An Approach to Reduce Side-Channel Timing Attack in Dragonfly Handshake of WPA3 for MODP Group

*1Ikramul Islam, 2Suvendu Barai & 3A.K.M. Fazlul Haque

*1Student, Department of Electronics and Telecommunication Engineering, Daffodil International University, Dhaka, Bangladesh

2 Student, Department of Electronics and Telecommunication Engineering, Daffodil International University, Dhaka, Bangladesh

3 Professor, Department of Electronics and Telecommunication Engineering, Daffodil International University, Dhaka, Bangladesh

Abstract

Wireless security has become a great concern in the era of 4th Industrial Revolution due to the expansion of wireless network and increased number of wireless devices. Blocking unauthorized users to a wireless network is a significant part of wireless security. The newly Wi-Fi Protected Access 3 (WPA3) announced as the successor of WPA2 by Wi-Fi Alliance, used to prevent unauthorized access in a wireless network. Dragonfly handshake is used by WPA3 for mutual authentication between a client and an access point. While systematically evaluating Dragonfly's security, a serious flaw namely 'Side Channel Timing Leak' in password conversion method, evidently released information about the password came across. Dragonfly supports both 'Modulo the Prime (MODP)' and 'Elliptic Curve Cryptography (ECC)' as password conversion method. In this paper, MODP group taken into consideration to generate Password Element (PE). Execution time differences are measurable during PE formation which results in certain plausibility to guess the password with the assistance of spoofed client mac addresses has been demonstrated. How leaked information creates signature of the password testified and to make it computationally intractable for an attacker to manipulate the flaw, three actions: Fixing iterations number, single password based PE database generation and fetching a PE of random choice from the database have been proposed. Finally, the complication for an attacker to hack the password is illustrated.

1. Introduction

Several attacks and vulnerabilities especially Key Reinstallation Attack (KRACK) [1, 2] led Wi-Fi alliance to release WPA3 as the successor of WPA2 protocol which also rolled out 14-year-old 4-way handshake by Dragonfly [3] i.e. a robust password based key exchange mechanism through for users authentication on a WPA3 router or access point [4, 5]. There exists a transition mode for backwards compatibility where both WPA2 and WPA3 are simultaneously supported. Dragonfly handles a great deal of responsibility as this handshake is supported by both EAP-pwd protocol and WPA3 certification used by some enterprises and personal Wi-Fi networks respectively to provide not only forward secrecy but also offline dictionary attacks protection [3, 6, 7].

1.1. Simultaneous Authentication of Equals (SAE)

For authentication and key management WPA3 uses a type of Dragonfly key exchange protocol known as SAE based on password-authenticated key exchange (PAKE) between client & access point (AP), first introduced by Dan Harkins to employ in WLAN mesh networks (IEEE 802.11s) [7–10]. As to access the internet using Wi-Fi, password or passphrase has become one of the basic methods for authentication [5, 11]. The same password for layer-2 authentication and generation of encryption keys leaves vulnerability in which an attacker obtains sufficient info [9, 12]. Unsurprisingly, failed to protect against active, passive and off-line dictionary attacks, enhanced protocol offered the protection after correction of the RFC 7664 standard in 2015 [7, 13, 14]. Leveraging discrete logarithmic and elliptic curve cryptography by Dragonfly handshake, this resistance is accomplished [15–18].

1.2. Deriving the Password Element

Rather than deriving the pairwise master key (PMK) for traffic encryption, SAE focuses on proper authentication of a device onto a network using password and MAC addresses. Diffie-Hellman key exchange (DHKE) method is incorporated in Dragonfly based on ECC which is zero-knowledge proof adds an authentication element [19, 20]. Dragonfly corroborates :

- ECC groups: Elliptic curve cryptography with elliptic curve over a prime field [6, 15].
- MODP groups: Finite field cryptography with multiplicative groups modulo the prime [21, 22].

Before initiation of the Dragonfly handshake a password element is formed from pre shared password using a hash-to-element method to compute keys. For MODP groups it is hash-to-group and hash-to-curve for elliptic curves [9, 22, 23]. The PE is determined at the time of the session with domain parameters for FFC groups p, large prime number dictating a prime field GF(p) and q which is another prime number in the multiplicative order of a group G, mutual agreement between client and AP using discrete logarithmic cryptography and follows a hunting-and-pecking technique with password as a seed value [8, 13].

Dragonfly's susceptibility to inherent side-channel attack leaks vital info about time [6, 24]. While producing password elements with MODP group shown how iteration depends on password and client MAC address influences the execution time [6, 22]. A minimum number of iterations recommended by Wi-Fi alliance, supposedly believe to solve the timing problem yet different variants of Dragonfly use different numbers (k=4/40) [25, 26]. Evidently, only minimum iterations can't fix the issue. Found out variation in execution time results with different passwords makes it easier and even simpler with the help of spoofed MAC addresses to catch the signature of the password [6].

1.3. Side-Channel Timing Leak

Recovering password of a Wi-Fi network with WPA3 considered infeasible for an adversary. Though recent researches revealed depending on the Access point's response time to commit frames, information relating password may be leaked [27–29]. As multiplicative groups that AP supports depends on the password being used which is natural in its algorithm and suffers from timing leak attack where it provides different response time. An adversary can abuse this information to produce a dictionary attack by performing simulation. How much time requires for an AP to process each password and compare this to observed timings [6, 27, 30, 31].

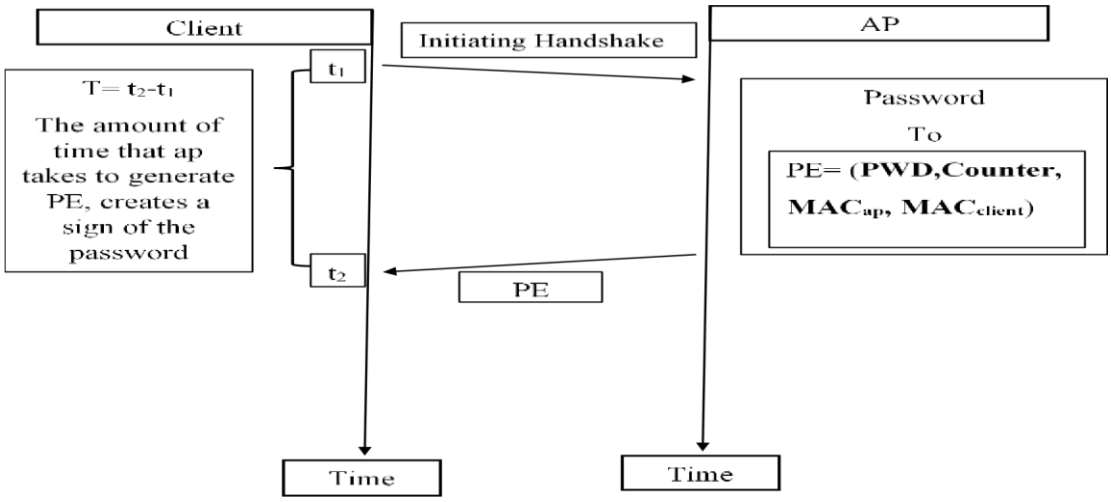

Figure 1. Simple Pictorial Representation of Side-Channel Timing Attack

More openness creating WPA3 and Dragonfly could have mitigated most attacks as while designing and analyzing MODP element technique assessed, this is essentially flawed. Exclusion of peer's identities or utilizing constant-time algorithms might have been a notable approach [6, 32–34]. Proposed scheme in this paper comes with:

- Fixating maximum and minimum iterations which makes it less vulnerable in secure practice.
- Database of PEs from a single password and arbitrary calling of a PE raise complexity for an adversary.
- Calculated the conditional probability to analyze the inverse functionality between increased number of PE and actual PE detection, only except requiring a memory for elements storage.

2. Materials & Method

2.1 Conversion of Password into Password Element

Conversion of a password into password element (PE) is described in Flow chart 1 (Figure 2). Here an input password is manually taken. The MAC address of the AP and the Client are also taken as inputs. Then base is calculated using the passwords, MAC addresses and the counter. SHA512 is used here. After that , Key Derivation Function (KDF) is calculated using the BASE and PKDF2, then the Seed is calculated using the KDF. At this point same length of bit string is hold by both of the Seed and prime (p). When the condition (seed<prime) is satisfied by the seed, Temp is calculated using the Seed. Finally, when Temp is greater than 1, then password element is given as output. But if both the condition is not satisfied by Seed and Temp respectively, then counter will be increased by 1 and Base will be calculated again [13].

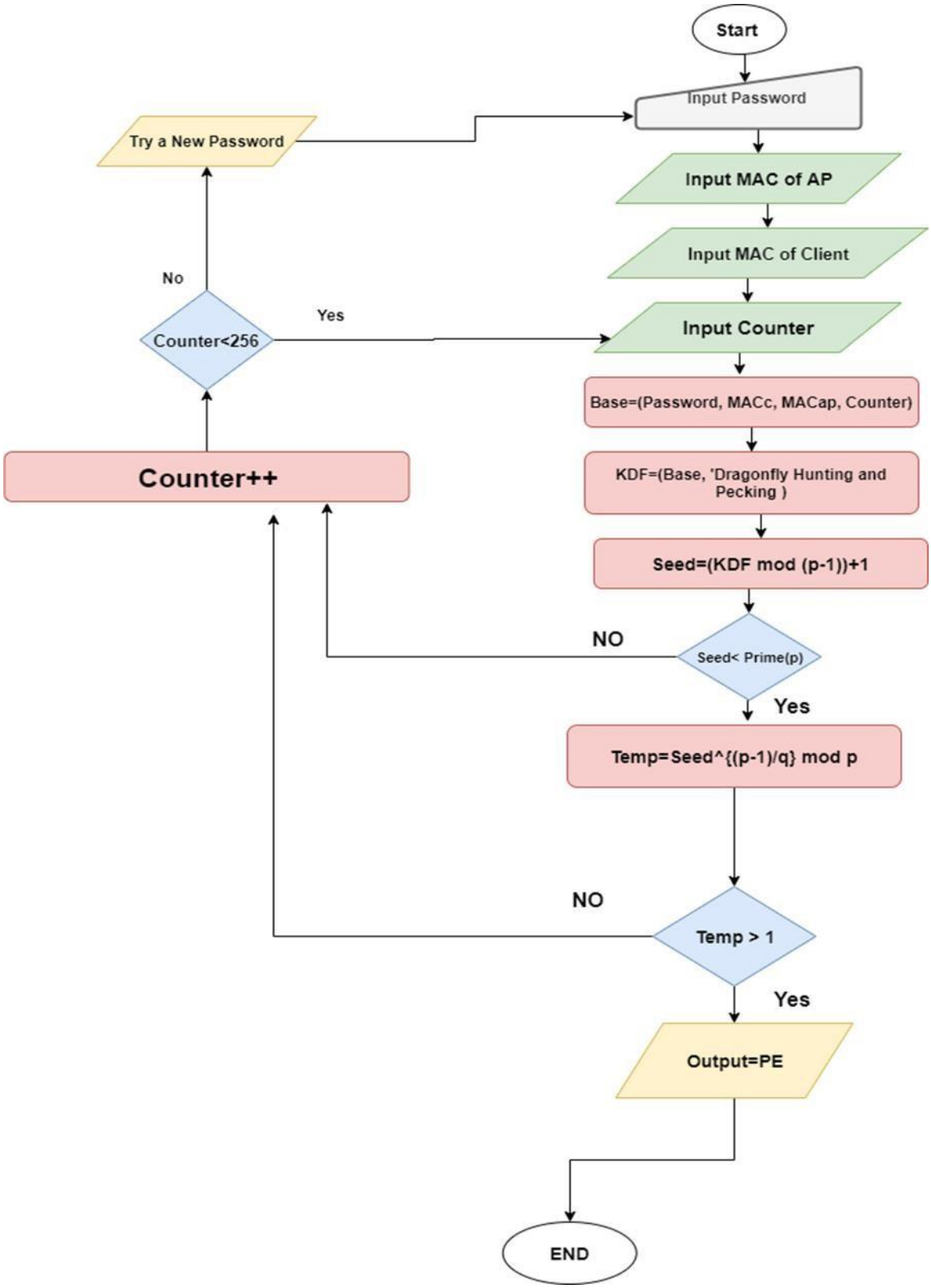

Figure 2. Flow Chart 1 Conversion of Password Into PE

2.2 Identifying The flaws

2.2.1 Iteration Depends on Password

It is clearly shown in the MODP algorithm that password influences the Base and the counter because MAC address is unique for both a Client and an AP device.

BASE = (Password, MAC of client, MAC of AP, Counter)

As base is influenced by the password, it can be easily said that KDF, Seed and Temp are also influenced by the password. If seed is not less than the prime or Temp is not greater than 1, then only the counter is incremented by 1 until counter is equal to 255. Here counter<256 is used because for MODP the probability of finding PE is near about zero when counter is greater than 255 [2, 6, 35].

So it can be said that more execution time of PE is taken by more counter or iterations/loops. If the conditions are not fulfilled then code that calculates the PE is skipped. Suppose $t_1,t_2,t_3,$…….time is taken by $n_1,n_2,n_3,$…… number of iterations respectively n_1,n_2,n_3 number of iteration/iterations is/are needed by totally different password p_1,p_2,p_3……and The signature or the information of the password is leaked by the variation of execution time (Table1). This timing leak information can be found by an adversary using false authentication message. Though the password is unknown to the adversary, still the execution time is found from the rejected message. Then Brute force attack can be easily run by exploiting this information.

2.2.2 Client MAC addresses influence the execution time

A vital role is played by the MAC address of the client in calculation of the base and in overall password element calculation.

Base = (Password, MAC of client, MAC of AP, Counter)

It is assumed that the access point MAC address is fixed. So, the algorithm or the password element is influenced by the client MAC addresses. Different client MAC addresses can be spoofed by an adversary in the input section of the algorithm. For different client MAC addresses the base will be different. Finally, different execution time is required for different PE (Table 2). On an average 17 MAC addresses are needed to find the signature of the password from a RockYou Database[6, 20, 35].

2.3 Proposed Scheme for preventing timing attack

It is tough to implement Dragonfly handshake without side-channel attack because of its design [6, 35–37] In this work, a potential approach is shown in the design section that can create paramount difficulties for an attacker to find the leaked time information. The attacker will be faced a labyrinth by the system of the access point (AP). Added that, the approach can only be used in the AP to authenticate the client.

Fixed number of Max and Min Iterations for a fixed number of Password Element

In MODP algorithm, maximum number of iterations is fixed at 256 as the probability of finding a PE after 255 iteration is almost zero [23, 25, 38]. It has been said to use a fixed number of minimum iterations 'K' in MODP algorithm for security purposes. But the exact number of K has not been suggested. It was found in the previous study that some uses variants K=4 and some uses k=40. In this work K_{min}=31 and K_{max}=255. It is not stated that K_{min} should be 31, rather than producing a fixed number of PE, 'i' is focused in the work. But again a K_{min} should be maintained. If fixed number of PE 'i' are not created and stored in the data base between 1 to K_{min} of iterations the counter will not be incremented. A random call will be taken place. Again, when a fixed number of PE is created, the counter will be stopped and random call will be taken place.

2.3.1 Random Call from the fixed number of PE

When a fixed number of password elements is generated (in this work i=20), a random call will take place. A random PE is selected and provided as the final output by this random call. If the iteration is equal to K_{min} (K_{min} =31 in this work) and enough number of PE elements are not produced, then the random call will still be taken place and choose a PE from the produced database. (Table 4)

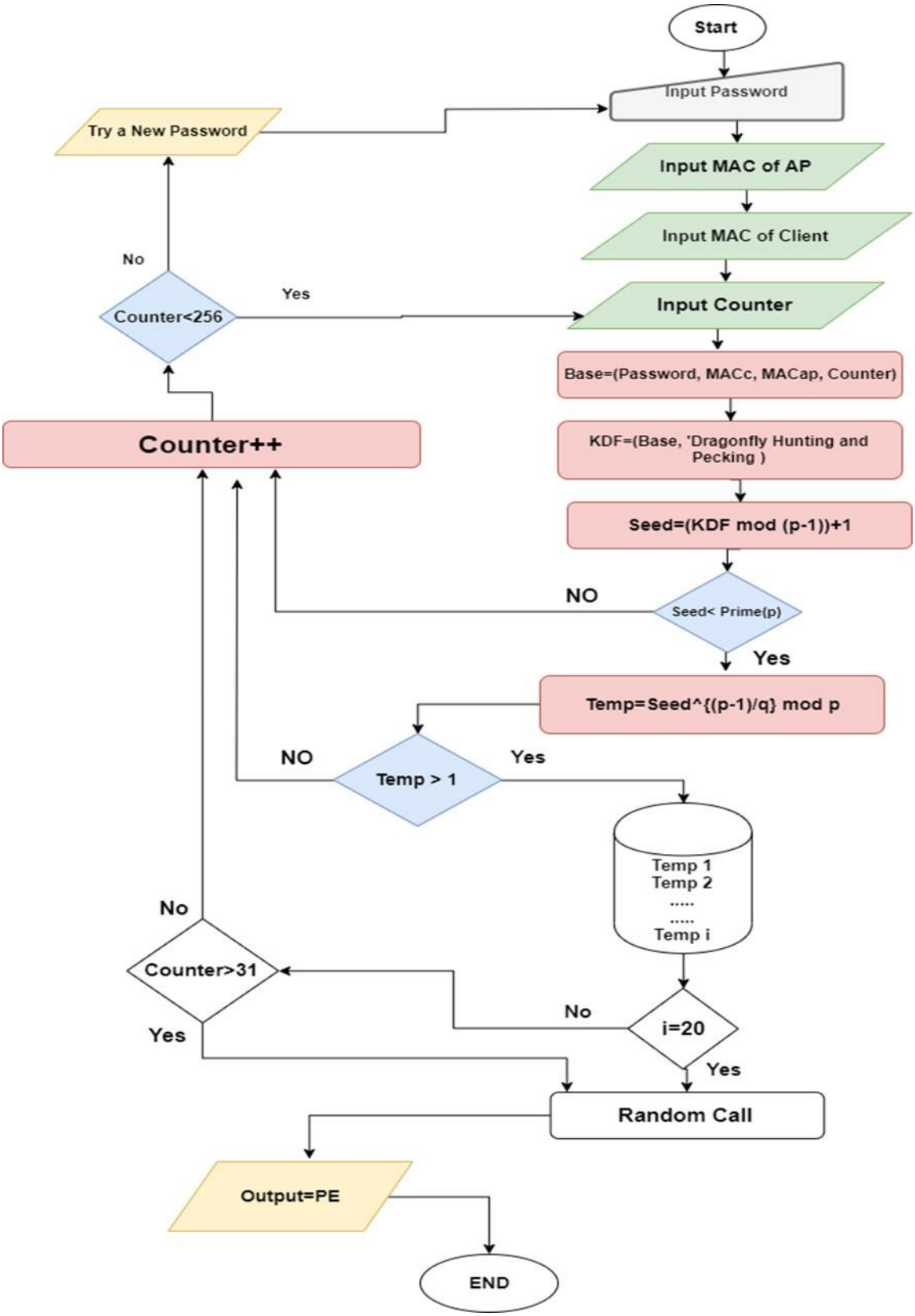

Figure 3. Flow Chart 2 (Proposed Scheme)

2.4 MODP Group-22

This work is done by taking MODP Group-22 as a reference. It is also called 1024-bit MODP Group with 160-bit Prime Order Subgroup-22 [21]. The parameters are given below.

The hexadecimal value of the prime is:

p = B10B8F96 A080E01D DE92DE5E AE5D54EC 52C99FBC FB06A3C6 9A6A9DCA 52D23B61 6073E286 75A23D18 9838EF1E 2EE652C0
13ECB4AE A9061123 24975C3C D49B83BF ACCBDD7D 90C4BD70

98488E9C 219A7372 4EFFD6FA E5644738 FAA31A4F F55BCCC0

A151AF5F 0DC8B4BD 45BF37DF 365C1A65 E68CFDA7 6D4DA708

DF1FB2BC 2E4A4371

The hexadecimal value of the generator is:

g = A4D1CBD5 C3FD3412 6765A442 EFB99905 F8104DD2 58AC507F D6406CFF 14266D31 266FEA1E 5C41564B 777E690F 5504F213
160217B4 B01B886A 5E91547F 9E2749F4 D7FBD7D3 B9A92EE1

909D0D22 63F80A76 A6A24C08 7A091F53 1DBF0A01 69B6A28A

D662A4D1 8E73AFA3 2D779D59 18D08BC8 858F4DCE F97C2A24

855E6EEB 22B3B2E5

The generator generates a prime-order subgroup of size:

q = F518AA87 81A8DF27 8ABA4E7D 64B7CB9D 49462353

2.5 Environment and Tools

The script was run on a Python 3.7 using Jupiter notebook. The processor was an Intel(R) Core (TM) i3-7100U 2.4GHz CPU including 4GB Random Access Memory. System type was 64 bit windows operating system. Though the processing power was too much high comparatively to a access point processor it was suitable enough to demonstrate the works. The data were collected in micro second where in AP the data came out usually in mili second. Different kinds of libraries, modules and functions like Haslib, Random, Binascii, and Crypto were used in the script.

3. Result and Analysis

3.1 Variation in Execution Time

Variations in loops or iterations are needed for the MODP algorithm to convert different passwords into password elements [7, 41]. Differences in execution time for different passwords actually leak very crucial information about the password. Brute force dictionary attack can be easily conducted by the leaked information. Different execution time for different password is shown in the Table 1.

Table 1. Variation in Execution Time

Serial Number	Loops/ Iteration	Execution Time	
		Execution Time Per Loop	Total Execution Time
01.	01	103 µs± 107 µs	103 µs
02.	02	53.8µs±17.8µs	107.68 µs
03.	03	53.8µs±25.2µs	161.4 µs
04.	04	112 µs±75.4µs	448 µs
05.	05	52.7µs±7.04µs	263.5 µs
06.	06	55.8µs±9.41µs	334.8 µs
07.	07	60.4µs±12.9µs	422.8 µs
08.	08	68.7µs±4.35µs	549.6 µs
09.	09	65.5µs±24.5µs	589.5 µs
10.	10	58.7µs±9.37µs	587 µs

3.2 Guessing Password using Different Client MAC Address

Here the MAC address of the AP is MAC AP - 00:10:6A:44:12:B4. The client MAC address is changed gradually from number 1 to 17 (Table 2), these MAC addresses are used randomly only for testing purpose in the script. Finally, it is found that by spoofing different MAC addresses from the client side, an unknown password can be identified using the leaked response time from the AP. Point to be noted that the password is same all the time.

Table 2. Guessing Password using Different Client MAC Address

Serial Number	MAC Address	Execution Time	
		Mean	Standard Deviation
01.	60:A8:AA:D0:EA:77	56.8µs	±15.9 µs
02.	AE:4E:E5:7F:D4:29	24.0µs	± 11.4 µs
03.	F8:3F:16:13:45:B7	61.8µs	± 7.32 µs
04.	B4:16:7A:E8:0E:68	110µs	±17.4 µs
05.	07:6E:73:1A:AD:E9	49.0µs	± 15.4 µs
06.	AD:94:E6:A7:D7:79	65.3µs	±13.8 µs
07.	85:F2:14:ED:0C:38	65.0µs	± 15.1 µs
08.	13:64:19:E6:BB:FF	63.6µs	± 33.8 µs
09.	90:AE:4E:A0:88:42	60.2µs	± 24.9 µs
10.	CA:A3:6A:D5:CF:2C	48.5µs	± 19.2 µs
11.	07:6E:73:1A:AD:E9	70.9µs	± 55.0 µs
12.	94:E6:A7:D7:79:85	52.7µs	± 9.18 µs
13.	14:ED:0C:38:A0:44	48.7µs	± 8.32 µs
14.	00:76:7C:D0:2C:FA	63.2µs	± 26.9 µs
15.	24:5B:E0:DC:73:53	64.3µs	± 13.6 µs
16.	37:D9:BB:E9:DB:28	91.7µs	± 23.4 µs
17.	AF:74:38:43:15:24	61.8µs	± 33.1 µs

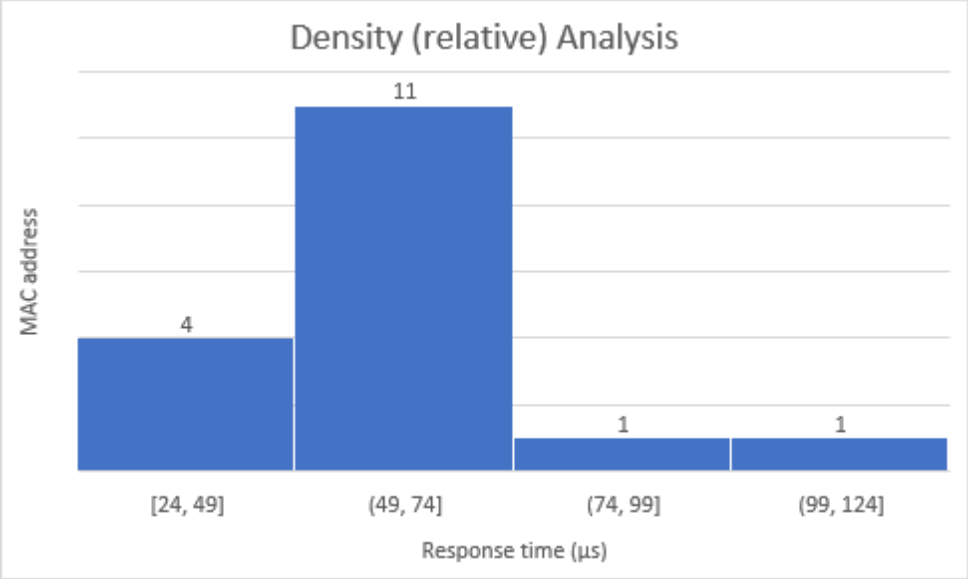

Figure 3. Density Analysis of The Response time for Different MAC Address

From figure 3 it was found that maximum number of response time against different client MAC address was between 49 to 74 micro seconds. Vital details about password was leaked from the information as it has become easier for an attacker to guess a password by brute-forcing dictionar. The words will be chosen then, those are executed between the leaked time information (49-74 microseconds). The actual password was executed near about 65 micro seconds with some standard deviation in the script.

3.3 Certain Number of Password Elements from Same Password

Table 3

Serial No.	Password	N_{th} Number of Iteration	Hex Code of Password Element (Rest are 0x0)	Execution Time Per Iteration (mean ± std. dev.)	Total Execution Time
01	TAposh24#92 ^!45%m1=+	1	0x8ae6fe948442d8	56.5 µs ± 35.1 µs (mean ± std. dev. of 7 runs, 1 loop each)	56.5 µs
02		2	0xa7d61a86105160	75.3 µs ± 45.1 µs (mean ± std. dev. of 7 runs, 2 loops each)	150.6 µs
03		3	0x32c9ce00367aca	81.6 µs ± 36.3 µs (mean ± std. dev. of 7 runs, 3 loops each)	244.8 µs
04		5	0xdba5126ad6877	90.8 µs ± 56.1 µs (mean ± std. dev. of 7 runs, 3 loops each)	454 µs
05		6	0x22f01d94a857a6	80.2 µs ± 26.7 µs (mean ± std. dev. of 7 runs, 6 loops each)	481.2 µs
06		7	0x2f1ad3a73283ee	115 µs ± 118 µs (mean ± std. dev. of 7 runs, 6 loops each)	805 µs
07		9	0x30a504d3505994	95.6 µs ± 9.82 µs (mean ± std. dev. of 7 runs, 9 loops each)	860.4 µs
08		11	0x74dcda37506a78	82.7 µs ± 13.6 µs (mean ± std. dev. of 7 runs, 11 loops each)	909.7 µs
09		12	0x8df17825e01630	85.3 µs ± 254 µs (mean ± std. dev. of 7 runs, 11 loops each)	1023.6 µs
10		15	0x16d9048e95de4b	80.6 µs ± 7.54 µs (mean ± std. dev. of 7 runs, 15 loops each)	1209 µs

Serial No.	Password	N^{th} Number of Iteration	Hex Code of Password Element	Execution Time Per Iteration (mean± std. dev.)	Total Execution Time
12	TAposh24#92 ^!45%m1=+	17	0x3dc3012e297d38	75.9 µs ± 7.34 µs (mean ± std. dev. of 7 runs, 17 loops each)	1290.3 µs
13		18	0x868703ba3f20e8	74.2 µs ± 16.2 µs (mean ± std. dev. of 7 runs, 18 loops each)	1335.6 µs
14		19	0x252af6b9733c8a	76.2 µs ± 11 µs (mean ± std. dev. of 7 runs, 19 loops each)	1447.8 µs
15		20	0x3cf16151e3c292	96.5 µs ± 61.2 µs (mean ± std. dev. of 7 runs, 20 loops each)	1930 µs
16		22	0x146203d5be8c73	111 µs ± 137 µs (mean ± std. dev. of 7 runs, 22 loops each)	2442 µs
17		26	0x355e5ad9bea6fc	97.3 µs ± 11.1 µs (mean ± std. dev. of 7 runs, 26 loops each)	2529.8 µs
18		27	0x3ca5f055f6ab9e	94.1 µs ± 24.4 µs (mean ± std. dev. of 7 runs, 27 loops each)	2540.7 µs
19		29	0x336f8ec3cdcd90	88.1 µs ± 49.2 µs (mean ± std. dev. of 7 runs, 29 loops each)	2554.9 µs
20		31	0x44deb294c0ff28	84.1 µs ± 18.1 µs (mean ± std. dev. of 7 runs, 31 loops each)	2607.1 µs

The designed work is to generate multiple password elements from a single password. All those PE in Table 3 are generated at different iteration with different execution time but the password was always same. Total 20 PEs are generated from 31 iterations and the final PE is provided after 31 iterations at near about 2500 micro seconds. A graphical representation of different iterations and execution time is shown in Figure 4.

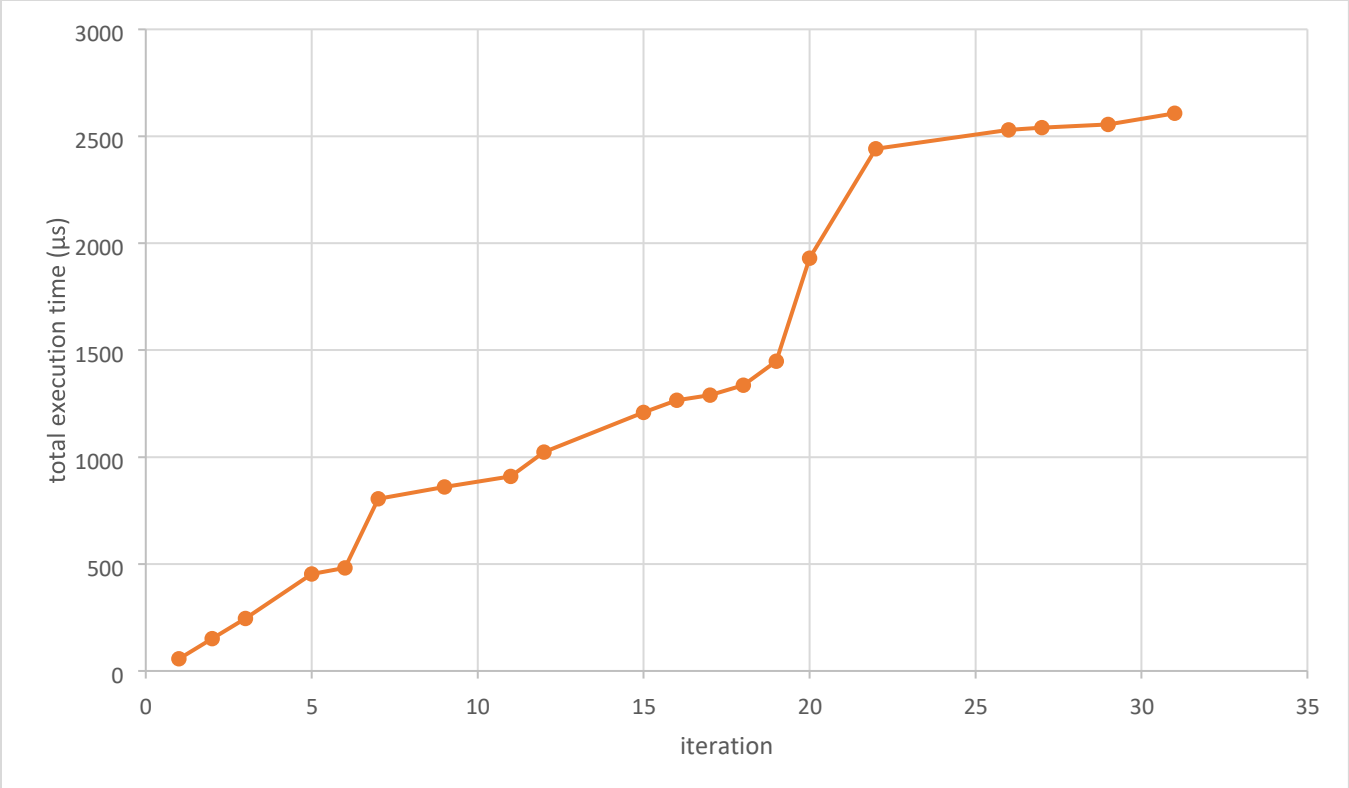

Figure 4 Graphical Representation of PEs Generation (Iteration Vs Execution Time)

3.4 Random Call

According to the proposed approach five random calls are done from the total number of generated password elements. Random module from Python 3.7 is used for this purpose. Different random call outputs different PE generated at different iteration and time is shown in Table 4.

Table 4. Random Calls

Random Call	Password Element	Iteration Number	Generation Time	Execution Time
1	0x16d9048e95de4b	15	1209 µs	2609.1 µs
2	0xdba5126ad68770	5	454 µs	2594.42 µs
3	0x252af6b9733c8a	19	1447.8 µs	2607.95 µs
4	0x3cf16151e3c292	20	1930µs	2625.3 µs
5	0x7151a7133a44c4	16	1265.6 µs	2689.49 µs

Conclusion

It is evident that number of PE generation at different iteration with different execution time from a single password is possible. Table 4 states that a random call delivers an unpredictable PE from the generated password element database. Hence, it is extremely difficult if not impossible for an adversary to run reverse engineering and manipulate the information of the output PE to find out the actual password because of the randomness. Though timing information can be obtained from the rejected responses, but the probability of matching the execution time and the executed PE is 1/i (i is the number of PE generated). The probability is 0.05 in this work. With the increase of the number 'i' the probability can be reduced to a great margin or even zero. Extra memory requirement for the storage of PE and the incurred computation cost is a challenge for this approach. Practical deployment on WPA3 supported AP and compatibility testing with the existing devices as well as sophisticated modifications for cost-effectiveness will be another exciting work.

*** This work is based on the report that was presented in Partial Fulfillment of the requirement of the Degree of Bachelors of Science in Electronics and Telecommunication Engineering at Daffodil International University in January, 2020 [22]

References

[1] M. Vanhoef and F. Piessens, "Key Reinstallation Attacks: Forcing Nonce Reuse in WPA2," in *Computer and Communications Security*, 2017, pp. 1313–1328, doi: 10.1145/3133956.3134027.

[2] M. Vanhoef and F. Piessens, "Release The Kraken: New kracks in the 802.11 standard," in *Proceedings of the ACM Conference on Computer and Communications Security*, 2018, pp. 299–314, doi: 10.1145/3243734.3243807.

[3] Wi-Fi Alliance, "Wi-Fi Alliance® introduces Wi-Fi CERTIFIED WPA3™ security." https://www.wi-fi.org/news-events/newsroom/wi-fi-alliance-introduces-wi-fi-certified-wpa3-security (accessed Oct. 24, 2019).

[4] K. M. Igoe, "[Cfrg] Status of DragonFly," 2012. https://mailarchive.ietf.org/arch/msg/cfrg/_BZEwEBBWhOPXn0Zw-cd3eSV6pY/ (accessed Oct. 26, 2019).

[5] D. Abdalla, M.; Pointcheval, “Simple Password-Based Encrypted Key Exchange Protocols,” in *BT - Topics in Cryptology – CT-RSA*, 2005, pp. 191–208.

[6] M. Vanhoef and E. Ronen, “Dragonblood: Analyzing the Dragonfly Handshake of WPA3 and EAP-pwd,” in *PROCEEDINGS - IEEE Symposium on Security and Privacy*, May 2020, vol. 2020-May, pp. 517–533, doi: 10.1109/SP40000.2020.00031.

[7] D. Harkins, “Simultaneous Authentication of Equals: A Secure, Password-Based Key Exchange for Mesh Networks,” in *Proceedings of the 2008 Second International Conference on Sensor Technologies and Applications*, 2008, pp. 839–844, doi: 10.1109/SENSORCOMM.2008.131.

[8] D. Harkins, “Secure Pre-Shared Key (PSK) Authentication for the Internet Key Exchange Protocol (IKE),” vol. 2, 2012.

[9] V. Smyshlyaev, S.; Alekseev, E.; Oshkin, I.; Popov, “The Security Evaluated Standardized Password-Authenticated Key Exchange (SESPAKE) Protocol,” pp. 1–51, 2017.

[10] G. Harkins, D.; Zorn, “Extensible Authentication Protocol (EAP) Authentication Using Only a Password,” 2010, [Online]. Available: https://datatracker.ietf.org/doc/html/draft-harkins-emu-eap-pwd-14 (accessed Oct. 29, 2019).

[11] B. Harsha, J. Blocki, J. Springer, M. Dark, and R. Morton, “Bicycle attacks considered harmful: Quantifying the damage of widespread password length leakage,” *Computers and Security*, vol. 100, 2021, doi: 10.1016/j.cose.2020.102068.

[12] M. Lass, “Security of WiFi networks,” 2019.

[13] D. Harkins, “Dragonfly Key Exchange,” pp. 1–18, 2015.

[14] J. Lancrenon and M. Škrobot, “On the provable security of the dragonfly protocol,” in *International Conference on Information Security*, 2015, vol. 9290, pp. 244–261, doi: 10.1007/978-3-319-23318-5_14.

[15] R. Schoof, “Elliptic Curves Over Finite Fields and the Computation of Square Roots mod p,” *Mathematics and Computation*, vol. 44, no. 170, p. 483, 1985, doi: 10.2307/2007968.

[16] L. M. Adleman and J. DeMarrais, “A subexponential algorithm for discrete logarithms over all finite fields,” in *Advances in Cryptology — CRYPTO’ 93*, 1994, vol. 773 LNCS, no. 203, pp. 147–158, doi: 10.1007/3-540-48329-2_13.

[17] E. Savaş and Ç. K. Koç, “Finite field arithmetic for cryptography,” *IEEE Circuits and Systems Magazine*, vol. 10, no. 2, pp. 40–56, 2010, doi: 10.1109/MCAS.2010.936785.

[18] N. P. Smart, “Discrete logarithm problem on elliptic curves of trace one,” *Journal of Cryptology*, no. 97–128, pp. 193–196, 1997.

[19] W. Diffie and M. Hellman, “Diffie–Hellman Key Exchange,” *Encyclopedia of Cryptography and Security*, pp. 342–342, 2011, doi: 10.1007/978-1-4419-5906-5_1293.

[20] M. Vanhoef and F. Piessens, “Predicting, Decrypting, and Abusing WPA2/802.11 Group Keys,” 2016, [Online]. Available: https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/vanhoef (accessed Oct. 15, 2019)

[21] T. Kivinen and M. Kojo "RFC 3526: More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE)," *The Internet Society*, May 2003. https://www.hjp.at/doc/rfc/rfc3526.html (accessed Oct. 10, 2019).

[22] I. Islam, S. Barai and A. H. Moon, "Reduced Side Channel Timing Attack in Dragonfly Handshake of WPA3 for MODP Group," 2020. Available: http://dspace.daffodilvarsity.edu.bd:8080/handle/123456789/6246

[23] M. Vanhoef and E. Ronen, "Dragonblood: A Security Analysis of WPA3's SAE Handshake," *Papers.Mathyvanhoef.Com*, 2018, [Online]. Available: https://papers.mathyvanhoef.com/dragonblood.pdf (accessed Oct. 19, 2019)

[24] Z. H. Jiang, Y. Fei, and D. Kaeli, "A novel side-channel timing attack on GPUs," *Proc. ACM Gt. Lakes Symp. VLSI, GLSVLSI*, vol. Part F1277, pp. 167–172, 2017, doi: 10.1145/3060403.3060462.

[25] W.-F. Alliance, *Wi-Fi Protected Access ® Security Considerations*, no. May. 2021.

[26] W.-F. Alliance, *WPA3™ Specification Version 3.0*. 2020.

[27] C. P. Kohlios and T. Hayajneh, "A comprehensive attack flow model and security analysis for Wi-Fi and WPA3," *Electronics (Switzerland)*, vol. 7, no. 11, 2018, doi: 10.3390/electronics7110284.

[28] J. Henry, "802.11s Mesh Networking," 2011.

[29] T. V. Goethem, C. Pöpper, W. Joosen, and M. Vanhoef, "Timeless timing attacks: Exploiting concurrency to leak secrets over remote connections," in *Proceedings of the 29th USENIX Security Symposium*, 2020, pp. 1985–2002.

[30] M. Appel and S. D.-I. Guenther, "WPA 3-Improvements over WPA 2 or broken again?," no. November, pp. 2–5, 2020, doi: 10.2313/NET-2020-11-1.

[31] M. Vanhoef and F. Piessens, "Advanced Wi-Fi Attacks Using Commodity Hardware," 2014.

[32] Y. Yarom, D. Genkin, and N. Heninger, "CacheBleed: a timing attack on OpenSSL constant-time RSA," *Journal of Cryptographic Engineering*, vol. 7, no. 2, pp. 99–112, 2017, doi: 10.1007/s13389-017-0152-y.

[33] B. Chevallier-Mames, M. Ciet, and M. Joye, "Low-cost solutions for preventing simple side-channel analysis: side-channel atomicity," *IEEE Transactions on Computers*, vol. 53, no. 6, pp. 760–768, Jun. 2004, doi: 10.1109/TC.2004.13.

[34] P. C. Kocher, "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems," in *Advances in Cryptology --- CRYPTO '96*, 1996, pp. 104–113.

[35] D. D. A. Braga, P. A. Fouque, and M. Sabt, "Dragonblood is Still Leaking: Practical Cache-based Side-Channel in the Wild," in *ACM International Conference Proceeding Series*, 2020, pp. 291–303, doi: 10.1145/3427228.3427295.

[36] N. Cubrilovic, "RockYou Hack: From Bad To Worse." https://techcrunch.com/2009/12/14/rockyou-hack-security-myspace-facebook-passwords/ (accessed Nov. 5, 2019).

[37] B. Brumley and N. Tuveri, "Remote Timing Attacks Are Still Practical," in *Computer Security* –

ESORICS 2011, 2011, pp. 355–371.

[38] H. Changhua and J. Mitchell, "Security Analysis and Improvements for IEEE 802.11i," in *In Proceedings of the 12th Annual Network and Distributed System Security Symposium*, 2005, pp. 90–110.

[39] J. Wichelmann, T. Eisenbarth, A. Moghimi, and B. Sunar, "MicroWalk: A framework for finding side channels in binaries," in *ACM International Conference Proceeding Series*, 2018, pp. 161–173, doi: 10.1145/3274694.3274741.

[40] B. Reddy and V. Srikanth, "Review on Wireless Security Protocols (WEP, WPA, WPA2 & WPA3)," *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, pp. 28–35, 2019, doi: 10.32628/CSEIT1953127.

[41] C. Koc, "Algorithms for Inversion Mod pk," *IEEE Transactions on Computers*, vol. 69, no. 6, pp. 907–913, Jun. 2020, doi: 10.1109/TC.2020.2970411.