experiments and analysis, the parameters of the graphics card except the memory size had effects on the cracking speed, where the influence of the memory frequency was very low. And as the SM number could be increased, however, the GPU and SP frequency were more or less equal among the high, medium, and low performance graphics cards. So, the SM number was the most important parameter.

In DMCG, the decision was made on the basis of the analytic hierarchy process [35, 36] to choose the graphics card. The hierarchy structure of the graphics card selection is shown in Figure 13.

The graphics cards were GeForce GTX465, 570, and 590. The parameter data of the three graphics cards were from Nvidia. The matrix of determination of the criterion layer (A_1) and the matrixes of determination of the scheme layer (B_1–B_4) was constructed. Wherein, the matrix construction adopted from one to nine and their reciprocals scale method (shown in table IV). Single and total hierarchical arrangements were processed.

R software was used to complete consistency check. First, the consistency check of a single hierarchical arrangement was processed. The check formula is shown in Equation (9), where $CI = (\text{maximal eigenvalue} - n)/n - 1$, and RI is the random consistency index. If $CR < 0.1$, the matrix of determination was acceptable.

$$CR = \frac{CI}{RI} \tag{9}$$

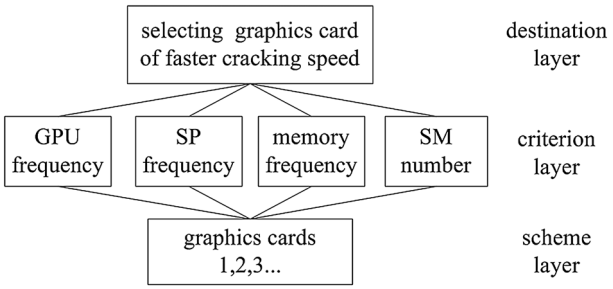

Figure 13. Hierarchy structure of graphics card selection.

Table IV. The construction of determination matrix.

Scale	Definition
1	Equal importance
3	Weak importance
5	Essential importance
7	Very strong importance
9	Absolute importance
2,4,6,8	Intermediate values

DOI: 10.1002/dac

$$A_1 = \begin{bmatrix} 1 & 1 & 7 & 1/5 \\ 1 & 1 & 7 & 1/5 \\ 1/7 & 1/7 & 1 & 1/9 \\ 5 & 5 & 9 & 1 \end{bmatrix}$$

$$B_1 = B_2 = B_3 = \begin{bmatrix} 1 & 1/2 & 1 \\ 2 & 1 & 2 \\ 1 & 1/2 & 1 \end{bmatrix}; B_4 = \begin{bmatrix} 1 & 1/3 & 1/7 \\ 3 & 1 & 1/5 \\ 7 & 5 & 1 \end{bmatrix}$$

Through computing R, the matrixes of A_1, B_1–B_3, and B_4 all passed the consistency check. Second, the consistency check of the total hierarchical arrangement was processed. The check formula is shown in Equation (10), and if $CR_{total} < 0.1$, the consistency check was passed.

$$CR_{total} = \frac{\sum_{j=1}^{n} a_j CI_j}{\sum_{j=1}^{n} a_j RI_j} \tag{10}$$

Through computing, the consistency check of the total hierarchical arrangement was also passed, the final rank of the three graphics cards was 0.145, 0.305, and 0.55. Therefore, the preference was GTX590, the second choice was GTX570, and the last choice was GTX 465.

7.3. Performance optimization

Reference [28] gave overall performance optimization strategies that were maximizing parallel execution, optimizing memory usage to achieve maximum memory bandwidth, and optimizing instruction usage to achieve maximum instruction throughput. Aiming DMCG, the execution configuration optimizations and memory optimizations were processed.

7.3.1. Execution configuration optimization. One of the keys to good performance was to keep the SM as busy as possible. The metric was the occupancy that was the ratio of the number of active warps per SM to the maximum number of possible active warps per SM. Higher occupancy did not always equate to higher performance; however, low occupancy always interfered with the ability to hide memory latency, resulting in performance degradation [28]. Especially, to hide latency arising from register dependencies, approximately 25% occupancy should be maintained as a minimum [28]. Occupancy was determined by several parameters such as TPB, registers per thread, shared memory per block, shared memory size of each SM, and register file size of each SM. The tool of CUDA occupancy calculator (COC) [37] could be used to compute the occupancy.

In DMCG, each thread used 34 registers, and each block is comprised of 192 threads without shared memory. The compute capability of GeForce GTX480 graphics card for PC1 was 2.0, LimitWarpsperSM was 48, and the other parameters were presented in Reference [37]. The theoretical occupancy computed by CUDA occupancy calculator is shown in Figure 14.

The occupancy of DMCG was 62.5% and satisfied the minimum optimization demand. However, in the practical application, the performance was also influenced by several other factors such as block switch, computational complexity of each thread, and memory latency. The block size should be determined from several optimal values (192, 230, and 960 as shown in Figure 14). The block size of DMCG was changed, and it was concluded that when the block size was 320, the performance was optimum and the cracking speed was improved from 31,186.68 to 34,734.24 (k/s).

DOI: 10.1002/dac

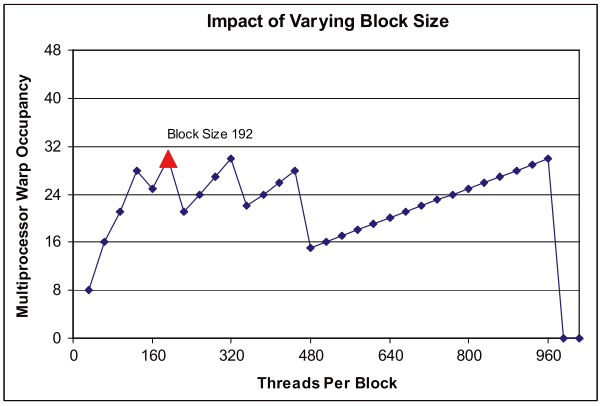

Figure 14. Theoretical occupancy of DMCG.

7.3.2. Memory optimization. The goal was to maximize the use of the hardware by maximizing bandwidth.

(1) Scheme 1 - Increasing SM occupancy

The shared memory was within a GPU chip, the access speed was much faster than the local/global memory and lower than a register. As shown in Scheme 1 (shown in Figure 15), part of the registers that each thread used was accessed by the shared memory in order to increase the SM occupancy. However, the occupancy of DMCG was 62.5%. In fact, once the occupancy of 50% had been reached, additional increases in the occupancy did not translate into improved performance [28]. Therefore, Scheme 1 was infeasible. Moreover, in DMCG, each thread computes 4095×2 of hash operations, and the registers should be accessed frequently. Hence, although increasing the SM occupancy, Scheme 1 could not compensate the longer latency that was brought by the usage of shared memory. The experimental result indicated that the cracking speed declined. Consequently, Scheme 1 failed.

(2) Scheme 2 - Global to shared memory access

In DMCG, in each time of hash operation, pre-computed f(IV, (K $\oplus$ ipad)) and f(IV, (K $\oplus$ opad)) that saved in the ipad and opad filed of the global memory respectively were fetched from the global memory, and the latency was large. In Scheme 2 (shown in Figure 16), shared memory was used instead of global memory, the pre-computed f(IV,(K $\oplus$ ipad)) and f(IV,(K $\oplus$ opad)) were fetched from global memory to

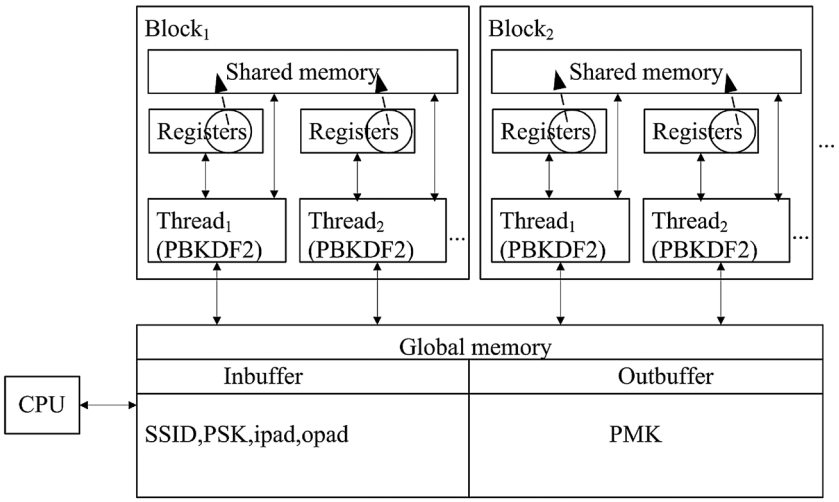

Figure 15. Scheme 1 - Increasing SM occupancy.

DOI: 10.1002/dac

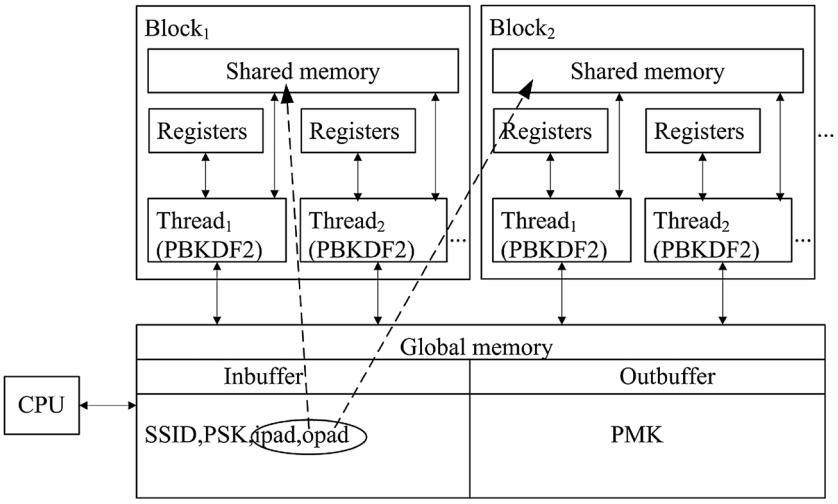

Figure 16. Scheme 2 - Global to shared memory access.

shared memory only once. In subsequent each time of hash operation, the access mode was changed from global memory access to shared memory access, so the memory bandwidth and performance would be increased. The experimental result indicated that the cracking speed was improved from 31,186.68 to 32,175.36 (k/s). The improvement was not much because the work load of access of pre-computed f (IV,(K $\oplus$ ipad)) and f(IV,(K $\oplus$ opad)) was much lower than that of the other hash operations.

Therefore, taking execution configuration and memory optimizations into account, the superimposed improvement was about 15%.

7.4. Comparison with other brute forcers

Table V summarized all the characteristics of cracking methods and the improvement effects of cracking speed that were discussed in this paper.

According to the discussion of the experiment and Table III, non-distributed DMCG could improve the cracking speed by two orders of magnitude approximately, compared with the computing power of one CPU core. The cracking speed could be improved by three orders of magnitude with GDMCG approximately, compared with the computing power of one CPU core. The cracking speed could be improved by four orders of magnitude with HDMCG approximately, compared with the computing power of one CPU core.

Compared with non-distributed Pyrit and EWSA, due to the performance optimization of Section 7.3, DMCG adopted the optimized block size and memory structure of program, and the performance improvement was about 15%.

Table V. The comparison of cracking methods.

Cracking methods	Multi-core CPU	GPU	Distributed method	Real time PSK list transmission	Improvement effect of cracking speed
One CPU core	Not	Not	Not		Two orders of magnitude (non-distributed DMCG); three orders of magnitude (GDMCG); four orders of magnitude (HDMCG)
Pyrit	Yes	Yes	Yes	Yes	Non-distributed Pyrit:15% (non-distributed DMCG); distributed Pyrit:100% (DMCG)
EWSA	Yes	Yes	Not		15% (non-distributed DMCG)
ZerOne	Yes	Yes	Yes	Yes	50% (DMCG)
DMCG	Yes	Yes	Yes	No	

DMCG, distributed multi-core CPU and GPU parallel cracking method; GDMCG; general performance DMCG; HDMCG, high-performance DMCG; EWSA, single-PC software wireless security auditor.

DOI: 10.1002/dac

Compared with Pyrit, DMCG adopted the automatic PSK list traversal (Section 7.1). The network overhead was low. We presumed that the update cycle of DC − t was 5 s, without regard to the performance improvement of non-distributed DMCG to non-distributed Pyrit, and the cracking speed of DMCG and Pyrit (with mainstream hardware) was 2×10^5 (k/s) (5×10^{-6} s per PSK). According to the discussion of Section 7.1, the in 100-M Ethernet environment, the network overhead of each PSK of DMCG was $0.08\times10^{-3}/(2\times10^5\times5)\approx8\times10^{-11}$ s. The cracking speed of DMCG was $1/(5\times10^{-6}+8\times10^{-11})\approx2\times10^5$ (k/s). The distributed Pyrit adopts real time PSK list transmission, the network overhead of each PSK was 64B/100 Mbps × 8 ≈ 5×10^{-6} s. The cracking speed of Pyrit was $1/(5\times10^{-6}+5\times10^{-6})\approx10^5$ (k/s). Therefore, the improvement effect of the cracking speed was about 100%.

Compared with ZerOne, the website of ZerOne said that the cracking speed of ZerOne was 2×10^6 (k/s) in GPU environment, when 100 PCs participated in the cracking [22]. The average cracking speed of each PC was 2×10^4 (k/s) lower than PC1 (32,921 (k/s)) in our experiment. The improvement effect of the cracking speed was about 50%. However, this project was internal, and the details of PC hardware could not be acquired. The comparison of the cracking speed might not be absolutely accurate. ZerOne was based on rainbow table, and PSKs were converted into PMKs beforehand, and the time-consuming hash operations were removed in the cracking process. However, compared with DMCG, the performance improvements were not shown from the cracking speed data from ZerOne. Therefore, ZerOne still required further study and optimization. Moreover, the cracking process of ZerOne would fail if the SSID was wrong in the rainbow table built beforehand.

8. CONCLUSION

This paper proposed a new method of cracking WPA/WPA2-PSK-DMCG. It adopted distributed technology and current widespread multi-core CPU and GPU to crack WPA/WPA2-PSK in parallel. Experimental result showed that the improvement effect was apparent. This method had two level parallelisms: cracking in parallel among PCs and among computing cores in single PC and furthest utilized the computing resource, especially, adopting GPGPU to improve the cracking speed greatly. In SWLAN, as the traditional cracking method was invalid, this method was still valid. Using DCM, this method could be expanded to cloud computing based on GPU. The factors affecting the cracking speed was analyzed, and accordingly, decision support was provided by the analytic hierarchy process. In the last part, the performance was optimized.

In the next work, we should study the means of defense against DMCG and the implementation and analysis of DCM.

REFERENCES

1. Lee Y, Lee M, Choi C-H. Arbitration interframe space-controlled medium access control: a medium access control protocol guaranteeing absolute priority in wireless local area networks. *International Journal of Communication Systems* 2013; **26**(7):832–852.
2. Park H, Shon T, Park S, Kim E. Adaptive anomaly control for alleviating the exclusive channel occupation in wireless networks. *International Journal of Communication Systems* 2013; **26**(6):720–731.
3. Jibukumar MG, Datta R, Biswas PK, Kumar BP. Busy tone contention protocol: a new high-throughput and energy-efficient wireless local area network medium access control protocol using busy tone. *International Journal of Communication Systems* 2012; **25**(8):991–1014.
4. IEEE Computer Society. IEEE Standard for information technology-telecommunications and information exchange between systems-local and metropolitan area networks-specific requirements-part 11: wireless LAN medium access control (MAC) and physical layer (PHY) specifications. IEEE Std 802.11TM-2007 (Revision of IEEE Std 802.11-1999), 2007.
5. Yonglei L, Zhigang J, Ying W. Survey on security scheme and attacking methods of WPA/WPA2. *Proceedings of 6th International Conference on Wireless Communications Networking and Mobile Computing*, Chengdu, CHN, 2010.
6. Beck M, Tews E, Practical attacks against WEP and WPA. *Proceedings of the 2nd ACM Conference on Wireless Network Security*, New York: ACM, 2009, 79–86.
7. Halvorsen FM, Haugen O. Cryptanalysis of IEEE 802.11i TKIP. Norwegian University of Science and Technology, 2009.

DOI: 10.1002/dac

8. Beck M. Enhanced TKIP Michael attacks http://download.aircrack-ng.org/wiki-files/doc/enhanced_tkip_michael.pdf, 2010.
9. Moen V, Raddum H, Hole KJ. Weaknesses in the temporal key hash of WPA. *ACM SIGMOBILE Mobile Computing and Communications Review* 2004; **8**(2):76–83.
10. RuiAC. Ferreira a probability problem arising from the security of the temporal key hash of WPA. *Wireless Personal Communications* 2012; 1–7.
11. Junaid M, Mufti M, Umar Ilyas M. Vulnerabilities of IEEE 802.11i Wireless LAN CCMP Protocol. *Transactions on Engineering, Computing and Technology* 2006; **11**:228–233.
12. Liu C, Yu JT, Brewster G. Empirical studies and queuing modeling of denial of service attacks against 802.11 WLANs, *IEEE International Symposium on "A World of Wireless, Mobile and Multimedia Networks"*, Montreal: IEEE Computer Society, 2010; 1–9.
13. Liu C, Yu JT. A solution to WLAN authentication and association DoS attacks. *IAENG International Journal of Computer Science* 2007; **34**(1):31–36.
14. Malekzadeh M, Ghani AAA, Subramaniam S. A new security model to prevent denial-of-service attacks and violation of availability in wireless networks. *International Journal of Communication Systems* 2012; **25**(7):903–925.
15. Mishra A, Arbaugh WA. An initial security analysis of the IEEE 802.1X Standard, http://www.cs.umd.edu/~waa/1x.pdf, 2002.
16. Malekzadeh M, Azim A, Ghani A, Desa J, Subramaniam S. Vulnerability analysis of extensible authentication protocol (EAP) DoS attack over wireless networks. *ICGST International Journal on Computer Network and Internet Research CNIR* 2009; **9**(1):39–46.
17. Pelechrinis K, Iliofotou M, Krishnamurthy SV. Denial of service attacks in wireless networks: the case of jammers, *IEEE Communications Surveys & Tutorials* 2011; **13**(2):245–257.
18. Yonglei L, Zhigang J, Ming G, Zhe C. ARJ: an IEEE 802.11g all-channel jammer with alterable jamming radius. *Chinese Journal of Computers* 2013; **36**(4):870–881. (In Chinese).
19. Aspyct.org, Aircrack-ng, http://www.aircrack-ng.org, 2013.
20. EWSA. http://www.elcomsoft.com/ewsa.html, 2013.
21. Pyrit. http://code.google.com/p/pyrit, 2013.
22. ZerOne. http://forum.anywlan.com/thread-12122-1-1.html, 2013.
23. Jensen K. *Colored Petri Nets: Basic Concepts, Analysis Methods and Practical Use*, Volume **1**. Monographs in Theoretical Computer Science, Berlin: Springer-Verlag, 1992.
24. Huang HJ, Kirchner H. Secure interoperation design in multi-domains environments based on colored Petri nets, *Information Sciences* 2013; **221**:591–606.
25. Nvidia. CUDA C Programming Guide. http://docs.nvidia.com/cuda/pdf/CUDA_C_Programming_Guide.pdf, 2013.
26. Zheng Z, Soon SH, Kwang QC, Jixiang S. GPU-accelerated real-time tracking of full-body motion with multi-layer search, *IEEE Transactions on Multimedia* 2013; **15**(1):106–119.
27. The Van Luong, Nouredine M, Talbi RI-G. GPU computing for parallel local search metaheuristic algorithms. *IEEE Transactions on Computers* 2013; **62**(1):173–185.
28. Nvidia. CUDA C best practices guide. http://docs.nvidia.com/cuda/pdf/CUDA_C_Best_Practices_Guide.pdf, 2013.
29. Amdahl GM. Validity of the single processor approach to achieving large-scale computing capabilities. *Proceedings of AFIPS Conference*, 1967; 483–485.
30. Rimal BP, Choi E. A service-oriented taxonomical spectrum, cloudy challenges and opportunities of cloud computing, *International Journal of Communication Systems* 2012; **25**(6):796–819.
31. Basic-English.http://www.basic-english.org/, 2013.
32. Yuan Y-D, Zhong W-L. *Contemporary Chinese Family Name (in Chinese)*. JiangXi People's Publishing House: Nanchang, Jiangxi, China, 2006.
33. Combs G. Wireshark, http://www.wireshark.org/, 2013.
34. Ren H, Zhang Y, Lin S. GPU-accelerated video copy detection based on incremental clustering. *Journal of Computer-Aided Design & Computer Graphics* 2010; **22**(3):449–456. (In Chinese).
35. Saaty TL. The analytic hierarchy process: planning, priority setting, resource allocation. McGraw-Hill, 1980.
36. Anuar NB, Papadaki M, Furnell SM, Clarke NL. Incident prioritisation using analytic hierarchy process (AHP): Risk Index Model (RIM). *Security and Communication Networks* 2013; **6**(9):1087–1116.
37. Nvidia. CUDA occupancy calculator. http://developer.nvidia.com/nvidia-gpu-computing-documentation, 2013.

DOI: 10.1002/dac