

# **Reduced Side Channel Timing Attack in Dragonfly Handshake of WPA3 for MODP Group**

**BY**

**IKRAMUL ISLAM**

**ID -161-19-1847**

**SUVENDU BARAI**

**ID-161-19-1845**

**MD. AHOSANUL HAQUE MOON**

**ID-161-19-1873**

This Report Presented in Partial Fulfillment of the Requirements of the Degree of  
Bachelor of Science in Electronics and Telecommunication Engineering

Supervised By

**Professor Dr. A.K.M. Fazlul Haque**

Professor, Department of ETE

&

Associate Dean, Faculty of Engineering

Daffodil International University



**Daffodil International University**


**Dhaka-1207**

**January 2020**

## APPROVAL

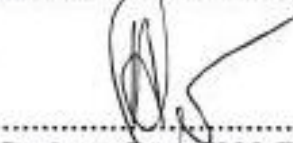
This project titled “**Reduced Side Channel Timing Attack in Dragonfly Handshake of WPA3 for MODP Group**” submitted by Ikramul Islam, Suvendu Barai and Md. Ahosanul Haque Moon to Department of Electronics and Telecommunication Engineering (ETE), Daffodil International University, has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of B.Sc. in Electronics and Telecommunication Engineering and approved as to its style and contents. This presentation was held in January, 2020.

### BROAD OF EXAMINER



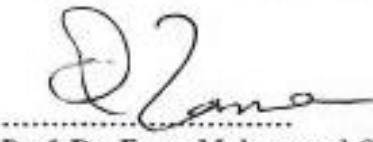
.....  
**Mr. Md. Taslim Arefin**  
**Associate Professor and Head**  
Department of ETE  
Faculty of Engineering  
Daffodil International University

**Chairman**



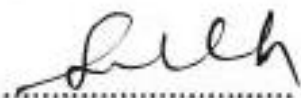
.....  
**Professor Dr. A.K.M. Fazlul Haque**  
**Professor and Associate Dean**  
Department of ETE  
Faculty of Engineering  
Daffodil International University

**Internal Examiner**



.....  
**Prof. Dr. Engr. Mohammad Quamruzzaman**  
**Professor**  
Department of ETE  
Faculty of Engineering  
Daffodil International University

**Internal Examiner**



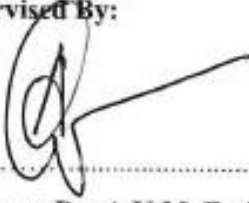
.....  
**Dr. Saeed Mahmudullah**  
**Professor**  
Department of EEE  
Dhaka University

**External Examiner**

## Declaration

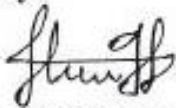
We hereby declare that this project is our own work and effort under the supervision of Professor Dr, A.K.M. Fazlul Haque, Professor, Department of Electronics and Telecommunication Engineering and Associate Dean, Faculty of Engineering, Daffodil International University, Dhaka. It has not been submitted anywhere for any award. Where other sources of information have been used, they have been acknowledged.

Supervised By:

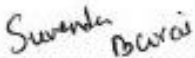


.....  
**Professor Dr. A.K.M. Fazlul Haque**  
**Professor, Department of ETE**  
**&**  
**Associate Dean, Faculty of Engineering**  
**Daffodil International University**

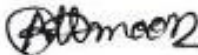
Submitted By



.....  
**Ikramul Islam**  
**Id-161-19-1847**  
**Department of ETE**  
**Daffodil International University**



.....  
**Suvendu Barai**  
**Id-161-19-1845**  
**Department of ETE**  
**Daffodil International University**



.....  
**Md. Ahsanul Haque Moon**  
**Id-161-19-1873**  
**Department of ETE**  
**Daffodil International University**

## ACKNOWLEDGEMENT

Fast of all, we would like to convey our gratitude to the Almighty and our parents, for giving us the right direction while attempting the task.

The real sprit of achieving a goal is through the way of excellence and austere discipline. we would have never been succeeded in completing our task without the cooperation, encouragement and help provided to us by various personalities.

This work would not have been possible without the support and guidance of Prof. **Dr. A.K.M Fazlul Haque, Professor**, Department of Electronics and Telecommunication Engineering and **Associate Dean**, Faculty of Engineering, Daffodil International University, Dhaka, under. Whose supervision we choose this topic and developed the project. Deep Knowledge and keen interest of our supervisor in the field of Wi-Fi security protocol influenced us to carry out of this project.

We would like to express our heartiest gratitude to **Md. Taslim Arefin, Associate Professor** and **Head**, Department of Electronics and Telecommunication Engineering, for this kind help to finish our thesis and also to other faculty members, the staffs of the ETE department of Daffodil International University. We must acknowledge with due respect the constant support and patient of our family members for completing this project.

**Ikramul Islam**

**Suvendu Barai**

**&**

**Md. Ahsanul Haque Moon**

## **ABSTRACT**

The scale of wireless network and the number of wireless devices are increasing day by day. Prevention of unauthorized access to protect the data in wireless network has become a vital part of wireless security. Wi-Fi alliance announced WPA3 as successor to WPA2 after the 'Key Reinstallation Attack' was run against WPA2. WPA3 uses Dragonfly handshake for mutual authentication between a client and an Access Point. But a serious vulnerability named side channel timing leak was found in the password conversion method of Dragonfly Handshake. MODP group has taken into consideration for the conversion of password element because dragonfly supports both MODP and ECC group. Any attacker in the range of a Wi-Fi network can run a brute-force dictionary attack using the leaked timing information. In this work, a method has proposed and designed to reduce the Timing Based Side Chanel Attack. This method is consist of three different parts: Fixing number of iterations, Generation of a database of Password Element, Random choice of a Password Element from the database. Leaked Timing Information Creation of the signature of password has been illustrated. Finally, the complications for an attacker to hack the password are stated.

## Table of Contents

<b>CONTENTS</b>	<b>PAGE</b>
APPROVAL.....	ii
DECLARATION.....	iii
ACKNOWLEDGEMENTS.....	iv
ABSTRACT.....	v
TABLE of CONTENTS.....	vi-viii
LIST OF FIGURES.....	ix-x
LIST OF TABLE.....	xi
CHAPTER 1 : INTRODUCTION .....	1
1.1 Overview.....	1
1.2 Motivation.....	2
1.3 About the Report.....	2
CHAPTER 2 : EVOLUTION OF WI-FI SECURITY PROTOCOLS.....	3
2.1 WEP.....	3
2.2 WPA.....	4
2.3 WPA2.....	5
2.4 WPA 3.....	6
2.4.1 WPA3-Personal.....	7
2.4.2 WPA3-Enterprise.....	7
2.5 OWE.....	8
CHAPTER 3 : BACKGROUND STUDY .....	9
3.1 EAPOL 4-Way Handshake .....	9
3.1.1 Temporal Keys in WPA-2.....	10
3.1.2 Key Reinstallation Attack in WPA2.....	10
3.2 SPEKE Protocol (Simple Password Exponential Key Exchange).....	12
3.2.1 Passwords used in WPA2 .....	13
3.2.2 Preventing offline dictionary attack.....	14
3.2.3 Password attack resistant key generation and mutual authentication .....	15

3.3 Dragonfly Handshake .....	17
3.3.1 Features of Dragonfly .....	18
3.3.2 Password conversion to group element.....	19
3.3.4 MODP Group.....	20
3.3.5 Additive MODP Group.....	21
3.3.6 Multiplicative MODP .....	22
3.3.7 Hunting and pecking with MODP group .....	23
3.4 Commit and Confirm Phase.....	23
3.5 Discrete Logarithm Problem.....	25
3.6 Diffie-Hellman Key Generation.....	26
3.7 Encryption Standard for WPA3 .....	27
3.7.1 The AES Key .....	28
3.7.2 The math of AES.....	29
3.8 Authentication.....	31
3.8.1 GCM and GMAC.....	32
3.8.2 Hash Function .....	33
3.8.3 SHA-512 .....	34
3.9 Popular Password Cracking Methods .....	35
3.9.1 Dictionary attack.....	35
3.9.2 Brute force attack.....	36
3.9.3 Rainbow table .....	36
CHAPTER 4 : VULNERABILITIES IN DRAGONFLY.....	37
4.1 Downgrade attacks .....	37
4.1.1 Exploits backward compatibility .....	37
4.1.2 Exploits dragonfly handshake.....	37
4.2 Side-channel attacks.....	37
4.2.1 Timing based.....	37

4.2.2 Cache based .....	38
4.3 Denial of service attack.....	39
CHAPTER 5 : MATERIAL AND METHODOLOGY .....	40
5.1 Convert Password to MODP Element .....	40
5.1.1 Iteration Depends on Password.....	42
5.1.2 Client MAC addresses influence the execution time .....	42
5.2 Proposed Scheme .....	43
5.2.1 Fixed number of Max and Min Iterations for a fixed number of Password Element.....	43
5.2.2 Random Call from the fixed number of PE .....	43
5.3 MODP Group-22 .....	45
5.4 Environment.....	46
CHAPTER 6 : RESULT AND ANALYSIS .....	47
6.1 Variation in Execution Time.....	47
6.2 Guessing Password using Different Client MAC Address .....	48
6.3 Certain Number of Password Elements from Same Password .....	51
6.4 Random Call .....	54
6.4.1 Random Call 1 .....	55
6.4.2 Random Call 2 .....	56
6.4.3 Random Call 3 .....	57
6.4.4 Random Call 4 .....	58
6.4.5 Random Call 5 .....	59
CHAPTER 7 :DISCUSSION.....	<b>Error! Bookmark not defined.</b>
CHAPTER 8 :CONCLUSION.....	60
CHAPTER 9 :REFERENCES .....	60
CHAPTER 10 :APPENDIX .....	63



## LIST OF FIGURES

FIGURE	PAGE
Figure 2.1 WEP Encryption Scheme .....	3
Figure 2.2 WPA Encryption Scheme .....	5
Figure 2.3 WPA 2 Encryption Scheme .....	6
Figure 2.4 WPA3 Encryption .....	7
Figure 3.1 4- Way Handshaking .....	9
Figure 3.2 PTK Reinstallation Attack.....	11
Figure 3.3 Replay Attack (GTK/IGTK Reinstallation).....	12
Figure 3.4 Offline Dictionary Attack on Password.....	13
Figure 3.5 Prevention of Offline Dictionary Attack 1 .....	14
Figure 3.6 Prevention of Offline Dictionary Attack 2 .....	15
Figure 3.7 Mutually Authenticated Key Generation to Resist Password Attack .....	16
Figure 3.8 Dragonfly Handshake Based on Elliptic Curve .....	18
Figure 3.9 Commit Phase.....	24
Figure 3.10 Confirm Phase .....	24
Figure 3.11 Diffie Hell-man Key Generation .....	26
Figure 3.12 GCM & GMAC .....	32
Figure 3.13 Hash Function a .....	33
Figure 3.14 Hash Function b.....	34
Figure 3.15 Dictionary Password to Hased Password .....	35
Figure 3.16 Password Matching.....	36
Figure 5.1 Flow Chart of Converting Password to MODP Element.....	41
Figure 5.2 Flow Chart of Integrated Proposed Scheme .....	44
Figure 6.1 Variation in Execution Time .....	<b>Error! Bookmark not defined.</b>
Figure 6.2 Graph of Iteration Vs Execution Time .....	48
Figure 6.3 Density Analysis of the Response Time against Different MAC Addresses .....	50
Figure 6.4 Random Call 1 (a).....	55
Figure 6.5 Random Call 1(b) .....	55
Figure 6.6 Random Call 2 (a).....	56
Figure 6.7 Random Call 2 (b) .....	56

Figure 6.8 Random Call 3 (a).....	57
Figure 6.9 Random Call 3 (b) .....	57
Figure 6.10 Random Call 4 (a) .....	58
Figure 6.11 Random Call 4 (b) .....	58
Figure 6.12 Random Call 5 (a) .....	59
Figure 6.13 Random Call 5 (b) .....	59

## LIST OF TABLES

<b>TABLES</b>	<b>PAGE</b>
Table 3-1 Illustration of Finite Field Crypto Group & Elliptic Curve Crypto Group.....	19
Table 3-2 Example of Additive MODP Group.....	21
Table 3-3 Example of Multiplicative MODP Group.....	22
Table 3-4 XOR.....	30
Table 3-5 table of Symbol for Hash Function.....	33
Table 6-1 Variation in Execution Time.....	47
Table 6-2 Execution Time against Different MAC Addresses.....	49
Table 6-3 Different PE from Same Password (a).....	51
Table 6-4 Different PE from A Same Password (b).....	52
Table 6-5 Random Calls.....	54

# CHAPTER 1 : INTRODUCTION

## 1.1 Overview

Securing wireless connectivity is the first priority as wireless networks are ubiquitous. Wi-Fi security has had a difficult route. Different security protocols such as WEP; WPA AND WPA2 works for the same purpose but provides distinct methodologies at the same time. As WEP offers 40 bit encryption key and small IV value (24 bits) which is apparently hack able and replay attack due to usage of symmetric algorithm to encrypt the payload of data but not header or trailer data [1]. Again both WPA and WPA2 suffer from several vulnerabilities which affect session key negotiation. A serious vulnerability in Wi-Fi Protected Access II protocol enabling attacker to modify the protocols 4-way handshake which is used by all latest, protected Wi-Fi enabled devices. The attack against the vulnerability is titled KRACK (Key Reinstallation Attack) which helps attacker to exploit the handshake, i.e. the initiation of the WPA2 connection [2]. Before joining the network a handshake is performed every time a client wants to get connected to that WPA2 Wi-Fi protected network in order to confirm both AP and client hold the correct credentials. A fresh encryption key to encrypt the subsequent traffic is established during handshake. While a key is programmed for one time usage, an adversary can trick a victim into reinstalling an already-in-use encryption key. This reinstallation forces two counters (transmit nonce's and receive replay) to reset to their initial value which results in replay, decrypt and/or forge packets. A man-in-the-middle attack can be performed by a potential attacker maintaining physical proximity with the protected Wi-Fi network. Interception/decryption of internet traffic without learning about the credentials of the network is also possible. Stealing information becomes simpler by combining downgrade attack to pull down a secure HTTPS connection to HTTP to those SSL/TLS sites that haven't applied security measures against downgrade attack [3]. Almost every Wi-Fi network setups both personal or enterprise can fall victim as this attack work against WPA, WPA2 and networks only use AES as security measure. Previously disclosed other handshakes such as group key, Peer Key, TDLS, fast BSS transition can be obstructed due to this KRACK technique. This attacks and vulnerabilities leads Wi-Fi alliance to release WPA3 which replaced 14-year-old handshake by dragonfly or in Wi-Fi standard Simultaneous Authentication of Equals (SAE).

## **1.2 Motivation**

Dragonfly handshake is susceptible to side channel (timing) attacks and may leak the shared password. The amount of times it takes for an AP to respond to commit frames may leak information about the password. This work is done to prevent the timing based side channel brute force attack to secure a WPA3 supported Wi-Fi network. The Dragonfly handshake prevents offline dictionary attacks and provides forward secrecy. It is a Password Authenticated Key Exchange (PAKE), meaning it turns a password into a high-entropy key. Before initiating the handshake, the pre-shared password is converted to a group element using a hash-to-element method. The hash-to-element method for MODP groups is called hash-to-group, and the one for elliptic curves is called hash-to-curve.

## **1.3 About the Report**

This report is consist of 8 chapters. The very first chapter provides the overview of this work. In chapter 2 evolution of Wi-Fi security protocol from WEP to WPA3 is illustrated. Background study of this work is described in chapter 3. Chapter 4 depicts the vulnerabilities in Dragonfly Handshake. Materials and Methodology are described in chapter 5. Result and Analysis, Discussion and Conclusions are described in chapter 6, 7 and 8 respectively.

# CHAPTER 2 : EVOLUTION OF WI-FI SECURITY

## PROTOCOLS

Distributing data using wireless medium means that anyone can capture the signal. There is no boundary, like in case of cable communication, which prevents unwanted access.

### 2.1 WEP

Since the birth of Wi-Fi in 1977, there was only one algorithm intended for it, named WEP-Wired Equivalent Privacy. Providing confidentiality equal to the wired transmission was its only priority. Data integrity and encryption are top priorities for security algorithms used in Wi-Fi. Integrity ensures the user that the message was not modified. Encryption prevents other users from knowing the message details. The below figure shows data flow in WEP encryption. WEP uses 32-bit Cyclic Redundancy Check (CRC32) for integrity purpose. Integrity Check Vector (ICV) is appended to the user data and forwarded to encryption. CRC32 method failed to provide reliability as it was meant to detect errors during transmission and with this the first flaw shows up [4]. RC4 is used for encryption which is a stream encryption algorithm, uses a pair of WEP key (password) and an initialization vector (IV) as a seed. WEP key might either have 40 or 104 bits and IV is always 24 bits long. So this becomes the seed either 64 or 128 bits long. Avoiding dealing with the same seed to encrypt different messages was IV's purpose but as it turned out within few minutes' number of packets can be captured as IV collision probability reaches 50% [5]. This leads to easy compromising of WEP key.

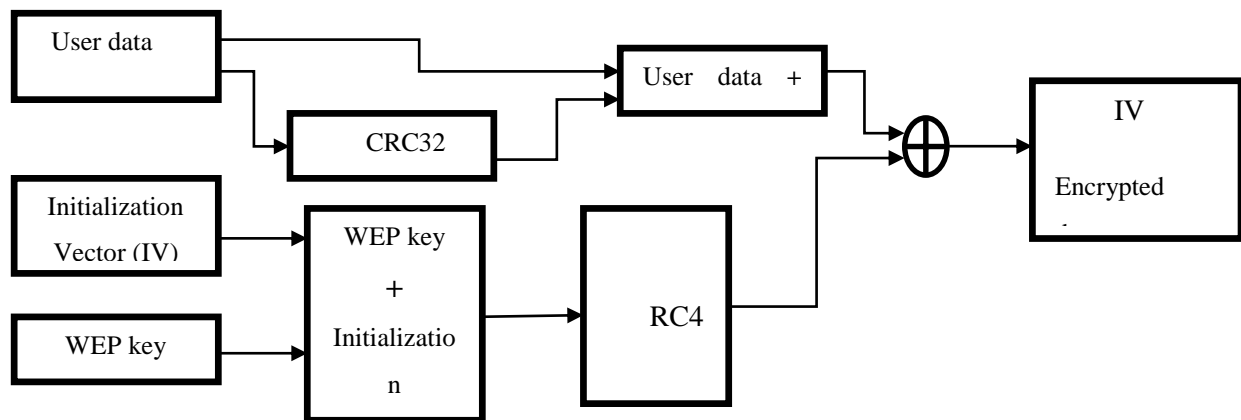


Figure 2.1 WEP Encryption Scheme

## 2.2 WPA

IEEE's response was 802.11i amendment known as WPA addressing WEP weaknesses and provided a number of security improvements. Only firmware up-gradation was required to uplift security aspect as it was based on same hardware as WEP. Mainly focused on improvement of two major flaws in WEP. Encryption was still based on RC4, since it was hardware implementation and couldn't be updated, but IV was extended to 48 bits Master key (password entered in AP and station) is not directly used rather it is mixed and encryption key is derived from master key which makes it hard to compromise the main key [4]. Output message is, as in case of WEP, X-OR operation of the user data and pseudorandom stream from RC4, but this time the key is much better protected and collision is less likely Message Integrity Check (MIC) algorithm is introduced which outplayed CRC32 by design sense. Algorithm worked with additional input such as transmitter address, destination address rather than only data that makes it harder to forge message with correct MIC vector. WPA protects well enough only against those who don't have a key. If a malicious user is legitimately connected to some network protected with WPA it can capture packets destined for others and decrypt them [5].

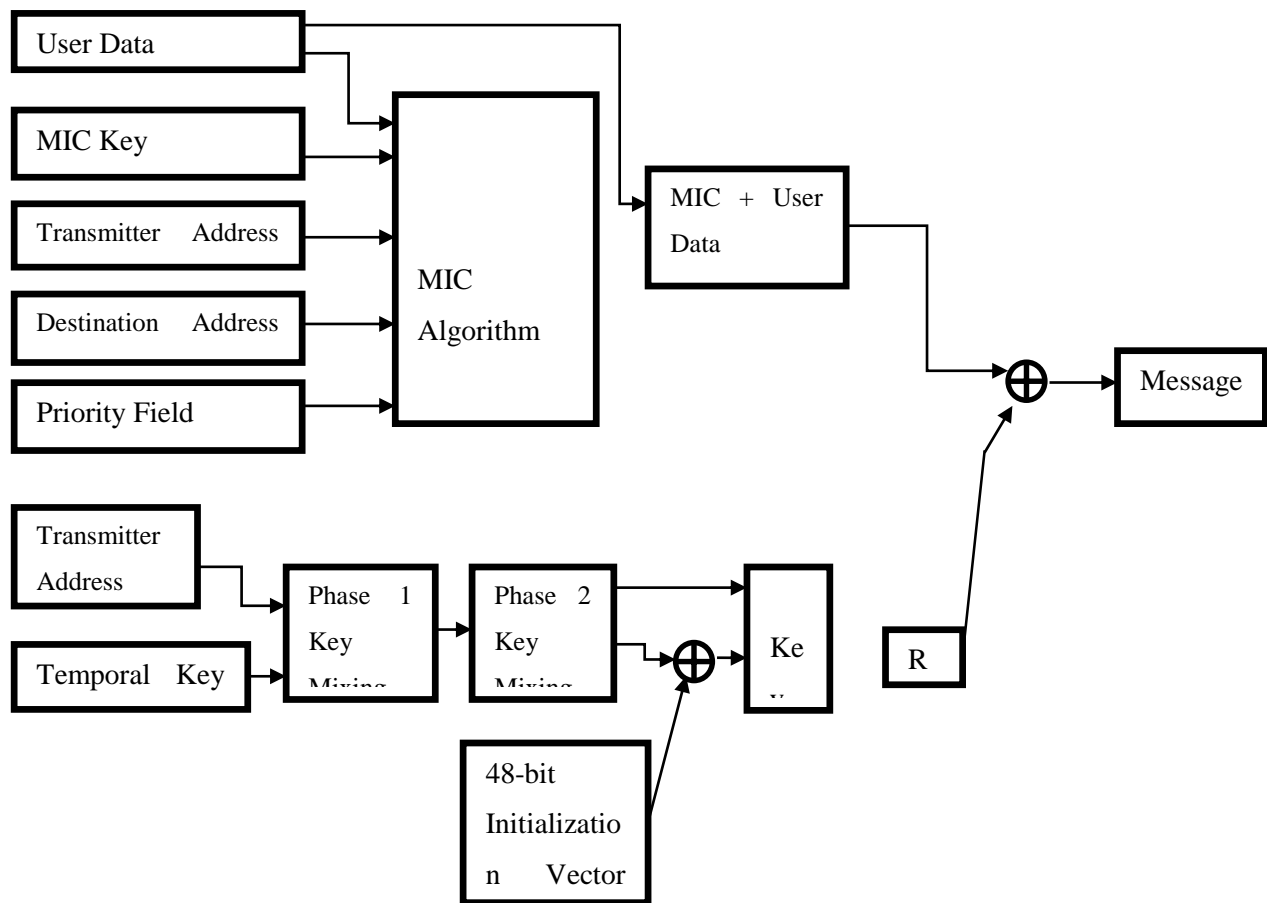


Figure 2.2 WPA Encryption Scheme

## 2.3 WPA2

WPA2 provides a significant security enhancement over WEP and WPA. The most important architectural change is the data encryption algorithm. In WPA2 the Counter Mode with Cipher block Chaining Message Authentication Code Protocol (CCMP) uses an Advanced Encryption Standard (AES).

Calculation of encrypted message and integrity makes WPA2 unique. Integrity calculation chain and encryption chain were easily distinguishable in previous systems. Single logical block-CCM is used for integrity and encryption check purpose in WPA2.

Both integrity check and encryption are based on Advanced Encryption Standard (AES) algorithm [1] [4]. This is a block-based algorithm, operating on 128 bits chunks. Input message is divided into 128 bits parts and used together with a previous result as an input to the AES algorithm. Whole message, including header



is taken into consideration for integrity check, while only payload is encrypted. Header must be transmitted as plain-text in order to enable frame detection and decoding [5].

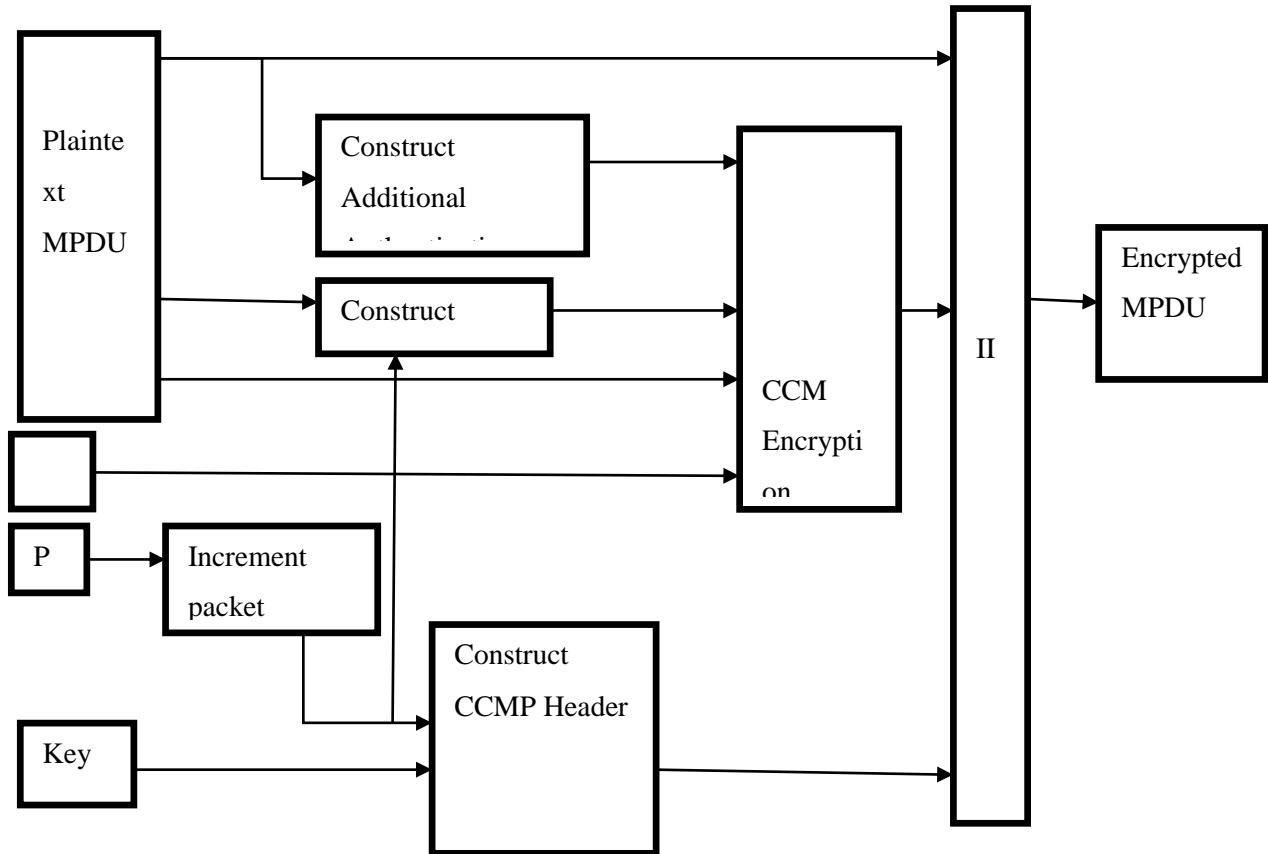


Figure 2.3 WPA 2 Encryption Scheme

## 2.4 WPA 3

WPA3 aims to simplify security, enable robust authentication and increase cryptographic strength. WPA3 comes in two variants WPA3-Personal and WPA3-Enterprise.

### 2.4.1 WPA3-Personal

This variant replaces PSK by SAE as it is based on key exchange, resists offline dictionary attack which do not allow to form a signature of password from previously recorded data [6]. This mechanism allows protecting the data even if the password is weak; meaning is short or using popular phrases.

### 2.4.2 WPA3-Enterprise

WPA3 is built upon WPA2, but it provides a number of improvements. Authenticated encryption is provided by 256-bit Galois/Counter Mode Protocol (GCMP-256). Key derivation uses 384-bit Hashed Message Authentication Mode (HMAC) with Secure Hash Algorithm (SHA). Key establishment uses Elliptic Curve Diffie-Hellman (ECDH) exchange and Elliptic Curve Digital Signature Algorithm (ECDSA). Frame protection based on 256-bit Broadcast/Multicast Integrity Protocol Galois Message Authentication Code (BIP-GMAC-256) [5].

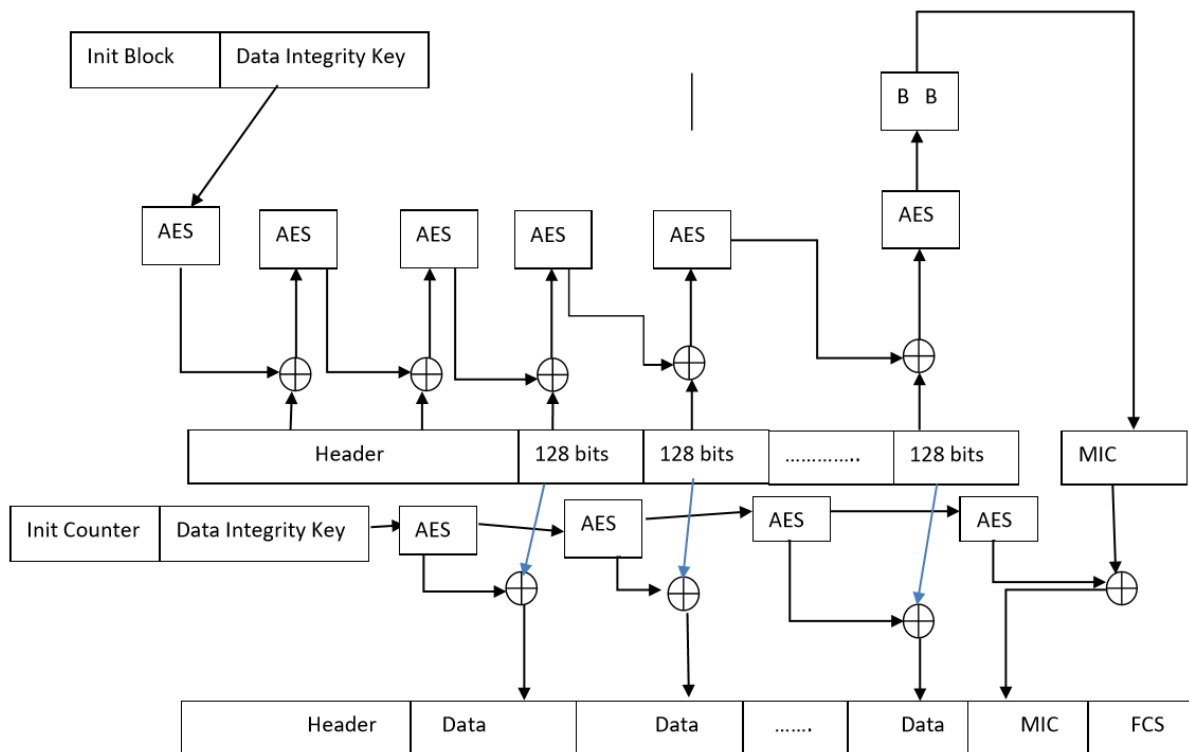


Figure 2.4 WPA3 Encryption

## **2.5 OWE**

Opportunistic Wireless Encryption (OWE) is an extension to IEEE 802.11 which adds a standard for opportunistic encryption for use with an open Wi-Fi network . Besides Wi-Fi enhanced it will solve open network problem by providing unauthenticated data encryption based on OWE (Opportunistic Wireless Encryption) [5].

## CHAPTER 3 : BACKGROUND STUDY

### 3.1 EAPOL 4-Way Handshake

The 4-Way HS occurs when client initially connecting to an AP or certain PTK is refreshed [7]. Now connecting to an AP either 802.1x which is EAP or PSK (password) some initial auth is done and ends up generating PMK on the both side and it never floats on the wireless. Now to generate PTK, AP first sends Nonce1 (long random value) and with this client creates PTK and sends its Nonce2 to the AP and successfully both parties generates PTK. With message 3, apart from confirming PTK, AP transports GTK and IGTK to learn what group keys the new clients are using. The 4<sup>th</sup> message indicates confirmation and PN (packet numbers) are reset to their appropriate values after all keys are installed.

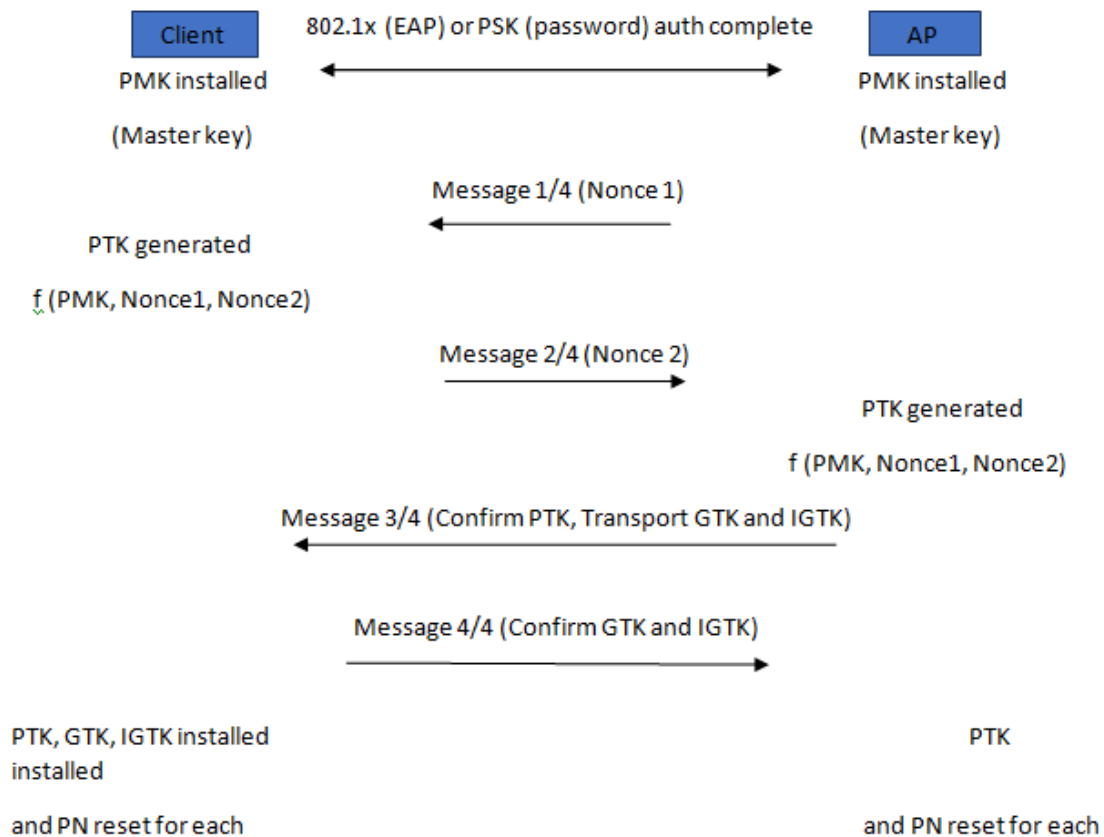


Figure 3.1 4- Way Handshaking

### 3.1.1 Temporal Keys in WPA-2

**PTK:** Pair wise Temporal Key

It protects unicast communication between AP and client. It is generated using EAPOL 4-Way HS. PTK is the function of (PMK, Nonce1, Nonce2)

**GTK:** Group Temporal Key

It protects broadcast/multicast communication from AP to clients. It is transported from AP to client in EAPOL 4-Way HS or Group 2-Way HS

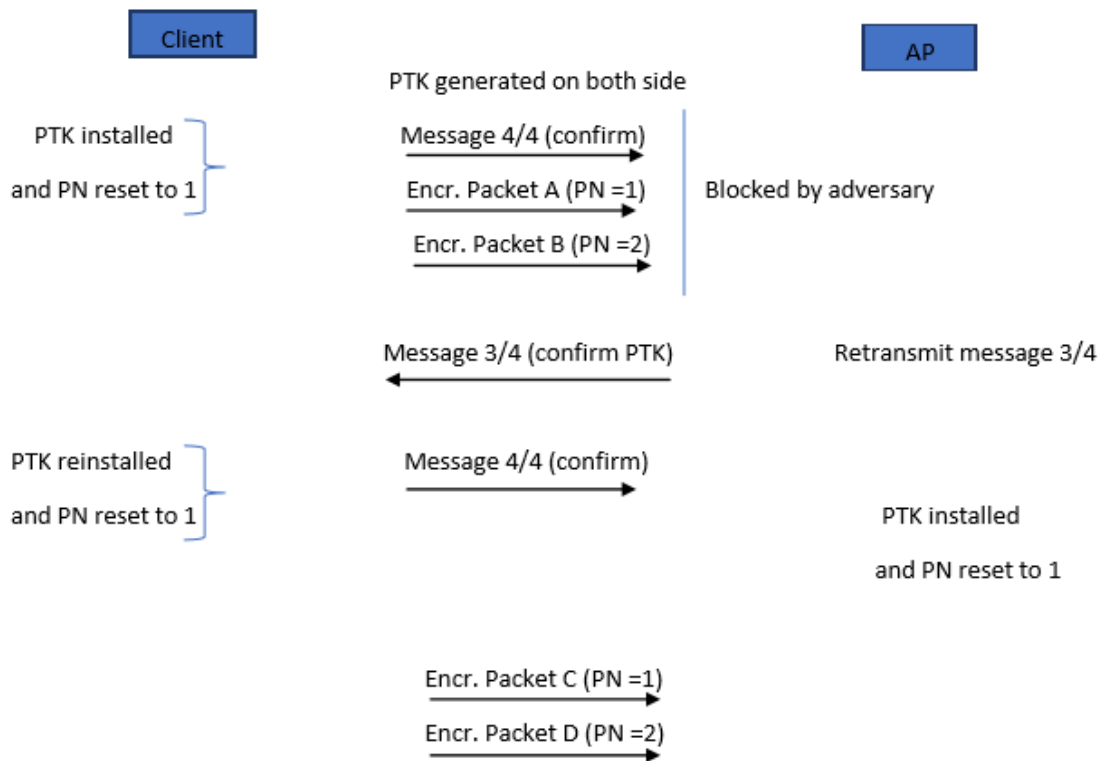
**IGTK:** Integrity Group Temporal Key

It protects of broadcast/multicast management frames from AP to client. It is transported from AP to client in EAPOL 4-Way HS or Group 2-Way HS

### 3.1.2 Key Reinstallation Attack in WPA2

**PTK Reinstallation Vulnerability (Decryption)**

The first 3 messages between AP to client has been established and PTK installed and PN reset to 1. Now with 4<sup>th</sup> message imagining an adversary who is in the middle like a man in the middle manipulating the back and forth messages as he captures packets with one radio and relay it on the other radio to the AP. Assuming the adversary can block the 4<sup>th</sup> message, AP never got the full confirmation and client starts sending encrypted packets thinking it has done all the 4 steps. So its packet numbers are reset. Reinstallation of PTK occurs as from AP's perspective the handshake hasn't completed yet. The client confirms 4/4 messages this time but for some install function it's resetting the packet number which goes without saying violates the packet number uniqueness. Packets (A and C) and (B and D) under the same PTK that means two different packets using the same PN which KRACKS open and because of the same PTK the adversary can XOR and gets a plaintext of XOR between both the data payloads. This overlapping of packet numbers are also used to perform replay attacks.



Reused PN means affected packets could be decrypted

Figure 3.2 PTK Reinstallation Attack

### GTK [IGTK] Reinstallation Vulnerability (Replay)

GTK and IGTK are being used for broadcast communication. So even though the HS hasn't here been completed yet from AP's stand point because of interception of adversary's block, AP doesn't necessarily have to stop broadcast multicast packets because they are for everyone inside the network regardless of the fact client is there or not. Broadcast data packet protected by GTK and broadcast management packet protected by IGTK has been shown here and also PN reset to x and y rather than 1 as GTK and IGTK probably were used by AP and so many broadcast packets may already have flown during absence of client. So the next one will be x+1 and y+1. AP hasn't received 4/4 so it retransmits 3/4 and client again reinstall the keys and PN reset to x and y. Now the man in the middle if he replays those two packets because x+1 is higher than x and y+1 is higher than y, he could get two copies of the packet whereas he was supposed to get only one which violates replay protection of CCMP.

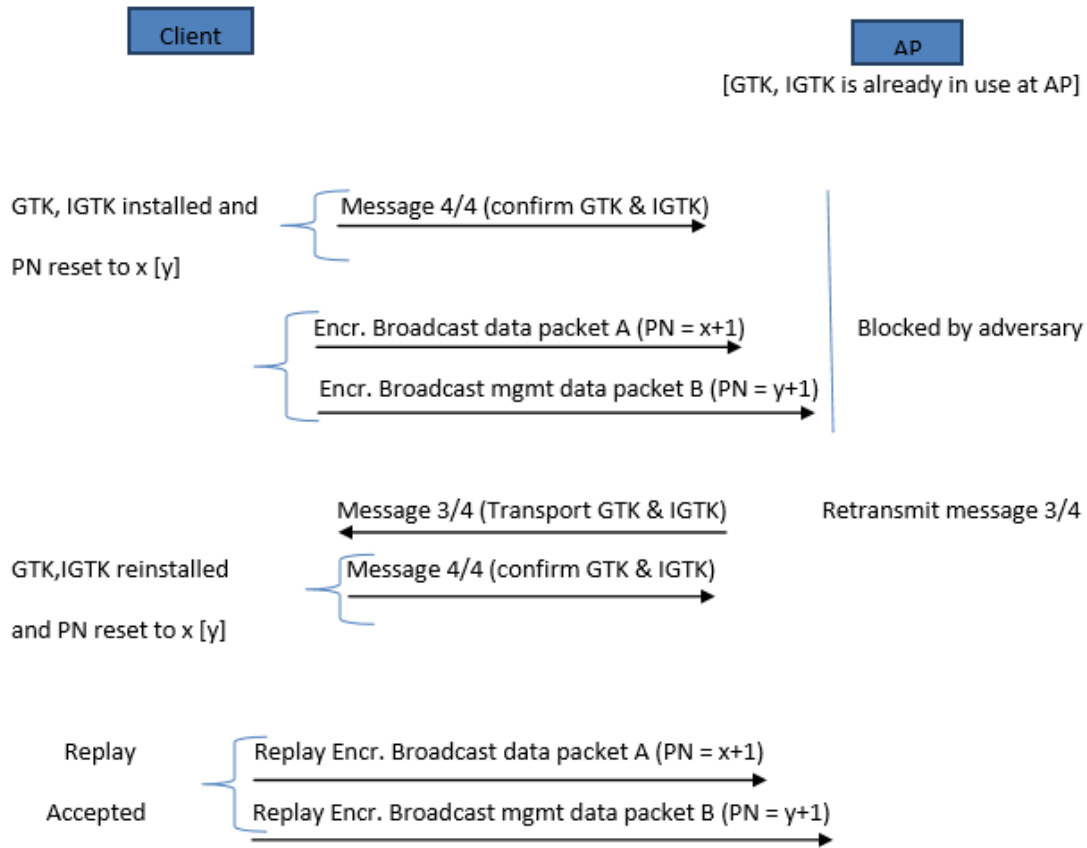


Figure 3.3 Replay Attack (GTK/IGTK Reinstallation)

### 3.2 SPEKE Protocol (Simple Password Exponential Key Exchange)

Dragonfly protocol, specified in IRTF RFC 7664 and in Section 12.4 (SAE) of the IEEE 802.11 standard, provides offline dictionary attack resistance for PSK. Dragonfly is similar to its proprietary predecessor called SPEKE [8]. Dragonfly makes offline password dictionary attack computationally intractable, irrespective of the size of the dictionary [9].

### 3.2.1 Passwords used in WPA2

On client side the password is first run through two well-known functions Password Based KDF2 (PBKDF2) and Key Based KDF (KBKDF) to compute PTK which is a combination of three keys (KEK, KCK, TK) and vice-versa for AP. Then there is a four way handshake where KCK part of it plays the role for authentication of messages and KEK is used for transportation of GTK. TK is used for encryption of data.

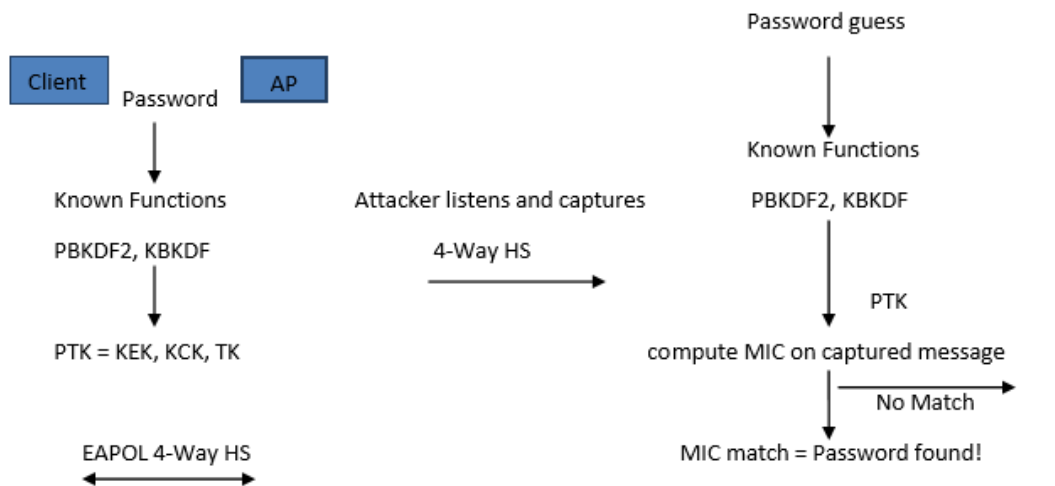


Figure 3.4 Offline Dictionary Attack on Password

Now how a password cracking attack is launched in this type of scheme. The left hand side shows previous figure with a more compressed form. What the attacker would do is sniff or captures the handshake and stores it. By running different guesses of password as the known functions are described in standard to how to apply them to password to come up with the keying. Looking at the captured messages if the computed message authentication code which is derived from the keying, matches with the captured traffic then the assumed password is correct can be said. Again using a weak password an adversary can guess the range of the password, i.e. password dictionary and cracking is then computationally tractable and deriving the right keys to decrypt all the communications that followed 4-way handshake. So this is the problem with current PSK method in Wi-Fi.



### 3.2.2 Preventing offline dictionary attack

Conceptually, there will be some known transformation which will convert the password into keying material. Likewise the attacker can take a password guess, run it through the known transformation and if he would have sniffed some kind of key handshake and match it with intermediate keying material (IKM) to check if the guessed password is correct. Now what if the check can be made computationally intractable.

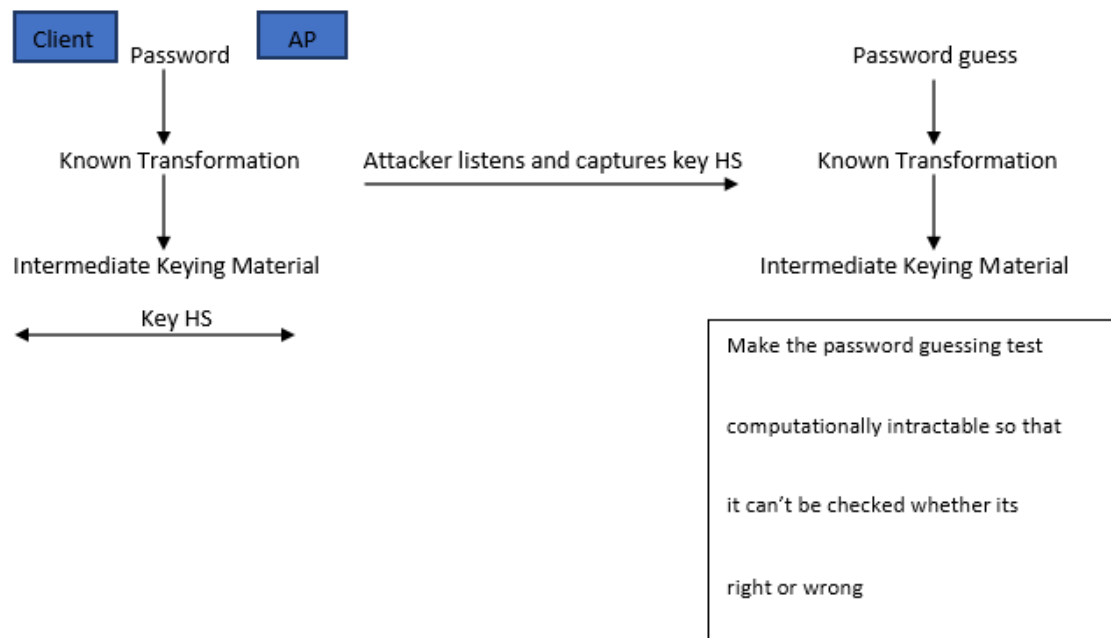


Figure 3.5 Prevention of Offline Dictionary Attack 1

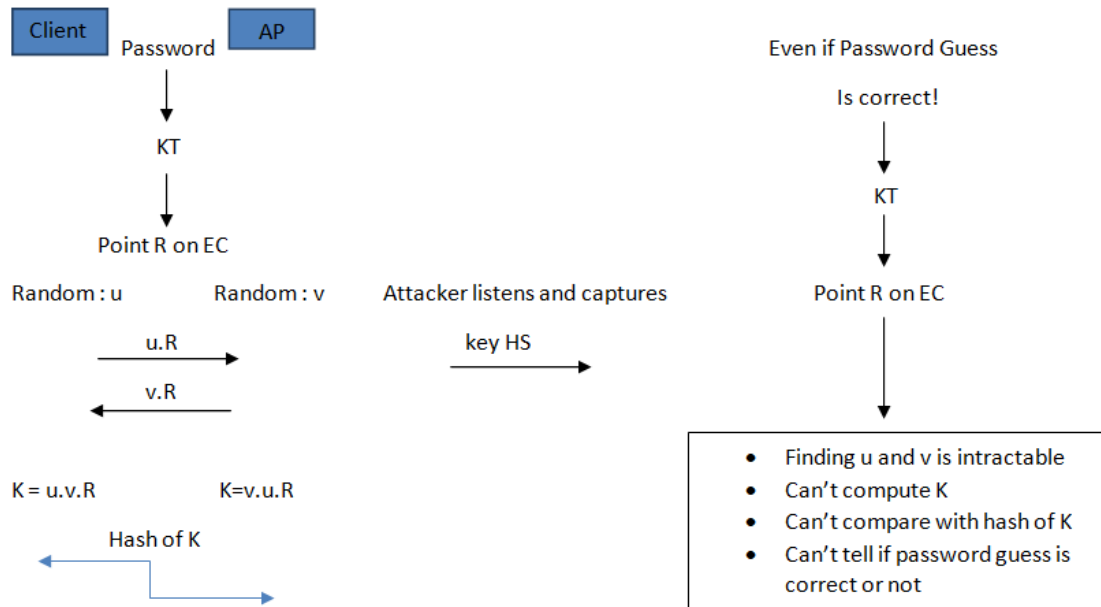


Figure 3.6 Prevention of Offline Dictionary Attack 2

### 3.2.3 Password attack resistant key generation and mutual authentication

Password will run through known transformations and then map into point on an elliptic curve **R**. Client generates a random number **u** and sends **(u.R)** to the AP which also does the same on its side as the transformations follow same standard and sends **(v.R)** to the client. From the discrete logarithm problem, just knowing **(u.R)** and **(v.R)** in the middle it is not possible to compute **u** and **v** and as the key generation formula requires knowledge of **u** and **v**. Hash of the key is sent to the AP side to check if the key matches which eventually proves both parties are in position of the same password to provide mutual authentication [10].

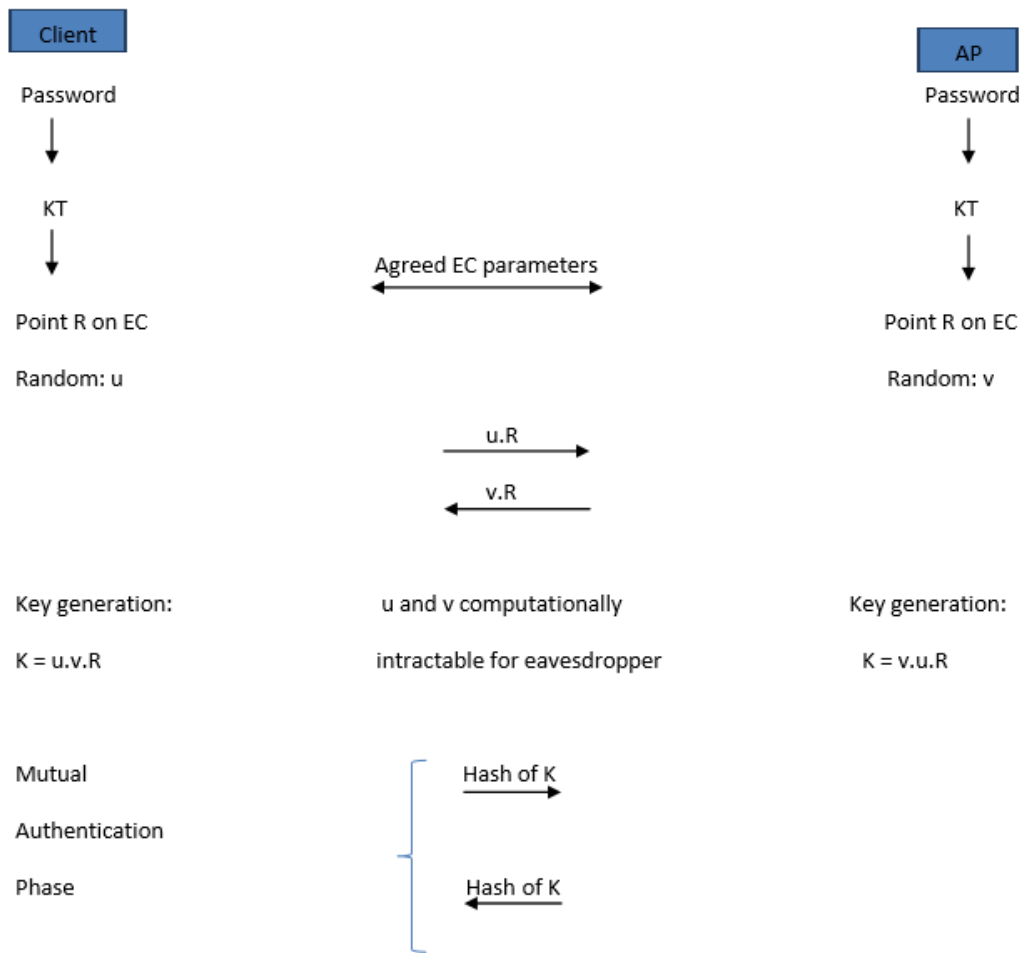


Figure 3.7 Mutually Authenticated Key Generation to Resist Password Attack

The four messages are floating on air and can easily be sniffed. It is resistant to dictionary attack in a way that even if the password guess is correct on the cracking side it will be computationally intractable. As with the known formulas password will be converted to point R and as the password guess is correct it will be same on client and AP side. Now the attacker only knows about R and he will see (u.R) has been transferred on the wirelessly. As the discrete logarithm problem is intractable to reverse, finding u is next to impossible. Similar with the v. And as he cannot compute K, comparison of hash of K to find if the password guess is correct or wrong cannot be told. But elliptic curve suffered from different execution time which eventually led to timing attack will be discussed later.

This is called SPEKE (Simple Password Exponential Key Exchange) protocol which has been here for more than 20 years until recently it has proprietary as some patterns has been resolved. Dragonfly protocol is slight modification of this. Generates random number as the sum of two number rather than a single number

### **3.3 Dragonfly Handshake**

Dragonfly is a robust password based key exchange mechanism through which users authenticate on a WPA3 router or access point [8] [9]. For backward compatibility there exists a transition mode where WPA2 and WPA3 are simultaneously supported. Not only the handshake prevents offline dictionary attacks but also provides forward secrecy and mutual authentication. The password authenticated key exchange (PAKE) procedure turns the password into a strong mechanized key. Harkins designed it in 2008 and adopted by both WPA3 and EAP-pwd.

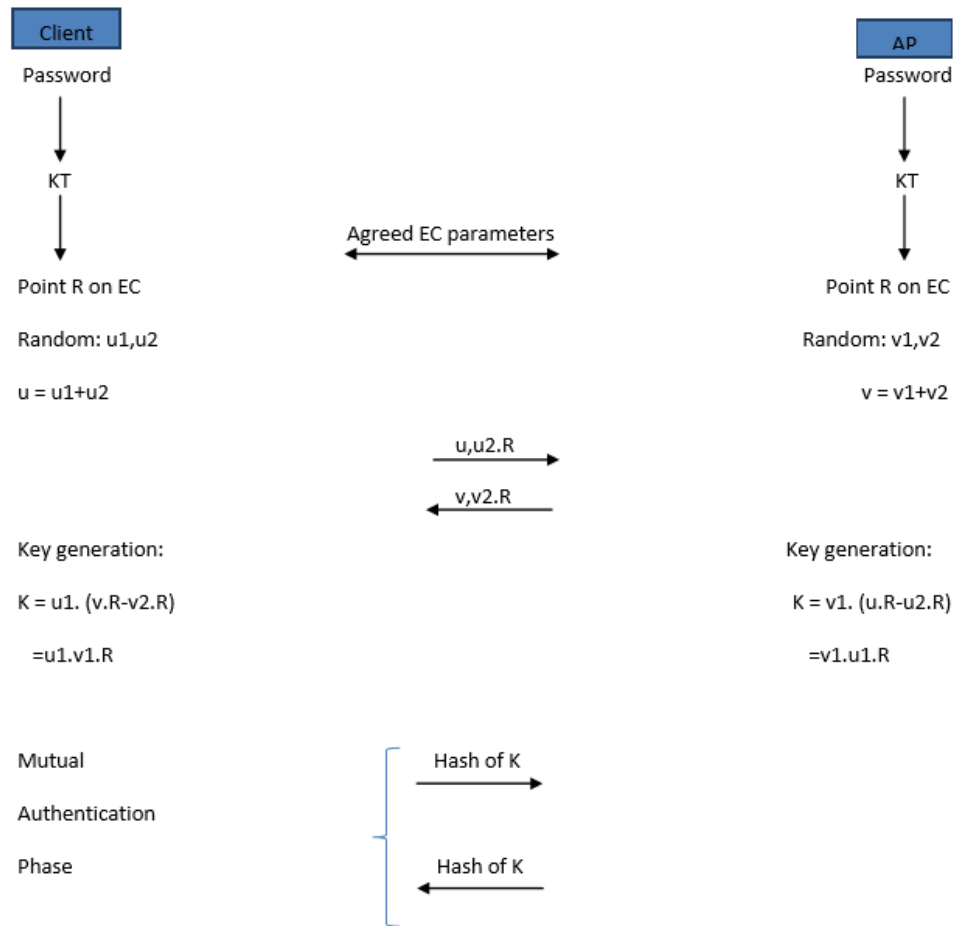


Figure 3.8 Dragonfly Handshake Based on Elliptic Curve

### 3.3.1 Features of Dragonfly

- Protection against dictionary attacks: User needs to be interacted with the network to be authenticated, only when the keys will be generated [9].
- Security for open and public networks: Traffic between each individual user and the access point will be encrypted with a unique password.
- Wi-Fi easy connects (DPP): Factory set asymmetric private key burned into its firmware with public key visible as QR codes on access points and IoT devices.
- Higher enterprise security: Encryption key strength 192 bit.

### 3.3.2 Password conversion to group element

SAE focuses on proper authentication of a device onto a network using a password and MAC addresses. Without learning of the password that the AP uses it should not be possible for an intruder to connect. Also guessing the password or the long-term authentication element of the password is highly uncertain. Though in WPA2 brute forcing the password from derived hashed value by eavesdropping the 4-way handshake is very probable. Diffie-Hellman key exchange method is used in dragonfly which is zero-knowledge proof adds an authentication element. Dragonfly corroborates.

- ECP groups: Elliptic curve cryptography with elliptic curve over a prime field.
- MODP groups: Finite field cryptography with multiplicative groups modulo a prime. [11]

	Finite Field Crypto (FFC)	Elliptic Curve Crypto (ECC)
Era	Classical	Modern
Math	MODP groups	Elliptic curves : P-256 (secp256r1), P-384 (secp384r1) etc
Referred as	Diffie-Hellman	Elliptic Curve Diffie-Hellman

Table 3-1 Illustration of Finite Field Crypto Group & Elliptic Curve Crypto Group

$y^2 = x^3 + ax + b \text{ mod } p$  Is the equation for elliptic curve where  $p$  is a prime and  $a, b, p$  depend on the curve being used. NIST and Brainpool curves are more popular elliptic curve to form group element. With NIST curve, a shared key is produced lead by usage of a pre-shared key and MAC addresses. While connecting to the AP an SAE exchange is performed. If successful, both client and AP create a cryptographically strong key through which session key is being derived. Cracking one session key will only affect one key unlike WPA2. Though at first it was believed brain pool curves are protected but later revealed that it requires extra defense which eventually cause vulnerability.

Before initiation of the dragonfly handshake a group element is formed from pre shared password using a hash-to-element method. For MODP groups it is hash-to-group and hash-to-curve for elliptic curves. In both algorithms a hash is computed over the password an incremental counter and the peer's identities (ID)

for the generation of password element P using try-and-increment strategy. With EAP-pwd, hash input contains a token which is generated from the server. Identified hash output as x coordinate of a point using hash-to-curve variant, it checks if any solution for y satisfies the equation  $y^2 = x^3 + ax + b \pmod p$ . If satisfies, the password element is the point (x,y). Otherwise the counter is raised to find solution for y deploying the new x value.

**3.3.4 MODP Group**

A field is a set of elements ‘F’ with two operations: addition and multiplication. Under addition, the elements are a commutative group. Under multiplication the non-zero elements are a commutative group. Also addition and multiplication are linked by a distributive property.

Set F with two operations: +, .

< F, + > is a commutative group

< F (x), . > is a commutative group

$a.(b+c) = a.b + a.c$

$(b+c).a = b.a + c.a$

Prime numbers: 2,3,5,7,11,13,19.....

In cryptography a finite field of integers modulo p (where modulo is the remainder of an integer division and p is a prime number, this defined as GF(p) or Galois field of p. For example,

$GF(3) = \{0,1,2\}$

$0+0 = 0$

$0 +1 = 1$

$0+2 = 2$

$1+2 = 0$

$2+2 = 1$

$1 \times 1 = 1$

$1 \times 3 = 0$

### 3.3.5 Additive MODP Group

Follows addition modulo (n) concept.

Let n be positive integer greater than 1 and  $a, b \in Z_n$ , where  $Z = \{0, 1, \dots, (n-1)\}$

$a+_nb$  = least non negative remainder when  $a+b$  is divided by n

$Z = \{0, 1, 2, 3, 4\}$  and by adding modulo 5

Table 3-2 Example of Additive MODP Group

$+_5$	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	0
2	2	3	4	0	1
3	3	4	0	1	2
4	4	0	1	2	3

#### Closure Property

- No repetition
- All elements from set Z

#### Commutatively Property

- Diagonal and symmetric above and below

#### Identity Element

- IE is 0 here because the of order being followed from set Z

To find inverse IE is acquiesced horizontally and vertically so that makes

$$4^{-1} = 1, 3^{-1} = 2, 2^{-1} = 3, 1^{-1} = 4, 0^{-1} = 0$$



### 3.3.6 Multiplicative MODP

Follows multiplication modulo (n) concept.

Let n be positive integer greater than 1 and a, b  $\in Z_n$ , where  $Z = \{0,1,\dots,(n-1)\}$

$a \times_n b$  = least non negative remainder when ab is divided by n

S = {2,4,6,8} and by multiplying modulo 10

Table 3-3 Example of Multiplicative MODP Group

$\times_{10}$	2	4	6	8
2	4	8	2	6
4	8	6	4	2
6	2	4	6	8
8	6	2	8	4

#### Closure property

- No repetition
- All elements from set S

#### Commutatively Property

- Diagonal and symmetric above and below

#### Identity Element

- IE is 6 here because the of order being followed from set S

To find inverse, IE is acquiesced horizontally and vertically So that makes

$$8^{-1} = 2, 6^{-1} = 6, 4^{-1} = 4, 2^{-1} = 8$$

### 3.3.7 Hunting and pecking with MODP group

The purpose of the MODP-specific hunting-and pecking technique is to derive a random element which, when used as a generator, forms a group from the same domain parameter set with the same order [12].

$$\text{Value} = ((p-1)/q)$$

p = prime

q = order from the domain parameter set

The secret generator is found by exponentiation the seed to the value. PE generates when the value is greater than one (1); if otherwise the counter is incremented, a new base and seed are formed and the hunting and pecking continues.

### 3.4 Commit and Confirm Phase

In the Commit Exchange, both sides commit to a single guess of the password. The peers generate a scalar and an element, exchange them with each other, and process the other's scalar and element to generate a common and shared secret [12].

To perform the  $-A$  power, an inverse mod q function is performed.

$$\text{Element A} = \text{inverse of } f_{(\text{pow}(PE,A),q)}$$

$$\text{Element B} = \text{inverse of } f_{(\text{pow}(PE,B),q)}$$

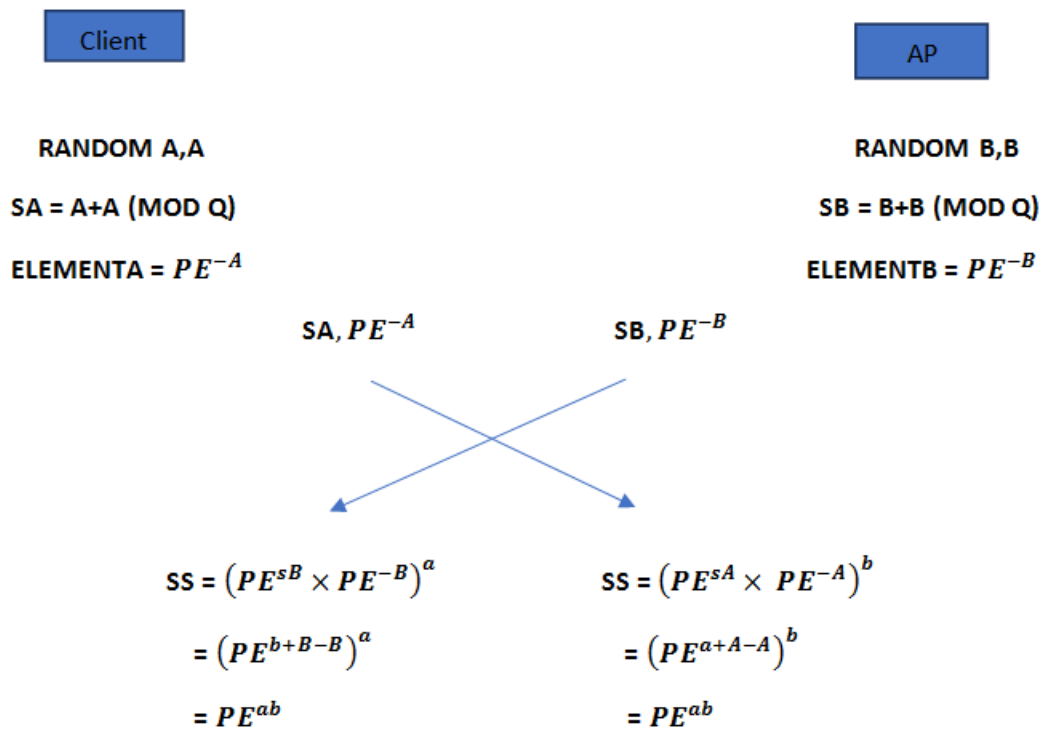


Figure 3.9 Commit Phase

In the Confirm Exchange, both sides confirm that they derived the same secret, and therefore, are in possession of the same password.

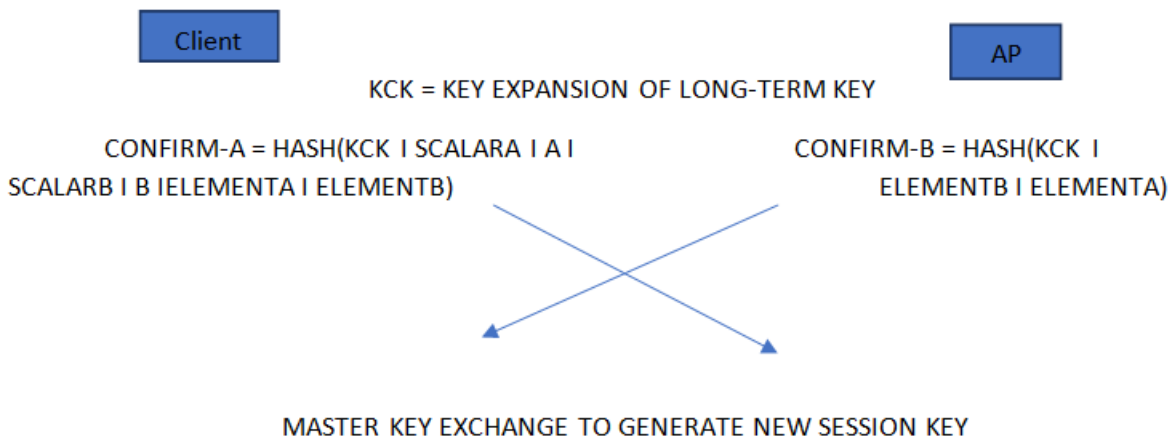


Figure 3.10 Confirm Phase

### 3.5 Discrete Logarithm Problem

Continuous log:  $a^x = b$

$$x = \log_a b$$

Discrete log:  $a^x = b \pmod n$

$$x = d \log_a b$$

The goal is to find  $x$ . But for certain choices of  $a$ ,  $b$  and  $n$  it would not exist. If  $a$  and  $n$  are generator and large prime number respectively then by definition it must exist. That means if the value of  $g$  is raised to each power  $g^1, g^2, g^3, \dots, g^{n-1}$  then what get is the permutation of the numbers  $1, 2, 3, \dots, n-1$  in the group  $\mathbb{Z}_n$ . To find a numerical procedure which is easier to perform in one direction but hard in the other way, modular or clock arithmetic comes to play.

$$x \pmod p$$

If a 46 unit's long rope is taken and wrap it around a clock of 12 units then where the rope ends is its solution. That makes

$$46 \pmod{12} \equiv 10$$

Now to make this work a prime modulus is taken like 17 and a primitive root is found which assumed 3 has a beautiful property because when raised to different exponents the solution distributes uniformly around the prime modulus.

$$3^x \pmod{17} \equiv \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16\}$$

The reverse procedure is hard though.

$$3^? \pmod{17} \equiv 12$$

For a hundred digits long prime modulus it becomes impractical to solve as with Earths all computational power integration it would take hundred years to run through all possibilities. So The strength of one way function is based on the time needed to reverse it.

### 3.6 Diffie-Hellman Key Generation

- Generates common secret between two parties.
  - No pre-shared secret required.
  - MITM cannot generate the common secret.
- It is based on public key cryptography [13].
- Used in SSH, TLS, IPsec and now in OWE, WPA3 and DPP.

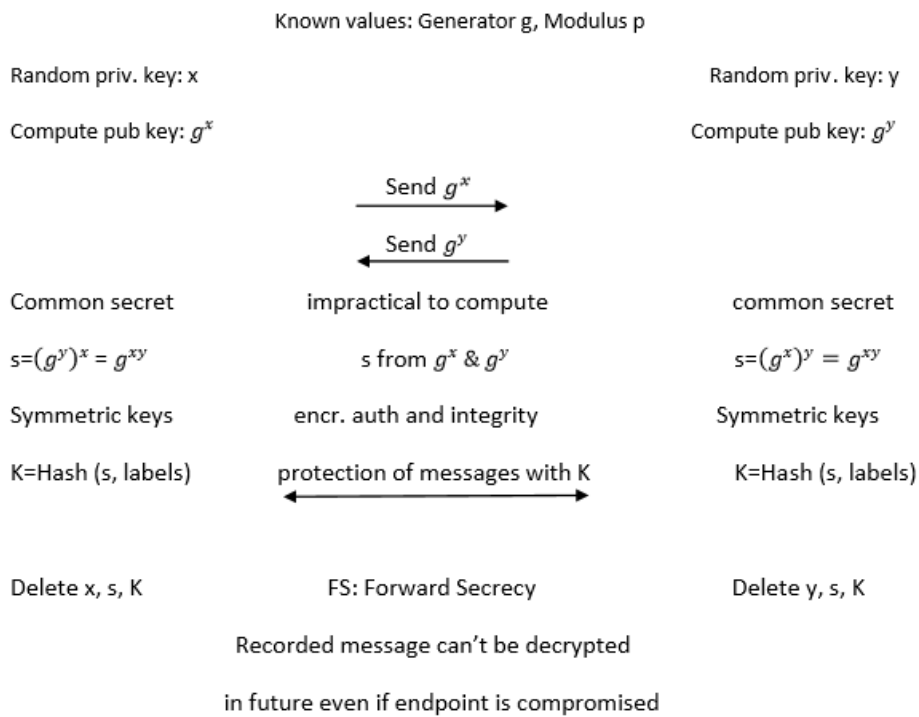


Figure 3.11 Diffie Hell-man Key Generation

### 3.7 Encryption Standard for WPA3

#### Modern Encryption Algorithms

- DES = Data Encryption Standard
- AES = Advanced Encryption Standard

DES adopted in 1971. Gradually became insecure as computing power increased. AES adopted in 2001 [14]. The NSA selected the algorithm created by Belgian mathematicians Joan Daemen and Vincent Rijmen. These algorithms are known as symmetric encryption. A single password or key used to encrypt and the same key used to decrypt the document so nonetheless the key is very important and high probability that this key might leak to a third party. The alternative to symmetric encryption is asymmetric encryption which uses two different keys to encrypt and decrypt the document, simplified a lock that accepts two different keys. One key is public key which is shared to anyone who wants to send message to the receiver and the private key is only kept with the receiving side to unlock the message as it doesn't flow on the internet. How it works has been shown below mathematically.

With Generator  $g=3$  and Prime  $p=17$

A (private key = 15)	B (private key = 13)
$3^{15} \text{ mod } 17 \equiv 6$ Public key of A = 6	$3^{13} \text{ mod } 17 \equiv 12$ Public key of B = 12
$12^{15} \text{ mod } 17 \equiv 10$	$6^{13} \text{ mod } 17 \equiv 10$

Thus shared secret = 10

Now the algorithm is being worked with called block cipher. Assumed there is a secret message which is written in plain text and want to encrypt. The first step is to convert the ASCII code or plain old letters into hexadecimal idea or could be put into binary format, either one, but here shown as HEX as it's much shorter to write and as it's a machine code nobody can read it.

### 3.7.1 The AES Key

The AES key comes in three flavours. 128, 192 and 256 as those numbers indicate the length of the key. So in the least secure model there is a 128 bit keys which goes in and blend the data. So for higher and highest security there are 192 and 256 bit length of keys respectively but 128 bit key is sufficient and widely used as with the increase of length of key size the algorithm starts running slower.

Original Message

---

Dear shumon,I am writing this confidential letter to you in hopes that we can arrange for a meeting. I know the sensitive nature of this message will cause some consternation.Do not worry. We will use encrypted messaging.



Convert to HEX



```
D7 F6 E1 90 CE 63 D1 3C 90 AE 4A A0 88 42 31 3D 9D F8 00 D4 52 05 EE 08 76 33 C1 73 F8 E8 C4 74 20 27 F7 42 A8 69 79
10 D1 F3 6F D5 12 30 17 32 6A D5 CF 4D DF 2C F2 AF D8 79 C9 A5 20 4F AE 2A 89 7C CC 45 75 83 A2 D6 F6 BC B4 90 1F 60
3E 6D E9 0F 2C B9 73 7B D7 7C E3 E1 49 CA A3 2F 6A FF 00 76 7C D0 2C FA 24 01 5B E0 DC 73 53 37 D9 BB E9 DB 28 E6 76
C7 D4 20 79 2C 34 78 5B E7 46 83 DF 99 D0 7F 96 20 85 1F 13 64 19 E6 BB 85 0E 68 07 EB 73 A6 B2 37 A0 DE 83 41 80 FF
AF 74 38 43 15 24 0C E6 E3 71 BD A5 30 01 AE 64 47 39 F4 5E B1 F0 5A 16 DD D6 3B 9B 41 B1 D7 A7 5F EE 77 36 9A 73 3C
07 6E 73 1A AD E9 AD 94 E6 A7 D7 79 85 F2 14 ED 0C 38 A0 44 60 A8 AA D0 EA FA 5D BE 65
```



Split into Blocks

D7 F6 E1 90	CE 63 D1	AE 4A A0	42 31 3D
	3C	88	9D

Each block (D7 F6 E1 90)

+  AES algorithm  Ciphred block (5A 16 DD D6)

Key (53 37 D9 BB E9 DB 28 E6)

Now the next thing is to split the chunks of data into blocks and here the block sizes are equal and four different collections of hex bytes. The blocks in practice are much bigger than this one but this gives an idea what a block looks like. A for loop is then implemented over the algorithm as for each block there will be an operation. A block of data and a key (password) will be taken and put them together through the algorithm. The AES algorithm helps to obfuscate or mix the data so it becomes untraceable. So what comes out as the output is a ciphered block which is of same size as the input block and has same number of bytes in it but completely scrambled. Now there will be a remainder key and it is being combined with the next block to produce the next ciphered block. So each block gets its own unique key as the first key has been generated by the user and every key after that has been computed.

### 3.7.2 The math of AES

Blocks of information are shuffled through multiple rounds of bit shifting, swapping and multiplying.

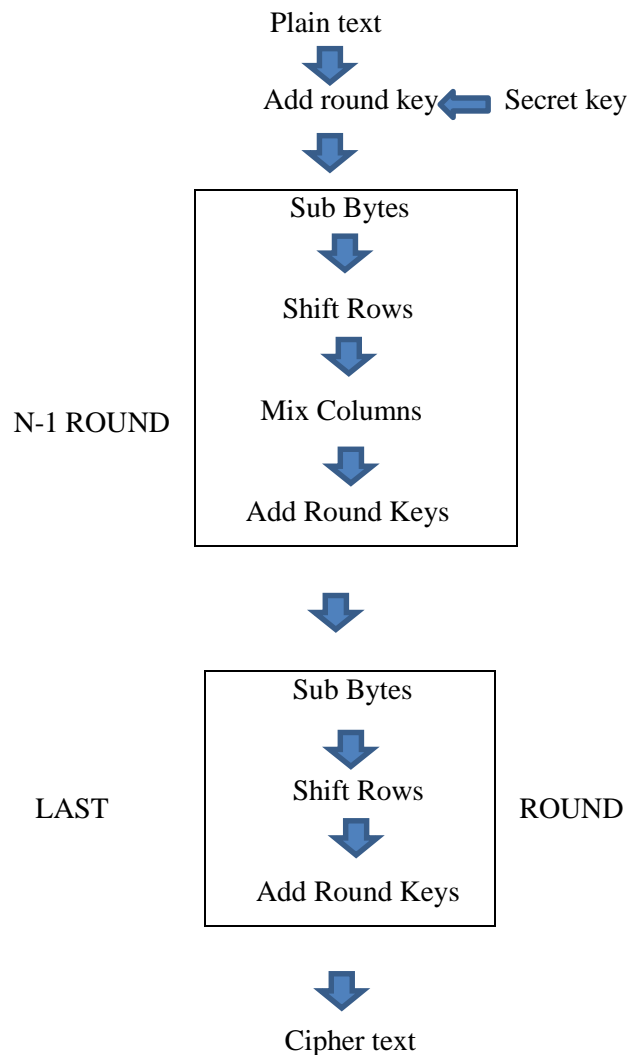
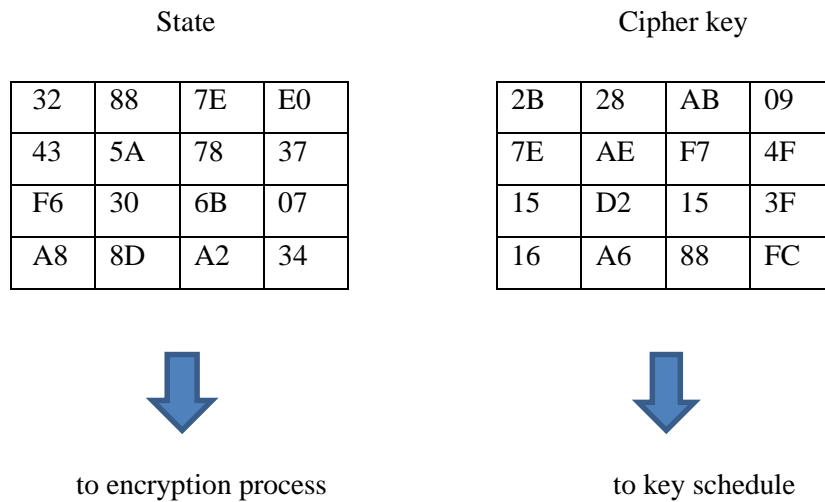




Table 3-4 XOR

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

So a plain text is taken and add a key to blend them together using XOR the operator. Number of rounds like 10,12 or even 15 depending on the key has been used and then there are four steps that do obfuscation and hence there is a final output.

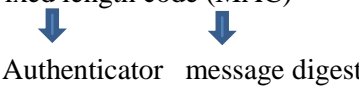


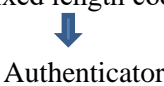
State = this is a block from the plaintext message to be encrypted.

The state represents here a block taken from the pain text which is unencrypted code and the plus circle means XOR and this will go through nine main rounds and in each round the bits will be scrambled and when it's done there will be completely encrypted file.

### 3.8 Authentication

3 types of authentication

- Message Encryption  
Cipher text = authenticator
- Message Authentication Code  
 $C(M,K) = \text{Fixed length code (MAC)}$   


The diagram shows the equation  $C(M,K) = \text{Fixed length code (MAC)}$  with two blue arrows pointing downwards from the words 'Fixed length code' and '(MAC)'. Below these arrows are the words 'Authenticator' and 'message digest' respectively.
- Hash Functions  
 $H(M) = \text{Fixed length code (Hash code 'h')}$   


The diagram shows the equation  $H(M) = \text{Fixed length code (Hash code 'h')}$  with a single blue arrow pointing downwards from the words 'Fixed length code'. Below this arrow is the word 'Authenticator'.

### 3.8.1 GCM and GMAC

In cryptography, Galois/Counter Mode (GCM) is a mode of operation for symmetric-key cryptographic block ciphers. The operation is an authenticated encryption algorithm designed to provide both data authenticity (integrity) and confidentiality. GCM is defined for block ciphers with a block size of 128 bits. Galois Message Authentication Code (GMAC) is an authentication-only variant of the GCM which can form an incremental message authentication code. Both GCM and GMAC can accept initialization vectors of arbitrary length [15].

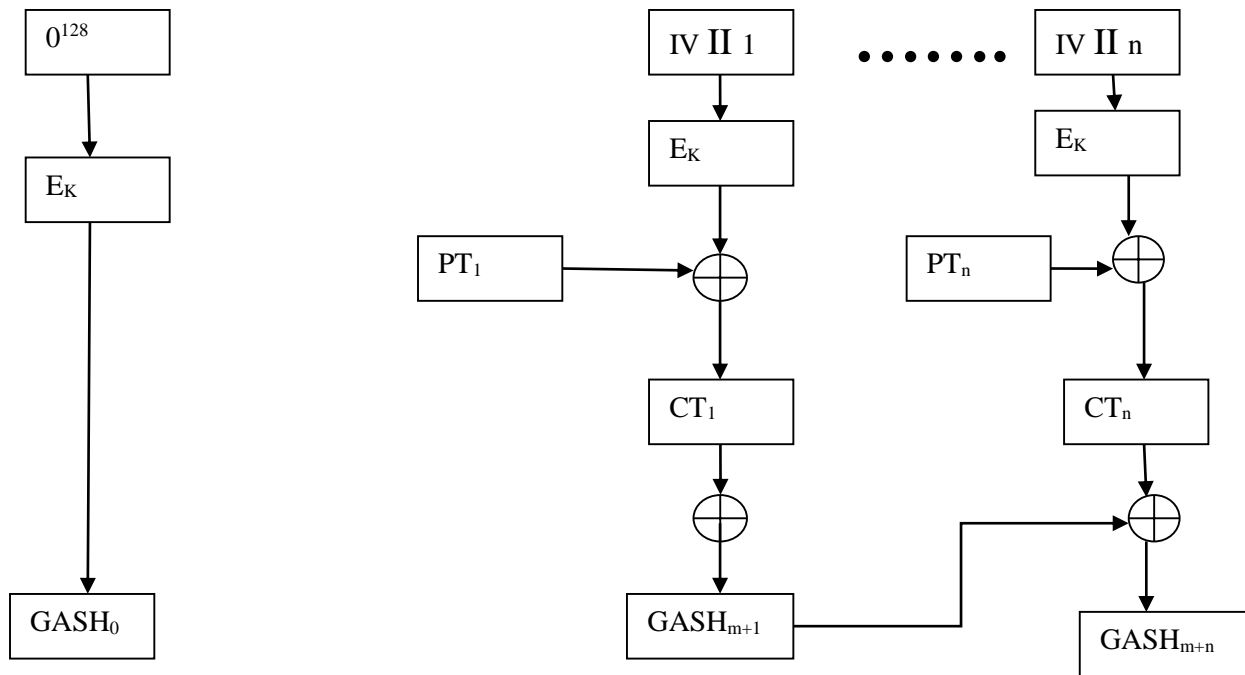


Figure 3.12 GCM & GMAC

$E_k$  is the encryption algorithm and key, which is AES 256

PT is Plain Text that gets encrypted into cypher text (CT)

All blocks are 128 bits in length

IV is a 96-bit Initialization Vector, which is a nonce

1<sup>st</sup> counter block is the IV followed by the 32-bit number “1”

### 3.8.2 Hash Function

Table 3-5 table of Symbol for Hash Function

Symbol	Meaning
H	Hash function
h	Hash code
PrA	Private key of A
PuA	Public key of A

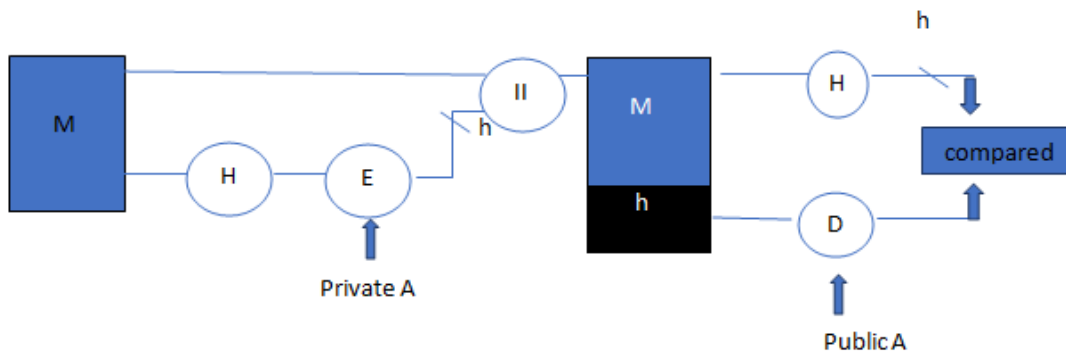


Figure 3.13 Hash Function a

Plain text is being conveyed through hash function to produce hash code and the hash code is encrypted with private key of A following appended on unencrypted plain text. On the receiving side the hash code is decrypted using A's public key and compared with computed hash code which has been generated by recombining plain text and hash function. Anyone can decrypt the hash code as user A's public key is available to everyone, creates a drawback [15].

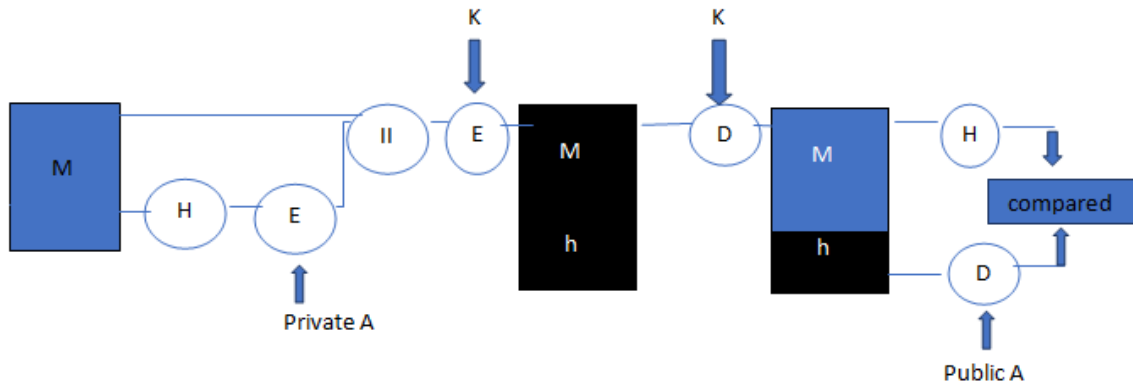


Figure 3.14 Hash Function b

Encryption is done on the appended hash value and plain text using secret key which is decrypted on the receiver side to compare the hashes.

### 3.8.3 SHA-512

SHA-512 pads the bits 100 so that length of P.T is  $128 < \text{multiple of } 1024$  bits. It appends 128 bit representation of original plain text such that length = multiple of 1024 bits. It initializes the buffers (a, b, c, d, e, f, g, h) 64 bit in hexadecimal. It processes each block of plain text in 80 rounds/steps. In each round there are 3 inputs. Content of 8 buffers, constant and the Q-word (64 bit)

The complete plain text is divided into 16 Q-words that is represented as  $w_0-w_{15}$  and from  $w_{16}$

$$W_t = \sigma_1^{512}(W_{t-2}) + W_{t-7} + \sigma_0^{512}(W_{t-15}) + W_{t-16}$$

Formula is used to generate the word. Output in buffers is hash code (512)

### 3.9 Popular Password Cracking Methods

In any major business database Google, Amazon or EBay all users' passwords should be stored in a digest form. They should never be stored in a plain text. However hackers and cyber criminals can still be able to steal these files and hack our passwords. They use many methods and strategies in which dictionary and brute force attack and combination of these two known as brute force dictionary attack or hybrid attack are more popular.

#### 3.9.1 Dictionary attack

A dictionary attack is the simple and fast password attack. By dictionary meaning digital resources that have words like Wikipedia.

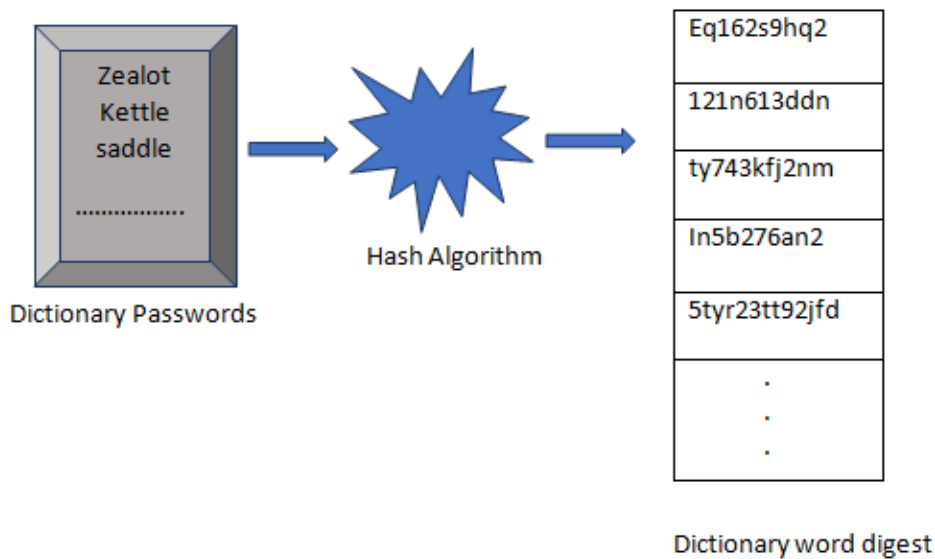


Figure 3.15 Dictionary Password to Hased Password

From a dictionary, hackers generate thousands of candidate digests. They create a huge lookup table with these candidate digests and their pre-matched plaintext passwords. Hackers compare these candidate digests to those in a stolen digest file. If there is a match they get the password.

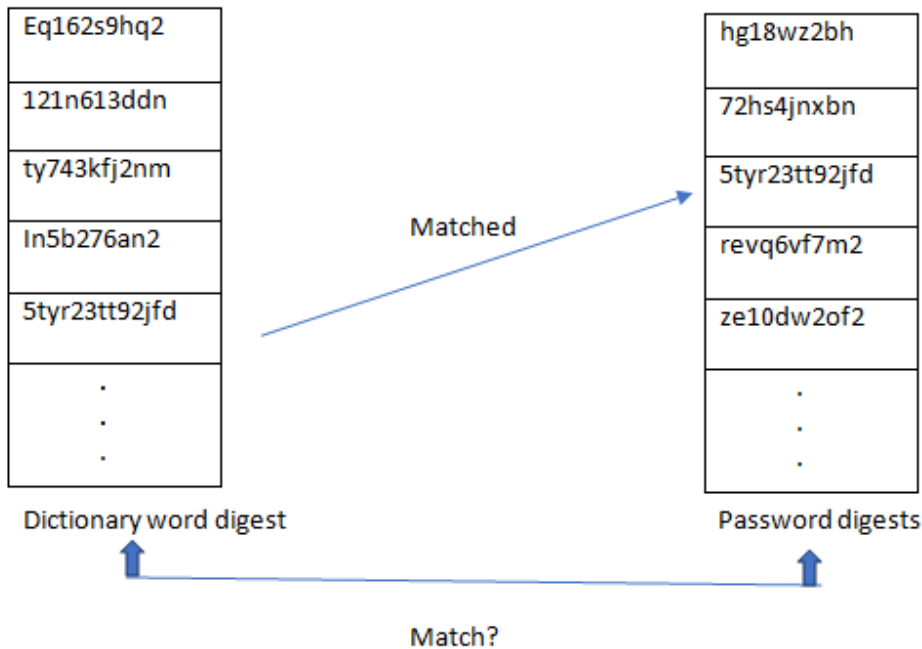


Figure 3.16 Password Matching

Although such approach would seem practical to do manually, computers can do this very fast and run through thousands of passwords in a few hours.

### 3.9.2 Brute force attack

Brute force attack is when hackers use computers to systematically cycle through each letter in a character set. A character set can be letter symbol number or anything the hackers want in. In the simplest terms, brute force attack is a trial and error method which attempts all the combinations for a password. This method is quite efficient for short passwords. But cracking all possible passwords is only a matter of time.

### 3.9.3 Rainbow table

In database passwords kept in hashed form of plain text. Huge pre-compiled hash file (GB/TB) is used to crack hashes to get the plain text password. One hash can be used for different passwords which saves time to find precise hash. If double encryption procedure is followed then encrypted format is cross-checked on them.

## **CHAPTER 4 : VULNERABILITIES in DRAGONFLY**

Though recent researches came up with some design flaws still remained in WPA3.

### **4.1 Downgrade attacks**

#### **4.1.1 Exploits backward compatibility**

During transition from WPA2 to WPA3, the Wi-Fi network supports WPA2 and WPA3 with an identical password. By creating rogue network and forcing WPA3 support client to join the network, WPA2 handshake can be captured to recover the network key. An example of these devices is Samsung galaxy S10 but more devices can be vulnerable to this.

#### **4.1.2 Exploits dragonfly handshake**

The second downgrade attack exploits dragonfly, the handshaking protocol used for WPA3. When a device wants to connect to a WPA3 network, it sends a commit frame to the network indicating the security group it wants to use. Now hackers can impersonate like an AP and tell the clients it doesn't support that security group, use a different one and once the clients use a lower security group hackers manipulate that chance using different methods and tools to get the key.

### **4.2 Side-channel attacks**

#### **4.2.1 Timing based**

Recovering password of a Wi-Fi network considered infeasible for an adversary. Though recent researches revealed depending on the AP's response time to commit frames, information relating password may be leaked. Security groups based on elliptic curve formation which all wpa3 devices are required to support, leaks no information about timing. However there is another optional multiplicative groups modulo a prime (MODP groups) which Access Point also supports depends on the password being used and suffers from timing leak attack as 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.



### **Timing attack on modular exponentiation:**

Timing attacks against a function  $f_k$  generally require two things:

1. Observing the target perform  $f_k(x)$  for a large number of sufficiently diversified known inputs  $x$ .
2. For each  $k$ , there are inputs  $x$  and  $x'$  such that  $f_k(x)$  and  $f_k(x')$  are expected to execute at different speed.

Diffie-Hellman private key operation is assumed to be  $f_k$ , where  $k$  is the private (fixed) exponent of the target and the inputs are also assumed to be 2048 bit integer values. Regardless of the value of  $x$  from operation  $f_k(x) = x^k \bmod p$ , the implementation proceeds in such way that it executed the exact same number of CPU instructions so there will be no timing difference to observe (presuming the CPU performs e.g. a MUL or DIV at a fixed number of clock cycles).

Now some  $x$  might be significantly smaller than the maximize size of 2048 bits. Implementation of modular multiplication is optimized to account for that. Hence few least or most significant bits of  $k$  might leak, depending on if the exponentiation implementation iterates the bits of the exponent from right to left or other way around. In the most unfavorable case of  $x=2$ , this will reveal at most 11 bits since the intermediate result grows exponentially.

To prevent this leak (again assuming the CPU performs a MUL instruction etc. at a fixed number of cycles independently of the value of the operands), all that is needed to prevent such leaks is at the starting of exponentiation function implementation justifying all inputs size to the modulus.

### **4.2.2 Cache based**

During construction of memory access patterns of dragonfly handshake an adversary might be able to observe those patterns which eventually reveals info about the password being used. By controlling application on victim's device or more simply JavaScript Code in the victims browser the observation can be exploited. The leaked patterns can be used to perform a dictionary attack, by simulating the memory access patterns associated to a guessed pass and comparing this to the measured memory access patterns.

### **4.3 Denial of service attack**

Since processing and computation of commit frame is heavy as it takes a lot of resources, estimated more than 16 frames will cause the AP to shut down or restart or just stop. With social engineering or evil twin attack a fake network is created in which clients get to join and end up leaking the password and valuable information.

The side-channel leakages are natural to Dragonfly as patched software was released after the initial disclosure which also affected by a novel side-channel attack. 'Multiplicative groups' feature is being exploited to perform timing based side channel attack. During handshaking time dragonfly uses certain multiplicative groups to encode the password, where the password encoding algorithm applies a variable number of iterations. The hash value which contains password being used and MAC addresses of AP and client decides the precise number of repetitions. A remote timing attack against password encoding algorithm reveals how many iterations were needed to encode the password. The recovered info then can

## CHAPTER 5 : MATERIAL AND METHODOLOGY

WPA3 certification is the successor to WPA2 in state of art Wi-Fi network. Though WPA2 has been used since last fourteen years, it has lost its integrity after Key Reinstallation Attack (KRACK) was run against it [7]. WPA3 and EAP-pwd which are used in home network and enterprise network respectively, both use the identical Password Authentication Key Exchange (PAKE) named Dragonfly Handshake. It is also known as Simultaneous Authentication Equals (SAE). Officially IEEE 802.11 and WPA3 adopt dragonfly handshake [16]. But Dragonfly Handshake of WPA3 also introduces some severe security flaws.

Timing based side channel attack, downgrade and dictionary attack against WPA2 - WPA3 compatible device during transition mode, security group downgrade attack, cache based side channel attack, denial of service attack are introduced in recent [17].

In this work, how timing based side channel attack can leak the information of the password of a WPA3 supported Wi-Fi network was introduced. An approach to mitigate the side channel timing attack was designed and partially tested also.

Timing based side channel attack is located in the password encoding algorithm which is the initial part of the four way dragonfly handshake. To derive the password element from a preshared password, using a certain domain parameter set dragonfly use discrete logarithm Cryptography. Discrete logarithm cryptography basically consists of Elliptic Curve Cryptography or ECC Groups and Finite Field Cryptography or FCC Groups which is basically the MODP Groups with finite field. In this work MODP group was taken in consideration [12].

### 5.1 Convert Password to MODP Element

Conversion of password element (PE) is described in Flow chart 1 below. Here an input password is manually taken. The MAC addresses both of the AP and the Client are also taken as inputs. Then base is calculated the 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 seed is calculated using the KDF. At this point same length of bit string is hold by both of the seed and prime (p). If the condition (seed<prime) is satisfied by the seed then TEMP is calculated using seed. Finally if TEMP is greater than 1, then password element was 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.

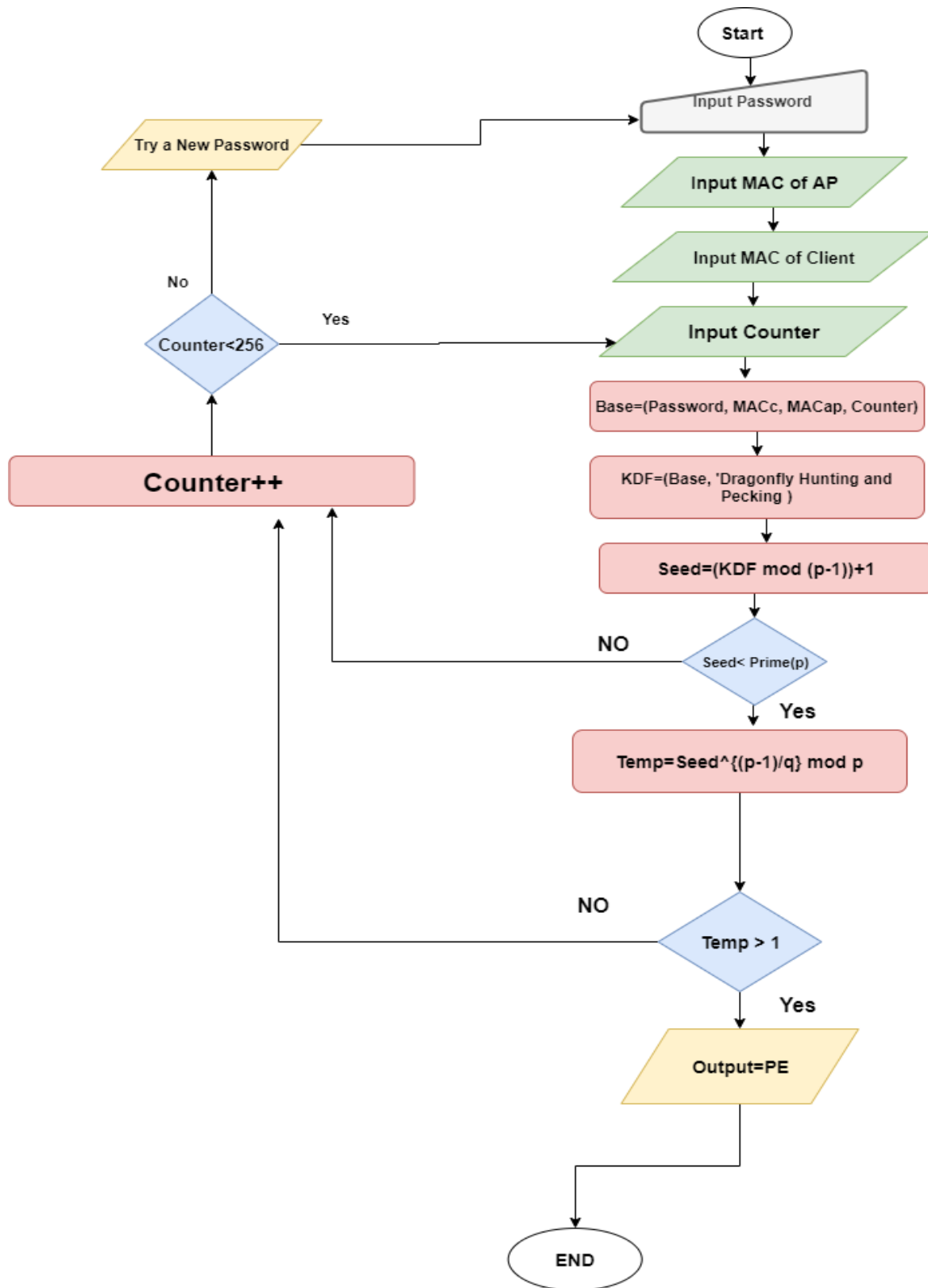


Figure 5.1 Flow Chart of Converting Password to MODP Element

### 5.1.1 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.

$$\text{BASE} = (\text{Password}, \text{MAC of client}, \text{MAC of AP}, \text{Counter})$$

As base is influenced by password, it can be easily said that KDF, Seed and Temp are also influenced by the password being used. 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.

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  $t_1, t_2, \dots$  time is taken by  $n_1, n_2, \dots$  number of iterations respectively  $n_1, n_2, n_3$  number of iteration/iterations is /are needed by totally different password  $p_1, p_2, p_3, \dots$  and . So the signature or the information of the password is leaked by the variation of execution time [Table 6-1]. These timing leak information can be finding by an adversary using false authentication message. Though the password was unknown to the adversary, still the execution time was found from the reject message. Then Brute force attack can be easily run by using this information.

### 5.1.2 Client MAC addresses influence the execution time

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

$$\text{BASE} = (\text{Password}, \text{MAC of client}, \text{MAC of AP}, \text{Counter})$$

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 6-2]. On an average 17 MAC addresses are needed to find the signature of the password from a ROckyup Dump Database [18].

## 5.2 Proposed Scheme

It is almost difficult to implement Dragonfly handshake without side channel attack because of its design. In this work an approach is shown in the design section that can create more difficulties for an attacker to find the leaked time information. The attacker is faced a labyrinth by the system of the Access point. Point to be noted that the approach can only be used in the Access point to authenticate the client.

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

In MODP algorithm maximum number of Iterations is fixed at number 256 as the probability of finding a PE after 255 iteration is almost zero. It has been said to use a fixed number of minimum iterations 'K' in MODP algorithm for security purposes [12] . But the exact number of K has not suggested. It is find in the study that some uses varients  $K=4$  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 stopped and random call will be taken place.

### 5.2.2 Random Call from the fixed number of PE

When a fixed number of password elements is generated (in this work  $i=20$ ) then a random call is taken place. A random PE is selected and provided as the final output by this random call. If the iterations is equal to  $K_{min}$  ( $K_{min}=31$  in this work) and still enough number of PE elements are not produced, then random call is also taken place and choose a PE from the produced password element. [Table 6-5]

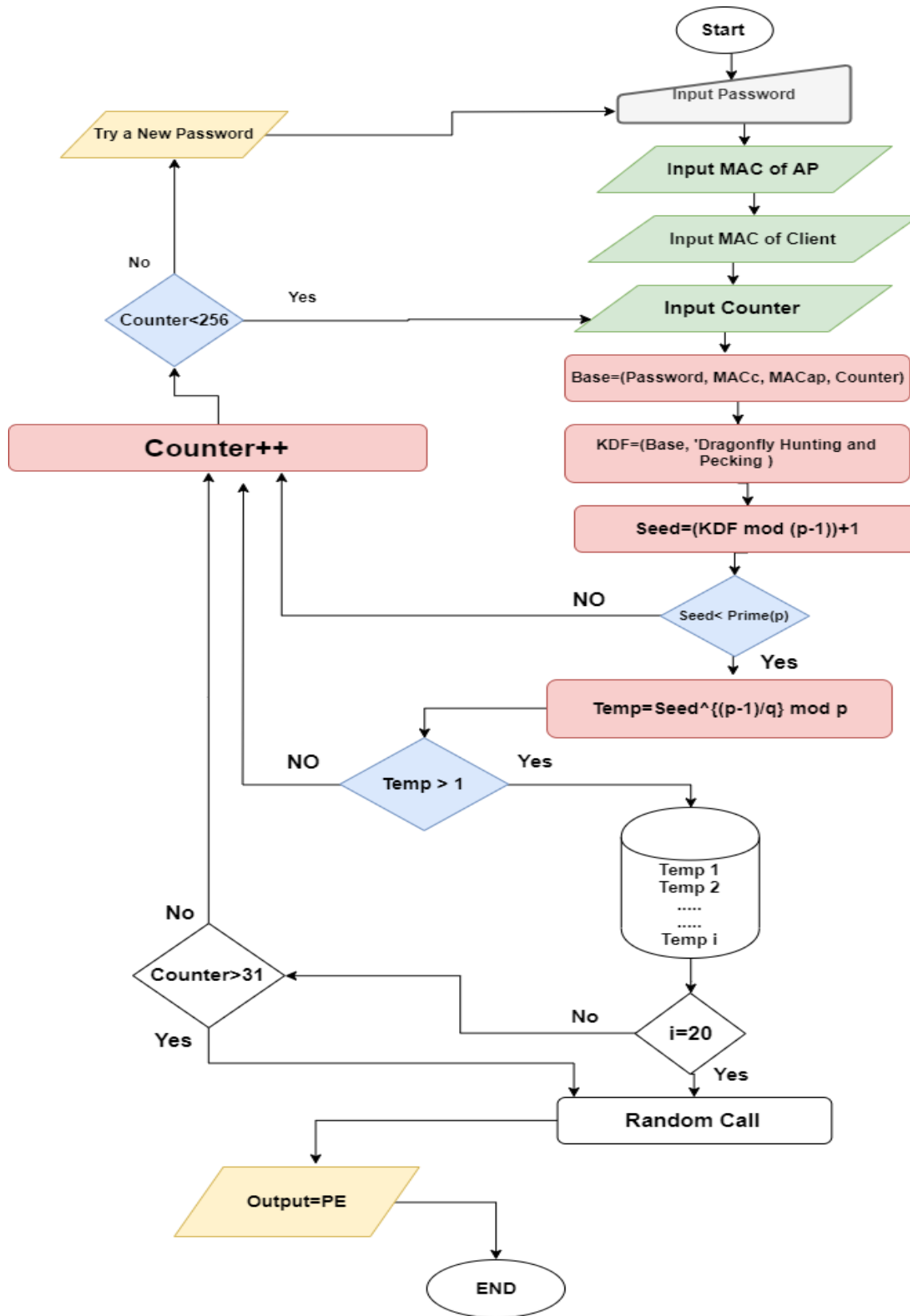


Figure 5.2 Flow Chart of Integrated Proposed Scheme

### 5.3 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. The parameters are given below [19] .

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 F55BCCCC  
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



## **5.4 Environment**

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 in mili second. Different kinds of libraries, modules and functions like Haslib, Random, Binascii, and Crypto were used in the script.

## CHAPTER 6 : RESULT AND ANALYSIS

### 6.1 Variation in Execution Time

Variations in loops or iterations are needed for the MODP algorithm to convert the different password into password element. 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.

Table 6-1 Variation in Execution Time

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

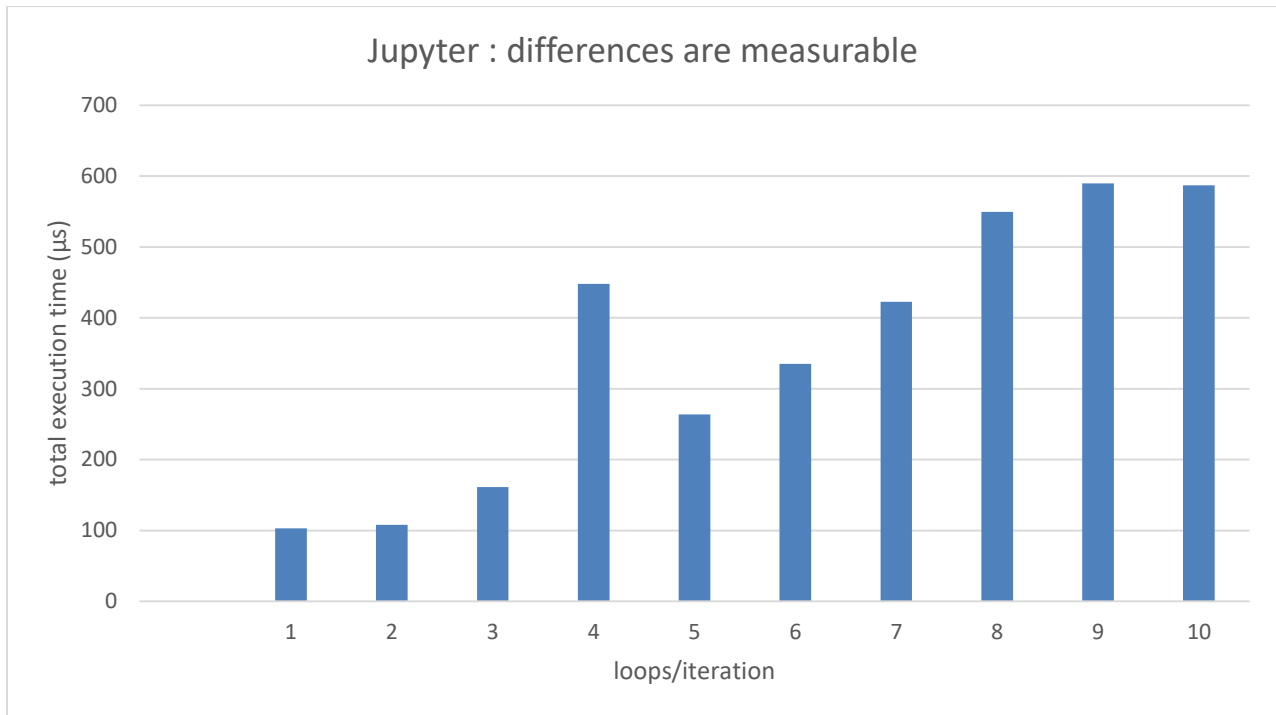


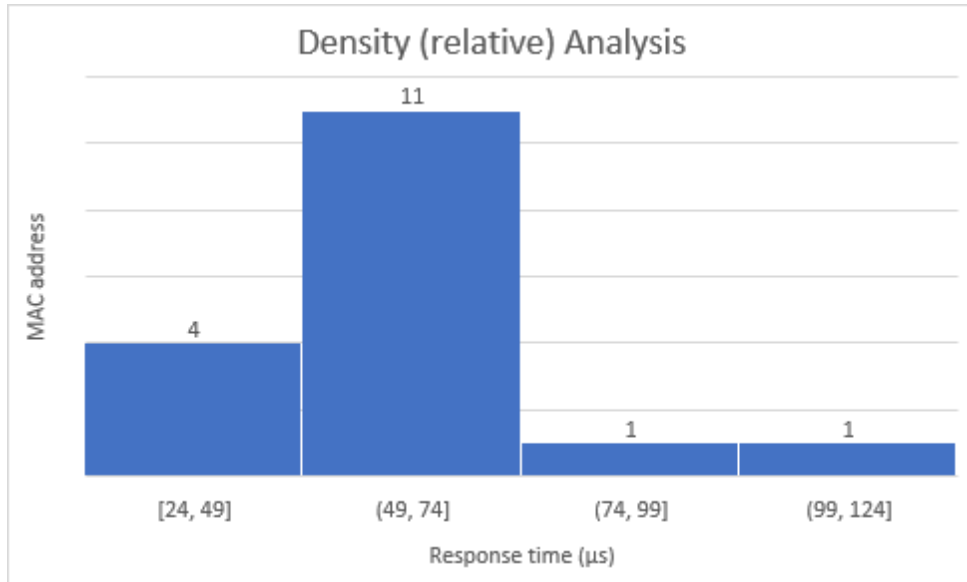
Figure 6.1 Graph of Iteration Vs Execution Time

## 6.2 Guessing Password using Different Client MAC Address

Here the Mac address of the access point is  $MAC_{AP} - 00:10:6A:44:12:B4$ . The client MAC address was changed gradually from number 1 to 17, these MAC addresses were used randomly only for testing purpose in the script. Finally it was 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 was same all the time.

Table 6-2 Execution Time against Different MAC Addresses

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



**Figure 6.2 Density Analysis of the Response Time against Different MAC Addresses**

From figure 6.2 it was found that maximum number of response time against different client MAC address was between 49 to 74 micro seconds. Vital information about the password was leaked from the information because it has become less difficult for an attacker to guess a password from the brute force dictionary. The words will be chosen then those are executed between the leaked time information. The actual password was executed near about 65 micro seconds with some standard deviation in the script.

### 6.3 Certain Number of Password Elements from Same Password

Table 6-3 Different PE from Same Password (a)

Serial No.	Password	N <sup>th</sup> Number of Iteration	Hex Code of Password Element	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	0xa7d61a8610516	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	0x8df17825e0163	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

Table 6-4 Different PE from A Same Password (b)

Serial No.	Password	N <sup>th</sup> Number of Iteration	Hex Code of Password Element	Execution Time Per Iteration (mean± std. dev. )	Total Execution Time
12	TAposh24#9 2 ^!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	0x336f8ec3cdcd9	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 approach which is proposed and designed in this work is to generate multiple password elements from a single password. All those PE in Table 6-3 and Table 6-4 were generated at different iteration with different execution time but the password was always same. In this work total 20 PE were generated from 31 iterations and the final PE was provided after 31 iterations at near about 2500 micro seconds. A graphical representation of different iterations and execution time is shown in Figure 6.4.

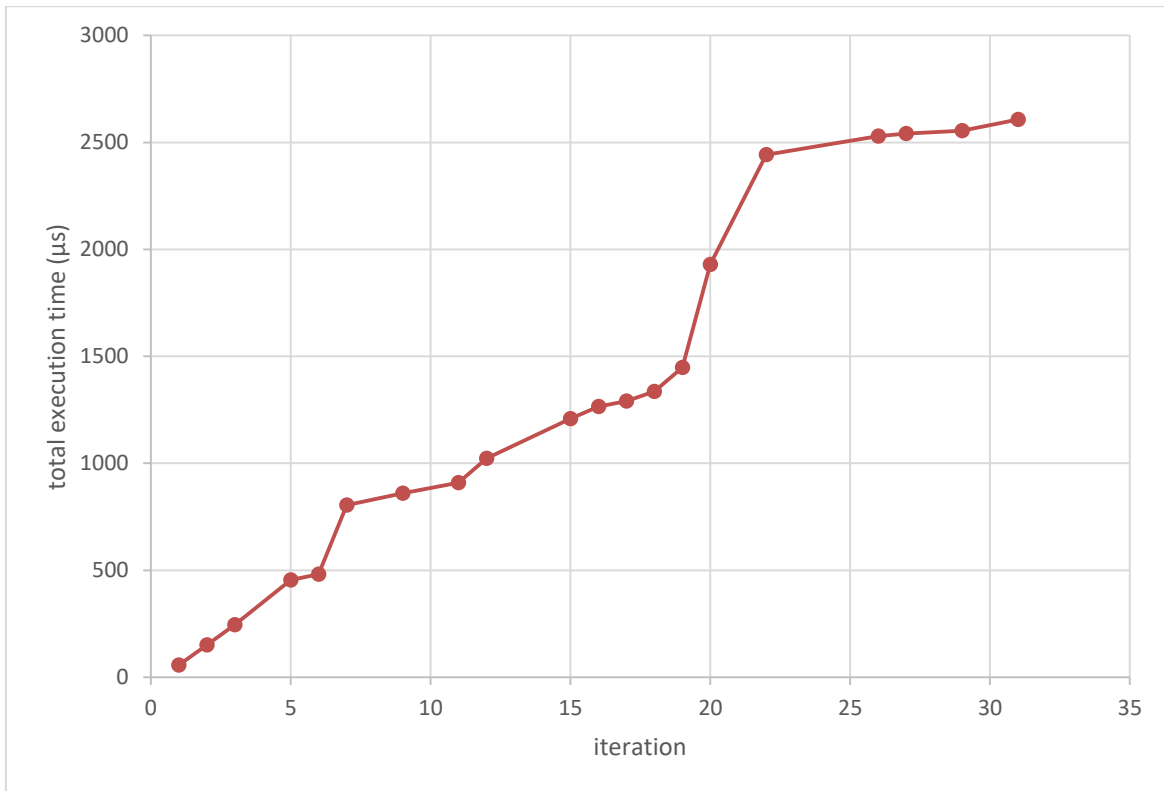


Figure 6.4 Iteration vs Total Execution Time of Table 6-3 and 6-4



## 6.4 Random Call

According to the proposed approach five random calls were done from the total number of generated password elements. Random module from Python 3.7 was used for this purpose. Different random call outputs, different PE generated at different iteration and time at a certain time is shown in Table 6-5.

Table 6-5 Random Calls

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

### 6.4.1 Random Call 1

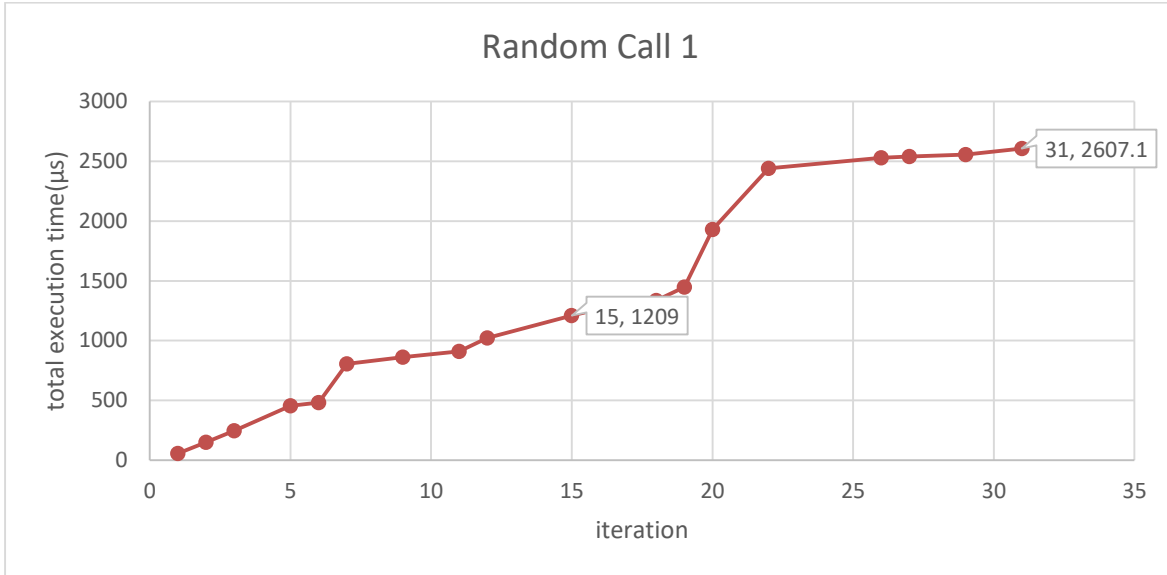


Figure 6.3 Random Call 1 (a)

From random call 1 the PE= 0x16d9048e95de4b was found, which was actually executed at 15<sup>th</sup> iteration, The PE was generated at 1209 μs but the PE was exposed at near about 2607.1 μs.

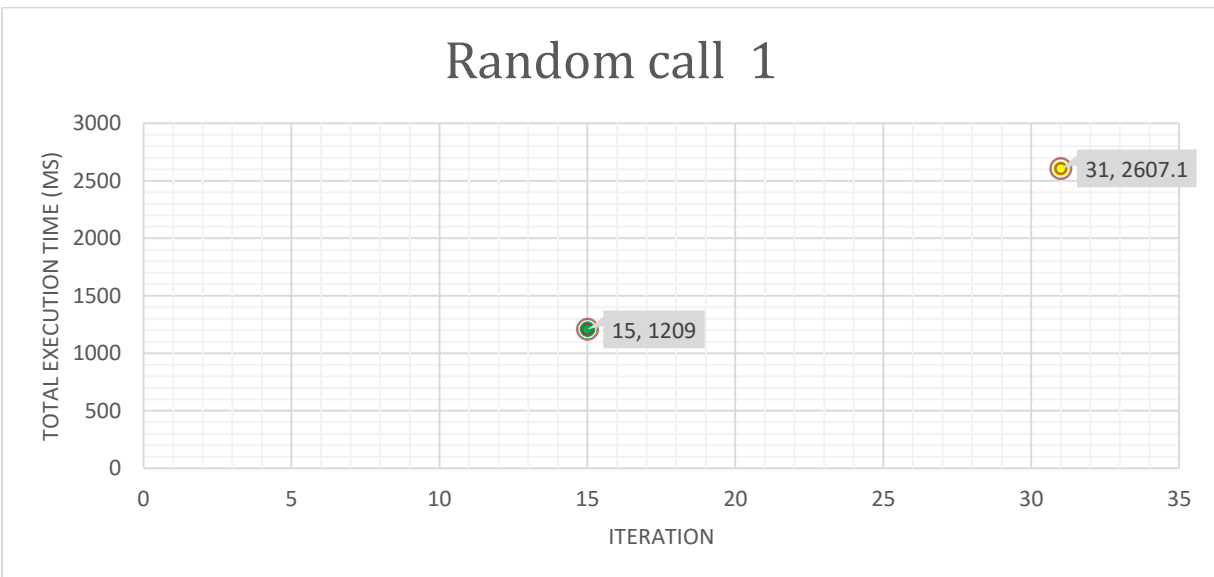


Figure 6.4 Random Call 1(b)

Here Green point denotes generation point of PE and Yellow point denotes execution point of PE.

## 6.4.2 Random Call 2

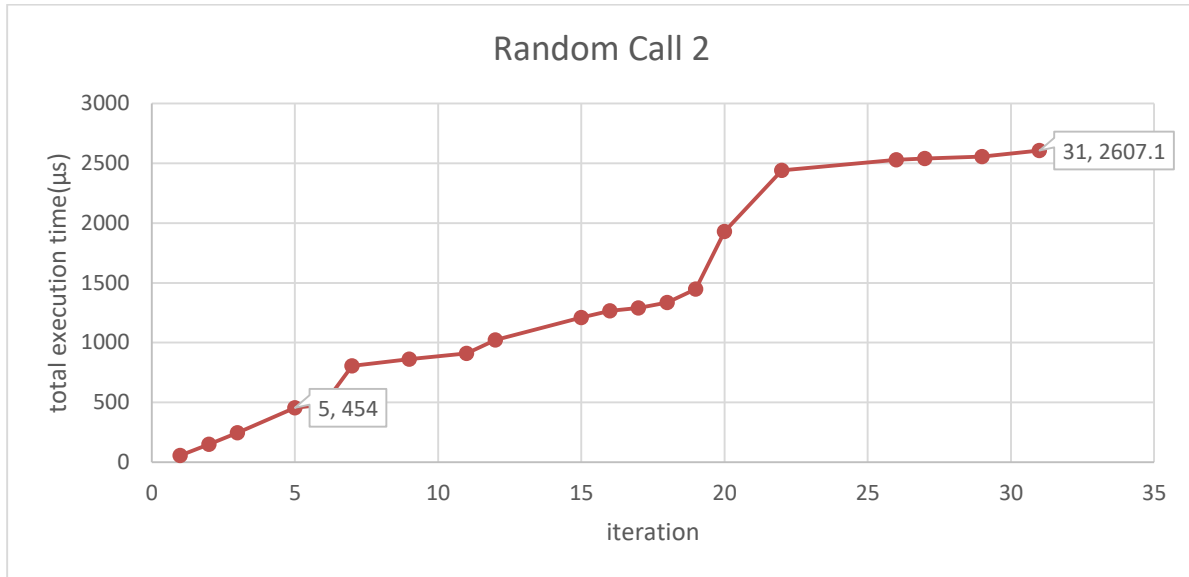


Figure 6.5 Random Call 2 (a)

From random call 1 the PE= 0xdba5126ad68770 was found, which was actually executed at 5<sup>th</sup> iteration, The PE was generated at 454 μs but the PE was exposed at near about 2607.1 μs .

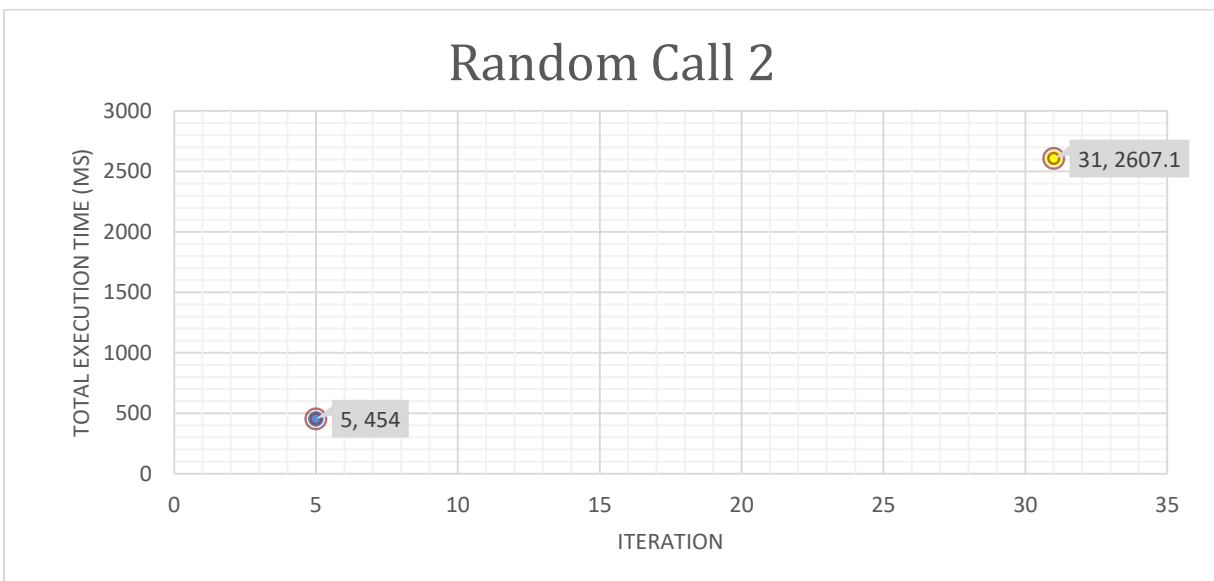


Figure 6.6 Random Call 2 (b)

Here Blue point denotes Generation Point of PE and Yellow point denotes Execution point of PE.

### 6.4.3 Random Call 3

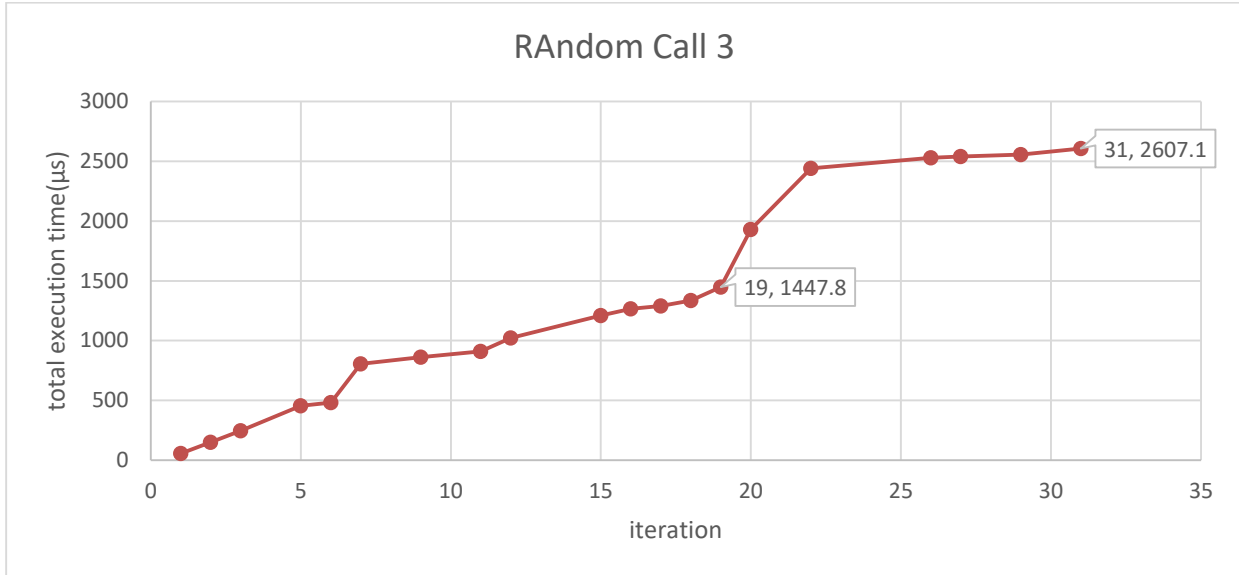


Figure 6.7 Random Call 3 (a)

From random call 1 the PE= 0x252af6b9733c8a was found, which was actually executed at 19<sup>th</sup> iteration, The PE was generated at 1447.8 μs but the PE was exposed at near about 2607.1 μs.

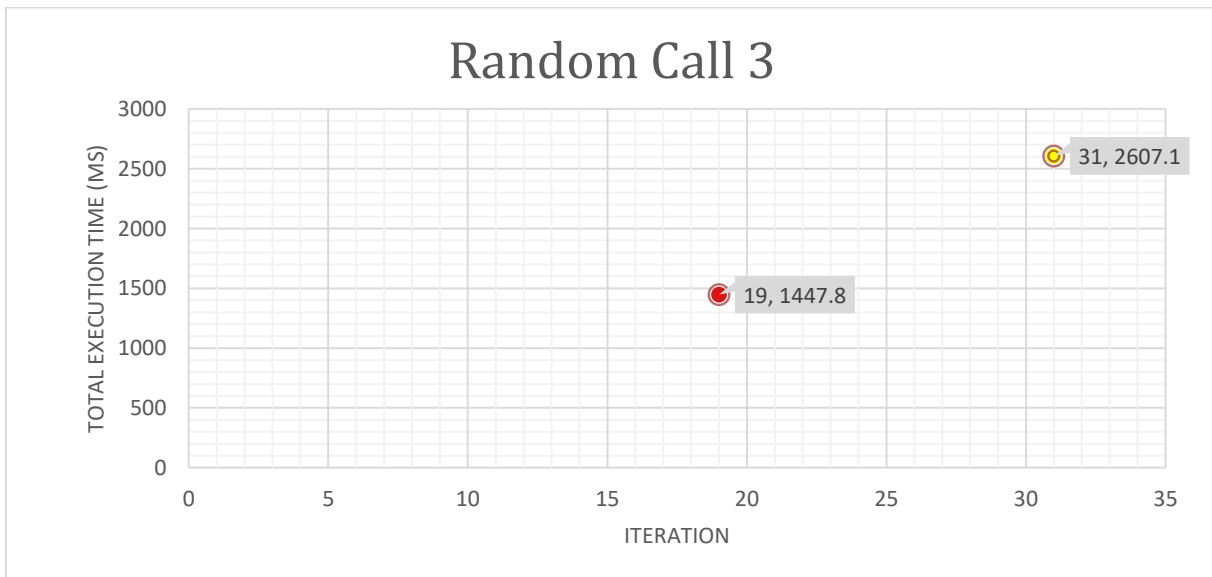


Figure 6.8 Random Call 3 (b)

Here Red point denotes Generation Point of PE and Yellow point denotes Execution point of PE

### 6.4.4 Random Call 4

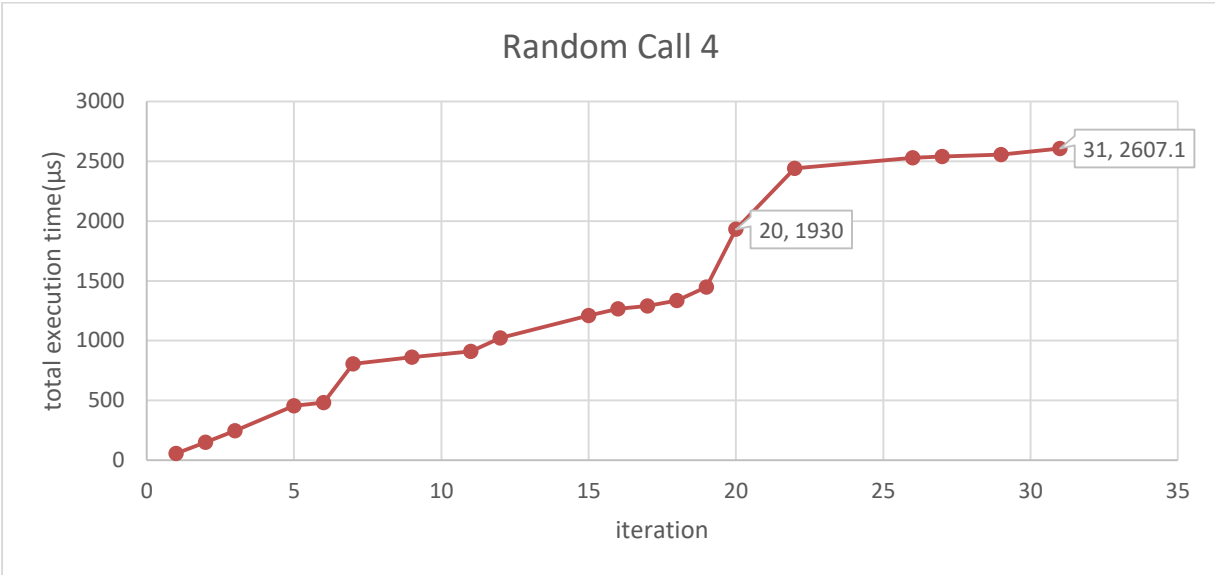


Figure 6.9 Random Call 4 (a)

From random call 1 the PE= 0x3cf16151e3c292 was found, which was actually executed at 20<sup>th</sup> iteration, The PE was generated at 1930μs but the PE was exposed at near about 2607.1 μs.

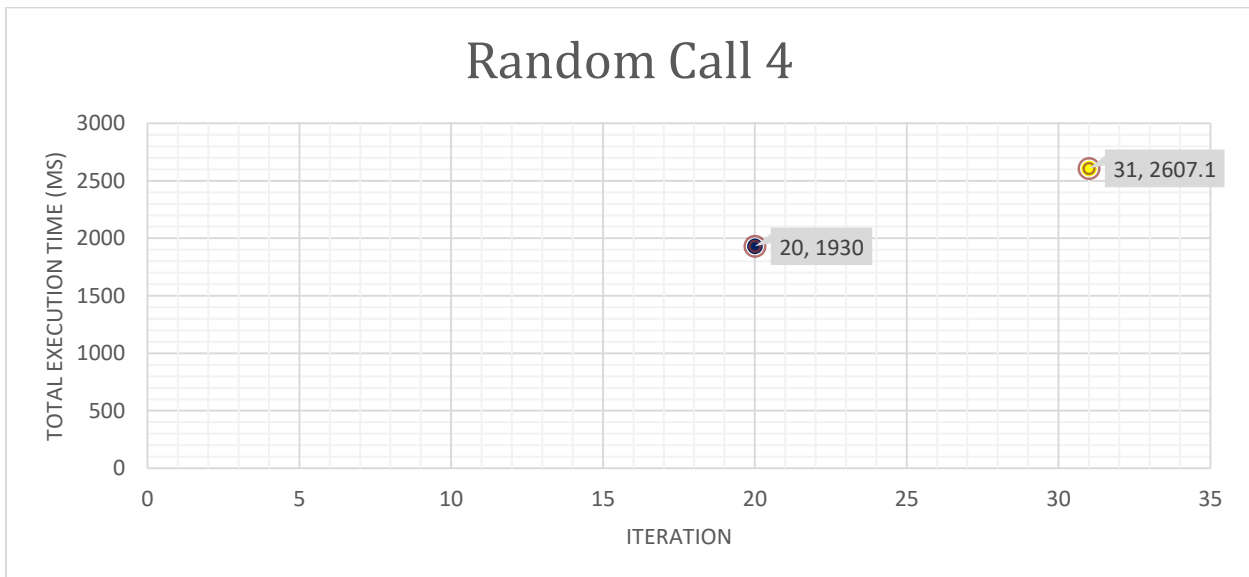


Figure 6.10 Random Call 4 (b)

Here Black point denotes Generation Point of PE and Yellow point denotes Execution point of PE.

### 6.4.5 Random Call 5

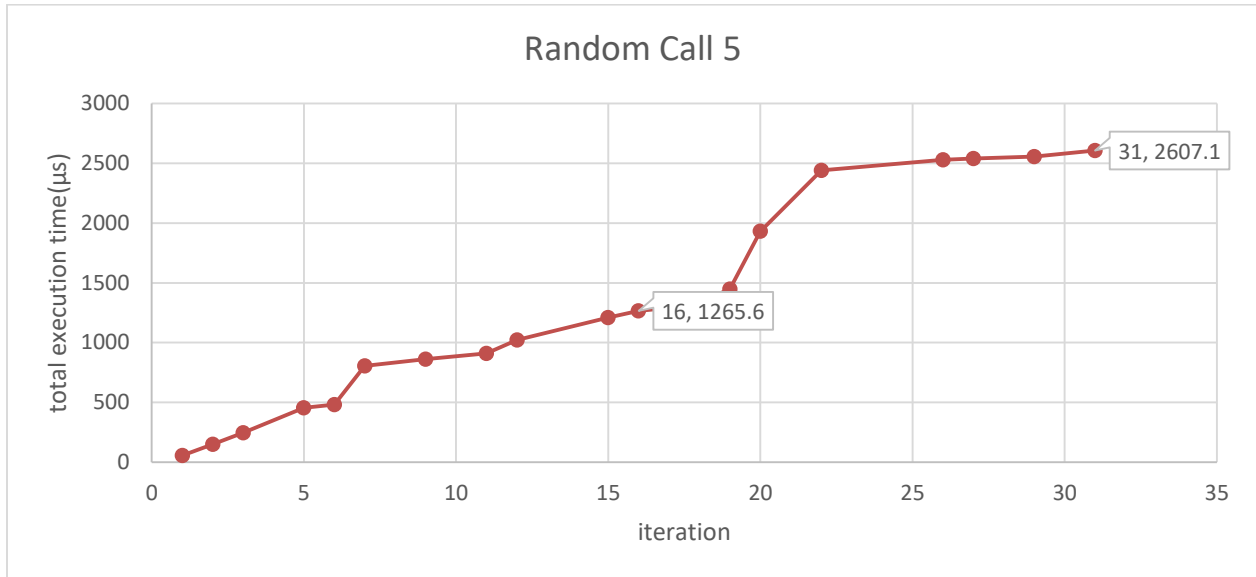


Figure 6.11 Random Call 5 (a)

From random call 1 the PE=0x7151a7133a44c4 was found, which was actually executed at 16<sup>th</sup> iteration, The PE was generated at 1265.6 μs but the PE was exposed at near about 2607.1 μs.

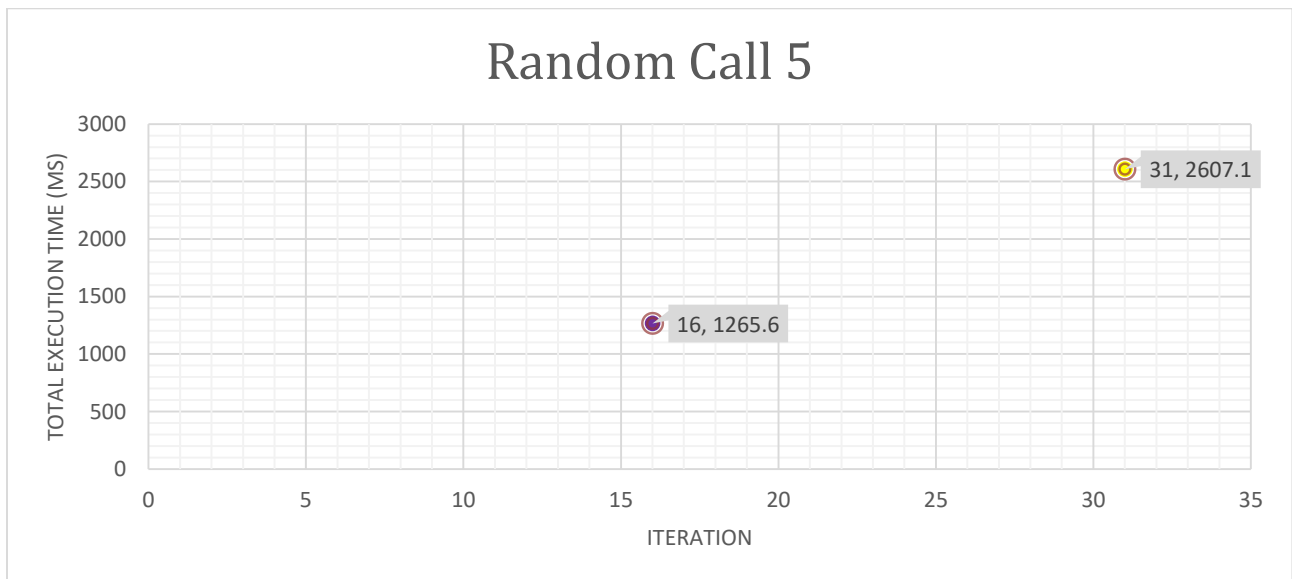


Figure 6.12 Random Call 5 (b)

Here Purple point denotes Generation Point of PE and Yellow point denotes Execution point of PE.

## **CHAPTER 7 : CONCLUSION**

In Conclusion, it can be said that generation of a number of PE from a single password at different iteration with different execution time is possible. Table 6-5 stated that a random call outputs an unpredictable PE from the database of generated PE. The outputs PE's information is totally unknown for an adversary as, the PE is nothing but a random number to attacker on which he/she can do reverse engineering to find the actual password. Though one can get a timing information from the rejection message but the probability of being the execution time is the exact execution of the certain PE is  $1/i$  . Here  $i$  is the number of PE generated, the probability is 0.05. By increasing the number of  $i$ , the probability can be achieved zero. The approach requires some extra memory to store the PE. In future it can be practically deployed on WPA3 access point or devices. Making it compatible with the existing devices will be another exciting work.

## **CHAPTER 8 : REFERENCES**

- [1] M. M. S. D. B. S. Arash Habibi Lashkari, "A survey on wireless security protocols (WEP, WPA and WPA2/802.11i)," 2009.
- [2] M. V. a. F. Piessens, "Key Reinstallation Attacks:," *Forcing Nonce Reuse in WPA2. In CCS.*, 2017.
- [3] D. G. a. N. H. Yuval Yarom, "Cache Bleed:A," *Timing Attack on Open SSL Constant Time RSA. In CHES (Lecture Notes in Computer Science)*,, vol. 9813, no. 346-367, 2016.
- [4] F. G. G. a. D. R. A. Oscar P. Sarmiento, " Basic security measures for IEEE 802.11 wireless networks," 2008.
- [5] M. Buczkowski, "<https://www.grandmetric.com/2018/07/06/ended-wpa3-wi-fi-security-evolution/>," grandmetric, 06 08 2018. [Online]. Available: <https://www.grandmetric.com/2018/07/06/ended-wpa3-wi-fi-security-evolution/>. [Accessed 2019].
- [6] "WPA3 AND ENHANCED OPEN: NEXT GENERATION WI-FI SECURITY," [Online].
- [7] Mathy Vanhoef and Frank Piessens, "Key Reinstallation Attacks: Forcing Nonce Reuse in WPA2," in *Computer and Communications Security (CCS)*, Dallas, TX, USA., 2017.
- [8] S. F. S. Feng Hao, "The SPEKE Protocol Revisited," *The SPEKE Protocol Revisited*, 2003.
- [9] N. A. a. O. B. Eman Alharbi, " An Enhanced Dragonfly Key Exchange Protocol against Offline Dictionary Attack.," *Journal of Information Security* 6, vol. 2, no. 69, p. 2, 2015.
- [10] D. Harkins., "Secure Pre-Shared Key (PSK) Authentication for the Internet Key Exchange Protocol (IKE)," *Secure Pre-Shared Key (PSK) Authentication for the Internet Key Exchange Protocol (IKE)*, vol. 6617, 2012.
- [11] D. Harkins, *Dragonfly Key Exchange*, Internet Research Task Force (IRTF) , November 2015.
- [12] D. Harkins, *Dragonfly Key Exchange. RFC 7664*, vol. 7664, 2015.
- [13] M. L. a. S. Kent., " Additional Diffie-Hellman Groups for Use with IETF Standards," *Additional Diffie-Hellman Groups for Use with IETF Standards*, vol. 5114, 2008.
- [14] D. J. Bernstein., " Cache-timing attacks on AES.," *Cache-timing attacks on AES*, 2005.



- [15] "MACs and hash functions: State of the art.," 1997. [Online]. Available: [https://doi.org/10.1016/S1363-4127\(97\)81327-1](https://doi.org/10.1016/S1363-4127(97)81327-1).
- [16] I. S. 8. Amendment 10, *IEEE Std 802.11s. 2011. Amendment 10: Mesh Networking.*, 2011.
- [17] Mathyvanhoef, "<https://wpa3.mathyvanhoef.com>," [Online].
- [18] M. V. a. E. Ronen, "Dragonblood: Analyzing the Dragonfly Handshake of WPA3 and EAP-pwd.," in *IEEE Symposium on Security and Privacy*, Oakland (San Francisco), May 2020.
- [19] S. K. M. Lepinski, *Additional Diffie-Hellman Groups for Use with IETF Standards*, IETF, January 2008.
- [20] [Online]. Available: <https://tools.ietf.org/html/rfc7664#page-12>.

## CHAPTER 9 : APPENDIX

Short form	Main word
AES	Advanced Encryption Standard
AP	Attack on Password
BIP-GMAC-256	256-bit Broadcast/Multicast Integrity Protocol Galois Message Authentication Code
CCMP	Cipher block Chaining Message Authentication Code Protocol
CRC32	32-bit Cyclic Redundancy Check
CT	Cypher text
DES	Data Encryption Standard
ECC	Elliptic Curve Crypto
ECDH	Elliptic Curve Diffie-Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm.
FFC	Finite Field Crypto
GCM	Galois/Counter Mode
GCMP-256	256-bit Galois/Counter Mode Protocol
GMAC	Galois Message Authentication Code
GTK	Group Temporal Key
h	Hash code
H	Hash function
HMAC	Hashed Message Authentication Mode
ICV	Integrity Check Vector
ID	Peer's identities
IGTK	Integrity Group Temporal Key
IKM	Intermediate keying material
KBKDF	Key Based KDF
KDF	Key Derivation Function
OWE	Opportunistic Wireless Encryption
p	Prime
PAKE	Password authenticated key exchange

PBKDF2	Password Based KDF2
PE	Password element
PrA	Private Key of A
PuA	Public key of A
PTK	Pair wise Temporal Key
q	Order from the domain parameter set
SHA	Secure Hash Algorithm
SPEKE	Simple Password Exponential Key Exchange
SAE	Simultaneous Authentication Equals

